Duplicate Detection in Probabilistic Relational Databases

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik von
Dipl.-Inform. Fabian Panse
aus Homberg/Efze

Hamburg, Juli 2014

In einer Welt voller Unsicherheit muß man eine Menge Dinge ausprobieren. Man kann nur hoffen, daß einige davon funktionieren

von Douglass North

Gutachter:

Prof. Dr. Norbert Ritter (Universität Hamburg)

Prof. Dr. Wolfgang Menzel (Universität Hamburg)

Prof. Dr. Ingo Schmitt (Brandenburgische Technische Universität Cottbus)

Tag der Disputation: 17. Dezember 2014

A revised version of this thesis can be found at
http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:gbv:18-228-7-211

Ich widme diese Arbeit meinen Eltern Michael und Viola, meinen Geschwistern Christina und Benjamin, sowie meinen Großeltern Günter, Rosemarie, Willy und Anita.

Zusammenfassung

Viele Anwendungen wie Texterkennungsverfahren oder Sensornetzwerke arbeiten häufig auf ungenauen, unscharfen, oder zweifelhaften Informationen. Ein aktueller Trend in der Datenbankforschung ist es, solche Unsicherheiten als wertvollen Bestandteil des Anwendungsergebnisses zu betrachten und somit die Ausgaben dieser Anwendungen entsprechend anzureichern. Ein weit entwickeltes Werkzeug zur Modellierung von Unsicherheiten in Anwendungsdaten sind probabilistische Datenbanken. Um verschiedene, heterogene probabilistische Datenbestände zu integrieren oder um einen einzelnen probabilistischen Datenbestand von Inkonsistenzen zu bereinigen, bedarf es Verfahren, um Datenobjekte zu erkennen, die dasselbe Realweltobjekt referenzieren. Herkömmliche Verfahren zum Aufspühren solcher sogenannten Duplikate wurden bisweilen allerdings nur für Datenobjekte konzipiert, die durch einzelne, exakte Attributwerte beschrieben sind und deren Zugehörigkeit zum betrachteten Gegenstandsbereich als sicher angesehen wird. Probabilistisch modellierte Datenobjekte können jedoch mehrere Werte pro Attribut aufweisen. Zudem kann ihre Relevanz für den betrachteten Gegenstandsbereich infrage stehen. Aus diesen Gründen ist eine Verwendung herkömmlicher Duplikaterkennungsverfahren für probabilistische Datenbanken ohne wesentliche Anpassungen nicht geeignet.

In dieser Dissertation widmen wir uns dem Erkennen von Duplikaten in probabilistischen relationalen Datenbanken. Der zentrale Forschungsaspekt dieser Arbeit ist dabei der Entwurf eines generischen Ansatzes, der eine Anpassung an individuelle Bedürfnisse erlaubt und somit ein Erkennen von probabilistischen Duplikaten in unterschiedlichen Anwendungsbereichen ermöglicht. Ein Vorteil probabilistischer Datenbanken ist, dass wir nicht gezwungen sind, auftretende Unsicherheiten über Duplikatsbeziehungen aufzulösen und solche Unsicherheiten stattdessen im Duplikaterkennungsergebnis modellieren können. Eine Vielzahl weit verbreiteter probabilistischer Datenmodelle, wie z.B. Tupel-unabhängige Datenbanken, sind allerdings nicht mächtig genug, um Unsicherheiten über Duplikatentscheidungen modellieren zu können. Aus diesem Grund unterscheiden wir zwischen deterministischen und indeterministischen Duplikaterkennungsverfahren. Erstere zeichnen sich dadurch aus, dass sie jegliche Unsicherheiten über Duplikatentscheidungen beseitigen, indem sie ein einziges Duplikat-Clustering als Ausgabe liefern. Das Ergebnis eines indeterministischen Verfahrens besteht hingegen aus einer Menge an möglichen Duplikat-Clusterings, die jeweils mit einer Wahrscheinlichkeit versehen sein können.

Wir identifizieren zwei sinnvolle Strategien, um herkömmliche Duplikaterkennungsverfahren an die inhärenten Unsicherheiten probabilistischer Daten anzupassen. Basierend auf diesen Strategien entwerfen wir zwei generische Ansätze zur deterministischen Duplikaterkennung in probabilistischen Datenbanken und präsentieren verschiedene Techniken, um die Berechnungskomplexität dieser beiden Ansätze zu verringern. In diesem Zusammenhang entwickelen wir ein Ähnlichkeitsmaß für diskrete Wahrscheinlichkeitsverteilungen, das eine performante Alternative zur Earth Mover's Distance darstellt.

Hinsichtlich des Konzepts der indeterministischen Duplikaterkennung beginnen wir mit einer formalen Definition. Anschließend präsentieren wir verschiedene Ansätze, die es gestatten, eine Menge von möglichen Duplikat-Clusterings in einer probabilistischen Datenbank zu modellieren und diskutieren mögliche Arten und Weisen in denen ein solches Deduplizierungsergebnis verarbeitet werden kann. Zudem präsentieren wir ein Clustering-Verfahren, das eine effiziente Berechnung von indeterministischen Duplikaterkennungsergebnissen ermöglicht. Darüber hinaus betrachten wir die Bedeutung der Qualität eines Duplikaterkennungsprozesses in Anbetracht unsicherer Duplikatentscheidungen, konzipieren verschiedene Maße, um diese Qualitätsinterpretationen zu beziffern und präsentieren Methoden, um diese Maße kostengünstig zu berechnen.

Den Abschluß dieser Arbeit bildet eine experimentelle Evaluierung, in der wir zunächst eine prototypische Implementierung präsentieren und dann verschiedene experimentelle Ergebnisse diskutieren.

Abstract

Many applications such as OCR systems or sensor networks have to deal with uncertain information. One trend in current database research is to accept uncertainty as a 'fact of life' and hence to incorporate it into such applications' results by producing probabilistic output data. To meaningfully integrate probabilistic data from multiple heterogeneous sources or to clean a single probabilistic database, duplicate database entities need to be identified.

Duplicate detection has been extensively studied in the past, but conventional duplicate detection approaches are designed for matching database entities that are described by certain values and certainly belong to the considered universe of discourse. In probabilistic databases, however, each database entity can have several alternative values per attribute and the membership of an entity to a universe can be questionable. As a consequence, conventional duplicate detection approaches cannot be used for probabilistic databases without adaptation.

In this thesis, we consider the challenge of duplicate detection in probabilistic relational databases. The central research aspect of this thesis is to develop a generic approach that enables detection of probabilistic duplicates in highly diverse application domains by allowing an adjustment to individual needs. The benefit of using a probabilistic database for modeling deduplication results is that we do not necessarily need to resolve uncertainty on duplicate decisions, but instead can incorporate emerging decision uncertainty into the output database. Nevertheless, many commonly used probabilistic representation systems such as tuple-independent probabilistic databases are not powerful enough to model uncertainty on duplicate decisions. For that reason, we distinguish between deterministic duplicate detection approaches that completely resolve uncertainty on duplicate decisions by producing a single duplicate clustering as a result and indeterministic duplicate detection approaches that provide a set of possible duplicate clusterings as output.

We identify two meaningful strategies for adapting conventional duplicate detection approaches to the uncertainty that is inherent in probabilistic data. According to these strategies, we propose two generic approaches for deterministic duplicate detection in probabilistic databases and present several techniques for reducing their computational complexity. In this context, we develop a similarity measure for discrete probability distributions that can be used as a fast alternative to the Earth Mover's Distance.

Additionally, we formalize the concept of indeterministic duplicate detection, propose approaches for representing an indeterministic deduplication result within a probabilistic database, discuss possible ways to meaningfully process indeterministic deduplication results, and present a clustering approach that can be used to efficiently compute a set of possible clusterings. Moreover, we discuss the meaning of detection quality in the presence of uncertain duplicate decisions, present measures for rating this meaning by numbers, and propose methods to compute these measures in an efficient way.

Finally, we present a prototypical implementation and the results of a set of experiments we conducted on several test databases in order to prove the concepts of our proposed approaches.

Danksagung

Diese Dissertation wäre ohne das Mitwirken und die Unterstützung einiger Personen nicht zustande gekommen. Der größte Dank gilt meinen Eltern Michael und Viola sowie meinen Geschwistern Christina und Benjamin, die mich während der vielen Jahre auf dem Weg zu meiner Promotion auf jede erdenkliche Art und Weise unterstützt haben. Großer Dank gilt auch Norbert, der mich mit den nötigen Ratschlägen durch meine Promotion führte, mir von Anfang an viele Freiheiten ließ, und mir gerade im letzten Jahr ein ruhiges und fokussiertes Niederschreiben dieser Dissertation ermöglichte.

Eine wichtige Stütze in all diesen Jahren war Anne. Sie stand mir immer mit Rat und Tat zur Seite und war jederzeit zu einem aufmunternden oder erfrischenden Gespräch bereit.

Bei Maurice mochte ich mich dafür bedanken, dass er es mir ermöglichte, einmal das Doktorrandenleben und den Forschungsalltag im Ausland kennen lernen zu dürfen. Des Weiteren bedanke ich mich für die vielen inspirierenden Diskussionen, die wir im Laufe der letzten Jahre führen konnten und hoffe, dass noch viele weitere folgen werden.

Mein Dank gilt auch zahlreichen Kollegen, die mich im Laufe der Zeit begleitet haben: Martin, der mich in die Gepflogenheiten des Mitarbeiter- und Doktorandenlebens einführte, Michael, Marc und Kristof, mit denen man sich jederzeit über alle möglichen Dinge unterhalten konnte, Kathleen, die immer für ein Lachen zu haben war, Wolfram und Steffen, die mir immer unterstützend zur Seite standen, Felix, der zwar allerlei Schmackes in den Armen hat, aber noch viel von mir lernen kann, Ante und Christoper für die vielen interessanten Gespräche über die Welt des Sports, Lars und Alexander, die dank ihrer langen Erfahrung immer einen guten Rat geben konnten, Volker, Kai und Claas für die technische Unterstützung, Sonja, die einfach Sonja ist, und natürlich Dirk, der jederzeit ein offenes Ohr für mich und meine Probleme hatte und mit dem man immer wunderbar scherzen kann.

Ich bedanke mich bei Dirk, Felix, Kai, Frank, Christian und Christina, die einen prüfenden Blick auf diese Ausarbeitung warfen und mir hilfreiche Hinweise gaben.

Außerdem bedanke ich mich bei all den Studierenden, die mit ihren wissenschaftlichen Arbeiten und/oder anregenden Diskussionen zu dieser Dissertation beigetragen haben.

Natürlich gab es auch viele Menschen, die mein Leben abseits des universitären Alltags geprägt und mir die nötige Abwechslung und Lebensfreude verschafft haben. Ich danke hierbei besonders Christian, Lennart, Tuba, Monique, Jessica und Viola.

Fabian Panse

Contents

Chapter 1

Introduction

Relational database systems have been playing an important role in digital information management for a long time because they provide mechanism to persist, query, and analyze data in an effective and efficient way. Conventional relational database systems [Cod83, UGMW02] are designed to deal with *deterministic* (*certain*) data that for example are produced by financial transactions, inventories, or customer relationship management systems. In such cases, every database tuple belongs to its corresponding database table with absolute certainty and fills each of the table's attributes with a single element of the attribute's domain. To deal with missing data, the certain data assumption is usually relaxed by the utilization of null values [Cod86, Cod87, UGMW02], i.e. special constructs that can be used to replace an attribute value if this value is not available.

Nowadays, however, many applications produce data as output that are not deterministic, but are rather incomplete, imprecise, or vague. Moreover, null values are usually not sufficient to deal with such uncertainty in an appropriate way [Pan09b]. For that reason, various extensions of the relational data model have been developed during the last decades that are based on different concepts of modeling uncertain information including fuzzy theory [Zad65] and probability theory [Kol60]. These approaches aim to represent uncertainty within the database without loosing the great benefits of relational database systems like vertical scalability and the support of complex queries. From all these approaches, probabilistic relational database systems [CP87, BGMP92, DS96, FR97, BSHW06, SORK11] attract the most attention in the database research community in recent years. The underlying idea of uncertain database systems is to manage and query a set of alternative database instances (called *possible worlds*) instead of a single database instance [DS07c]. Probabilistic database systems extend this concept by ascribing probabilities to the individual possible worlds. Although an incorporation of uncertainty into data representation and data querying increases complexity, probabilistic database systems can provide powerful mechanism for efficient data management [SORK11].

It is obvious that probabilistic databases suffer from similar quality problems as conventional databases and therefore can lack consistency. An often considered type of data consistency concerns the unique representation of real-world entities like persons or products within the database. In this case, a database becomes inconsistent if a real-world entity is mistakenly modeled in the database for several times and hence if the database is corrupted by so-called *duplicates* [EIV07, NH10, Chr12]. In many applications, duplicates can have disastrous consequences. For instance, in the medical area a non-detected

duplicate can lead to a situation where a patient gets several non-compatible medications and hence her health or even her life can be threatened by the unintended interdependencies between the administered medications. In order to ensure consistency, the database needs to be cleaned from duplicate entity representations [EIV07, NH10, Chr12].

Another important application for duplicate detection is an integration of data that originate from numerous autonomic sources [Len02, HRO06, LN06, BJ07, HDI12]. If no universal identifier is available or if the values of such an identifier are often missing or error-prone, duplicate detection is required to meaningfully link tuples from the different sources that capture (maybe different) aspects of the same real-world entities. As a consequence, methods for detecting duplicates in probabilistic databases are an essential step towards a meaningful integration of probabilistic data.

Finally, duplicate detection can be used to identify semantic correspondences between the schemas of different databases [BN05, Bil06] and therefore can be beneficial to improve the quality of schema matching [RB01, BBR11] in particular and data integration in general.

Duplicate detection is a well-known operation for conventional (certain) databases [EIV07, NH10, Chr12], but to this day got only less attention for probabilistic databases [MBGM06, PvKdKR10]. However, whereas some detection aspects are quite similar for certain databases as well as probabilistic databases and therefore can be adopted with only less adaptations, duplicate detection in probabilistic databases poses some new challenges; functional as well as semantical. In this thesis we analyze the semantics of duplicate detection in probabilistic databases and propose approaches for effectively and efficiently detecting duplicates in a probabilistic database based on the analyzed semantics.

This chapter is structured as follows. First we provide an insight into the variety of applications that produce probabilistic data or at least have to deal with probabilistic information for producing a deterministic outcome. Second, we shortly present the principles of probabilistic databases and duplicate detection on the basis of a simple example on criminal suspects. Then, we discuss the research questions that are considered in this thesis and point out several challenges that we need to face for answering these questions. Finally, we list the contributions that are provided by this thesis and give an outline of the remainder of this thesis.

1.1. Probabilistic Data Applications

Probabilistic database systems are designed to manage and query uncertain information in a structured way. Applications that produce (or at least need to deal with) uncertain information/data can be found in many areas.

- In sensor networks multiple sensors are coupled to produce a global picture on a considered scenario by combining a large number of local views. Nevertheless, it can happen that different sensors report contrary data values on the same objects. Moreover, sensors are not immune to measurement errors, especially if the considered objects dynamically change their observed state, e.g. moving objects in spatiotemporal databases [CP03, Tao09]. For that reason, the produced information is often imprecise and need to be managed and queried by taking this imprecision into account [CP03, CKP04, Tao09, YTX+10, EKM+12]. In RFID management [GBF+06, KBS08, TSC+09]

it can be useful to incorporate the noise of scanned RFID tags into the query results by ascribing confidence scores to the output tuples.

- In data integration, database schemas need to be matched and automatic schema matching approaches [RB01, BBR11] are far away from being perfect [Gal06b]. Since an introduction of manual effort is often not possible due to time or monetary cost requirements, some schema matching approaches (relational [MG07, DHY09, SDH09, Gal11a, ZCJC13] or XML [GCC12]) incorporate emerging uncertainty on correct matches into the resultant schema mappings. Another source of uncertainty in data integration is the elimination of duplicates, because it can be uncertain whether or not two database tuples represent the same real-world entity. Moreover, detected duplicates can have conflicting values and it is often not clear which merge of these values represents reality best. Beskales et al. [BSIBD09], Ioannou et al. [INNV10, Ioa11], van Keulen and de Keijzer [vKdK09], and Panse et al. [PvKR13] present approaches that incorporate uncertainty on duplicate decisions into the detection result. Several approaches [DeM89, T+92, AFM06, HM09, BCMP10, BBC+11] resolve conflicts in merging duplicate tuples by modeling all plausible values inside the resultant database.

- In the area of information extraction, applications aim to extract structured information from a spate of unstructured data. This area becomes especially important since the world wide web enables a collective access to the information that is provided by millions of webpages. Because the extraction of concrete information from a bulk of unstructured information is a hard challenge for automatic systems (sometimes even for humans), many extraction approaches as for example Conditional Random Fields [LMP01] make use of probabilistic models to capture all the plausible ways for interpreting an unstructured text. Information extraction approaches that incorporate emerging uncertainty into the extraction result have been described in several proposals [GS06, DBS09, WMF+10, vKH11].

  An interesting application of information extraction is the Never-Ending Language Learner[1] (short NELL) that extracts structured data of the form (*entity*,*relation*,*value*) from millions of unstructured webpages. Since the reliability of webpages as well as the correct interpretation of the parsed texts are uncertain, the resultant facts are annotated with confidence scores.

  Another example of applications that have to deal with uncertainty in the extraction process are those that strive to solve the *Name Disambiguation Problem* [FGL12] (also known as *Named Entity Recognition* [WMM10]) or the *Topology Disambiguation Problem* [HvK12] in particular. These applications try to map names that are mentioned in unstructured text document to the real-world entities the considered names actually refer to. For illustration, without analyzing context information it is often not clear whether the name 'Chelsea' refers to the identically named area of London, refers to the London football club 'F.C. Chelsea', or refers to 'Chelsea Clinton' who is a daughter of Bill Clinton a former president of the United States of America.

  Upcoming applications for information extraction are product analyses and market investigations because it can be useful to analyze the millions of twitter feeds in order to get feedback on already

[1] http://rtw.ml.cmu.edu/rtw/

sold products or to find a frequently required market niche [SORK11]. However, for automatic solutions it is often not clear whether a feedback is positive or negative (especially if the feed is maybe meant ironic) and it is often not obvious to which product a considered twitter feed actually refers to.

- A particular area of sensor-based applications for information extraction is the field of optimal character recognition (short OCR). The digital revolution is still young and many documents that get dusty in the archives of companies, public institutes, or libraries are manually written (or are written by type writers) and therefore are not digitally available yet. OCR systems get manuscripts or typewriter documents as input, scan these documents, and produce digital text documents as output. It is obvious that the automatic detection of correct words and numbers is challenging, if handwritings are hard to read or documents are old and their contents already start to fade out. For these reasons, OCR systems can benefit from using probabilistic data approaches [CKZ94, KR11] that interpret the scanned documents in several possible ways. Similar potentials hold for applications that aim to detect objects in photos and videos. Mandelbaum et al. [MKM98] propose a stochastic approach to identify objects in images and several proposals [BI98, ORP98] make use of probabilistic and/or statistical techniques to identify actions or interactions between objects in images. Gee et al. [GBBH95] use a Bayesian framework to solve the *atlas problem*, i.e. to match a template that represents the structural anatomy of the human brain with the anatomic brain images from a particular person.

- Many scientific applications produce uncertain data due to the inherent uncertainty of their measurements. Suciu et al. study the use of probabilistic databases in the field of astronomy [SCH09]. Several works in the area of bioinformatics show the suitability of uncertain database systems to operate on protein chemistry data [NJ02], in the prediction of protein functions [DGL+09, ZLGZ10], or to model protein-protein interactions [PBGK10]. Moreover, probabilistic databases can be useful to model uncertainty in biological image analyzation [LS09].

- A yet less considered but potential field for uncertain data producing applications is the field of criminal investigations. The statements of witnesses are often vague (e.g. 'the suspect was tall'), imprecise (e.g. 'the suspect was around forty years'), or conflicting due to subjective perceptions, poor memory powers, or intended lies. Evidences that are found on the crime scene as for example fingerprints, DNA evidences, or shoeprints are often of a poor quality and do not enable a conclusion of deterministic information. Moreover, a set of evidences can be usually interpreted in a variety of ways. In the digital era a large amount of criminal investigations is realized by analyzing web activities. Nonetheless, regardless of whether it is based on physical evidences or it is based on virtual evidences criminal investigations are often full of uncertainties and considering these uncertainties within the investigation process can increase the crime clearance rate to a large extent. As an example, credit card fraud detection using hidden Markov models has been considered by Srivastava et al. [SKSM08].

- Finally, in many areas such as risk assessment [AJP+10, JXW+08, JXW+11, XBE+09], business intelligence [SWF+09], or predictions of future events [QPB13] uncertainty does not arise be-

cause of imperfect data production, but is inherent in the nature of the produced data. In logistics, the transportation times need to be estimated by the shipping companies in order to determine appropriate delivery schedules. Other examples concern forecasting the traffic on streets or data networks, predicting prices on stock markets, or forecasting the number of cell phones within the coverage region of a cell phone tower [RSG05]. Moreover, in many areas, current prices are based on estimations on future demands. For instance, in deregulated electricity markets [Wol98] the price of electricity determined by a specific company depends on the predicted energy demand in an upcoming time period which in turn relies upon other uncertain factors such as the upcoming weather conditions, but the price also depends on other uncertain factors as for example the estimated bids of the company's competitors [RSG05]. Besides future demand and bids of competitors, prices on petrol additionally depend on complex factors including political situations (e.g. possible crisis in the middle-east) or the available reservoirs of fossil oil in particular areas. In general, specific events as forthcoming strikes or legislative changes can influence the process of price estimations in a variety of application areas.

As illustrated by the discussion presented above, the need for probabilistic data management and hence the need for probabilistic databases has been emerged in many application areas. Consequently, cleaning probabilistic databases or integrating probabilistic databases from different sources becomes more and more important in today's data management.

1.2. Probabilistic Data

Before going into detail on modeling uncertain information within a probabilistic database, we consider an illustrative scenario that produces uncertain information by integrating data that his gathered from several uncertain sources. This scenario is depicted in Figure 1.1 and considers the collection of information on criminal suspects whereby the information on the individual suspects originate from three different sources.

Witness Statements: First, we got information from witnesses. The information provided by the witnesses is rather vague than exact, because it is based on subjective perceptions, individual degrees of belief (the witness is possibly prejudiced), and 'sensors' of poor quality (e.g. the size or the age of a suspect is only estimated instead of exactly measured). The information provided by a witnesses can be distorted if the witness has a poor memory power. Moreover, we do not know whether or not we can trust the gathered information because it can be the case that some witnesses give untrue statements by purpose in order to cover up their own secrets. Despite all these uncertainties, witnesses can provide useful information on the age, the size, the gender, or the hair color of the wanted criminals.

Evidence on the Crime Scene: The second information source is the crime scene itself, because it often provides traces like fingerprints, shoeprints, or blood drops. Shoeprints can disclose the shoe size of the criminals, blood drops can be used to detect DNA sequences as well as blood types, and fingerprints are important evidences to identify criminals. In the real world, however, such evidences are often not in a perfect state. Fingerprints can be incomplete or blurred so that they

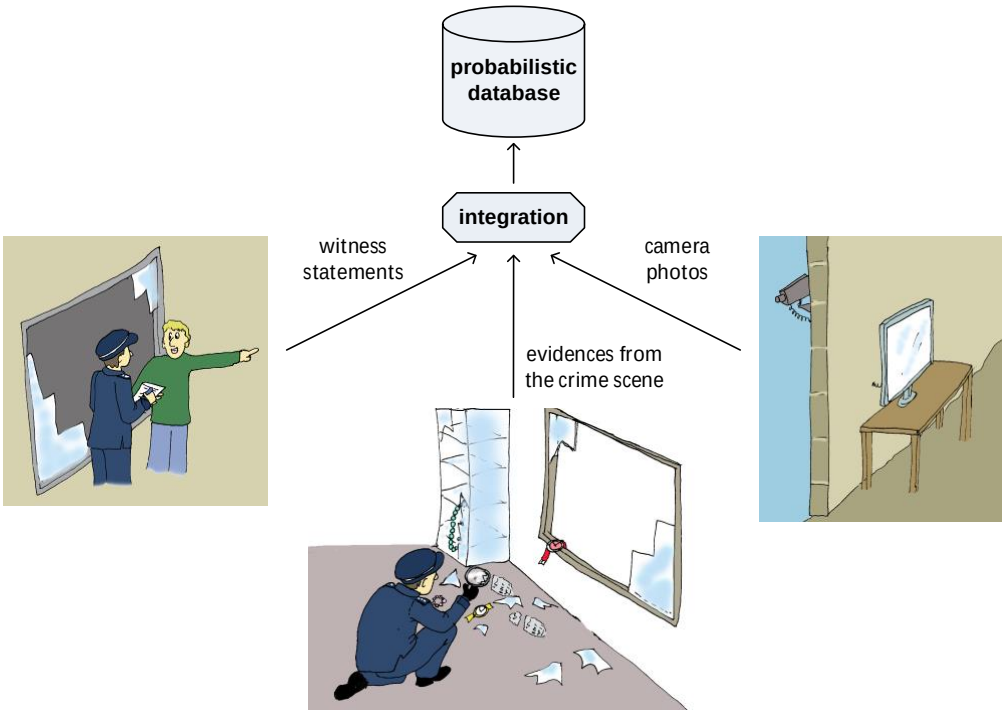


Figure 1.1.: Crime scenario[2]

cannot be used to clearly identify the criminal, but can be only used to exclude some persons from the suspect list for sure. Moreover, fingerprints that have been found on the crime scene must not necessarily belong to the wanted criminal, but can belong to any other person. In the case of blood traces we can extract a full DNA sequence. However, such a sequence is often useless in an early state of an investigation process because we have only less data to compare it. Nevertheless, such blood drops can be also used to extract the gender of the criminal or to extract her blood type. Since the crime scene can contain several blood traces and these traces can be spoiled, we often can only derive a set of alternative values instead of a single one.

Observation Cameras: The last source includes information that is extracted from observation cameras that are installed close to the crime scenes and that sometimes capture a picture from one or more of the involved criminals. Besides concrete photos we can additionally derive information on size, age, gender, and hair color from these video recordings. Nevertheless, observation cameras often only produce gray scaled pictures so that we can indeed distinguish blond hair from black hair, but cannot distinguish dark brown hair from black hair and cannot distinguish blond hair from gray hair. Moreover, photos extracted from these videos are often blurry, the criminals are masked, and they usually did not look directly into the camera. Finally, although we can derive information on size and age from the video records, this information is still vague, because it results from subjective estimations that are made by manually analyzing the camera pictures.

[2]The artwork in this figure was provided by Christina Panse.

1.2.1. Probabilistic Data Representation

To evaluate the information on the criminal suspects that we have gathered from the individual sources we integrate them into a probabilistic database. As mentioned before, a probabilistic database theoretically corresponds to a probability distribution on a finite set of conventional database instances that are typically called the database's *possible worlds* [SORK11], i.e. one of these instances is assumed to represent the 'true' state of the real world, but it is not completely known which of them. Since the number of possible worlds can be considerably large, a separate storing of all these worlds is usually impractical and a more succinct representation needs to be used instead.

During the last decades several probabilistic representation systems have been proposed [Gre09, SBH+09, SD09, AKO09, SORK11]. These systems differ in their compactness, modeling power, and representation/query complexity. It generally holds that a high compactness and a low representation/-query complexity implicates a low modeling power, i.e. not every set of possible worlds can be represented by this system. For that reason, the choice of the used representation system always depends on the requirements of the given application scenario. In our illustrative example, we work on information that is highly uncertain and therefore require a system that enables a high compact representation. For this reason, we use a simple representation system that we call *AOR?-databases* in this thesis[3]. AOR?-databases model uncertainty on two levels. First, they ascribe a probability to each database tuple (tuple-level uncertainty) and consider all these tuples as mutually independent events. Second, they consider the attribute values of one tuple as mutually independent probabilistic events (attribute-level uncertainty). The probability of a tuple describes the likelihood that the tuple belongs to the database's 'true' world. The probability of an attribute value describes the likelihood that the corresponding tuple has this value in the corresponding attribute in the case the tuple belongs to the database's 'true' world. Thus, the probability of an attribute value is always conditioned by the probability of its corresponding tuple. In order to differentiate between the tuples from an AOR?-database and the tuples from a conventional database we adopt the notation from Das Sarma et al. [SBH+09] and denote the first as A-tuples. Moreover, an A-tuple that has a probability lower than one is denoted to be *maybe*.

The independence assumptions (between attribute values as well as A-tuples) enable the system to decompose the probability distribution over possible worlds into several probability distributions over alternative attribute values and therefore is the key for the more compact representation that is provided by AOR?-databases. However, because these assumptions AOR?-database are not powerful enough to model every potential set of possible worlds. Despite of these shortcoming, they are useful in many application scenarios.

Example 1 *An AOR?-database that may result from the crime scenario is presented in Figure 1.2. This database contains a single table* 'Suspect' *that has five A-tuples and ten attributes. The first attribute is a surrogate key and the last attribute contains the A-tuples' probabilities. Since the attribute* 'SID' *serves as database internal identifier for all A-tuples of the considered table, we sometimes write* t_i *to refer to the A-tuple that has the value* i *in the attribute* 'SID'. *In contrast to these two attributes, all*

[3] A uniform naming of representation systems is sometimes missing in the literature on probabilistic databases and we therefore name systems by their significant properties if a universal name is missing. For instance, the name *AOR?-databases* was selected, because this representation system is based on the use of so-called *attribute-ORs* and it allows tuples to be maybe (maybe-tuples are typically labeled with a question mark).

Suspect

SID	gender		size	age	hair color		blood type		shoe size		fingerprint		photo		p
1	male	:0.9	1.70 – 1.80	25-35	brown	:1.0	A	:0.7	⊥	:1.0		:1.0	⊥	:1.0	1.0
	female	:0.1					AB	:0.3							
2	female	:0.7	1.65 – 1.70	ca. 45	brown	:1.0	B	:0.8	39	:0.7		:0.9	⊥	:1.0	1.0
	male	:0.3					0	:0.2	40	:0.3	⊥	:0.1			
3	⊥	:1.0	ca. 1.75	30-40	brown	:0.8	B	:1.0	40	:0.5	⊥	:1.0	⊥	:1.0	0.8
					blond	:0.2			41	:0.5					
4	male	:1.0	1.75 – 1.80	20-30	brown	:0.6	⊥	:1.0	40	:0.9		:1.0		:1.0	1.0
					black	:0.4			⊥	:0.1					
5	female	:1.0	1.71 – 1.72	45-50	gray	:1.0	B	:0.7	⊥	:1.0		:0.9	⊥	:1.0	1.0
							AB	:0.2				:0.1			
							0	:0.1							

Figure 1.2.: Probabilistic database table *'Suspect'* modeling uncertain information on criminal suspects

other attributes of the considered table are uncertain, i.e. an A-tuple can have several alternative values in each of these attributes. For instance, the blood type of A-tuple t_5 is either B (probability 0.7), AB (probability 0.2), or 0 (probability 0.1). Notice that the fingerprint of A-tuple t_2 is only incomplete. Moreover it was not clear whether or not this fingerprint actually belongs to the considered criminal. For that reason, a probability of 0.9 is ascribed to this fingerprint. Since it is not completely clear whether or not the person that is described by A-tuple t_3 is the actually wanted criminal (maybe it is a person that was near the crime scene accidentally), t_3 is associated with a probability lower than one. Note, for representation purposes we describe the uncertain age and the uncertain size of the suspects by interval values (e.g. the age of the first suspect is between 25 and 30 years) or approximate values (e.g. the second suspect is around 45 years old) in this table, but assume that the corresponding data values are internally stored by the use of probability distributions. Moreover, we simplify the databases by assuming that for each suspect only a single fingerprint and a single photo need to be stored. As in conventional relational databases, a null value $\perp$ is used to represent a missing attribute value.

Due to the assumed mutual independence between the A-tuples and due to the assumed mutual independence between the attribute values of one A-tuple, a single possible world can be constructed from an AOR?-database in two steps:

- First, we select all non-maybe A-tuples and select some of the maybe A-tuples.
- Second, we choose an alternative value per attribute for each of the selected A-tuples.

The probability of a possible world is computed by multiplying the probabilities of the selected A-tuples with the probabilities of the selected attribute values and with the the inverse probabilities of all non-selected maybe A-tuples. Note, null values are used as in conventional databases and therefore do not need to be resolved for constructing a possible world, i.e. the null value from A-tuple t_1 in the attribute *'shoe size'* remains a null value in all the database's possible worlds.

Similar to the possible worlds of a probabilistic database, an A-tuple models a set of possible instances, i.e. the non-empty possible worlds of an AOR?-database that only contains this A-tuple. A

Suspect$^{<1>}$

SID	gender	size	age	hair color	blood type	shoe size	fingerprint	photo
1	male	1.72	26	brown	A	⊥		⊥
2	female	1.67	44	brown	B	39		⊥
3	⊥	1.76	31	brown	B	40	⊥	⊥
4	male	1.79	29	black	⊥	40		
5	female	1.70	46	grey	B	⊥		⊥

Suspect$^{<2>}$

SID	gender	size	age	hair color	blood type	shoe size	fingerprint	photo
1	male	1.70	30	brown	AB	⊥		⊥
2	male	1.70	41	brown	0	40	⊥	⊥
4	male	1.76	20	brown	⊥	⊥		
5	female	1.71	50	grey	B	⊥		⊥

Figure 1.3.: Two possible worlds of the probabilistic database table *'Suspect'*

possible instance of an A-tuple is constructed by selecting one of its alternative values per attribute. Since all attributes are considered to be mutual independent events, the probability of a possible instance is computed by multiplying the probability of the A-tuple with the probabilities of the selected values. As a consequence, the world construction process described above corresponds to a process that selects a possible instance per non-maybe A-tuple and selects a possible instance or none instance per maybe A-tuple.

Example 2 *For illustration, Figure 1.3 presents two possible worlds of the sample database from Figure 1.2. Notice, since A-tuple* t_3 *is only maybe, it is missing in some of the possible worlds, e.g. it is missing in the possible world that is presented in Figure 1.3. Altogether, if we assume that each value of the attribute* 'size' *is internally stored at the granularity of centimeters and if we assume that each approximate value represents a probability distribution on ten alternative values, the sample AOR?-database represents a set of* 3.89×10^{12} *possible worlds (e.g. A-tuple* t_1 *has already* $2 \times 11 \times 11 \times 1 \times 2 \times 1 \times 1 \times 1 = 484$ *possible instances). This number is already large although we only consider information on five suspects and it is up to the reader to imagine what incredible number of worlds is compactly modeled in a similar AOR?-database with hundreds or thousands of suspects (or A-tuples respectively).*

This small example already shows that a separate storing of possible worlds is usually not manageable in practice and therefore emphasizes the importance of compact representation systems.

1.2.2. Querying Probabilistic Databases

In querying probabilistic databases, we have to distinguish between queries whose results are directly presented to the user and queries whose results are used as input to another query, as for example in the case of views. Whereas the results of the first need to be presented in a simple way, the results of the second need to be compositional, i.e. the query results must represent a valid probabilistic database.

For answering queries in a compositional way, the *possible worlds semantics* [AKG87, DS07c] is typically used as a reference. According to this semantics, the result of querying a set of possible worlds

SID	gender		hair color		p
2	female male	:0.7 :0.3	brown	:1.0	0.8
3	⊥	:1.0	brown blond	:0.8 :0.2	0.8
5	female	:1.0	gray	:1.0	0.7

(a) Compositional query result

	SID	gender	hair color	p
t_{A1}	5	female	gray	0.7
t_{A2}	3	⊥	brown	0.64
t_{A3}	2	female	brown	0.56
t_{A4}	2	male	brown	0.24
t_{A5}	3	⊥	blond	0.16

(b) User presented query result

Figure 1.4.: Result of the sample query evaluated under the two different semantics

is defined as another set of possible worlds where each output world corresponds to the query result of any of the input worlds. The probability of an output world corresponds to the accumulative probability of all input worlds that produce this output world as query result. In the case where a probabilistic database is modeled in a compact representation system, the result of a query is expected to be the probabilistic database that is modeled within the same system and that exactly represents the same set of possible worlds that would result from posing this query to the possible worlds of the input database.

Example 3 *For illustration, we consider a query that requests the* SID*, the gender, and the hair color of all suspects that have the blood type B. By evaluating this query according to the possible worlds semantics the query result is the AOR?-database that is presented in Figure 1.4(a)*[4]*. Notice, the probability of each output A-tuple results from multiplying the probability of its corresponding input A-tuple and the probability that this input A-tuple has the value 'B' in attribute* 'blood type'*. Since the A-tuples t_1 and t_4 do not have the value 'B' in the attribute* 'blood type' *in any of their possible instances, they do not belong to the query result.*

In contrast, if the query result is presented to the user, it is usually modeled as a list of *tuple-probability pairs* where the tuple is an ordinary database tuple (not an A-tuple!) and the probability is the likelihood that its corresponding tuple belongs to the 'true' world of the compositional query result. For reasons of presentation, the tuple-probability pairs of the query result are usually sorted by their probabilities in decreasing order [SORK11].

Example 4 *For illustration, we consider the same query as we have considered in Example 3, but this time we evaluate it as a user query. The corresponding query result is presented in Figure 1.4(b) and contains five tuple-probability pairs (we denote the output tuples as t_{A1} to t_{A5}). Note, the output tuples t_{A2} and t_{A5} result from querying the maybe A-tuple t_3. Therefore the probabilities of these two tuples are computed by multiplying the probability of t_3 with the probability that t_3 has the corresponding value in the attribute* 'hair color'*. It is obvious that the simpler presentation comes to the price of losing information on tuple correlations because the mutual exclusion between the output tuples t_{A2} and t_{A5} as well as t_{A3} and t_{A4} is not reflected by the query result.*

[4]Note, because null values remain null values in every possible world, A-tuple t_4 does not belong to the query answer in any of these worlds although its value in the attribute *'blood type'* cannot be excluded to be B for sure.

1.3. Duplicate Detection

Duplicate detection [EIV07, NH10, HCML09] is the process of identifying data objects that refer to the same real-world entity and is often also denoted as *record linkage* [NK62, FS69, Jar89, Win02, BG04, HSW07], *object matching* [DLLH03, ZSC10], or *data matching* [Chr12]. In order to clean a database from duplicates, duplicate detection needs to be followed by a process that merges the detected duplicates to single data objects. This process is usually denoted as *duplicate merging* [BGMM+09] or *data fusion* [MA06, BN08]. A combined execution of both processes is usually denoted as *duplicate elimination* [LLL01, SB04], *deduplication* [SB02, CG07a, ARS09], the *merge/purge problem* [HS95], or *entity resolution* [BGMM+09, BG07c, Tal11, BBKL12, GM12, GM13].

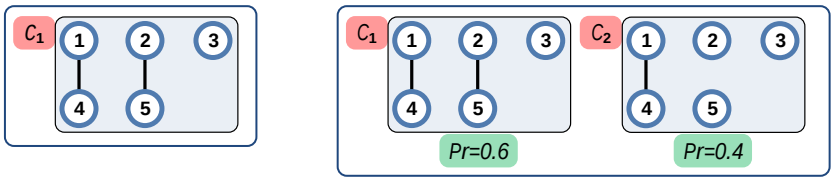
Duplicate detection scenarios are typically restricted to single relational tables. In this case, each database tuple is supposed to represent another real-world entity. In complex databases, however, real-world entities often need to be represented by tuples in several tables (we denote such situations as *multi-table memberships*). For that reason, we introduce the term *database entity* in this thesis in order to encompass to all the information within a database that is considered to represent the same real-world entity. As a consequence, we consider duplicate detection as the identification of different database entities that refer to the same real-world entity. We will introduce the concept of database entities in more detail in Section 2.1. Until then we consider database entities and database tuples (or A-tuples respectively) synonymously.

Real-world equivalence is a transitive relation. Thus, if the database entity e_r represents the same real-world entity as the database entities e_s and e_t (and hence is a duplicate of them), e_s and e_t need to represent the same real-world entity (and hence need to be duplicates) as well. Due to this transitivity, the result of a duplicate detection process is a partition (called *clustering*) of all entities of the input database into non-empty and disjoint partition claesses (called *clusters*) where each cluster refers to another real-world entity. Since the clusters are disjoint and each database entity is assigned to exact one cluster, the result is deterministic and we therefore speak about *deterministic* duplicate detection in such cases.

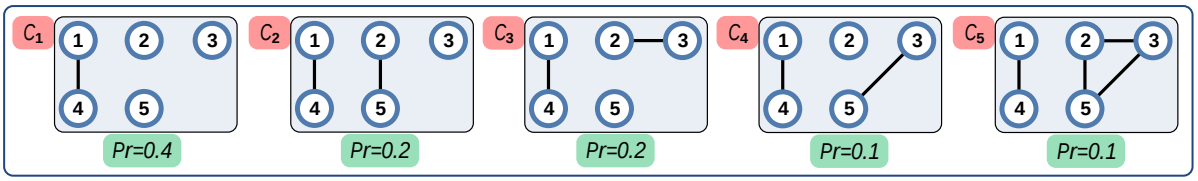
Example 5 *In the illustrating example of this chapter, duplicate detection corresponds to the task of identifying suspects that actually describe the same person. Individual criminals typically participate in multiple crimes and it is often not obvious whether or not two suspects are actually the same person. Thus, it is only natural that different A-tuples of the table* 'Suspect' *refer to the same person and hence the table is commonly corrupted by many duplicates*[5].

Although the final detection result is a clustering, duplicate decisions are typically made in a pairwise fashion before combining them to a globally consistent result, i.e. the duplicate clustering. In the pairwise comparison step, each entity pair is assigned to the class of MATCHES (the supposed duplicates) or is assigned to the class of UNMATCHES (the supposed non-duplicates) based on the similarities of their attribute values. The underlying idea of the pairwise comparison approach is to use the similarity of the attribute values as an indication for the real-world equivalence between the corresponding database

[5] Notice that the A-tuples in the considered sample table only contain information on the properties of the suspected person and information on the crime or information on the crime scene are assumed to be stored separately. Thus, each A-tuple represent a person and two A-tuples are considered to be duplicates if they represent the same person.



(a) Certain duplicate clustering (b) Probabilistic duplicate clustering (small)



(c) Probabilistic duplicate clustering (large)

Figure 1.5.: A certain clustering and two probabilistic clusterings of the A-tuples from Table *'Suspect'*

entities. Nevertheless, because data is often corrupted by many errors and database entities are oftentimes outdated, it can happen that the attribute values of some non-duplicate entities are more similar than the attribute values of some duplicate entities (note, in the most data applications we do not have a property like the fingerprint that enables a certain implication of real-world equivalence). This circumstance makes duplicate detection a challenge, because it is often impossible to clearly demarcate duplicate pairs from non-duplicate pairs. As a consequence, duplicate detection approaches are vulnerable for two kinds of errors [MWGM10]: (a) *false positives*, i.e. actual non-duplicates that have been incorrectly detected as duplicates, and (b) *false negatives*, i.e actual duplicates that have not been detected as duplicates. For the sake of completeness, actual duplicates that have been correctly detected as duplicates are called *true positives* and actual non-duplicates that have been correctly detected as non-duplicates are called *true negatives*. Relaxing the matching criteria usually decreases the number of false negatives, but also increases the number of false positives. Therefore, duplicate detection is always a trade-off between both kinds of errors that essentially depends on the considered application scenario because some applications consider false positives to be worse than false negatives and some applications do the contrary.

Example 6 *If we adopt the pairwise matching concept from conventional databases to probabilistic databases, duplicate decisions on A-tuples are made by comparing the A-tuples' attribute values. In order to use all attribute values that are available in our crime scenario, we require specific similarity measures for fingerprints and photos. It is obvious that the similarity in some attributes is more important than the similarity in other attributes. For instance, whereas the same age or the same size is only a small indication for a duplicate because there are many people having this age (or size respectively), an identical fingerprint is already a clear proof of real-world equivalence if we can assume that the compared values are free from errors. If we consider the sample table* 'Suspect' *and if we assume an error free database, i.e. for each attribute value the 'true' value is among the alternative values, we see that the A-tuples* t_1 *and* t_4 *are certainly duplicates (a hard* MATCH*) because they share the same fingerprint.*

Suspect (Cleaned)

SID	gender	size	age	hair color	blood type	shoe size	fingerprint	photo	p
1,4	*male* :1.0	*1.75 – 1.80*	*25-30*	*brown* :1.0	*A* :0.7 *AB* :0.3	*40* :0.9 ⊥ :0.1	 :1.0	 :1.0	1.0
2,5	*female* :1.0	*1.65 – 1.72*	*ca. 47*	*brown* :0.5 *gray* :0.5	*B* :0.8 *0* :0.2	*39* :0.7 *40* :0.3	 :0.9 :0.1	⊥ :1.0	1.0
3	⊥ :1.0	*ca. 1.75*	*30-40*	*brown* :0.8 *blond* :0.2	*B* :1.0	*40* :0.5 *41* :0.5	⊥ :1.0	⊥ :1.0	0.8

Figure 1.6.: Possible cleaned version of *'Suspect'*

The A-tuples t_2 and t_5 are highly similar in their sizes, ages and blood types. Moreover, they likely have a high similar fingerprint and therefore are possibly a duplicate. Nevertheless, since we cannot classify them as a MATCH *without doubt, we call them a* POSSIBLE MATCH *instead. Note that in deterministic duplicate detection* POSSIBLE MATCHES *are usually resolved by manual reviews of domain experts and therefore have to be small in numbers. Interestingly, t_3 and t_4 are somewhat similar. However, t_1 and t_3 are a hard* UNMATCH *because they have different blood types and therefore we can conclude that t_3 and t_4 must be a hard* UNMATCH *as well. A potential duplicate clustering of the five A-tuples is presented in Figure 1.5(a). As shown by this figure, we usually present a clustering in this thesis as a graph that has one node per database entity (in this case A-tuples) and connects two nodes by an edge if they belong to the same duplicate cluster.*

The main purpose of deduplication is to remove duplicate entities from the database and hence to make the database (more) consistent. Nevertheless, merging duplicate database entities has a useful side effect, because each of these entities can contain information that is not present in the others and combining their information can lead to new conclusions.

Example 7 *Thus, in the considered crime scenario we benefit from deduplication in two ways. First, by removing duplicates we reduce the number of investigations that are need to be made. Second, deduplication can help to catch and arrest the wanted criminals because the data values of the merged A-tuples become more informative and we therefore can make conclusions on the identities of the suspects that could not been made by only considering the A-tuples of the uncleaned database.*

The value of merging duplicates can be perfectly illustrated on the basis of the hard MATCH *between t_1 and t_4 because their information is complementary. Whereas t_1 has a known blood type, t_4 has a photo and a shoe size. Merging is usually performed on an attribute-by-attribute basis and can be realized in different ways. For illustration, we use a merging approach that computes a probabilistic version of the set intersection operator for attribute values that do not contradict, e.g. the non-contradictory intervals $25-35$ and $20-30$ are merged to the interval $25-30$, and computes a probabilistic version of the set union operator in the case of contradictions, e.g. the certain but contradictory values 'brown' and 'gray' are merged to a uniform probability distribution on both values. By assuming that t_1 and t_4 as well as t_2 and t_5 are classified as duplicates, Figure 1.6 presents a 'cleaned' version of the initial table* 'Suspect' *by using a variant of the above described merging approach. As we can see, the new A-tuples*

contain much more information that the original A-tuples from Figure 1.2 and hence can considerably speed up the investigation process.

Detection of duplicates can only be based on matching attribute values, but can also profit from the information on relationships to other database entities.

Example 8 *For instance, let us assume a second table* 'Crime' *that stores information on the crime scenes (e.g. place, time, date, and crime type) and let us assume a third table that connects the A-tuples from the table* 'Suspect' *with tuples of the table* 'Crime'. *Criminals often work by a specific pattern. Consequently, descriptions on the committed crimes can help to identify duplicate suspects. Moreover, we can use information on crime scenes to exclude duplicates for sure (and hence to reduce the number of false positives). For example, a criminal cannot commit two crimes at a similar time at two far afield places. Thus, if two A-tuples from the table* 'Suspect' *refer to tuples from the table* 'Crime' *that are not compatible, we can classify them as an hard* UNMATCH *although the attribute values of these A-tuples are similar to a large extent. Of course, such conclusions are always based on the condition that the provided information on the crime, the crime scene, and the crime participation of individual suspects is known to be correct.*

1.3.1. Indeterministic Duplicate Detection

Until now, we consider duplicate detection as a process that eventually resolves any kind of uncertainty on the made duplicate decisions because it produces a single clustering as a final result. In the context of probabilistic databases, however, such a deterministic resolution is not necessarily required, but ambiguous decisions can be probabilistically stored in the database instead. A so-called *indeterministic* duplicate detection process [PvKR13, PR12, BSIBD09, INNV10] rather produces a probability distribution on a set of possible clusterings (called as *probabilistic clustering* in this thesis) instead of a single clustering if some of the underlying duplicate decisions cannot be made with absolute confidence.

Example 9 *A probabilistic duplicate clustering of the five A-tuples from the sample table* 'Suspect' *is presented in Figure 1.5(b). In this case, we take into account that the two A-tuples* t_2 *and* t_5 *are a* POSSIBLE MATCH *and therefore avoid a risky decision by modeling both plausible clusterings in the detection results. Of course, the number of possible clusterings can be larger in numbers. For instance, let us assume that the similarity between* t_2 *and* t_3 *as well as the similarity between* t_3 *and* t_5 *is within such a range that we cannot conclude with absolute certainty whether they are duplicates or not. For considering all these uncertainties in the detection result, we need to create a probabilistic duplicate clustering with five possible clusterings as presented in Figure 1.5(c).*

It is important to note that an indeterministic deduplication result cannot be represented by an AOR?-database because it requires a modeling of complex correlations between the existence of tuples, but the A-tuples of an AOR?-database are considered to be mutual independent. For that reason, we will extend our consideration to more powerful (but although more complex) representation systems in Chapter 3 of this thesis.

In general, uncertainty on duplicate decisions can be caused by four reasons. First, uncertainty can be introduced into the detection process by the processed data themselves as it is the case if we consider a

probabilistic database as input. Second, it is often not known which process configuration detects duplicates best, because the quality of configurations essentially depends on the given database and essentially depends on the considered application domain. Moreover, beyond a perfect detection the meaning of quality is not even clearly defined and we therefore often cannot determine which process configuration is most suitable for a given scenario. Third, high data similarity does not correspond to real-world equivalence, but is only an indication for it. Thus, we cannot make all duplicate decisions with absolute confidence even if a certain input database is processed and the best possible process configuration is available. Finally, in contrast to the assumption that we made in the examples of the previous section we usually cannot assume that the database is free of errors. Instead data values can contain typographical errors, attribute values of one tuple can be confused and values can be even confused across tuples. Thus even if a clear proof on real-world equivalence such as two identical fingerprints is given some amount on duplicate uncertainty always remain.

The advantage of indeterministic duplicate detection is that we are not forced to make deterministic duplicate decisions in ambiguous situations, but can incorporate emerging decision uncertainty into the output database. This uncertainty in turn can be gradually resolved afterwards by introducing new evidence time after time and therefore the indeterministic deduplication result can be considered as a prompt cleaning answer (or integration answer respectively) that can be queried instantly (note that in many applications, e.g. virtual data integration, the deduplication result must be immediately provided to the user and there is no time left for manual reviews of domain experts or other time-consuming activities). Moreover, indeterministic duplicate detection enables the usage of context information that was not available at detection time. For instance, the deduplication result can be provided to a community and the collective knowledge of this community can help to resolve indeterministically modeled decisions afterwards [dKvK07b, vKdK09].

Example 10 *For demonstrating the benefits of indeterministic duplicate detection, we reconsider the probabilistic clustering of the illustrative example that is presented in Figure 1.5(b). The A-tuples t_2 and t_5 are a* POSSIBLE MATCH*, i.e. they belong to the same cluster in the first possible clustering, but belong to different clusters in the second possible clustering. In a deterministic duplicate detection approach, we need to chose one of these possible clusterings as a final result and hence have to classify these A-tuples either as a* MATCH *or have to classify them as an* UNMATCH*. In the first case, we risk a false positive and hence risk a merging of information that actually does not describe the same person. This incorrect merging can have serious consequences because we would only investigate for one suspect instead of two suspects. Moreover, the investigation on this single suspect would be rely on incorrect information such as properties that actually do not belong together (e.g. brown hair and blue eyes). As a consequence, the investigations possibly become misdirected so that we do not suspect the actual criminal (even if we known her) because she does not fit to the used searching scheme. In the second case, however, we risk a false negative and therefore risk the chance of arresting the corresponding criminal because we do not combine the information provided from both A-tuples. Of course, we can store the suspicion that both A-tuples maybe refer to the same criminal in a separate document or database table. In that case, however, this information is not directly incorporated into query evaluation and needs to be queried separately instead.*

In contrast, if we use an indeterministic deduplication approach, the original A-tuples as well as the merged A-tuple belong to the database and all of them are considered in answering the posed database queries. For instance, if we request for all male suspects that are 45 *years old, A-tuple* t_2 *belongs to the query result, because it satisfies both conditions in some of the database's possible worlds. On the other hand, if we request for all suspects that have gray hair and shoe size* 39*, the newly created A-tuple* $t_{2,5}$ *belongs to the query result, because the value in the attribute* 'hair color' *is either brown or gray and the value in the attribute* 'shoe size' *is either* 39 *or* 40*. Note that if we would use a deterministic deduplication approach we would either miss tuple* t_2 *in the result for the first query (*t_2 *and* t_5 *are a* MATCH*) or we would miss tuple* $t_{2,5}$ *in the result of the second query (*t_2 *and* t_5 *are an* UNMATCH*).*

Recall that an indeterministically modeled decision is not considered as a final result, but should be rather considered as a prompt answer that can be resolved gradually. Thus, every time we find a new evidence about the two suspects, e.g. a shoeprint or a fingerprint, the indeterministically modeled decision becomes more clear (in which direction however) until it is eventually resolved.

1.4. Research Questions

As we have presented throughout the previous sections, probabilistic data management is useful in several application domains. Like a conventional database, a probabilistic database can be corrupted by duplicates and eliminating these duplicates can considerably increase the quality of the database. Moreover, duplicate database entities need to be detected for meaningful integrating data that originate from different sources and can be beneficial in matching database schemas.

Therefore, the goal of this thesis is to develop a generic approach for duplicate detection (deterministic as well as indeterministic) in uncertain databases. Duplicate decisions are typically based on the similarities between attributes values and/or relationship information. The meaning of similarity, however, can strongly vary from one domain to another. Moreover, in which way we can conclude real-world equivalence from data similarity considerably depends on the quality of the considered data because the portion of data errors and the volume of missing values significantly affects the similarity between two duplicates and significantly affects the similarity between two non-duplicates respectively. Finally, a perfect detection of duplicates is usually unrealizable and whether or not false positives are preferred to false negatives depends on the considered database applications. Thus, although many duplicate detection algorithm are entitled to be domain-independent [ME97, BM02, ZH06, Leh06, SBY10] the actually used duplicate detection process is not because some of the algorithms' input parameters such as similarity measures or thresholds need to be adapted to the given detection scenario. As a consequence, configurating a duplicate detection process in an appropriate way strongly depends on the considered domain and strongly depends on some quality characteristics (e.g. completeness or accuracy) of the deduplicated database. For that reason, we strive for a detection approach that (a) provides a variety of meaningful functionalities, and (b) allows a flexible composition of these functionalities so that the user becomes enabled to design a duplicate detection process that fits best to her purpose.

The research question that is primarily considered in this thesis is therefore:

- *In which ways we can effectively and efficiently detect duplicates in probabilistic relational databases?*

Based on this question, we can derive a set of subquestions. It is obvious that the uncertainty of the input data affects the detection process. Nevertheless, it is not completely clear in which way this process is affected by uncertainty on tuple-level and is affected by uncertainty on attribute-level. This implicates the following subquestion:

- *How to match database entities in the presence of tuple-level uncertainty and attribute-level uncertainty?*

Strong probabilistic representation systems offer the opportunity to model uncertainty on duplicate decisions in the detection result, but several representation systems do not. For this reason, we have to distinguish between deterministic duplicate detection approaches and indeterministic duplicate detection approaches. To produce a deterministic duplicate detection result we have to resolve the uncertainty of the input database at some moment of the detection process. This poses two subquestions:

- *At what moments we can resolve input data uncertainty?*
- *By which methods we can resolve input data uncertainty at a particular moment?*

The underlying idea of indeterministic deduplication is to model uncertainty on duplicate decisions within the output database. This leads us to the following subquestions:

- *What types of applications can benefit from indeterministic deduplication results?*
- *How we can efficiently compute indeterministic duplicate detection results?*
- *How we can efficiently represent indeterministic deduplication results within a probabilistic relational database?*

Decreasing the number of ambiguous duplicate decisions comes to the price of decreasing the certainty of the output database. As a consequence, indeterministic duplicate detection is always a trade-off between these two contrary goals. To quantify this trade-off in numbers we need measures for rating the quality of indeterministic duplicate detection results. This implicates the following subquestion:

- *What does detection quality mean in the presence of uncertain duplicate decisions?*

1.5. Challenges

For answering the research questions presented above we have to face a set of challenges:

Type mismatch

The first challenge concerns the semantics of duplicate detection in relational data. Duplicate detection itself is entity-based, i.e. it maps database entities to real-world entities, but the relational data model is not because a database tuple cannot only represent an entity, but can also represent a relationship between entities, or can represent the value of a multi-valued attribute. For detecting duplicates in certain relational databases, this type mismatch usually does not matter because duplicate detection is typically restricted to single database tables or foreign key references from one database table to another database table are always treated as relationship information. One aspect

of probabilistic relational databases, however, is the uncertain membership of tuples to database tables (i.e. tuple-level uncertainty) and therefore it is not unusual that entities are represented by several tuples in different tables. For that reason, we require an entity-based interpretation of relational data in general and require an entity-based interpretation of probabilistic relational data in particular.

Representation heterogeneity

Probabilistic representation systems are used to compactly represent a probability distribution over a set of possible worlds. Obviously, by using different representation systems, same information (i.e. the same probability distribution over the same set of possible worlds) can be represented in different ways. In this thesis, however, we strive for detection approaches that are independent from the used representation and therefore always produce the same output clustering if same input information is given. This property is especially necessary if we integrate probabilistic data from several sources where some sources use another system for probabilistic data representation than others. As a consequence, one challenge is to overcome the heterogeneity that is caused by the usage of different representation systems.

Resolution of data uncertainty

For accomplishing deterministic duplicate detection results we need to resolve the uncertainty of the input database. In theory, input data uncertainty can be resolved at different moments of the detection process. For instance, the first moment is to aggregate all the possible worlds of the probabilistic input database to a single world that is a certain database and the last moment is to aggregate all possible duplicate clusterings to a single clustering. Intuitively, an aggregation of worlds seems to be most efficient because duplicate detection can then be performed in a certain database. In contrast, an aggregation of the duplicate clusterings seems to be most effective because uncertainty can be considered in any of the detection phases. Nevertheless, there are several other moments where uncertainty can be resolved and it is not clear which of these moments provides a trade-off between effectiveness and efficiency that is best fitting to a specific application scenario. For that reason, we have to examine aggregation methods for different domains such as database instances, similarity scores, duplicate decisions, or clusterings. Moreover, different applications can have different quality requirements, e.g. some applications consider false positives to be worse than false negatives or vice versa. As a consequence, we need to incorporate such quality requirements into the aggregation process.

Matching database entities in the presence of uncertainty

If the uncertainty of the input database is not resolved beforehand (e.g. by an aggregation of possible worlds), we need to deal with this uncertainty in the following detection phases. This particularly concerns the pairwise matching of database entities. Although we cannot use the possible worlds semantics as a reference for duplicate detection in probabilistic databases if we need to compute a deterministic outcome, it seems most intuitive to classify two database entities as a MATCH if they are similar in the majority of possible worlds. By doing so we produce a deterministic detection result by aggregating the results that have been computed in the individual worlds. However, this approach implicates a set of further challenges:

- We strive for an efficient detection approach. Nevertheless, as we have already illustrated in the previous part of this thesis, the number of possible worlds of a probabilistic database is usually extremely large and matching entities in each of these worlds is therefore impractical. As a consequence, we need to develop detection approaches that process a set of possible worlds without processing each of these worlds separately.
- Due to tuple-level uncertainty, some database entities are missing in some of the possible worlds and it is not clear how to incorporate such absences into the aggregation process.
- Due to incorrect tuple dependencies, it can occur that duplicate entities do not coexist in any possible world, i.e. duplicate-free worlds do not necessarily imply a duplicate-free probabilistic database, and therefore will never be classified as a MATCH if the final detection result is only based on the detection results of the individual worlds. Moreover, incorrect tuple dependencies can cause that two duplicate entities are dissimilar in all possible worlds.
- If the schema of a probabilistic database is designed by using an approach that has been developed for designing certain database schemas, it can contain incorrect dependencies from attributes and/or relationship roles to table memberships. For instance, a database table *'Student'* can contain the attribute *'name'*, but the name of a person actually does not depend on the fact whether or not she is a student. In certain databases such incorrect dependencies do not matter, but in probabilistic databases such dependencies can negatively affect the detection process if the membership to the considered table can be uncertain because in this case an entity can only have a value in the incorrectly depending attribute if it belongs to the corresponding table. As a consequence, the uncertainty whether or not a database entity belongs to a table can influence the finally made duplicate decisions stronger than useful.
- Most often the possible instances of a database entity are very similar to each other. Nonetheless, this is not a mandatory property and sometimes even the contrary is the case. In this thesis we call such database entities to be *scattered* and call two scattered database entities as *scattered* duplicates if they represent the same real-world entity. A scattered duplicate can simply emerge by mistakenly inserting a scattered database entity into the database for several times and therefore can be a not uncommon error. The accumulative probability of all the possible worlds in which two scattered duplicate are similar, however, can be extremely low if these entities are independent and each of them has a large number of possible instances. For illustration, let e_r and e_s be two database entities that both represent a uniform probability distribution on the same set of ten possible instances. If the pairwise similarities of these ten instances are low, both entities are only similar in these possible worlds in which they have the same possible instance. As a consequence, the accumulative probability of the worlds in which these entities are similar is only $10 \times 0.1 \times 0.1 = 0.1$.

Modeling indeterministic deduplication results within probabilistic databases

After computing the probabilistic clustering, each duplicate cluster of any possible clustering is merged to a single database entity by a conventional merging approach. To correctly represent an indeterministic deduplication result into a probabilistic database, however, we cannot simply insert the new entities into the database, but need to adopt cluster correlations that are inherently

modeled in the probabilistic clustering from the duplicate clusters to the newly created entities. For that reason, we need to study in which way such correlations can be efficiently modeled within a probabilistic representation system.

Quality of indeterministic duplicate detection results

In order to compare two indeterministic duplicate detection results or to compare an indeterministic duplicate detection result with a deterministic duplicate detection result, we demand for measures that rate the quality of probabilistic duplicate clusterings. For that purpose, we need to identify the criteria that are significant for the quality of an indeterministic duplicate detection result. Moreover, we need measures that can be used to quantify these criteria by numerical values. In this context, it is not completely clear if we can reuse measures from other application domains, e.g. duplicate detection in certain data, or if we need to design some measures by our own. Finally, the quality of a database is typically considered as its fit for use. As a consequence, there is a high correlation between the meaning of quality and the database application that need to work on the deduplicated database. However, it is not clear in which way this correlation affects the selection of the significant quality criteria and in which way this correlation affects the selection of the quality measures that are used to quantify these criteria.

Factorization of probabilistic clusterings

In theory, an indeterministic duplicate detection process produces an uncertain clustering of all database entities as output. However, the number of plausible duplicate clusterings can become large and storing (over even enumerating) all these clusterings becomes infeasible already for a small number of uncertain pairwise decisions. For that reason, we require for a more compact representation of probabilistic clusterings. Since most duplicate decisions are mutual independent, compactness can be achieved by *factorizing* the probabilistic clustering into its independent components. Nevertheless, a subsequent factorization is commonly not practical and we require detection approaches that directly produce the factorized representation of a probabilistic clustering as output. Moreover, factorization influences the way we have to represent an indeterministic deduplication result within a probabilistic database. Finally, quality measures are usually defined with a non-factorized uncertain clustering in mind and we therefore need methods to efficiently compute the quality of an uncertain clustering based on its factorized representation.

1.6. Contribution

For answering the research questions described in Section 1.4, we need to solve the challenges presented in Section 1.5. The main contributions of this thesis can be therefore summarized as follows:

Entity-based interpretation of (probabilistic) relational data

Duplicate detection is per definition entity-based, but the relational data model is not. For that reason, we propose an entity-based interpretation of relational data and extend it to probabilistic relational data. In the context of the latter, we discuss the use of foreign keys in several probabilistic representation systems and present methods to model the concept of inheritance within these systems.

Framework for deterministic duplicate detection in certain databases

We propose a framework for duplicate detection in certain databases. This framework is called HaDDeF and serves as a fundamental baseline for the two approaches for detecting duplicates in probabilistic data that are presented in this thesis. HaDDeF contains a seven-phase model for duplicate detection that is based on a pairwise matching of entity descriptions. With this framework, we extend current research on duplicate detection in certain relational databases in three ways: (a) we present a generic approach for incorporating relationship information into the detection process, (b) we propose the concept of *impact values* that can be used to improve the effectiveness of a duplicate detection process, and (c) we propose a method for modeling and matching database entities with multi-table memberships.

Uncertain Value Theory

We propose a formalism called *Uncertain Value Theory* that extends the possible worlds semantics from database level to arbitrary domains and hence can be used for modeling and managing uncertainty at different levels of granularity (including functions). This formalism serves as a baseline for all the formal definitions that we make in this thesis.

Deterministic duplicate detection in probabilistic data

We study several semantical issues that emerge in the context of deterministic duplicate detection in probabilistic databases and elaborate a set of challenges that need to be considered for developing effective duplicate detection approaches.

A deterministic detection result need to be accomplished by the use of aggregation methods. For that reason, we extensively discuss methods for aggregating (interim as well as final) detection results. Furthermore, we conceptualize an approach that can be used to incorporate quality requirements into the most of these aggregation methods.

We propose two generic approaches for deterministic duplicate detection in probabilistic databases. Both approaches are based on the reuse of existing methods for detecting duplicates in certain databases. The first approach is based on the principle idea of the possible worlds semantics and therefore processes a set of sample worlds separately before aggregating the worlds' detection results to a deterministic one. We refer to this approach as *world-based detection approach* in this thesis. The second approach considers uncertainty not on database level, but considers uncertainty on description level and is therefore called as *description-based detection approach*. This approach is based on pairwise matching probabilistic entity descriptions and thus processes all the possible worlds per entity pair collectively. To make deterministic duplicate detection scalable, we propose techniques that can be used to improve the efficiency of matching two probabilistic entity descriptions. Moreover, we provide a theoretical and an experimental comparison of both detection approaches.

A particular challenge is the detection of scattered duplicates. Because probabilistic entity descriptions can be considered as probability mass functions, we survey existing methods for measuring the similarity between two probability mass functions. Since none of these methods is suitable for our purpose, we present a probabilistic extension of the Monge-Elkan Similarity (called *Probabilistic Monge-Elkan Similarity*) and extend it to the use of entity correlations.

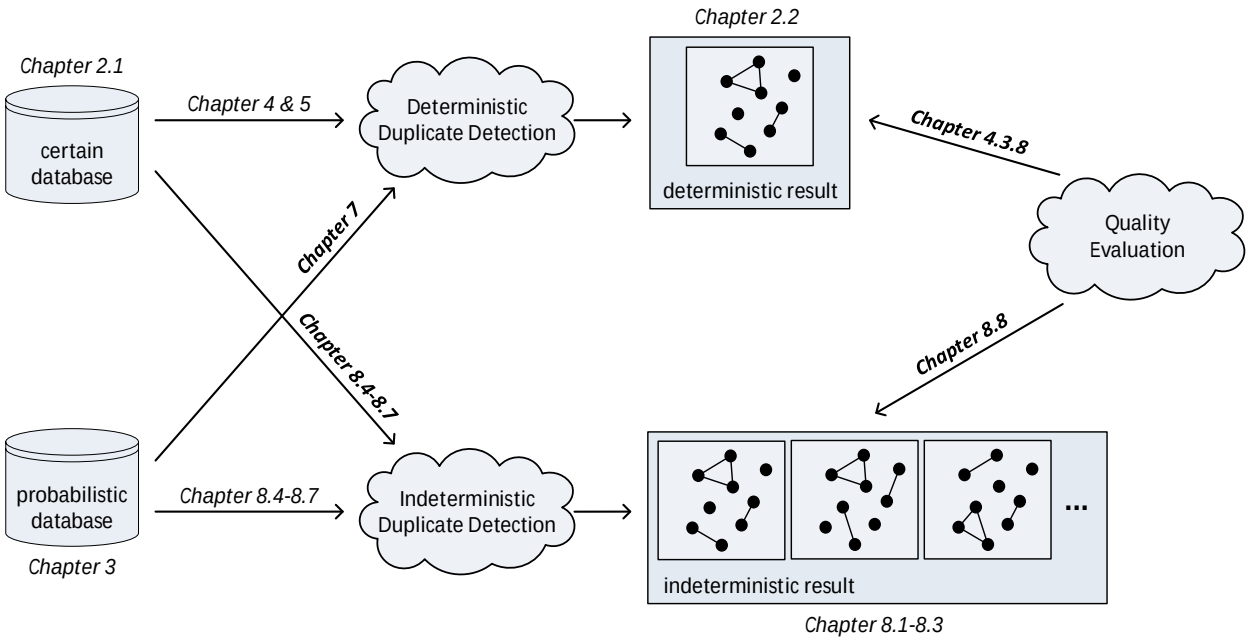


Figure 1.7.: Overview on the connections between some of the chapters and sections of this thesis

The challenge of dealing with incorrect tuple dependencies and the challenge of poor schema design require both a specific preparation of the input databases. For this reason, we propose two preparation activities in this thesis. Whereas the first activity extracts entity correlations from the input database, the second activity transforms the schema of input database in order to remove incorrect dependencies from attributes and/or relationship roles to table memberships.

Finally, we study sources of uncertainty within a duplicate detection process and present methods to incorporate process uncertainty into the description-based detection approach.

Indeterministic duplicate detection

We provide a formalization of probabilistic clusterings, indeterministic duplicate detection processes, and probabilistic clustering factorizations. Moreover, we introduce an approach for modeling an indeterministic deduplication result within a probabilistic database and provide a discussion on the different types of database applications that can profit from modeling decision uncertainty within the database. Since a subsequent factorization of probabilistic clusterings is not practical, we propose a clustering approach that directly computes a probabilistic clustering in a factorized way. Finally, we study the meaning of detection quality in the presence of uncertain duplicate decisions and propose measures that can be used to rate this quality. Because a probabilistic clustering is usually given by its factorized representation, we elaborate properties that can be exploited to efficiently compute the introduced quality measures based on the clustering's factorization.

1.7. Outline

The rest of this thesis is structured in eleven chapters. Figure 1.7 graphically illustrates the connections between some of these chapters and brings them into line with the topic of this thesis.

- In **Chapter 2** we introduce some preliminaries that are fundamentally required for this thesis. First, we present an entity-based interpretation of relational databases in Section 2.1. Because duplicate detection results are clusterings of database entities and we sometimes require operations on clusterings we then introduce some background on clusterings in Section 2.2.

- **Chapter 3** contains an extensive and formal presentation of probabilistic databases. First, we formally introduce the fundamental principles of probabilistic databases such as the possible worlds semantics in Section 3.2, then discuss some commonly used representation systems in Section 3.3, adopt our entity-based interpretation from certain relational databases to probabilistic relational databases in Section 3.4, give an overview on techniques for querying probabilistic databases in Section 3.5, and conclude this chapter with some further remarks in Section 3.6.

- **Chapter 4** is devoted to background information on duplicate elimination in general and duplicate detection in particular. We first present a formal definition of both processes in Section 4.1 and then shortly discuss the two most important application fields of duplicate elimination, namely data cleaning and data integration, in Section 4.2. We precede with a detailed presentation of conventional approaches for duplicate detection in certain databases in Section 4.3 and conclude this chapter by a short discussion on existing approaches for duplicate merging in Section 4.4.

- **Chapter 5** introduces the duplicate detection framework HaDDeF. First, we present methods for describing database entities in Section 5.1. Then, we introduce the concept of impact values and present a seven-phase model for duplicate detection that is based on a pairwise matching of entity descriptions in Section 5.2. Section 5.3 describes our approach for matching database entities with multi-table memberships.

- In **Chapter 6** we present the principles of the Uncertain Value Theory. Moreover, we present existing approaches for measuring data uncertainty in Section 6.10.

- In **Chapter 7** we present our research on deterministic duplicate detection in probabilistic databases. We first provide a study about semantical detection issues and list several detection challenges in Section 7.1. Section 7.2 contains an extensive discussion on aggregation methods that can be used for uncertainty resolution. Then, we present the world-based detection approach as well as the description-based detection approach in Section 7.3 and Section 7.4 respectively. A detection of scattered duplicates by using similarity measures for probability mass functions and therefore the introduction of the Probabilistic Monge-Elkan Similarity are part of Section 7.5. Section 7.6 describes several techniques for improving the efficiency of matching probabilistic entity descriptions. The two new preparation activities are then discuss in Section 7.7. In Section 7.8 we consider an incorporation of process uncertainty into the detection process Finally, we conclude this chapter by comparing the world-based detection approach and the description-based detection approach in Section 7.9.

- **Chapter 8** covers our research on indeterministic duplicate detection. Section 8.1 and Section 8.2 formally introduce the concepts of probabilistic clusterings as well as indeterministic duplicate detection processes and discuss the challenges that are caused by both concepts. A factorization of

probabilistic clusterings is considered in Section 8.3. Then we present methods to model indeterministic deduplication results within a probabilistic database in Section 8.4 and discuss different types of applications that can profit from indeterministic duplicate detection results in Section 8.5. In Section 8.7 we propose a duplicate clustering approach that instantly produces a probabilistic clustering in a factorized way. Finally, we conclude this chapter, by an extensive study on the meaning and the computation of the quality of indeterministic duplicate detection results in Section 8.8.

- In **Chapter 9** we shortly describe our prototypical implementation called HaDES (Hamburg Duplicate Elimination System) and present the results of several experiments that we have conducted in order to prove the concepts of the different approaches and techniques that we have proposed in this thesis.

- In **Chapter 10** we present several works that are related to this thesis and compare these works with the contributions that are provided by this thesis.

- **Chapter 11** concludes this thesis in Section 11.1 and gives a short outlook on open challenges as well as upcoming research in Section 11.2.

Chapter 2

Preliminaries

In this chapter we introduce some preliminaries that are required in the remainder of this thesis. Due to duplicate detection is entity-based by nature but the relational data model is not, we first present an entity-based interpretation of relational data in Section 2.1. The result of a duplicate detection process is a clustering of database entities. For that reason, we introduce some formal background on clusterings, properties of clusterings, and operations on clusterings in Section 2.2.

2.1. Entity-Based Interpretation of Relational Data

The purpose of a database is to model a part of the real-world that is of interest for a specific class of applications. Consequently, a database can be considered as an image or a view of the real-world. As in many real-life areas, this image is a result of the subjective perceptions of the people that are responsible to design the database and to fill it with data. Therefore, we have to distinguish between the actual entities of the real-world and the entities the database considers to be real.

Definition 1 (Real-World Entity): *A real-world entity is a distinguishable thing (or object) of the considered universe of discourse. The set of all real-world entities, i.e. the real-world itself, is denoted as* $\mathfrak{W}$.

Examples for real-world entities are persons, movies, books, cities, and so on. Note, the concrete set of considered real-world entities always depends on the considered universe of discourse.

Definition 2 (Database Entity): *A database entity is a digital representation of a real-world entity. Thus, a database entity is a distinguishable thing (or object) of all digital images of the real-world.*

Since data representation depends on the used data model (e.g. relational, XML, etc.) and because data representation can be extremely complex, a database entity can be considered as a (logical) construct that encapsulates all the database information that is assumed to describe the same real-world entity and hence is associated with the same database entity identifier. As we will see in the remainder of this section, we differentiate between a database entity itself, i.e. its identity, and its instance, i.e. its time-variant properties. Whereas the first is defined by the aforementioned database entity identifier, the latter

is defined by a set of data values. As a consequence, the latter depends on the used data model, but the first does not.

Duplicate elimination is the process of removing database entities from the database that refer to the same real-world entity. Whereas duplicate elimination is entity-based (it maps database entities to real-world entities), the relational data model is only semi entity-based, because not every tuple of a relational database corresponds to an own database entity, but can also represent a relationship between several database entities or can represent a multi-valued attribute. Consequently, we have a type mismatch between the entity-based consideration of the duplicate elimination process and the semi entity-based modeling of the underlying data.

In most existing approaches, duplicate elimination is only applied to single database tables so that each database entity can be adequately described by an ordinary database tuple and the aforementioned type mismatch does not matter. Nevertheless, there are many cases in which duplicate elimination need to be performed in a database with multiple tables that are linked by foreign keys. In such cases, however, a single database tuple is most often not suitable to describe a database entity sufficiently and the type mismatch need to be resolved.

Whereas the relational data model is not entity-based, the *entity-relationship model* (short ERM) is entity-based. Since relational databases are usually designed by transforming an initially developed entity-relationship schema into a set of database tables, we can use the transformation rules to interpret a relational database in an entity-based way. For that purpose we first introduce the principles of the entity-relationship model and the relational data model, then present rules for transforming an entity-relationship schema (short ER-schema) into a relation database schema (short RM-schema) and finally elaborate an entity-based interpretation of relational data.

2.1.1. Entity-Relationship Model

The entity-relationship model is a data model that was originally introduced by Chen [Che76] and that is typically used in the conceptual design of a database in order to model the considered universe of discourse in an implementation independent manner [Vos08, UGMW02]. The advantage of the entity-relationship model is that it is simple to understand even for people from outside the database community, because it is based on the native concepts of *entities* and *relationships* between entities. Moreover it has a straightforward graphical notation that helps to minimize the communication problem between the application informed, but database technology unfamiliar users and the application unfamiliar database designers.

To model entities and their interactions, the entity-relationship model is based on four main components:

- **Attribute:** An *attribute* describes a real-world property as a name, a date, or a place and can be associated to an entity as well as a relationship between entities. An attribute has a name and a domain. It can be single-valued, i.e. its instance is a single domain element, or multi-valued, i.e. its instance is a subset of the domain. Examples of single-valued attributes are age or weight, because very object has a single age and a single weight (of course this uniqueness is restricted to a single measure). Examples of multi-valued attributes are hobbies or qualifications, because every

person can have more then one hobby or can have several qualifications. Moreover, an attribute can be atomic or can be composed by a set of further attributes. A typical example of a composed attribute is the address of a person, because it is often composed by the attributes *'street'*, *'house number'*, *'zip-code'*, and *'city'*.

The concrete value of an entity e in an attribute A is in the following denoted as $e[A]$. The same holds for relationships. To make that notion useable across types (entity type as well as relationship type) we assume that every two database attributes A_1 and A_2 are clearly distinguishable by their names.

- **Entity Type:** An *entity set* is a collection of entities that share some characteristics, i.e. attributes and relationship roles. For instance, all students have a matriculation number and a library identifier (attributes). Moreover, they take exams and attend to lectures (relationships). An *entity type* models the time-invariant structure of the characteristics shared by the entities of a specific entity set and hence can be considered as the set's schema. Since all entities of an entity set are distinguishable the corresponding entity type must contain an attribute or a set of attributes whose values are unique for all entities that possibly belongs to the considered set at any time. This attribute set is called the entity type's primary key.

- **Relationship Type:** Relationships model interactions and connections between different entities. A relationship type models the time-invariant structure of a set of semantically equivalent relationships. Since entities can be involved in a relationship in several ways, each involvement is usually annotated with a role name. A relationship can be *one-to-many*, i.e. one of the relationship roles is unique and one entity can participate in that role maximally once, or can be *many-to-many*, i.e. none of the roles is unique and entities can be participate in each role for several times. Due to interactions between entities are often not only described by the involved entities themselves, relationships can be associated with attributes. For instance, two persons meet each other at a specific place at a specific time.

- **Inheritance:** Entity sets can be formed at somebody's leisure. Consequently, an entity set can be a subset of another entity set and its corresponding entity type can therefore be a specialization of another entity type. Such kinds of inheritance are incorporated into the entity-relationship model by a specific *is-a*-relationship that connects an entity type (the *subtype*) with another entity type (the *supertype*) if the first is a specialization of the second. A set of *is-a*-relationships that refer to the same supertype can be disjoint (an entity of the supertype can only belong to one of the subtypes), non-disjoint, total (an entity of the supertype belongs to at least one of the subtypes), or partial (i.e. non-total).

To resolve the considered type mismatch, we need to differentiate between an entity that we already have defined as a distinguishable thing and its instance, i.e. its time-variant characteristics. Obviously, an entity can belong to several entity sets and hence its characteristics can be described by several entity types. The description of an entity by a specific entity type is called an *entity tuple* that in turn is defined as an element of the Cartesian Product of the domains of all the attributes of the corresponding entity

type. Since at each time the characteristic of an entity is unique, each entity can only be described by one entity tuple per entity type.

Definition 3 (Database Entity Instance): *In the entity-relationship model, the instance of a database entity is defined as the set of all entity tuples that contains the entity's identifier in the attribute DEI. Let e be a database entity and let db be a database that is represented by using the entity-relationship model, we denote the instance of e in db at time τ, i.e. the set of entity tuples in db at time τ that describe e, as Inst(e, db, τ).*

Notice, many database books, as for example [Vos08], do not distinguish between and entity and its instance. In our work, however, such a distinction is inevitable, because we consider the identity of distinguishable things of the real-world and not the identity of tuples.

Whereas the entity's instance is described by a set of entity tuples, we need a universal identifier to describe the entity itself. This leads us to the problem that primary keys are usually defined for single entity types and guarantee uniqueness only for the type's tuples. To identify entities, however, we need keys that are unique beyond single entity types and hence are unique for the whole universe of discourse. For that reason, we introduce a special attribute *database entity identifier* (short *DEI*) that is considered to be unique for all considered entities whether or not they originate from the same database. Moreover, we assume that the *DEI* is the primary key of each entity table. As we think, this assumption is not too restrictive, because without having the ability of universally identifying entities, detecting duplicates (especially across sources) is per definition meaningless. Furthermore, in cases where an entity type already has a primary key, this key can be further used as a secondary key.

Since we distinguish between entities and their instances, we also need such a distinction on the level of entity types. The set of entity tuples that belong to an entity type is called as its instance and the set of entities that is described by an entity type is called as the type's extension. Whereas the schema of an entity type is time-invariant, the instance and the extension are not, because the membership of a single entity to a specific entity set as well as its attribute values and relationship roles can change over time. For example, a person can be a student at one time and can be a professor at another time. Moreover, her age increases by one once in each year.

Definition 4 (Entity Type Instance): *The instance of an entity type is the set of entity tuples that belong to this type. Let E be an entity type, we denote the instance of E at time τ as Inst(E, τ).*

Definition 5 (Entity Type Extension): *The extension of an entity type is the set of database entities that are described by this type. Let E be an entity type, we denote the extension of E at time τ as Ext(E, τ). Since database entities are described by entity tuples, the extension of an entity type can be derived from its instance, i.e. Ext$(E, \tau) = \{e \mid t \in$ Inst$(E, \tau), t[$DEI$] = e\}$.*

The instance of a database contains all tuples that belongs to the instance of any of its entity types. Accordingly, the extension of a database contains all database entities that belongs to the extension of any of its entity types. Let *db* be a database, the instance and the extension of *db* at time τ are denoted as *Inst*(db, τ) and *Ext*(db, τ) respectively.

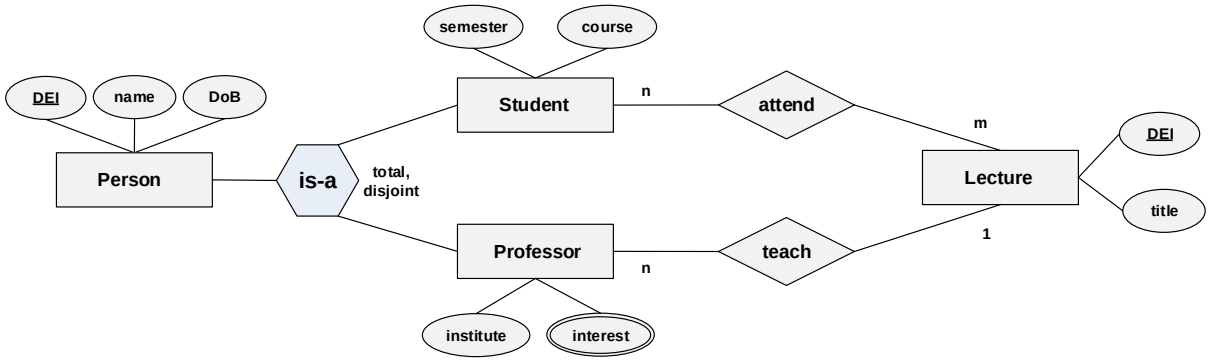


Figure 2.1.: Sample ER-schema

Specializations can be formalized by using the notion of extension. If an entity type E_l is connected to an entity type E_k by an *is-a*-relationship, it holds that at each time the extension of E_l is a subset of the extension of E_k, i.e., $\forall \tau \in \mathit{Time}\colon \mathit{Ext}(E_l, \tau) \subseteq \mathit{Ext}(E_k, \tau)$. Moreover, a set of *is-a*-relationships that connects a set of subtypes $\mathcal{E}$ to a supertype E_k is

(i) *total*, if: $\forall \tau \in \mathit{Time}\colon \forall e \in \mathit{Ext}(E_k, \tau)\colon \exists E_l \in \mathcal{E}\colon e \in \mathit{Ext}(E_l, \tau)$ and *partial* else

(ii) *disjoint*, if: $\forall \tau \in \mathit{Time}\colon \forall E_{l_1}, E_{l_2} \in \mathcal{E}\colon \mathit{Ext}(E_{l_1}, \tau) \cap \mathit{Ext}(E_{l_2}, \tau) = \emptyset$ and *non-disjoint* else.

Note, in cases where the point of time is clear from context, e.g. in duplicate elimination we only consider the starting time of the elimination process, we sometimes omit the quantification of time by simply writing $\mathit{Inst}(e, db)$, $\mathit{Inst}(E)$, $\mathit{Ext}(E)$, $\mathit{Inst}(db)$, and $\mathit{Ext}(db)$.

Obviously, the membership to the extension of an entity type models a characteristic of an entity just as an attribute. Nevertheless, since collections of entities are often formed based on coincident properties of attribute values, the concepts of entity type membership and attributes cannot be clearly distinct, but instead whether an entity characteristic is represented by type membership or is represented by an attribute depends on the used approach for designing the entity-relationship schema. For instance, the gender of a person can be modeled by an attribute *'gender'*, but can be also implicitly modeled by the membership to one of the entity types *'Male'* and *'Female'*. In general, both characteristics are often correlated, e.g. a person can only belong to the extension of an entity type *'Senior'* if its value for the attribute *'age'* is 65 or greater.

Example 11 *For illustration we consider an entity-relationship schema that is graphically depicted in Figure 2.1. The schema has the four entity types* 'Person'*,* 'Student'*,* 'Professor'*, and* 'Lecture' *that each has several attributes. Whereas all other attributes are single-valued, the attribute* 'interest' *from type* 'Professor' *is multi-valued. The types* 'Student' *and* 'Professor' *are specializations of the type* 'Person' *and hence inherit all its attributes and relationship roles. Since no person can be a student and a professor at the same time, the specializations are disjoint. Moreover, we do not model person that are neither students nor professors. Thus the set of both specializations is total. Due to the given case of inheritance, the extensions of the types* 'Person' *and* 'Student' *as well as* 'Person' *and* 'Professor' *are overlapping and hence some entities belong to the extensions of multiple types. The entity type* 'Student' *is connected with the entity type* 'Lecture' *by the relationship type* 'attend' *and the entity type* 'Professor'

is connected with 'Lecture' *by the relationship type* 'teach'. *Whereas the first relationship type is many-to-many, the second type is only one-to-many. Both relationship types do not have attributes.*

Note, in this example the roles of each relationship type belong to different entity types. For that reason, we can use the names of the entity types to clearly name the corresponding roles and therefore do not need to define the role names explicitly.

2.1.2. Relational Data Model

The relational data model [UGMW02, KE99, Vos08] is currently one of the most popular data models and was first introduced by Codd et al. [Cod83]. A relational database consists of a set of database relations which are commonly also called as database tables (in the rest of this thesis, we will use both terms interchangeably). Besides tables, a relational database can possess a set of constraints that are either restricted to single tables, e.g. primary keys, or consider multiple tables, e.g. foreign keys.

Similar to entity types, a database table has a time-invariant schema and a time-variant instance. By considering first normal form [UGMW02, KE99] the schema of a relational database table only consists of a set of non-composed single-valued attributes, a primary key, and a set of foreign keys. Accordingly to entity tuples, a database tuples of a specific database table is an element of the cross-product of all the table's attributes' domains. To model missing information these domains are usually extended by the null value to which we will simply refer as '$\perp$' or 'NULL'.

Note, besides its primary key value each database tuple has a numerical value, called *tuple identifier*, that can be used to identify a tuple within the complete database. We sometimes use that identifier to uniquely address tuples across tables in an easy way. This usage is especially useful if some of the considered tuples have same key values as it can be the case if the primary key of one table is a foreign key that references to the primary key of another table. Since that identifier is a database internal value and hidden from the user, we do not represent it by an own attribute, but instead annotate the tuple that has the identifier i with the label t_i if required (for demonstration, see Figure 2.2).

A relational database schema is usually designed by transforming an entity-relationship schema into the relational data model. According to the ER-RM transformation rules presented in [Vos08], an entity-relationship schema is transformed into a relational database schema as follows:

- Each entity type of the entity-relationship schema is mapped to a database table where each single-valued attribute of this entity type is modeled by one of the table's attributes. Moreover, single-valued composed attributes are decomposed into their subattributes. Since such tables represent entity types, we call them *entity tables*.

  Note, due to normalization it can happen that an entity type is represented by multiple tables. In that case, we either assume a logical denormalization to a single entity table, or we assume that each of these tables represents another entity type that are connected by relationship types. Since normalization[1] is only applied in the case of independence between some of the table's attributes, an entity table is only split into multiple entity tables if the original entity type was 'poorly' modeled because it combined mutually independent information.

[1] We only consider normalizations into the second or third normal form in this thesis.

- Whereas the single-valued attributes of an entity type are modeled in the corresponding entity table, multi-valued attributes are not, but in contrast are each modeled by an extra table instead. Since such tables represent multi-valued attributes, we call them *multi-value tables*. If a multi-valued attribute is non-composed, the resultant multi-value table has exact one attribute except the foreign key attributes that references to the entity table of the corresponding entity type and has several non foreign key attributes else.

- The relationship types of an entity-relationship schema are modeled either by foreign keys in the entity tables and/or multi-value tables, or are modeled by an extra table. Usually, an extra table is only used, if the relationship is many-to-many or more than two roles are involved. Since such tables represent relationship types, we call them *relationship tables*. A relationship table contains one foreign key per involved role where each of these foreign keys references to an entity table. The primary key of a relationship table is the combination of all its foreign keys.

- *Is-a*-relationships can be transformed from the entity-relationship model into the relational data model in several ways (for more detailed information we refer to any of the database books [UGMW02, KE99, Vos08]). Note, each of these approaches has its advantages and disadvantages with respect to the complexity of different queries classes and space requirements. Therefore, the best choice always depends on the considered use case. In this thesis, however, we restrict our consideration to two approaches. The first of them does not require the use of foreign keys and therefore can be adopted to probabilistic databases even if a simple representation system is used. In contrast, the second one is standardly used in database literature and requires the least space in graphical representations.

  In the first approach, we create an extra entity table for each entity type that does not inherit from any other entity type. Each of these tables does not only contain the attributes and relationship roles of the entity type itself but also contains the attributes and the relationship roles from all of its subtypes (directly or indirectly). For modeling the memberships to the individual subtypes, the table additionally gets a boolean attribute per subtype that we call *membership attribute* in this thesis. Since this approach models a complete inheritance hierarchy within a single database table, we call it the *Single Table Approach* in the rest of this thesis.

  In contrast, the second approach creates one entity table per entity type. The table contains only the attributes and relationship roles of the modeled type. The *is-a*-relationship is then realized by defining the primary key of an entity table that represents a subtype as a foreign key that references to the primary key of the entity table that represents the direct supertype of the considered subtype. Since this approach partitions the set of attributes and relationship roles that describes a single entity and assigns each partition class to another entity table, we call it the *Vertical Partitioning Approach* in the rest of this thesis.

Following the description above, the tables of a relational database can be partitioned into three classes: (a) *entity tables* that represent entity types, (b) *relationship tables* that represent relationship types, and (c) *multi-value tables* that represent multi-valued attributes. Note that there is a fluent passage between the boundaries of the three classes. For instance, depending on the considered universe of discourse, the

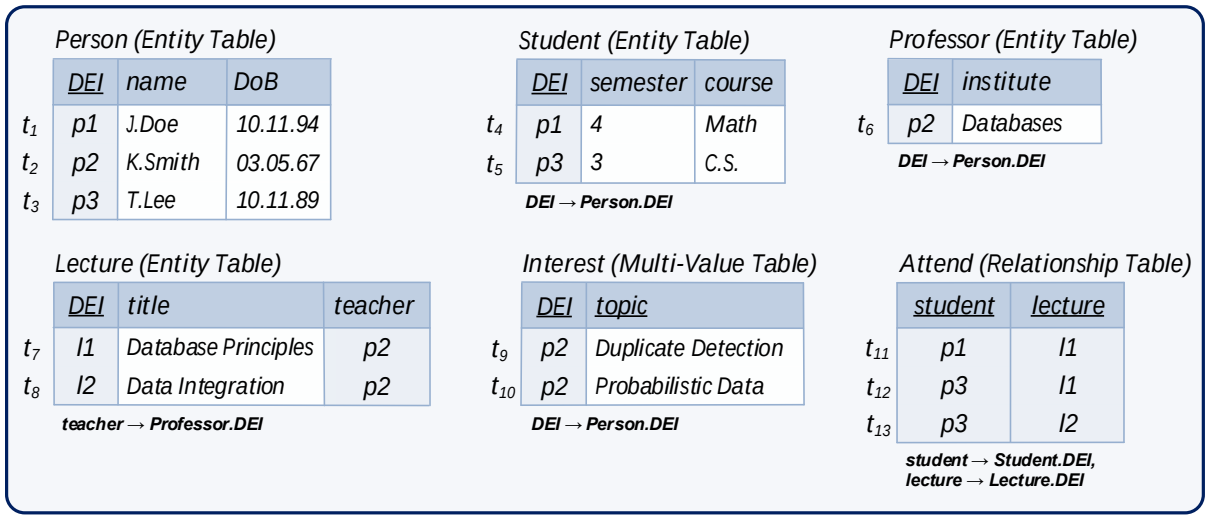


Figure 2.2.: Sample RM-schema with database instance

address of a person can be modeled in an entity-relationship schema by a composed attribute that has the subattribute *'city'*, can be modeled by a relationship that connects the entity type *'Person'* with an entity type *'City'*, but can be also modeled as an own entity type that is in a relationship with the type *'Person'*. This example illustrates that only a simple change on the underlying entity-relationship schema can have a large affect on the classification of the tables of the transformed relational database. This fact, however, is not necessarily a negative one, because it is not untypical that same (or similar) information can be modeled in several ways.

Since we assume *DEI*s for the entity-relationship model, we assume all primary keys of an entity table to consist of a single attribute, that is the *DEI*. Foreign keys most often reference to an entity table (the single exception is a multi-value table that reference to a relationship table). Consequently, we consider (most) foreign keys to consist only of a single attribute as well. Moreover, in this thesis we do not consider multi-value tables.

Following the above described mapping between entity types and entity tables, we can adopt the notions of database entity instances, entity type instances and entity type extensions from the entity-relationship model to the relational data model and therefore distinguish between an entity table's *schema*, i.e. the time-invariant structure of all shared characteristics like attributes or relationship roles, its *instance*, i.e. its set of database tuples, and its *extension*, i.e. the set of the tuple's corresponding database entities. Since an entity can belong to the extensions of several entity tables, the instance of a database entity can consists of several tuples from different entity-tables, but contains at most one tuple per entity table. If clear from context, we usually simply write $t \in T$ instead of $t \in \mathit{Inst}(T)$ if we refer to a tuple t from the instance of a database table T.

Example 12 *Figure 2.2 presents the relational database schema that results from transforming the ER-schema from Figure 2.1 into the relational data model by using the Vertical Partitioning Approach for modeling inheritance. From each entity type one entity table is derived. Thus, the schema contains four entity tables. Whereas the relationship type* 'Attend' *is modeled by an extra table, i.e. the table* 'Attend'*, the relationship type* 'teach' *is modeled within the entity table* 'Lecture' *by using a foreign key*

that references to the entity table 'Professor'. *Finally, the multi-valued attribute* 'interest' *is transformed into an extra table* 'Interest'.

This example shows that tables can be simply classified into entity tables, relationship tables, and multi-value table by considered the structure of their primary keys. The primary key of an entity table only consists of the DEI. In contrast, the primary key of a relationship table consists of several foreign keys that each references to entity tables. In the case of a multi-value table, the primary key is composed by one or more attributes and a foreign key that references to an entity table.

An instance of the database is represented in the same figure. This sample instance illustrates that tuples from different tables, e.g. tuple t_1 and tuple t_4, can share the same key value and we therefore require the tuple identifier to distinguish them. The corresponding extensions of the entity tables are Ext('Person') $= \{p_1, p_2, p_3\}$, *Ext*('Student') $= \{p_1, p_3\}$, *Ext*('Professor') $= \{p_2\}$, *and Ext*('Lecture') $= \{l_1, l_2\}$. *As we can see, some of the entities belong to the extensions of several tables, e.g. entity p_2, and hence are represented in the database by multiple tuples. Since we consider the DEI as primary key in each table, the tuples representing the same database entity are exactly the tuples that share the same key value.*

In this thesis, database queries will be either represented by expression of the relational algebra or by SQL-Statements. We use a version of the relational algebra that has the basic operations σ (selection), π (projection), $\times$ (cross product), $\bowtie$ (join), and δ (renaming) as it is described in [KE99]. SQL queries are based on the 2003 SQL-Standard [SQL03].

2.2. Clustering Background

Duplicate detection results are represented by the use of clusterings and a variety of quality measures for duplicate detection are based on comparing two clusterings. For that reason, we introduce some preliminaries on clusterings and operations on clusterings in this section.

Definition 6 (Cluster & Clustering): *A cluster $C = \{a, \ldots, l\}$ is a set of base elements (e.g. database entities). A clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ is a set of clusters. The range of a clustering $\mathcal{C}$ is defined as the set of all base elements that belong to any of its clusters, i.e. $rng(\mathcal{C}) = \bigcup \mathcal{C}$. The set C-All$(S)$ is the set of all clusterings that can be defined on an element set S, i.e. $\forall \mathcal{C} \in$ C-All(S): $rng(\mathcal{C}) = S$.*

Theoretically, C-All(S) contains a clustering for each possible combination of subsets of S so that each element $a \in S$ is contained in at least one of these subsets, e.g. $\mathcal{C}_1 = \{\langle a, b, c\rangle\}$, $\mathcal{C}_2 = \{\langle a\rangle, \langle a, b, c\rangle\}$, $\mathcal{C}_3 = \{\langle a, b\rangle, \langle a, b, c\rangle\}$, $\mathcal{C}_4 = \{\langle a, b\rangle, \langle b\rangle, \langle a, b, c\rangle\}$ and so on.

Since this number is extremely large even for small element sets, we make a restriction and declare that a cluster C is only allowed to belong to a clustering $\mathcal{C}$, if $\mathcal{C}$ does not contain a cluster C' that is a strict superset of C, i.e. $C' \supsetneq C$. Note, this restriction does not affect any of the aspects that are considered in this thesis.

For ease of presentation and according to existing work on duplicate detection (e.g. [MWGM10]), we use angle brackets to embrace base elements that belong to the same cluster within a clustering. To pictorially present clusterings in this thesis, we use a graph based notation where each base element is represented by a node and two elements are connected by an edge, if they belong to the same cluster.

(a) $\mathcal{C}_1 = \{\langle a\rangle, \langle b\rangle, \langle c\rangle\}$ (b) $\mathcal{C}_2 = \{\langle a, b\rangle, \langle c\rangle\}$ (c) $\mathcal{C}_3 = \{\langle a, c\rangle, \langle b\rangle\}$ (d) $\mathcal{C}_4 = \{\langle a\rangle, \langle b, c\rangle\}$

(e) $\mathcal{C}_5 = \{\langle a, b\rangle, \langle a, c\rangle\}$ (f) $\mathcal{C}_6 = \{\langle a, b\rangle, \langle b, c\rangle\}$ (g) $\mathcal{C}_7 = \{\langle a, c\rangle, \langle b, c\rangle\}$ (h) $\mathcal{C}_8 = \{\langle a, b, c\rangle\}$

Figure 2.3.: The possible clusterings $\mathcal{C}_1$ to $\mathcal{C}_8$ of $S = \{a, b, c\}$

Example 13 *For illustration, we consider the set $S = \{a, b, c\}$. The eight possible clusterings of S are depicted in Figure 2.3.*

Two clusters C_i and C_j are disjoint, iff they do not overlap. For simplification, we introduce a short notion of set disjointness and set overlap by using the symbols '$\otimes$' and '$\odot$', i.e., $C_i \otimes C_j \Leftrightarrow C_i \cap C_j = \emptyset$ (C_i and C_j are disjoint) and $C_i \odot C_j \Leftrightarrow C_i \cap C_j \neq \emptyset = \neg(C_i \otimes C_j)$ (C_i and C_j overlap). A clustering is called to be *cluster-disjoint*, if all its clusters are pairwise disjoint. In mathematics, a cluster-disjoint clustering of the element set S is called a partition of S. However, since in duplicate detection literature the term 'clustering' is primarily used, we will do the same but still have in mind that a cluster-disjoint clustering of S always corresponds to a partition of S.

Definition 7 (Cluster-disjoint Clustering): *A clustering $\mathcal{C}$ is called to be cluster-disjoint, iff all its clusters are pairwise disjoint. A cluster-disjoint clustering $\mathcal{C}$ is a partition on $rng(\mathcal{C})$ and its clusters are partition classes. The set C-All$_{\otimes}(S)$ is the set of all cluster-disjoint clusterings that can be defined on an element set S.*

In the graph-based notation, a clustering is cluster-disjoint if its graphical representation is equivalent to its transitive closure.

Example 14 *For illustration, from the eight possible clusterings depicted in Figure 2.3 only the clusterings $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$, and $\mathcal{C}_8$ are cluster-disjoint.*

Sometimes we are only interested in a part of a clustering. For that reason, we introduce the projection operator.

Definition 8 (Projection of Clusterings): *The result of projecting a clustering $\mathcal{C}$ on the element set $S \subseteq rng(\mathcal{C})$ is the clustering $\pi_S(\mathcal{C}) = \{C \cap S \mid C \in \mathcal{C}\}$.*

Note, since each cluster of the projection result is a subset of a cluster of the projected clustering, the projection of a cluster-disjoint clustering is always cluster-disjoint as well.

Example 15 *The projection of the clustering $\mathcal{C} = \{\langle a, b\rangle, \langle c, d\rangle, \langle e\rangle\}$ on the element set $S = \{a, b, d\}$ results in the clustering $\mathcal{C}' = \pi_S(\mathcal{C}) = \{\langle a, b\rangle, \langle d\rangle\}$.*

Sometimes we need to select some particular clusterings from a set of clusterings. For that purpose, we introduce the clustering selection operator.

Definition 9 (Clustering Selection): *Let CS be a set of clusterings and let ϕ be a predicate that can be applied to clusterings of CS. Selecting elements of the set CS by using the predicate ϕ is defined as:*

$$\sigma_{\phi}(CS) = \{\mathcal{C} \in CS \mid \phi(\mathcal{C}) = true\}$$

Example 16 *Let $CS = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ be a set containing the clusterings $\mathcal{C}_1 = \{\langle a, b\rangle, \langle c, d, e\rangle\}$, $\mathcal{C}_2 = \{\langle a, b\rangle, \langle c\rangle, \langle d, e\rangle\}$, and $\mathcal{C}_3 = \{\langle a, b, c\rangle, \langle d, e\rangle\}$. Moreover, let $\phi_1 \colon |\mathcal{C}| = 3$ be a boolean predicate that returns 'true' if the evaluated clustering $\mathcal{C}$ has exact three clusters and returns 'false' otherwise. From processing the selection $\sigma_{\phi_1}(CS)$ only the clustering $\mathcal{C}_2$ results. In contrast from processing the selection $\sigma_{\phi_2}(CS)$ with the boolean predicate $\phi_2 \colon \{\langle a, b\rangle, \langle c\rangle\} \prec \mathcal{C}$ that returns 'true' if the evaluated clustering dominates the clustering $\{\langle a, b\rangle, \langle c\rangle\}$ and returns 'false' otherwise, the clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$ result.*

A clustering is dominated by another clustering, if the second clustering contains the information of the first clustering.

Definition 10 (Domination of Clusterings): *A clustering $\mathcal{C}_i$ is dominated by a clustering $\mathcal{C}_j$ (in symbols $\mathcal{C}_i \prec \mathcal{C}_j$), iff $rng(\mathcal{C}_i) \subseteq rng(\mathcal{C}_j) \wedge \pi_{rng(\mathcal{C}_i)}(\mathcal{C}_j) = \mathcal{C}_i$.*

Example 17 *For illustration, the clustering $\mathcal{C}_1 = \{\langle a, b\rangle, \langle c\rangle, \langle d, e\rangle\}$ dominates the clustering $\mathcal{C}_2 = \{\langle a, b\rangle, \langle d\rangle\}$ because $\pi_{\{a,b,d\}}(\mathcal{C}_1) = \mathcal{C}_2$, but does not dominate the clustering $\mathcal{C}_3 = \{\langle a, b\rangle, \langle d\rangle, \langle e\rangle\}$ because $\pi_{\{a,b,d,e\}}(\mathcal{C}_1) = \{\langle a, b\rangle, \langle d, e\rangle\} \neq \mathcal{C}_3$.*

Note, a clustering always dominates itself. Moreover, a disjoint clustering can only dominate disjoint clusterings, but in turn can be dominated by non-disjoint clusterings. Let $\mathcal{C}_i$ and $\mathcal{C}_j$ be two clusterings. If $rng(\mathcal{C}_i) \subseteq rng(\mathcal{C}_j)$, it holds that $S \cap rng(\mathcal{C}_i) \subseteq S \cap rng(\mathcal{C}_j)$ for every set S. Thus, if clustering $\mathcal{C}_j$ dominates clustering $\mathcal{C}_i$, the projection of $\mathcal{C}_j$ on a set S dominates the projection of $\mathcal{C}_i$ on S for every set S. We say that a clustering $\mathcal{C}_j$ strongly dominates another clustering $\mathcal{C}_i$, if $\mathcal{C}_i \prec \mathcal{C}_j$ and $rng(\mathcal{C}_j) \supsetneq rng(\mathcal{C}_i)$.

Multiple clusterings can be integrated to a single clustering by assuming a negative relationship (the elements are assumed to belong to different clusters) for all elements that only belong to the range of one the integrated clusterings.

Definition 11 (Integration of Clusterings): *Two clusterings $\mathcal{C}_i$ and $\mathcal{C}_j$ can be integrated to the single clustering $\mathcal{C}_{ij}$ (in symbols: $\mathcal{C}_{ij} = \mathcal{C}_i \boxtimes \mathcal{C}_j$) by the union of their clustersets: $\mathcal{C}_{ij} = \mathcal{C}_i \boxtimes \mathcal{C}_j = \mathcal{C}_i \cup \mathcal{C}_j$. The integration operator is reflexiv, symmetric and associativ. The n-ary integration of the k clusterings $\mathcal{C}_1, \ldots, \mathcal{C}_k$ is defined as a subsequent execution of $k-1$ binary integrations, i.e.: $\boxtimes_{i=1}^{k} \mathcal{C}_i = ((\mathcal{C}_1 \boxtimes \mathcal{C}_2) \boxtimes \ldots) \boxtimes \mathcal{C}_k$.*

Example 18 *The integration of the two clusterings $\mathcal{C}_1 = \{\langle a, b\rangle\}$ and $\mathcal{C}_2 = \{\langle a, c\rangle, \langle d, e\rangle\}$ is the clusteirng $\mathcal{C}_{12} = \mathcal{C}_1 \boxtimes \mathcal{C}_2 = \{\langle a, b\rangle, \langle a, c\rangle, \langle d, e\rangle\}$.*

Theorem 1 *The clustering $\mathcal{C}_{ij} = \mathcal{C}_i \boxtimes \mathcal{C}_j$ is a cluster-disjoint clustering, iff $\mathcal{C}_i$ and $\mathcal{C}_j$ are cluster-disjoint clusterings and either their ranges are disjoint or the elements of the ranges' overlap only form clusters that belongs to $\mathcal{C}_i$ and $\mathcal{C}_j$, i.e. $\forall C_1 \in \mathcal{C}_i \colon \forall C_2 \in \mathcal{C}_j \colon C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$.*

Theorem 1 is proved by Proof 13 in Section A.1.

Example 19 *For illustration, we consider the three cluster-disjoint clusterings $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$:*

$$\begin{aligned} \mathcal{C}_1 &= \{\langle a,b\rangle, \langle c\rangle\} & \mathcal{C}_1 \boxtimes \mathcal{C}_2 &= \{\langle a,b\rangle, \langle c\rangle, \langle e,f,g\rangle, \langle h\rangle\} \\ \mathcal{C}_2 &= \{\langle e,f,g\rangle, \langle h\rangle\} & \mathcal{C}_1 \boxtimes \mathcal{C}_3 &= \{\langle a,b\rangle, \langle c\rangle, \langle d,h\rangle\} \\ \mathcal{C}_3 &= \{\langle a,b\rangle, \langle d,h\rangle\} & \mathcal{C}_2 \boxtimes \mathcal{C}_3 &= \{\langle a,b\rangle, \langle d,h\rangle, \langle e,f,g\rangle, \langle h\rangle\} \end{aligned}$$

The integration of $\mathcal{C}_1$ and $\mathcal{C}_2$ results in the clustering $\mathcal{C}_1 \boxtimes \mathcal{C}_2$ that is cluster-disjoint, because $\mathcal{C}_1$ and $\mathcal{C}_2$ do not have an element in common. Although the clusterings $\mathcal{C}_1$ and $\mathcal{C}_3$ have two elements in common, i.e. 'a' and 'b', their integration result is cluster-disjoint, because the shared elements form in both clusterings the same cluster. In contrast, the result of integrating $\mathcal{C}_2$ and $\mathcal{C}_3$ is not cluster-disjoint, because they both contain the element 'h' in their range, but whereas 'h' forms a cluster by its own in $\mathcal{C}_2$ it forms a cluster with the element 'd' in $\mathcal{C}_3$. Therefore, the clustering $\mathcal{C}_2 \boxtimes \mathcal{C}_3$ contains 'h' in two clusters and thus is not cluster-disjoint.

Theorem 2 *Let $\mathcal{C}$ be a clustering and $S_i, S_j \subseteq rng(\mathcal{C})$ be two disjoint subsets of the clustering's range. If there is no cluster $C \in \mathcal{C}$ its elements belong to S_i and S_j, i.e. $\forall C \in \mathcal{C} \colon C \circledast S_i \vee C \circledast S_j$, it holds: $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) = \pi_{S_i \cup S_j}(\mathcal{C})$.*

Theorem 2 is proved by Proof 14 in Section A.1.

Example 20 *For illustration, we consider the clustering $\mathcal{C} = \{\langle a,b\rangle, \langle a,c\rangle, \langle d\rangle, \langle e,f\rangle\}$ and consider the three sets $S_1 = \{a,b\}$, $S_2 = \{d,e\}$, and $S_3 = \{c,f\}$:*

$$\begin{aligned} \pi_{S_1}(\mathcal{C}) \boxtimes \pi_{S_2}(\mathcal{C}) &= \{\langle a,b\rangle, \langle a\rangle, \langle d\rangle, \langle e\rangle\} & \pi_{S_1 \cup S_2}(\mathcal{C}) &= \{\langle a,b\rangle, \langle a\rangle, \langle d\rangle, \langle e\rangle\} \\ \pi_{S_1}(\mathcal{C}) \boxtimes \pi_{S_3}(\mathcal{C}) &= \{\langle a,b\rangle, \langle a\rangle, \langle c\rangle, \langle f\rangle\} & \pi_{S_1 \cup S_3}(\mathcal{C}) &= \{\langle a,b\rangle, \langle a,c\rangle, \langle f\rangle\} \end{aligned}$$

The integration of $\pi_{S_1}(\mathcal{C})$ and $\pi_{S_2}(\mathcal{C})$ results in a clustering that is equivalent to $\pi_{S_1 \cup S_2}(\mathcal{C})$ because there is no cluster in $\mathcal{C}$ that contains elements that belongs to S_1 and S_2. In contrast, the result of integrating $\pi_{S_1}(\mathcal{C})$ and $\pi_{S_3}(\mathcal{C})$ is not equivalent to $\pi_{S_1 \cup S_3}(\mathcal{C})$ because the cluster $\langle a,c\rangle$ has an element of S_1, i.e. 'a', and has an element of S_3, i.e. 'c'. In other words, by performing projection on S_1 and S_3 separately first, the cluster $\langle a,c\rangle$ is split into the clusters $\langle a\rangle$ and $\langle c\rangle$ that is not re-merged by the integration operator.

Theorem 3 *Let $\mathcal{C}$ be a clustering and let $S = \{S_1, \ldots, S_k\}$ be a partition of $rng(\mathcal{C})$. If there is no cluster $C \in \mathcal{C}$ its element belongs to different partition classes, i.e. $\forall C \in \mathcal{C} \colon \forall S_i, S_j \in S \colon C \circledast S_i \vee C \circledast S_j$, it holds: $\bigboxtimes_{S_i \in S}\big(\pi_{S_i}(\mathcal{C})\big) = \mathcal{C}$.*

Theorem 3 is proved by Proof 15 in Section A.1.

Definition 12 (Factorization of Clusterings): *A set of clusterings $\mathcal{F} = \{\mathcal{C}_{F_1}, \ldots, \mathcal{C}_{F_k}\}$ is called a factorization of the clustering $\mathcal{C}$, iff $\mathcal{C}$ can be reconstructed from $\mathcal{F}$ by using the integration operator, i.e. $\mathcal{C} = \bigboxtimes\mathcal{F}$. The clusterings in $\mathcal{F}$ are called factors and the sets in $\mathcal{F}^{set} = \{rng(\mathcal{C}_{F_i}) \mid \mathcal{C}_{F_i} \in \mathcal{F}\}$ are called factor sets.*

Theorem 3 exactly describes the cases in which a partition of a clustering range can be used to construct a factorization of that clustering. Thus, let $\mathcal{C}$ be a clustering and let $S = \{S_1, \ldots, S_k\}$ be a partition of $rng(\mathcal{C})$. If there is no cluster $C \in \mathcal{C}$ whose elements belong to different partition classes, the set $\mathcal{F} = \{\pi_{S_i}(\mathcal{C}) \mid S_i \in S\}$ is a factorization of $\mathcal{C}$ and the partition classes $S_1, \ldots, S_k$ are the factor sets.

Chapter 3

Probabilistic Databases

In this chapter we survey existing approaches for modeling and managing probabilistic data. We start with a formal definition of incomplete databases and probabilistic databases, and give a formal consideration on the possible worlds semantics in Section 3.2. Then we discuss several systems for succinctly representing probabilistic databases in Section 3.3. In Section 3.4, we adapt the entity-based interpretation from certain relational data to probabilistic relational data. Section 3.5 contains an overview on principles and techniques for querying probabilistic databases. Finally, in Section 3.6 we discuss some further interesting aspects as referential integrity, schema design, sources of probabilities, and other work on probabilistic databases that has not been covered in the main part of this chapter.

3.1. Possible Worlds

In Chapter 1, we introduce the concept of probabilistic databases only informally by presenting a simple example. For that reason, we present the underlying theory of modeling and querying uncertain databases in a more formal way in this chapter. Although uncertainty can be modeled by using other concepts as possibility theory [Zad78] or Dempster-Shafer theory [Sha76], we primarily treat uncertain data as incomplete data or probabilistic data in this thesis. Research on incomplete data [IL84, SORK11] and probabilistic data [CP87, BGMP92, Pit94, DS96, FR97, LLRS97] starts in the mid-eighties and to date has been covered by several works.

According to Suciu et al. [SORK11], an *incomplete database* is defined as follows:

Definition 13 (Incomplete Database): *An* incomplete database *idb is a set of possible worlds* $\boldsymbol{W} = \{W_1, \ldots, W_k\}$ *where each world* $W \in \boldsymbol{W}$ *is a conventional database instance and all worlds are defined on the same database schema.*

The set **W** is also called the *possible world space* of *idb*. In this thesis we restrict to incomplete databases with finite sets of possible worlds. Note, in several research directions such as certain query answering, the alternative database instances of an incomplete database are often also denoted as the database's *possible repairs* [Ber11, DPW12]. According to Definition 13, all possible worlds of an incomplete database are defined on the same schema. In order to distinguish this schema from the schema of the compact representation (see Section 3.3), we denote it as *world schema*.

The concept of probabilistic databases extends the concept of incomplete databases by ascribing probabilities to the individual worlds.

Definition 14 (Probabilistic Database): *A probabilistic database pdb is a probability space*[1] $(\boldsymbol{W}, Pr)$ *where* $\boldsymbol{W}$ *is a world space and* Pr *is a discrete probability distribution over these worlds, i.e.* $Pr\colon \boldsymbol{W} \rightarrow (0,1]$ *is a function so that* $\sum_{W \in \boldsymbol{W}} Pr(W) = 1$.

A tuple belongs to an uncertain database, if it belongs to at least one of its possible worlds. The marginal probability of a tuple t in a probabilistic database $pdb = (\mathbf{W}, Pr)$ is described by the probability mass function p and results in the accumulative probability of all possible worlds the considered tuple belongs to, i.e. $p(t) = \sum_{W \in \mathbf{W}, t \in W} Pr(W)$. A tuple t is called a *maybe-tuple* if it belongs to at least one possible world, but does not belong to all possible worlds, i.e. $p(t) \in (0,1)$. It is called a *certain-tuple* if it belongs to all possible worlds, i.e. $p(t) = 1$. Moreover, let $p_\wedge$ and $p_\vee$ be two probability mass functions that describe the probability that the 'true' world of *pdb* contains all tuples of tuple set T (function $p_\wedge$) or contains at least one tuple of tuple set T (function $p_\vee$) respectively, i.e. $p_\wedge(T) = \sum_{W \in \mathbf{W}, T \subseteq W} Pr(W)$ and $p_\vee(T) = \sum_{W \in \mathbf{W}, T \cap W \neq \emptyset} Pr(W)$.

In an incomplete database, two maybe-tuples t_r and t_s are called *independent*, if there exists at least one possible world that contains none of them, there exists at least one possible world that contains t_r but does not contain t_s, there exists at least one possible world that contains t_s but does not contain t_r, and there exists at least one possible world that contains both. Moreover, two maybe-tuples t_r and t_s are called *exclusive* if they do not coexist in a possible world and a tuple t_s is called to be *included* by a tuple t_r if all possible worlds that contains t_r contains t_s as well.

In a probabilistic database, correlations (in this thesis, we use the terms *correlation* and *dependency* interchangeably) are defined in a more subtler way because they also depend on the worlds' probabilities. For instance, two maybe-tuples t_r and t_s are independent if $p_\wedge(\{t_r, t_s\}) = p(t_r) \times p(t_s)$ and therefore $p_\vee(\{t_r, t_s\}) = 1 - (1 - p(t_r)) \times (1 - p(t_s))$. Table 3.1 lists the conditions that need to hold for the probability mass functions $p_\wedge$ and $p_\vee$ if the two maybe-tuples t_r and t_s are in one of the three above discussed states of dependency.

Independence:	$p_\wedge(\{t_r, t_s\}) = p(t_r) \times p(t_s)$	$p_\vee(\{t_r, t_s\}) = 1 - (1 - p(t_r)) \times (1 - p(t_s))$
Exclusion:	$p_\wedge(\{t_r, t_s\}) = 0$	$p_\vee(\{t_r, t_s\}) = p(t_r) + p(t_s)$
Inclusion (t_r includes t_s):	$p_\wedge(\{t_r, t_s\}) = t_r$	$p_\vee(\{t_r, t_s\}) = t_s$

Table 3.1.: Dependencies between probabilistic database tuples

[1] Actually, a probability space [Kol60, CK05] is a triple $(\Omega, \mathcal{F}, P)$ where Ω is a sample space, $\mathcal{F} \subseteq 2^\Omega$ (the set of all events) is a σ-algebra, and P is a probability measure defined on $\mathcal{F}$. In the case of a probabilistic database, Ω corresponds to the set of possible worlds $\mathbf{W}$. Moreover, if consideration is restricted to discrete probability distributions on finite sets of worlds, the sample space is always to be at most countable, $\mathcal{F}$ is usually implied to be 2^Ω, and Pr is a probability mass function that ascribes probabilities to points of Ω. In that case, the probability of an event results in the accumulative probability of all outcomes $o \in \Omega$ that belong to this event. For that reason, the notation $(\mathbf{W}, Pr)$ is a commonly used shorthand for $(\mathbf{W}, 2^\mathbf{W}, P(X) \mapsto \sum_{W \in X} Pr(W))$.

SELECT	*	**SELECT**	*	**SELECT**	*
FROM	People	**FROM**	People	**FROM**	People
WHERE	residence = 'Hamburg'	**WHERE**	residence $\simeq$ 'Hamburg'	**WHERE**	residence = 'Hamburg'
					WITH PROB $>= 0.8$

(a) Conventional db query (b) Uncertainty including db query (c) Uncertainty processing db query

Figure 3.1.: A sample query for each of the three query classes that can be considered in uncertain data querying

Note that three pairwise independent tuples t_r, t_s, and t_u does not need to be mutually independent, i.e. although $p_\wedge(\{t_r, t_s\}) = p(t_r) \times p(t_s)$, $p_\wedge(\{t_r, t_u\}) = p(t_r) \times p(t_u)$, and $p_\wedge(\{t_s, t_u\}) = p(t_s) \times p(t_u)$ it can hold that $p_\wedge(\{t_r, t_s, t_u\}) \neq p(t_r) \times p(t_s) \times p(t_u)$.

3.2. Possible Worlds Semantics

Queries of uncertain databases can be categorized into three classes:

- Queries that are designed with certain data in mind and hence do not address uncertainty in any of their conditions or operations. Since each of these queries has an equivalent in certain data processing, we simply refer to them as *Conventional Database Queries*.
- Queries that do not directly address uncertainty of the input database in any of their clauses, but use imprecise operators in their conditions and hence introduce uncertainty into the query answers by themselves, i.e. they produce uncertain data as output even if certain data is given as input. Note, since no uncertainty of the input data is considered, these queries can be posed to a conventional database as well. We call these queries *Uncertainty Including Database Queries*.
- Queries that directly address data uncertainty in their clauses have no equivalents in certain data processing because they are especially designed to process uncertain data. We call these queries *Uncertainty Processing Database Queries*.

Example 21 *For demonstration, we consider the three queries that are depicted in Figure 3.1. The first query* 'Which people live in Hamburg?' *(see Figure 3.1(a)) is a conventional database query because it does not address any kind of uncertainty. In contrast, the query* 'Which people live near to Hamburg?' *(see Figure 3.1(b)) belongs to the second class (uncertainty including database query) because the meaning of the geographical relation 'near' is not clearly defined so that the query is vague and the answer becomes uncertain*[2]*. Finally, the query* 'Which people live in Hamburg with a probability of at least 80%?' *(see Figure 3.1(c)) is an uncertainty processing query because it explicitly address uncertainty in its request.*

It is obvious that queries of the third class are especially designed for uncertain databases and are used to extract information on the uncertainty of a specific part of the data. Thus, whereas queries of

[2]Of course, the query answer can be considered to be certain if a specific distance measure and a specific threshold are used to separate all people living near to Hamburg from all people living not near to Hamburg. However, in this case the relation 'near' cannot be considered as vague anymore.

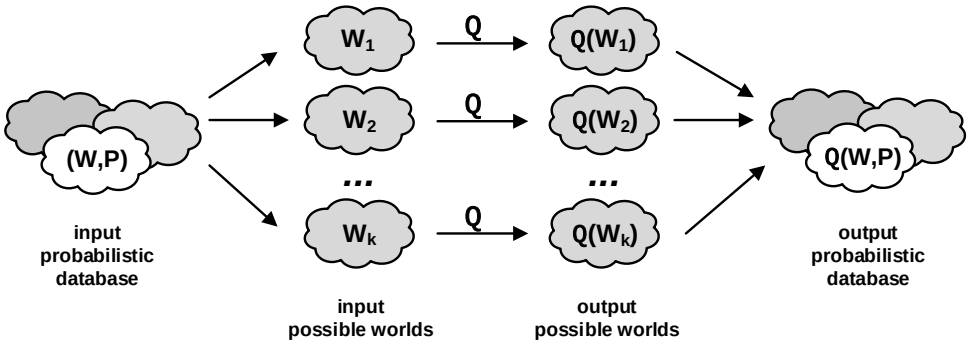


Figure 3.2.: Querying a probabilistic database within the possible worlds semantics

the first two classes consider information on uncertainty of the queried database as meta data and expect a certain database (a single possible world) as input, queries of the third class consider the database's uncertainty as instance data and hence expect to process a set of possible worlds in a collective fashion. Consequently, the semantics of a query of the third class depends on the way uncertainty need to be processed by this query. For that reason, we restrict our consideration to queries of the first two classes if not explicitly stated otherwise.

With some exceptions [LD09, GJS10], querying a probabilistic database with a query of the first two classes traditionally follows the *Possible Worlds Semantics* [AKG87, DS07c]. This semantics prescribes that a world space is queried by querying each of its worlds separately. As a consequence, the result of posing a query to a set of worlds is a second set of worlds where each of the output worlds results from querying one of the input worlds. This principle is graphically illustrated in Figure 3.2.

Logically, redundant worlds of the query result are consolidated to a single one. Thus, the probability of an output world is equivalent to the accumulative probability of all input worlds that produce this world as query result. Note, that in the case of conventional database queries from each input world only one output world can result, but several input worlds can produce the same output world. For that reason, the number of output worlds is always equal to or lower than the number of input worlds and querying can only decrease uncertainty, but can never increase uncertainty.

In summary, the possible worlds semantics can be defined as follows:

Definition 15 (Possible Worlds Semantics): *Let $pdb = (\boldsymbol{W}, Pr)$ a probabilistic database and let Q be a database query, the result of posing Q to pdb is the probabilistic database $pdb_Q = (\boldsymbol{W}_Q, Pr_Q)$ with*

$$\boldsymbol{W}_Q = \{Q(W) \mid W \in \boldsymbol{W}\}$$
$$\textit{and} \qquad \forall W \in \boldsymbol{W}_Q \colon Pr_Q(W) = \sum\nolimits_{W' \in \boldsymbol{W}, Q(W')=W} Pr(W')$$

3.3. Representation Systems

The number of possible worlds of a probabilistic database is often tremendous. Moreover, many of these worlds overlap to a large extent. Thus, storing each of these worlds separately is usually not only impractical, but also unnecessary. For that reason, several representation systems have been developed

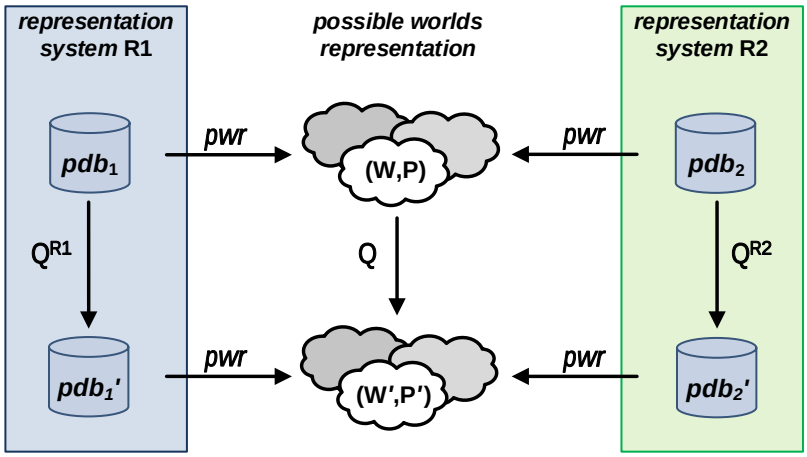


Figure 3.3.: The concept of system-specific queries to guarantee conformity to the possible worlds semantics

[Gre09, SBH+09, SD09, AKO09, SORK11] that are able to represent a probabilistic database in a more succinct way.

In a narrow sense, storing all possible worlds separately is a representation system by itself. For that reason, we consider it as a naive system that we call the *possible worlds representation*. Although the possible worlds representation is not suitable as an actually used representation system, it usually serves as a reference for all other representation systems because of its simplicity and its meaningful semantics.

Theoretically, the representation of a probabilistic database can be transformed from one representation system to another. Nevertheless, due to the different modeling power of the individual systems, a lossless transformation is not always available. A lossless transformation into the possible worlds representation, however, is a mandatory requirement of each probabilistic representation system. For that reason, we introduce the mapping *pwr* that maps a probabilistic database that is modeled in any arbitrary representation system $\mathfrak{R}$ into its possible worlds representation. Moreover, we introduce the mapping *pws* that directly maps a probabilistic database that is modeled in any arbitrary representation system $\mathfrak{R}$ to its possible world space. Thus, let *pdb* be a probabilistic database and let $pwr(pdb) = (\mathbf{W}, Pr)$ be its possible worlds representation, the mapping *pws* maps *pdb* to $\mathbf{W}$, i.e. $pws(pdb) = \mathbf{W}$.

To stay conform with the possible worlds semantics, even if the probabilistic database *pdb* is modeled in an arbitrary representation system $\mathfrak{R}$, for each database query Q we need a system-specific query $Q^{\mathfrak{R}}$ that directly computes the succinct represented probabilistic database that would result from applying Q to the possible worlds representation of *pdb*, i.e. $pwr(Q^{\mathfrak{R}}(pdb)) = Q(pwr(pdb))$. This principle is illustrated in Figure 3.3 for two different representation systems $\mathfrak{R}_1$ and $\mathfrak{R}_2$.

As previously mentioned, a several number of representation systems has been proposed in recent decades. The individual systems differ in several properties as *compactness*, *modeling power*, and *representation/query complexity* (see Figure 3.4). In general, the more compact the system, the less powerful it becomes or the more complex become representing and querying the database that is modeled by this system. In contrast, the less complex the system's representation and querying mechanisms are, the less compact or the less powerful the system becomes. Indeed, these three properties are pairwise contrary to each other, because improving one of them usually comes at the price of diminishing the others. Thus,

selecting a representation system that is suitable for a given purpose is always a trade-off between these three properties.

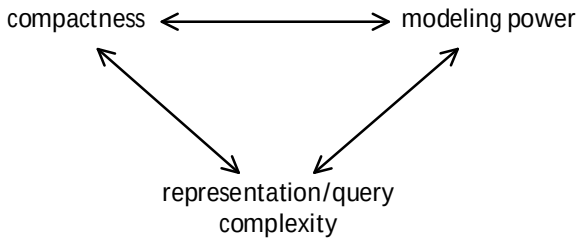


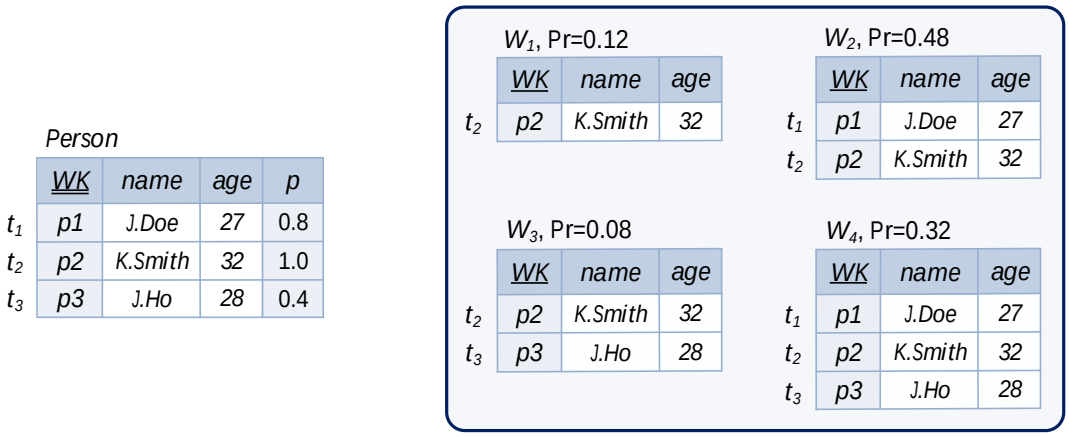
Figure 3.4.: The three contradicting system properties compactness, representation/query complexity, and modeling power

Representation systems that extends the relational data model typically model uncertainty on two levels [SORK11]. By using tuple-level uncertainty, a database tuple can be considered as a boolean random variable, i.e. it is uncertain whether the tuple belongs to the database's 'true' world or not. In contrast, by using attribute-level uncertainty, an attribute value can be considered as a random variable, i.e. the 'true' value of a tuple in an attribute is not exactly known. Whereas some representation systems such as AOR?-databases use both modeling concepts to compactly represent a probability distribution over a set of possible worlds, several representation systems only uses tuple-level uncertainty. Interestingly, attribute-level uncertainty can be converted into tuple-level uncertainty by constructing a set of mutually exclusive tuples.

In the remainder of this section, we present five probabilistic representation systems. We start with a compact, less representation/query complex, but also less powerful system and close with a powerful system that is less compact and highly representation/query complex. Therefore, by introducing these representation systems one after another, we increase the modeling power at the cost of compactness and representation/query simplicity step by step. Finally, we shortly discuss the differences in modeling power of all these models in more detail. Querying a succinctly represented probabilistic database is separately considered in Section 3.5.

Recall from Definition 13, an incomplete database corresponds to a set of possible worlds without probabilities. Thus, for each of the discussed representation systems an incomplete version can be defined accordingly by simply omitting probabilities. In the rest of this thesis, we will denote the incomplete versions of these systems by the same name as its probabilistic equivalent.

Before going into representation details, we shortly need to discuss the meaning of key attributes in probabilistic representation systems. In probabilistic databases, we have to distinguish between the world schema and the schema of the probabilistic representation and therefore have to distinguish between the conventional primary key of a database table and the primary key of the table's probabilistic representation. Whereas the first is only unique in every possible world, i.e. no two tuples that belong to the same world have the same value in this key attribute, the latter is unique for all possible tuples of this table. For a better differentiation we call the first as *world (primary) key* (short *WK*) and call the second as *representation (primary) key* (short *RK*). Whether or not a table's world key can be used as its representation key, i.e. uniqueness in worlds implicates uniqueness in the representation, depends on



Person

	WK	name	age	p
t_1	p1	J.Doe	27	0.8
t_2	p2	K.Smith	32	1.0
t_3	p3	J.Ho	28	0.4

(a) TI-database

W_1, Pr=0.12

	WK	name	age
t_2	p2	K.Smith	32

W_2, Pr=0.48

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32

W_3, Pr=0.08

	WK	name	age
t_2	p2	K.Smith	32
t_3	p3	J.Ho	28

W_4, Pr=0.32

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Ho	28

(b) Possible worlds representation

Figure 3.5.: Sample TI-database along with its possible worlds representation

the individual representation systems. Note, if the representation key does not correspond to the world key, the representation key is a system-specific attribute that is required to model uncertainty within the database and hence does not belong to the schema of the possible worlds. We consider world keys to be always certain, i.e. all tuples of one table that represent the same database entity share the same world key. If this certainty is not given, we simply assume that an additional surrogate key is considered as world key instead. In the rest of this thesis, we mark world keys by a single underline (according to the usual practice of primary keys) and mark representation keys by a double underline. Of course, if its world key corresponds to its representation key, a database table does not contain a single underlined attribute and we double underline the world key instead. As in certain databases each tuple is annotated by a database unique tuple identifier (see Section 2.1.2) that enables an identification across tables.

3.3.1. Tuple-Independent Probabilistic Databases

The simplest way to model uncertainty in the relational data model is to assign each database tuple t with its marginal probability $p(t) \in]0, 1]$ [FR97]. By doing so, each tuple is considered as an independent probabilistic event (boolean random variable) that belongs to the extension of its corresponding database table with its given probability. This representation system only uses the concept of tuple-level uncertainty and is commonly known as *tuple-independent database* [SORK11] (short *TI-database*), but is also denoted as *Probabilistic-? Table* [Gre09]. Since the tuples of a TI-database are mutually independent, each two tuples belong to at least one same world. As a consequence, the world key of a table is automatically unique for all possible tuples of this table and we actually do not need to distinguish between the world key and the representation key.

Recall, each tuple with a marginal probability lower than one is called a *maybe-tuple* and each tuple with a probability of exact one is called a *certain tuple*. Due to the certain-tuples are certain events, the set of possible worlds of a TI-database is implicated by the presence and absence of the maybe-tuples. Let *pdb* be a TI-database, let $pdb^!$ be the set of all certain-tuples of *pdb*, and let $pdb^?$ be the set of all

Person

	WK	name		age	
t_1	p1	J.Doe	:1.0	27	:0.8
				28	:0.2
t_2	p2	K.Smith	:1.0	32	:1.0
t_3	p3	J.Doe	:0.7	28	:0.5
		J.Ho	:0.3	29	:0.5

(a) AOR-database

W_1, Pr=0.28

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Doe	28

W_2, Pr=0.28

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Doe	29

W_3, Pr=0.12

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Ho	28

W_4, Pr=0.12

	DEI	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Ho	29

W_5, Pr=0.07

	WK	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Doe	28

W_6, Pr=0.07

	WK	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Doe	29

W_7, Pr=0.03

	WK	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Ho	28

W_8, Pr=0.03

	DEI	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Ho	29

(b) Possible worlds representation

Figure 3.6.: Sample AOR-database along with its possible worlds representation

maybe-tuples of *pdb*, the possible world space of *pdb* is defined as:

$$\mathbf{W} = pws(pdb) = \{pdb^! \cup S \mid S \subseteq pdb^?\} \tag{3.1}$$

The probability of a possible world is computed as the product of the probabilities of all tuples that belong to this world multiplied with the inverse probabilities of all tuples that do not belong to this world, i.e. the probability of world $W \in \mathbf{W}$ is defined as $Pr(W) = \prod_{t \in W} p(t) \times \prod_{t \in pdb, t \notin W} (1 - p(t))$.

Since the space contains one possible world per possible subset of all maybe-tuples, the number of possible worlds that is represented by a TI-database *pdb* is $|\mathbf{W}| = 2^{|pdb^?|}$.

Example 22 *Figure 3.5(a) presents a TI-database with a single table and three tuples. Whereas the tuples t_1 and t_3 are maybe, tuple t_2 is not. Consequently, this database represents the four possible worlds W_1 to W_4 that are depicted in Figure 3.5(b).*

The most probable world of a TI-database can be simply computed by removing all tuples that have a marginal probability lower than 0.5 (note, if at least one tuple has a probability of exact 0.5, several most probable worlds exist).

Note, in the incomplete version of TI-databases, maybe-tuples are distinguished from certain-tuples by using a boolean attribute and are graphically marked by an annotated '?' symbol.

3.3.2. Attribute-OR Probabilistic Databases

Another simple representation system is the concept of attribute-OR databases (short *AOR-databases*). AOR-databases become popular by Barbara et al. [BGMP90, BGMP92] and are also denoted as *Probabilistic or-set-tables* [Gre09] (or-set tables without probabilities are also known from [INV91, LW93]). In contrast to TI-databases these databases consider all database tuples as certain events (no tuple-level uncertainty), but consider all values in non-key attributes as mutually independent random variables, i.e. each tuple can have several alternative values per non-key attribute where each of these values is ascribed with a probability that prescribes the likelihood that this value is the 'true' value (attribute-level uncertainty). Tuples in AOR-databases can be therefore considered as sets of random variables and are also denoted as A-tuples [SBH+09]. Due to the mutual independence of the attribute values, the database cannot capture correlations between the attribute values of different A-tuples and cannot even capture correlations between the attribute values of the same A-tuple. Since every tuple is present in every possible world, the world key can be used as representation key.

A possible world is constructed from an AOR-database by selecting for each A-tuple one of its alternative values per attribute. Thus, let *pdb* be an AOR-database with a single table T, let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be the attributes of table T, and let $Prob(t[A] = a)$ be the probability that A-tuple t has the alternative value a in attribute A. The set of possible instances of an A-tuple $t \in T$ is defined as:

$$pws(t) = \{(a_1, \ldots, a_m) \mid \forall i \in \{1, \ldots, m\} \colon Prob(t[A_i] = a_i) > 0\} \tag{3.2}$$

where each possible instance $t^{\langle * \rangle} \in pws(t)$ has the probability $p(t^{\langle * \rangle}) = \prod_{i=1}^{m} Pr(t[A_i] = t^{\langle * \rangle}[A_i])$.

Since all A-tuples are certain and because the possible instances of the individual A-tuples are mutually independent, the possible world space of an AOR-database *pdb* corresponds to the set of all minimal hitting sets[3] for the collection $C = \{pws(t) \mid t \in pdb\}$. The probability of each world $W \in \mathbf{W}$ is computed as $Pr(W) = \prod_{t^{\langle * \rangle} \in W} p(t^{\langle * \rangle})$.

Example 23 *Figure 3.6(a) presents an AOR-database with a single table and three A-tuples. Whereas the A-tuples t_1 and t_3 have probabilistic attribute values, e.g. the age of A-tuple t_1 is either 27 (probability 0.8) or 28 (probability 0.2), A-tuple t_2 has only certain ones. The corresponding possible world space is presented in Figure 3.6(b) and consists of eight worlds.*

The most probable world of an AOR-database can be simply computed by selecting the most probable alternative value per attribute for each A-tuple. Obviously, if at least one A-tuple has several most probable alternative values in at least one attribute, more than one most probable world exist.

3.3.3. Attribute-OR Probabilistic Databases with Maybe-Tuples

A more complex representation system that we already have used in Chapter 1 to present the concept of probabilistic databases are attribute-OR databases with maybe-tuples (short *AOR?-databases*) that are also denoted as *p-or-set-?-tables* [Gre09]. AOR?-databases enable a modeling of uncertainty on table

[3] According to Garey and Johnson [GJ79], a set H is a hitting set for a collection of sets C if $H \subseteq \bigcup C$ and $H \cap S \neq \emptyset$ for each $S \in C$ and is a minimal hitting set for C if no strict subset of H is a hitting set for C, i.e. $|H \cap S| = 1$ for each $S \in C$.

Person

	WK	name		age		p
t_1	p1	J.Doe	:1.0	27 28	:0.8 :0.2	0.8
t_2	p2	K.Smith	:1.0	32	:1.0	1.0
t_3	p3	J.Doe J.Ho	:0.7 :0.3	28 29	:0.5 :0.5	1.0

(a) AOR?-database

W_1, Pr=0.224

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Doe	28

W_2, Pr=0.224

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Doe	29

W_3, Pr=0.096

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Ho	28

W_4, Pr=0.096

	WK	name	age
t_1	p1	J.Doe	27
t_2	p2	K.Smith	32
t_3	p3	J.Ho	29

W_5, Pr=0.056

	WK	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Doe	28

W_6, Pr=0.056

	WK	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Doe	29

W_7, Pr=0.024

	WK	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Ho	28

W_8, Pr=0.024

	WK	name	age
t_1	p1	J.Doe	28
t_2	p2	K.Smith	32
t_3	p3	J.Ho	29

W_9, Pr=0.07

	WK	name	age
t_2	p2	K.Smith	32
t_3	p3	J.Doe	28

W_{10}, Pr=0.07

	WK	name	age
t_2	p2	K.Smith	32
t_3	p3	J.Doe	29

W_{11}, Pr=0.03

	WK	name	age
t_2	p2	K.Smith	32
t_3	p3	J.Ho	28

W_{12}, Pr=0.03

	WK	name	age
t_2	p2	K.Smith	32
t_3	p3	J.Ho	29

(b) Possible worlds representation

Figure 3.7.: Sample AOR?-database along with its possible worlds representation

membership as well as uncertainty on attribute values by combining the ideas of TI-databases as well as AOR-databases and therefore by using tuple-level uncertainty and attribute-level uncertainty. As an AOR-database an AOR?-database is a set of A-tuples. However, in contrast to AOR-databases these A-tuples are not longer considered as certain events, but instead are considered as mutual independent events that are associated with its marginal probability as we know it from TI-databases. Note, due to the mutual independence of the individual A-tuples, the world key can be used as representation key.

In contrast to AOR-databases, the probability $Prob(t[A] = a)$ only denotes the likelihood that A-tuple t has the value a in attribute A if t exists. Thus, this probability is not absolute anymore, but is conditioned by the existence of the corresponding A-tuple. As a consequence, the probability that an A-tuple t with the marginal probability $p(t)$ is represented in a possible world of a probabilistic database *pdb* by its possible instance $t^{\langle * \rangle} \in pws(t)$ results in $Prob(t^{\langle * \rangle} \in pdb) = p(t) \times p(t^{\langle * \rangle})$.

Therefore, let $pdb^!$ be the set of all certain A-tuples and let $pdb^?$ be the set of all maybe A-tuples of an AOR?-database *pdb*. The possible world space of *pdb* corresponds to the union of all minimal hitting

sets for one of the collections

$$C \in \{\bigcup_{t \in pdb^!} \{pws(t)\} \cup S \mid S \subseteq \bigcup_{t' \in pdb^?} \{pws(t')\}\} \tag{3.3}$$

The probability of each world $W \in \mathbf{W}$ is computed as

$$Pr(W) = \prod_{t^{\langle * \rangle} \in W} Prob(t^{\langle * \rangle} \in pdb) \times \prod_{t \in pdb^?, pws(t) \cap W = \emptyset} (1 - p(t)) \tag{3.4}$$

Example 24 *Figure 3.7(a) presents an AOR?-database with a single table and three A-tuples. The only difference to the AOR-database from Figure 3.6(a) is that the first A-tuple is maybe. As a consequence, instead of eight worlds the corresponding world space consists of twelve worlds. These worlds are presented in Figure 3.7(b).*

The most probable world of an AOR?-database can be simply computed by selecting the most probable possible instance per certain A-tuple and by selecting the most probable state (most probable possible instance or no instance) per maybe A-tuple. Of course, if at least one A-tuple has several most probable possible instances (or states respectively), more than one most probable world exist.

An AOR?-database can be specialized to a TI-database by constraining that each A-tuple has a single possible instance and can be specialized to an AOR-database by constraining that each A-tuple is certain. Note, as in TI-databases, in the incomplete version of AOR?-databases, maybe A-tuples are annotated with a question mark.

3.3.4. Block-Independent-Disjoint Probabilistic Databases

A more powerful representation system is the class of the so-called *Block-Independent-Disjoint Probabilistic Databases* [DS96, DRS09, SORK11] that are also shortly denoted as *BID-databases* or even *BID-tables* if they contain a single table. Another name that has been used for such databases is *tuple-disjoint independent databases* [DK10a].

Although BID-databases only use tuple-level uncertainty they extent AOR?-databases to a modeling of correlations between the attribute values of one A-tuple. Such correlations are incorporated into the database by utilizing so-called tuple-ORs. A tuple-OR is a block of ordinary tuples (not A-tuples!) that each is associated with its marginal probability. All tuples of one block are considered to be mutually exclusive probabilistic events and tuples from different blocks are considered to be mutually independent probabilistic events. Since the tuples of one block are mutually exclusive, their sum in probability must be lower than or equal to one. A block is called *maybe* if the accumulative probability of its tuples is lower than one and is called *certain* otherwise.

Example 25 *Figure 3.8(a) presents a BID-database with a single table and five tuples that are grouped into three blocks (attribute* 'BNo.'*). For a better graphical representation, tuples that belong to the same block do not only share the same value in the attribute* 'BNo.'*, but are also not separated by horizontal lines. Whereas the probabilities of the tuples of the last two blocks each sum up to one, the first block is maybe. The corresponding possible world space is presented in Figure 3.8(b) and consists of six worlds (note that person p_1 is missing in the worlds W_3 and W_6 due to its corresponding block is maybe).*

Person

	RK	BNo.	WK	name	age	p
t_1	1	1	p1	J.Doe	27	0.6
t_2	2	1	p1	J.Doe	28	0.2
t_3	3	2	p2	K.Smith	32	1.0
t_4	4	3	p3	J.Doe	28	0.8
t_5	5	3	p3	J.Ho	29	0.2

(a) BID-database

W_1, Pr=0.48

	WK	name	age
t_1	p1	J.Doe	27
t_3	p2	K.Smith	32
t_4	p3	J.Doe	28

W_2, Pr=0.16

	WK	name	age
t_2	p1	J.Doe	28
t_3	p2	K.Smith	32
t_4	p3	J.Doe	28

W_3, Pr=0.16

	WK	name	age
t_3	p2	K.Smith	32
t_4	p3	J.Doe	28

W_4, Pr=0.12

	WK	name	age
t_1	p1	J.Doe	27
t_3	p2	K.Smith	32
t_5	p3	J.Ho	29

W_5, Pr=0.04

	WK	name	age
t_2	p1	J.Doe	28
t_3	p2	K.Smith	32
t_5	p3	J.Ho	29

W_6, Pr=0.04

	WK	name	age
t_3	p2	K.Smith	32
t_5	p3	J.Ho	29

(b) Possible worlds representation

Figure 3.8.: Sample BID-database along with its possible worlds representation

In contrast to the representation systems that we have discussed in the previous sections, tuples in a BID-table can share the same world key value. Since they need to be mutual exclusive, tuples with same world key value have to belong to the same block. Although, not all tuple of one block need to have the same world key value, we usually restrict blocks to single world key values in our examples and therefore typically consider the block number (attribute *'BNo.'*) and the world key value to represented by the same attribute.

Several proposals on BID-databases do not use a specific representation key attribute to identify tuples within a BID-table, but consider the block number as world key, additionally introduce a numerical attribute that enumerates all tuples per block, and consider the combination of both attributes as representation key. Nevertheless, this approach can lead to larger schemas if we define a foreign key of one table as a reference to the representation key of another table. For that reason, we decide to use an extra attribute as representation key instead. However, the notation $t_{i,j}$ that is commonly used to refer to the j-th tuple of the i-th block can be useful in some situations. Thus, we sometimes adopt this notation and consider that $t_{i,j}$ refers to the tuple that has the j-th smallest representation key value among all tuples from block i.

An AOR?-database can be transformed into a BID-database by constructing one block per A-tuple and by filling this block with all the A-tuple's possible instances, i.e. for each A-tuple t of the AOR?-database we construct a block B that contains the ordinary tuples $pws(t)$. It is obvious that compared to AOR?-databases the improved modeling power of BID-databases comes to the price of compactness, because a

block corresponds to a plain representation of all the A-tuples' possible instances. For example, an A-tuple with 5 attributes and 3 alternative values per attribute (in overall 15 values) needs to be represented by a block with $3^5 = 243$ ordinary tuples that each has 5 attribute values (in overall 1215 values). For that reason, BID-databases are internally often stored similar to AOR?-databases by dividing the set of all attributes into independent subsets [BSHW06].

A possible world is constructed by selecting one tuple per certain block and by selecting one or no tuple per maybe block. Thus, let *pdb* be a BID-database, let $\mathcal{B}^!$ be the set of all certain blocks of *pdb*, and let $\mathcal{B}^?$ be the set of all maybe blocks of *pdb* where each block B has the probability $p(B) = \sum_{t \in B} p(t)$, the possible world space of *pdb* is defined as the union of all minimal hitting sets for one of the collections $C \in \{\mathcal{B}^! \cup S \mid S \subseteq \mathcal{B}^?\}$. The probability of each world $W \in \mathbf{W}$ is computed as $Pr(W) = \prod_{t \in W} p(t) \times \prod_{B \in \mathcal{B}^?, B \cap W = \emptyset} (1 - p(B))$.

The most probable world of a BID-database can be simply computed by selecting the most probable tuple per certain block and by selecting the most probable state (most probable tuple or no tuple) per maybe block. Of course, if at least one block has several most probable tuples (or states respectively), more than one most probable world exist.

BID-databases are closely related to the ULDB model that has been proposed by Benjelloun et al. [BSHW06] (see next section). Lakshmanan et al. introduce the concept of FP-Relations [LLRS97]. FP-Relations are generalizations of BID-tables that use interval probabilities instead of point probabilities.

As for the previously presented representation systems that can model uncertainty on tuple membership, in the incomplete version of BID-databases maybe-tuples are annotated with a question mark.

3.3.5. Probabilistic Conditional Databases

A powerful representation system for incomplete databases is the concept of *conditional tables* (short *c-tables*) [IL84, LSV02, Gra09] that has been originally introduced by Imielinski and Lipski [IL84] in the eighties. In c-tables, each tuple is associated with a boolean condition[4] over discrete random variables (tuple-level uncertainty) where the domain of each variable theoretically can be infinite. If a tuple is associated with a tautology (e.g. the boolean constant *true*), this tuple belongs to every possible world of the incomplete database and we typically omit its condition in our examples.

Theoretically, the tuples' conditions can be restricted to disjunctions of atomic events without losing any modeling power [SORK11]. In that case, however, it can happen that we require one variable per possible world and therefore run into the same problem as in the possible worlds representation because the number of variables can be infeasible in size.

Since same random variables can appear in the conditions of several tuples, they can be used to incorporate correlations between these tuples. Moreover, random variables can be used as attribute values (attribute-level uncertainty) which can increase compactness further on. Nevertheless, since the representation system does not lose any modeling power if variables are restricted to the tuples' conditions [Koc09] and the use of variables in attribute values complicates data processing and data modeling, we will consider only c-tables with constant attribute values in this thesis.

[4] In c-tables, a boolean condition is considered as a logical expression that is built up by atomic conditions that are connected by the logical operators $\vee$, $\wedge$, and $\neg$. Atomic conditions are either the boolean values *true* and *false*, or are of the form $X \theta Y$ or $X \theta c$ where X and Y are variables, c is a constant, and $\theta \in \{=, <\}$ [LSV02].

Person

	RK	WK	name	age	condition
t_1	1	p1	J.Doe	27	X=1
t_2	2	p1	J.Doe	28	X=2
t_3	3	p2	K.Smith	32	Y=1
t_4	4	p2	S.Kmith	32	Y=2
t_5	5	p3	J.Doe	28	X=1 v X=3
t_6	6	p3	J.Ho	29	X=2

Variables

var	value	Prob
X	1	0.6
X	2	0.2
X	3	0.2
Y	1	0.8
Y	2	0.2

(a) Probabilistic conditional database

W_1, Pr=0.48

	WK	name	age
t_1	p1	J.Doe	27
t_3	p2	K.Smith	32
t_5	p3	J.Doe	28

W_2, Pr=0.16

	WK	name	age
t_2	p1	J.Doe	28
t_3	p2	K.Smith	32
t_6	p3	J.Ho	29

W_3, Pr=0.16

	WK	name	age
t_3	p2	K.Smith	32
t_5	p3	J.Doe	28

W_4, Pr=0.12

	WK	name	age
t_1	p1	J.Doe	27
t_4	p2	S.Kmith	32
t_5	p3	J.Doe	28

W_5, Pr=0.04

	WK	name	age
t_2	p1	J.Doe	28
t_4	p2	S.Kmith	32
t_6	p3	J.Ho	29

W_6, Pr=0.04

	WK	name	age
t_4	p2	S.Kmith	32
t_5	p3	J.Doe	28

(b) Possible worlds representation

Figure 3.9.: Sample probabilistic conditional database along with its possible worlds representation

Grahne [Gra84] proposes an extension of c-tables that is also able to capture constraints beyond single tables from which we abstract in this thesis. Well known restrictions of c-tables are v-tables and Codd-tables [IL84]. Both systems uses variables only as attribute values and do not annotate tuples by conditions. Whereas in v-tables same variables can be assigned to different attribute values and hence can be used to model correlations between these values, in Codd-tables the variables from different attribute values need to be distinct. As a consequence, v-tables can be considered as databases with labeled null values and Codd-tables can be considered as databases with unlabeled null values. Codd-tables roughly correspond to the currently standardized mechanism for using NULL values in relational database systems.

Probabilistic conditional tables [SORK11] (short *pc-tables*) are the probabilistic extensions of c-tables, i.e. each random variable is associated with a probability distribution. To avoid confusions between databases and single database tables, we will refer to databases with several tables where the tuples of all tables are defined on a single set of random variables as *pc-databases* in this thesis.

If probabilistic databases are restricted to finite sets of possible worlds, the possible values of the variables need to be finite, too. For that reason, the probability distribution of each random variable corresponds to a probability mass function. Let dom_X be the domain of the random variable X and let $Prob(X = x)$ be the probability that X corresponds to the value $x \in dom_X$, the set of possible values of X is defined as $\{x \mid x \in dom_X, Prob(X = x) > 0\}$.

Example 26 *Figure 3.9(a) presents a pc-table with six tuples. For ease of presentation, tuples that share the same world key value are not separated by horizontal lines. Each tuple is associated with a condition that contains one of the two variables X and Y. The probability distributions on these variables are stored in a separate table. This database represents the six possible worlds that are depicted in Figure 3.9(b). Note that this database contains complex correlations. For instance, the existence of tuple t_1 includes the existence of tuple t_5.*

An assignment θ is a function that maps each random variable to one of its possible values. The mapping of the random variable X to its possible value x by the assignment θ is therefore denoted as $\theta(X) = x$ and is sometimes also written as $X \mapsto x$. Let $\mathbf{X}$ be the set of all random variables that are used in the considered pc-database, due to these variables are mutually independent, the probability of an assignment θ is defined as:

$$Prob(\theta) = \prod\nolimits_{X \in \mathbf{X}} Prob(X = \theta(X)) \tag{3.5}$$

Example 27 *For instance, the assignment $\theta(X \mapsto 1, Y \mapsto 1)$ consists of the two atomic assignments $X \mapsto 1$ and $Y \mapsto 1$. Thus, its probability results in $Prob(\theta) = Prob(X = 1) \times Prob(Y = 1) = 0.48$.*

Theoretically, for each variable assignment another possible world can be constructed from the pc-database where the constructed world contains each tuple whose condition is satisfied by the selected assignment. Let *pdb* be a pc-database, let Φ_t be the boolean condition of tuple t, and let θ be the used assignment, the corresponding possible world results in:

$$W^{\theta}_{pdb} = \{t \mid t \in pdb, \Phi_t(\theta) = true\} \tag{3.6}$$

Consequently, let Θ be the set of all assignments that are possible for the given set of random variables $\mathbf{X}$, the possible world space of a pc-database *pdb* can be constructed as:

$$\mathbf{W} = pws(pdb) = \bigcup\nolimits_{\theta \in \Theta} \{W^{\theta}_{pdb}\} = \{W^{\theta}_{pdb} \mid \theta \in \Theta\} \tag{3.7}$$

The probability of a possible world results in the accumulative probability of all assignments that lead to this world. Consequently, the probability of world $W \in \mathbf{W}$ is computed as:

$$Pr(W) = \sum\nolimits_{\theta \in \Theta, W^{\theta}_{pdb} = W} Prob(\theta) \tag{3.8}$$

Example 28 *For instance, the possible world W_1 from Figure 3.9(b) results from using the assignment $\theta(X \mapsto 1, Y \mapsto 1)$ and therefore has the probability $Pr(W_1) = Prob(X = 1) \times Prob(Y = 1) = 0.48$.*

If each variable assignment leads to another possible world, the most probable world of a pc-database can be simply computed by selecting the most probable value per random variable and then by removing all tuples whose conditions are not satisfied by the selected assignment. Of course, if at least one random variable has several most probable values, more than one most probable world exist. If different variable assignments can lead to the same world, the computation process is much more complicated. In the worst

case it requires an enumerating of all possible worlds which is impractical already for small databases. In such cases, the most probable world can often only be approximated instead of exactly computed.

By definition, the marginal probability of a tuple corresponds to the accumulative probability of all possible worlds this tuple belongs to and hence results in the accumulative probabilities of all assignments that satisfy the tuple's condition. Therefore, the marginal probability of a tuple t with the condition Φ_t can be computed as:

$$p(t) = \sum_{\theta \in \Theta, \Phi_t(\theta)=true} Prob(\theta) \tag{3.9}$$

Example 29 *For demonstration, tuple t_4 from Figure 3.9(b) is present in the possible worlds W_4, W_5, and W_6. These worlds in turn result from the assignments $\theta_4(X \mapsto 1, Y \mapsto 2)$, $\theta_5(X \mapsto 2, Y \mapsto 2)$ and $\theta_6(X \mapsto 3, Y \mapsto 2)$. Therefore, the marginal probability of tuple t_4 results in:*

$$\begin{aligned} p(t_4) &= Prob(X{=}1) \times Prob(Y{=}2) + Prob(X{=}2) \times Prob(Y{=}2) + Prob(X{=}3) \times Prob(Y{=}2) \\ &= (Prob(X{=}1) + Prob(X{=}2) + Prob(X{=}3)) \times Prob(Y{=}2) \\ &= Prob(Y{=}2) \\ &= 0.2 \end{aligned}$$

Each BID-database can be transformed into a pc-database by introducing one variable per block and by setting the condition of each of the block's tuple to another assignment of the block's corresponding variable. In contrast, not every pc-database can be transformed into a BID-database. For instance, the possible world space presented in Figure 3.9(b) cannot be represented by a BID-database, because the inclusion of tuple t_5 from tuple t_1 cannot be modeled in this system. As a consequence, the concept of pc-databases is a more powerful representation system as the concept of BID-databases.

Two well known probabilistic database management systems are MayBMS [KO08, HAKO09, Koc09] and Trio [Wid09, ABS+06, MTdK+07]. Both systems use a representation system that is based on the concept of pc-databases.

- MayBMS uses **U-databases** as representation system. A U-database is a set of U-relations. As in pc-databases, in U-relations each tuple is associated with a condition on discrete variables. In U-relations the use of variables is restricted to the tuples' conditions and these conditions are restricted to conjunctions of k atomic events where each atomic event $X = x$ compares a random variable X with one of its possible values x. For storing conditions, an U-relation uses two extra columns **V** and **D** (*condition columns*) per atomic event (note, the restriction to k atomic events follows from the restriction to k pairs of condition columns). The column **V** contains the variable and the column **D** contains the variable's possible value of its corresponding atomic event, i.e. an atomic condition $X = x$ is stored by setting its corresponding condition columns to $\mathbf{V} = X$ and $\mathbf{D} = x$. Disjunctions of conjunctions are modeled by the use of several tuples with same attribute values, but different settings in the condition columns. The representation key of a U-relation is the combination of all condition columns and the world key. The possible values of the variables along with their probabilities are separately stored in a ternary relation that is called *World-Table*.

Person[name]

	V	D	RK	WK	name
t_1	X	1	1	p1	J.Doe
t_2	X	2	2	p1	J.Doe
t_3	Y	1	3	p2	K.Smith
t_4	Y	2	4	p2	S.Kmith
t_5	X	1	5	p3	J.Doe
t_6	X	3	6	p3	J.Doe
t_7	X	2	7	p3	J.Ho

Person[age]

	V	D	RK	WK	age
t_8	X	1	1	p1	27
t_9	X	2	2	p1	28
t_{10}	⊥	⊥	3	p2	32
t_{11}	X	1	4	p3	28
t_{12}	X	3	5	p3	28
t_{13}	X	2	6	p3	29

World-Table

V	D	Pr
X	1	0.6
X	2	0.2
X	3	0.2
Y	1	0.8
Y	2	0.2

(a) Sample U-Database

Person

	XID	AID	WK	name	age	lineage	p
t_1	1	1	p1	J.Doe	27	λ(1,1)=(4,1)	(0.6)
t_2	1	2	p1	J.Doe	28	λ(1,2)=(4,2)	(0.2)
t_3	2	1	p2	K.Smith	32	-	0.8
t_4	2	2	p2	S.Kmith	32	-	0.2
t_5	3	1	p3	J.Doe	28	λ(3,1)=(4,1) ∨ (4,3)	(0.8)
t_6	3	2	p3	J.Ho	29	λ(3,2)=(4,2)	(0.2)

Correlation Table

	XID	AID	value	p
t_7	4	1	1	0.6
t_8	4	2	2	0.2
t_9	4	3	3	0.2

(b) Sample ULDB

Figure 3.10.: Sample U-Database (MayBMS) and sample ULDB (Trio) both having the possible worlds representation depicted in Figure 3.9(b)

To increase compactness further on, U-databases use vertical partitioning to decompose tables into several probabilistic independent sets of attributes and then store each of these sets using an own table. This process is also denoted as normalization [SORK11].

Due to its architecture, U-databases are especially convenient to model the answers of *UCQ* queries (Unions of Conjunctive Queries) to other U-databases or BID databases [SORK11]. Since queries of this class can be evaluated efficiently in most cases, this property can be useful in many applications.

- The concept of **Uncertainty-Lineage Data Bases** (short ULDBs) [BSHW06] is the representation system that is used in the Trio system. A ULDB is a set of so-called x-relations that each consists of a set of x-tuples. The concept of x-tuples corresponds to the concepts of tuple-ORs and hence is a block of mutually exclusive ordinary tuples, which are simply called alternatives. Each x-tuple is identified by an x-tuple identifier (short XID) and each alternative of one x-tuple is identified by an alternative identifier (short AID). Thus, the XID serves as world key and the combination of the XID and the AID serves as representation key. Each x-tuple alternative can be annotated with a lineage formula (noted by λ) that is a boolean condition like the conditions that are used in pc-tables. Although these formulas have the primary purpose to model the lineage of query output alternatives to query input alternatives, i.e. a lineage formula models in which way a set of input alternatives have to be combined in order to compute the formula's corresponding

output alternative, they can be also used to model correlations between the alternatives of different x-tuples. For this purpose we need to construct a separate x-relation whose x-tuples are used as random variables. We will refer to this x-relation as *correlation table*. ULDBs distinguish between base-tuples, which are x-tuples whose alternatives do not have any lineage, and non-base-tuples, which are x-tuples whose alternatives have a lineage because these tuples have been derived from some base-tuples. For that reason, the lineage of each non-base tuple alternative directly or indirectly refers to a set of base-tuples it is derived from. Whereas the probabilities of the base-tuples' alternatives are directly associated to them, the probabilities of the non-base-tuples' alternatives are computed based on the lineage conditions and the probabilities of the referenced base-tuples. As we will see in Section 3.5, the latter corresponds to the mechanism of computing the probabilities of query output tuples in pc-databases. The first is a consequence from the fact that ULDB assigns AIDs to the x-tuple alternatives and hence inherently model the mutual exclusion between them without using random variables. This kind of inherent modeling, however, is only possible for base-tuples, i.e. the x-tuple's alternatives have an empty lineage and therefore are independent from the alternatives of all other x-tuples. Note, an x-relation without lineage conditions corresponds to a BID-table.

Example 30 *A sample U-database is presented in Figure 3.10(a). Due to normalization the database consists of two U-relations, i.e. one for the attribute* 'name' *and one for the attribute* 'age'. *In contrast, a sample ULDB is presented in Figure 3.10(b) (the probabilities of all non-base tuples are written in brackets because they are actually not stored in this table). The similarity between the world-table, the correlation table, and the variable map of the pc-database from Figure 3.9(a) cannot be denied. Moreover, the resemblance of the boolean conditions that are modeled by the condition columns (U-database) or that are modeled by the lineage formulas (ULDB) to the conditions that are modeled in the pc-database is obvious. Indeed, the possible worlds representations of both databases are exact the same as the one depicted in Figure 3.9(b).*

3.3.6. Properties of Representation Systems

Two important properties that describe the modeling power of probabilistic representation systems are *completeness* and *closeness*.

According to [SORK11], a representation system for probabilistic databases is called to be *complete*, if each possible worlds representation can be losslessly transformed into this system.

Definition 16 (Completeness): *Let $\mathfrak{R}$ be a representation system, let $\mathfrak{R}^{db}$ be the set of all possible representations of that system, $\mathfrak{R}$ is complete if it can represent any probabilistic database $(\boldsymbol{W}, Pr)$, i.e.* $\forall(\boldsymbol{W}, Pr)\colon \exists pdb_2 \in \mathfrak{R}^{db}\colon pwr(pdb_2) = (\boldsymbol{W}, Pr)$.

It has been proven that the concept of pc-databases is a complete representation system [SORK11]. The same holds for U-Databases [Koc09, SORK11] and ULDBs [BSHW06]. In contrast, TI-databases, AOR-databases, AOR?-databases, and BID-databases are no complete representation system as it can be simply illustrated by the world space that is depicted in Figure 3.9(b), because it cannot be represented by any of these systems. Although BID-databases are not complete, they are stronger than AOR?-databases

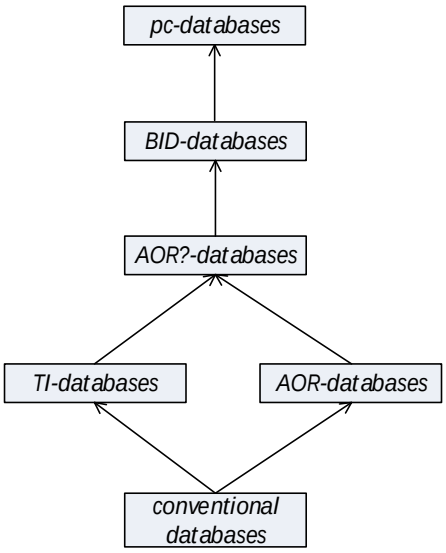


Figure 3.11.: The semi-ordering of the modeling power of the representation systems considered in this thesis

because the world space that is depicted in Figure 3.8(b) cannot be represented by the latter system. AOR?-databases are in turn stronger than TI-databases and AOR-databases which can be demonstrated by the world space that is depicted in Figure 3.7(b), because none of the latter two systems can be used to represent this space. The possible world space presented in Figure 3.5(b) cannot be modeled by an AOR-database and the possible world space presented in Figure 3.6(b) cannot be modeled by a TI-database. Thus, none of both systems has a greater modeling power than the other. In summary, by considering the modeling power of the representation systems that we have presented in this chapter, we get the semi-ordering that is presented in Figure 3.11.

According to Green [Gre09] a representation system for probabilistic databases is *closed* under a query language, if every probabilistic database that can result from posing a query of this language to a probabilistic database that is represented in this system can be represented in this system as well.

Definition 17 (Closeness): *Let $\mathfrak{R}$ be a representation system, let $\mathfrak{R}^{db}$ be the set of all possible representations of that system, and let $\mathcal{L}$ be a database query language. The system $\mathfrak{R}$ is closed under $\mathcal{L}$ if for any query $Q \in \mathcal{L}$ and any database $pdb_1 \in \mathfrak{R}^{db}$ there is a database $pdb_2 \in \mathfrak{R}^{db}$ that represents the result of posing Q to pdb_1, i.e. $pwr(pdb_2) = Q(pwr(pdb_1))$.*

It is obvious that completeness implicates closeness. Therefore, pc-databases are closed under the relational calculus. The other four systems are not closed under the join operator[5] as we can illustrate by a simple example.

[5] By considering the possible worlds semantics, AOR-databases are even not closed under the selection operator. For that reason, former research on AOR-databases (e.g. [BGMP92]) introduce a definition of the selection operator that selects complete A-tuples instead of alternative instances of A-tuples. By doing so, however, the operator is not conform to the possible worlds semantics as we have presented it in Section 3.2 because it retains worlds in which some tuples do not satisfy the condition of the applied selection operator.

SELECT t.name **AS** nameA, u.name **AS** nameB
FROM Person t, Person u
WHERE t.WK $\neq$ u.WK
AND t.age $<$ u.age

(a) Sample query Q_4

W_A, Pr=0.48

	nameA	nameB
t_6	J.Doe	K.Smith
t_7	J.Doe	J.Doe

W_B, Pr=0.3

	nameA	nameB
t_6	J.Doe	K.Smith
t_8	J.Ho	K.Smith
t_9	J.Doe	J.Ho

W_C, Pr=0.22

	nameA	nameB
t_6	J.Doe	K.Smith

(b) The possible worlds representation that results from applying query Q_4 to the BID-table *'Person'* from Figure 3.8(a)

Figure 3.12.: Sample for illustrating that BID-databases are not closed under the join operator

Example 31 *For the purpose of illustration we consider the database query Q_4 that is presented in Figure 3.12(a) and pose this query to the BID-database from Figure 3.8(a). This query computes the is-younger-than relationship between two persons by using a theta join and returns the name of the younger person followed by the name of the older person. Recall that the join needs to be processed in each possible world separately. Therefore, the resultant possible world space contains the three possible worlds presented in Figure 3.12(b). Note that world W_A results from querying world W_1, world W_B results from querying one of the worlds W_2 and W_5, and world W_C results from querying one of the worlds W_3, W_4, and W_6. Since* J.Doe *can only be younger than* J.Ho *if tuple t_5 belongs to the processed world and because the single tuple with WK $= p_2$ belongs to all of the processed worlds, the presence of tuple t_8 implies the presence of tuple t_9 and vice versa. These implications, however, cannot be represented within a BID-database. Consequently, BID-databases are not closed under the join operator which in turn implies that TI-databases, AOR-databases, and AOR?-databases are not closed under the join operator as well.*

3.3.7. Coupling Representation Systems with Views

Example 31 moreover demonstrates that queries can introduce correlations between tuples that are not present in the source data. At a first glance, this property looks like a great disadvantage because using a less powerful representation system automatically restricts the set of queries that is applicable to the represented database. On a second look, however, we will see that this property can be also useful, because it enables us to use a less powerful representation system even if we need to represent highly complex correlations.

In relational databases, views are used to recompose tables that have been decomposed into several small tables because of functional dependencies (normalization). As a consequence, queries are utilized to introduce correlations between the tuples of different tables. Obviously, we can apply the same principle to probabilistic data and hence can define views to introduce tuple correlations into probabilistic database by purpose. Therefore, instead of modeling all correlations into the database we can use a less

powerful representation system and then shift complexity from the database to a set of queries, i.e. the particularly defined views. Consequently, we can make use of the expressiveness of a query to reduce the complexity of the stored database.

Suciu et al. [SORK11] show that TI-databases and BID-databases become complete representation systems if we couple them with views. Whereas TI-databases require views using the complete expression spectrum of the relational calculus, BID-databases only require views that are conjunctive queries, i.e. queries that do not use negations or disjunctions. It is important to note that although these combinations are complete representation systems, it can happen that they require as many tuples as possible worlds and hence are sometimes impractical. Nevertheless, in many use cases they are suitable to represent a highly complex database with high compactness and a manageable representation/query complexity.

Example 32 *For illustration, we cannot represent the possible world space from Figure 3.12 in a BID-database, but we can represent it with a BID-database if we combine it with a view. For that purpose we simply retain the BID-database from Figure 3.8(a) and define a view that corresponds to query* Q_4.

3.4. Entity-based Interpretation of Probabilistic Data

As the conventional relational data model, the above presented representation systems are not inherently entity-based. As introduced in Section 2.1, we consider each database entity to be identified by its *DEI*. Since an entity can only belong to every table once, all tuples with the same *DEI* need to be mutually exclusive, i.e. in every possible world only one of them is allowed to be present. Thus, the *DEI* corresponds to the world key of each table and we therefore assume in the rest of this thesis that each table of a probabilistic database uses the *DEI* as world key (recall in certain databases, we assume that each table uses the *DEI* as primary key).

In general, in an entity-based interpretation, we can distinguish between *membership uncertainty* and *value uncertainty*. Whereas the first models the uncertainty whether or not an entity is represented in a particular entity table (and hence whether or not it belongs to the table's corresponding extension), the second models uncertainty on the entity's attribute values. Membership uncertainty and value uncertainty should not be confused with tuple-level uncertainty and attribute-level uncertainty. The first describe the uncertainty on some properties of a database entity and only depend on the considered world schema, but does not depend on the used representation system. In contrast, tuple-level uncertainty and attribute-level uncertainty are modeling concepts and whether they are used or not depend on the used representation system, but does not depend on the considered world schema. Because the membership of an entity to a specific collection can be described by a membership attribute and because an attribute can be represented by the memberships to a set of collections (recall Section 2.1.1) membership uncertainty can be transformed into value uncertainty and vice versa by changing the database schema. Membership uncertainty can be stored as value uncertainty by adding an extra boolean attribute to the schema of the considered table. The value of this attribute is 'true' if the tuple (and hence the corresponding entity) belongs to the table and is 'false' otherwise. Note, if several tables have mutual exclusive memberships a categorical attribute can be used instead of a boolean attribute. In the opposite direction, the uncertainty

on the value of a tuple (database entity) in a specific attribute can be transformed into membership uncertainty by creating a separate table per element of the attribute's corresponding domain.

Example 33 *For illustration, value uncertainty on the gender of a person can be transformed into membership uncertainty by splitting the table* 'Person' *into the two tables* 'Man' *and* 'Woman'. *On the contrary, the uncertainty of the membership to the tables* 'Man' *and* 'Woman' *can be transformed into value uncertainty by combining both tables to the table* 'Person' *and by adding an attribute* 'gender' *that has the two domain elements 'man' and 'woman' to the table's schema.*

In Section 2.1, we denote the set of tuples that represent a database entity e in a database *db* as the entity's instance $\mathit{Inst}(e, \mathit{db})$. It is obvious that in a probabilistic database an entity can have several possible instances. The probability of an instance results in the accumulative probability of all the database's possible worlds that contain this instance.

Definition 18 (Possible Entity Instance): *Let $(\boldsymbol{W}, Pr)$ be the possible worlds representation of a probabilistic database pdb and let $\mathfrak{I}_{pdb}$ be the space of all entity instances that are conform to the given database schema. The probability that an entity e has the instance $I \in \mathfrak{I}_{pdb}$ results in:*

$$Prob(\mathit{Inst}(e, pdb) = I) = \sum\nolimits_{W \in \boldsymbol{W}, \mathit{Inst}(e,W)=I} Pr(W)$$

The set of possible instances of e in pdb is therefore defined as $\mathit{Inst}_{Poss}(e, pdb) = \{I \mid I \in \mathfrak{I}, Pr(\mathit{Inst}(e, pdb) = I) > 0\}$.

The extension of a probabilistic database (or table of a probabilistic database respectively) can be uncertain, if the underlying system is able to model membership uncertainty. Whereas the set of possible database entities of a probabilistic database is defined as the set of all database entities that belong to the extension of at least one of the database's possible world, the probability that an entity belong to the 'true' extension of the probabilistic database results in the accumulative probability of all the database's worlds whose extensions contain this entity.

Definition 19 (Probabilistic Database Extension): *Let $(\boldsymbol{W}, Pr)$ be the possible worlds representation of a probabilistic database pdb. The 'true' extension of pdb is represented by the probability space $\mathit{Ext}(pdb) = (\boldsymbol{W}_{Ext}, Pr_{Ext})$ where*

$$\boldsymbol{W}_{Ext} = \{\mathit{Ext}(W) \mid W \in \boldsymbol{W}\}$$
$$\textit{and} \qquad \forall \mathfrak{E} \in \boldsymbol{W}_{Ext} \colon Pr_{Ext}(\mathfrak{E}) = \sum\nolimits_{W \in \boldsymbol{W}, \mathit{Ext}(W)=\mathfrak{E}} Pr(W)$$

The probability that a database entity e belongs to the 'true' extension of pdb is therefore defined as:

$$Prob(e \in \mathit{Ext}(pdb)) = \sum\nolimits_{\mathfrak{E} \in \boldsymbol{W}_{Ext}, e \in \mathfrak{E}} Pr_{Ext}(\mathfrak{E}) = \sum\nolimits_{W \in \boldsymbol{W}, e \in \mathit{Ext}(W)} Pr(W)$$

The probability that a database entity belong to the extension of a single database table is defined accordingly.

The possible extension of pdb, i.e. the set of database entities that possibly belong to the 'true' extension of pdb, is denoted as $\mathit{Ext}_{Poss}(pdb)$ and is defined as the set of all database entities that belong to the extension of at least one world in $\boldsymbol{W}$, i.e.:

$$\mathit{Ext}_{Poss}(pdb) = \bigcup\nolimits_{W \in \boldsymbol{W}} \mathit{Ext}(W) = \{e \mid Prob(e \in \mathit{Ext}(pdb)) > 0\}$$

The 'true' extension and the set of possible database entities of an incomplete database are defined accordingly.

Due to the *DEI* must be unique per table in every possible world, we make the following conventions for the individual representation systems:

- **TI-databases:** In tuple-independent databases we cannot model exclusion between tuples. As a consequence, all tuples of one table must represent another database entity. This in turn implicates that this representation system is only able to model the uncertain membership of a database entity to a database table, but is not able to model uncertainty on any of the entity's attribute values.

- **AOR-databases:** In attribute-OR databases, all A-tuples are present in every world. Therefore, all A-tuples of one table need to represent another database entity and the table membership of all entities is certain. However, since each A-tuple can have several alternative values per attribute, this system is able to model value uncertainty.

- **AOR?-databases:** In attribute-OR databases with maybe-tuples, A-tuples are independent to each other. For that reason, all A-tuples of one table need to represent another database entity. In contrast, to TI-databases and AOR-databases, however, an AOR?-database is able to model uncertainty on table membership as well as uncertainty on the entity's values. Let T be a table of an AOR?-database and let t be an A-tuple that represents a database entity e in T, the probability that e belongs to the extension of T results in the probability of t, i.e. $Prob(e \in \mathit{Ext}(T)) = p(t)$.

- **BID-databases:** In block-independent-disjoint databases, the tuples of one block are mutually exclusive. Thus, all tuples of one block can represent the same entity, and BID-databases are able to model uncertainty on table membership (blocks can be maybe) and uncertainty on the entity's values (the tuples of one block disagree in some attributes). Let B be the block of tuples that represent entity e in table T, the probability that e belongs to the extension of T results in the probability of B, i.e. $Prob(e \in \mathit{Ext}(T)) = p(B) = \sum_{t \in B} p(t)$.

  Of course, a block can be also used to represent a mutual exclusion between different entities. In this thesis, however, we consider each block to represent a single database entity and instead use a pc-database if mutual exclusions between tuples from different entities are required.

- **PC-databases:** In probabilistic conditional databases, several tuples of the same table can represent the same database entity, but their conditions need to be exclusive. Thus, let $\varkappa(\Phi)$ be the set of all variable assignments that satisfy the condition Φ, a pc-database is only consistent if for each tuple set $\{t_1, \ldots, t_k\}$ that represent the same database entity in one table, the sets $\{\varkappa(\Phi_{t_1}), \ldots, \varkappa(\Phi_{t_k})\}$ are pairwise disjoint. This condition cannot be ensured by the representation system itself, but need to be considered by the user or need to be considered by the application that is used for manipulating the stored data (insert, update, delete). Nonetheless, database constraints can be used to prevent data changes that lead to inconsistent results. In this thesis, we always assume that the considered pc-databases are consistent.

  Let $\{t_1, \ldots, t_k\}$ be the tuples of a pc-table T that represent entity e. Since the satisfying assignments of all these tuples need to be disjoint, the probability that e belongs to the extension of T results in $Prob(e \in \mathit{Ext}(T)) = \sum_{i=1}^{k} \sum_{\theta \in \varkappa(\Phi_{t_i})} Prob(\theta) = \sum_{\theta \in \bigcup_{i=1}^{k} \varkappa(\Phi_{t_i})} Prob(\theta)$.

Due to no correlations between the tuples of different entities can be modeled, we sometimes refer to TI-databases, AOR-databases, AOR?-databases, and BID-databases as entity-independent probabilistic databases.

In general, an entity can be represented by a tuple in several entity tables. In the aforementioned entity-independent probabilistic representation systems, however, tuples from different tables are mutually independent. Thus, we cannot even model correlations between the attribute values of the same entity if the corresponding attributes belong to different tables and we cannot model correlations between the memberships of one entity to the individual tables. From the latter follows that we cannot model a membership exclusion between different entity tables as for example an exclusion between the two tables *'Student'* and *'Professor'*. As we will demonstrate in Section 3.6.2, this property has a strong influence on the strategy by which the database schema need to be designed if the underlying ER-schema contains some inheritance hierarchies.

In ULDB, each x-tuple is usually considered to represent an own database entity. To simplify the notion of BID-databases and pc-databases, we adopt the concept of x-tuples and x-tuple alternative to these two systems and therefore sometimes call all tuples of the same table (BID-table or pc-table) that share the same *DEI* as alternatives of the same x-tuple.

Moreover, we adapt the relational concept to store variable assignments that is used by U-databases and denote the database table that stores all variable assignments in a pc-database as World-Table.

3.5. Querying Probabilistic Databases

Querying a probabilistic database has two main challenges. The first challenge is to compute the query answers efficiently even if the database is large and the query is complex. The second challenge is to present the query answers to the user in a simple but informative way. For presentation reasons we start with the second challenge and then continue with a discussion on the first.

3.5.1. Presentation of Query Answers in Probabilistic Databases

Following the possible worlds semantics, from querying a probabilistic database another probabilistic database results. Since each of the resultant worlds serves as a possible set of query answers, this semantics is also denoted as *possible answer sets semantics* [SORK11]. The advantage of the possible worlds semantics is that it is compositional, i.e. the result of one query can be used as input to another query. This compositional property is important because it enables the use of views. The disadvantage of this semantics is that due to its possibly large number of worlds, a query result usually cannot be presented to the user in the possible worlds representation, but need to be presented in a more compact representation instead. However, even if the query result is presented to the user in its succinct representation such as a pc-table, it can be difficult to understand (especially for unfamiliar user).

For that reason, from a usability perspective it is more practical to present the user a list of all result tuples, i.e. all possible query answers, along with their probabilities regardless to which worlds these tuples belong and hence by ignoring any kind of tuple dependency in the final presentation. Obviously, by doing so the query result does not correspond to an accurate probabilistic database anymore and hence

cannot be used as input to another query. For that reason, this query semantics is not compositional and should be only used to evaluate queries whose answers are directly passed to the user.

The idea behind this query semantics is that the user gets a simple presentation of all possibly answer tuples and if she is interested in a specific dependency she has to request this dependency by another query. For instance, if the user wants to know the probability that two query answers belong to the same possible world she can pose a second query that explicitly ask for tuple pairs and hence the result of this query will reflect the correlation between both tuples. This query semantics is usually called as *Possible Answers Semantics* [SORK11] and is defined as:

Definition 20 (Possible Answers Semantics): *Let $pdb = (\boldsymbol{W}, Pr)$ a probabilistic database and let Q be a database query, the result of posing Q to pdb_1 in the possible answers semantics is a set of tuple-probability pairs $Q_{poss}(pdb) = \{(t_1, p(t_1)), \ldots, (t_k, p(t_k))\}$ where t is a tuple and $p(t)$ is its marginal probability so that for each pair $(t_i, p(t_i)) \in Q_{poss}(pdb)$ holds:.*

$$t_i \in \{t \mid \exists W \in \boldsymbol{W} \colon t \in Q(W)\}$$
$$\textit{and} \qquad p(t_i) = \sum\nolimits_{W \in \boldsymbol{W}, t_i \in Q(W)} Pr(W)$$

Each tuple $t \in Q_{poss}(pdb)$ is called a possible answer of query Q to database *pdb*. Moreover, each tuple that belongs to the set $Q_{cert}(pdb) = \{t \mid \forall W \in \mathbf{W} \colon t \in Q(W)\} = \{t \mid t \in Q_{poss}(pdb), p(t) = 1\} \subseteq Q_{poss}(pdb)$ is called a certain answer of query Q to database *pdb*.

To increase the quality of presentation further on, the possible query answers are usually returned in the decreased order of their marginal probabilities. Moreover, often only the k most probable answers are required. In this case, the computation complexity of query evaluation can be considerably reduced. The challenge of efficient *top-k query answering* has been studied in several works [RDS07, SIC07, GZM09, LCH13a, SLG13].

Note, if the set of all possible query answers Q_{poss} is presented in a table-based fashion its representation corresponds to a table of a TI-database, but its semantics is different, because from such a set of tuple-probability pairs we cannot derive a possible worlds representation as we can derive it from a TI-database.

3.5.2. Query Evaluation in Probabilistic Databases

The possible worlds semantics is defined on the possible worlds representation and hence only gives a reference for query evaluation instead of a concrete solution for any compact representation system. A tuple is a possible query answer if it results from the considered query in any of the probabilistic database's possible worlds. Moreover, each world consists of a subset of all tuples of a probabilistic database. Consequently, a tuple can only be a possible query answer if there are some tuples in the probabilistic database that produce this tuple as query output. Note that this condition is necessary, but not sufficient because it can happen that the corresponding input tuples do not coexist in any of the possible worlds. Nonetheless, because of this condition we can compute all possible answers of a query that does not use aggregation by querying all the database's tuples collectively, i.e. by ignoring any of the underlying correlations. To compute the concrete set of possible query answers or to compute the concrete marginal probabilities of the output tuples, however, we need to take all tuple correlations into

IsYoungerThan

	nameA	nameB	lineage	p
t_6	J.Doe	K.Smith	$(\epsilon_1 \wedge \epsilon_3) \vee (\epsilon_2 \wedge \epsilon_3) \vee (\epsilon_4 \wedge \epsilon_3)$	1.0
t_7	J.Doe	J.Doe	$(\epsilon_1 \wedge \epsilon_4)$	0.48
t_8	J.Ho	K.Smith	$(\epsilon_3 \wedge \epsilon_5)$	0.3
t_9	J.Doe	J.Ho	$(\epsilon_1 \wedge \epsilon_5) \vee (\epsilon_2 \wedge \epsilon_5)$	0.3

Person (World Table)

	DEI	name	age	p
t_1	p1	J.Doe	27	0.8
t_2	p1	J.Doe	28	0.2
t_3	p2	K.Smith	32	1.0
t_4	p3	J.Doe	28	0.6
t_5	p3	J.Ho	29	0.3

Figure 3.13.: A pc-database having the possible worlds representation depicted in Figure 3.12

account (correlations modeled in the queried database as well as correlations introduced by the query). For that reason, query evaluation is based on the concept of lineage. As we have already introduced in Section 3.3.5, the lineage of an output tuple traces back from which input tuples the considered output tuple is derived from. Moreover, we have shown that lineage can be used to represent tuple correlations. Thus, lineage can be also used to represent tuple correlations that have been introduced by a query.

Example 34 *This is demonstrated by the pc-database presented in Figure 3.13. Notice that this pc-database exactly models the possible worlds representation from Figure 3.12 and recall that this query result cannot be represented by a BID-database without using views.*

Apparently, this pc-database looks different to this pc-database we have presented in Figure 3.9(a). The world table is substituted by the input database, i.e. the BID-table 'Person' *from Figure 3.8(a). Moreover, instead of comparisons of variables with values, the tuples' conditions are build on atomic events where each event ϵ_i refers to an input tuple t_i and is true iff t_i belongs to the input database's 'true' world. In other words, these events correspond to boolean random variables and the set of possible assignments to all these variables, i.e. events, is defined by the set of possible worlds of the input database, i.e. each possible world of the input database corresponds to another variable assignment. The marginal probabilities of all possible answers can be computed based on their lineage. For instance, tuple t_9 is a true query answer, if its lineage formula Φ_{t_9} is satisfied which in turn exactly corresponds to the cases where the tuples t_1 and t_5 or the tuples t_2 and t_5 belong to the input database. Since t_1 and t_2 are exclusive and both independent to t_5, the marginal probability of t_9 is computed as:* $p(t_9) = (p(t_1) + p(t_2)) \times p(t_5) = (0.8 + 0.2) \times 0.3 = 0.3$

As illustrated by this example, the problem of computing the marginal probability of a possible query answer can be reduced to the problem of computing the probability of a propositional formula. However, for computing the probability of such a formula, we need to detect all combinations of events (e.g. variable assignments in pc-databases) that satisfy this formula. This well known problem of probabilistic inference can be solved in several ways. One way is to use one of the traditional inference algorithms that have been developed in the research of artificial intelligence [Dar09]. Another way is to directly integrate inference into the query evaluation process.

- **Intensional Query Evaluation:** By using the intensional query semantics, query evaluation and probabilistic inference are considered as two separate processes. First a database engine evaluates

the query on the database and produces query answers along with their lineage formulas. Second a general-purpose algorithm for probabilistic inference as for example Monte-Carlo Simulation [KLM89] (approximate), Variable Elimination [Dec96] (exact) or the Junction Tree Algorithm [HD96] (exact) is used to compute the marginal probabilities of the possible query answers based on their lineage. Nevertheless, probabilistic inference is generally known to be a hard problem [Rot96] and even approximative approaches usually scale bad if large lineage expressions need to be exploited. Due to this expensiveness marginal probabilities are often only computed for individual tuples ad-hoc on user request [Wid09].

- **Extensional Query Evaluation:** In extensional query evaluation, probabilistic inference is incorporated into query evaluation, i.e. instead of performing a separate process for computing probabilities, these probabilities are directly computed by the database engine. This approach, however, is not an exact inference method and only works correctly for specific cases.

We start with a short introduction of intensional query evaluation and then discuss the underlying idea, benefits and shortcomings of extensional query evaluation. Note, although the ULDB model uses the symbol λ to denote lineage formulas, we will adopt the general notion of pc-databases and therefore will denote a lineage formula by Φ.

3.5.2.1. Intensional Query Evaluation

Since for computing the tuple probabilities based on the lineage formulas traditional methods of probabilistic inference are used, we focus on the database-specific part of intensional query evaluation, i.e. the query-specific construction of the lineage formulas of the output tuples from the lineage formulas of the input tuples. Logically, the construction depends on the individual semantics of the relational operators. For instance, the join operator can only join two tuples, if both tuples exist in the joined tables. As a consequence, given the two input tuples t_r and t_s with the lineage formulas Φ_{t_r} and Φ_{t_s} respectively, the join operator constructs the lineage formula of the output tuple $t = t_r \bowtie t_s$ as $\Phi_t = \Phi_{t_r} \wedge \Phi_{t_s}$. The lineage construction rules of some other relational operators (here we restrict to the basic operators) are listed in Table 3.2 and are based on the rules given by Fuhr and Rölleke [FR97].

Relational Operator		**Lineage Construction Rule**	
Projection:	$t \in \pi_{\mathcal{A}}(Q)$	$\Phi_t = \bigvee_{t_r \in Q, \pi_{\mathcal{A}}(t_r)=t} \Phi_{t_r}$	
Selection:	$t \in \sigma_c(Q)$	$\Phi_t = \Phi_{t_r}$	where $t_r \in Q, t_r = t$
Cross Product:	$t \in Q_1 \times Q_2$	$\Phi_t = \Phi_{t_r} \wedge \Phi_{t_s}$	where $t_r \in Q_1, t_s \in Q_2, t_r \times t_s = t$
Set Union:	$t \in Q_1 \cup Q_2$	$\Phi_t = \bigvee_{t_r \in Q_1, t_r=t} \Phi_{t_r} \vee \bigvee_{t_s \in Q_2, t_s=t} \Phi_{t_s}$	
Set Difference:	$t \in Q_1 - Q_2$	$\Phi_t = \begin{cases} \Phi_{t_r} \wedge \neg(\Phi_{t_s}) & \text{if } \exists t_s \in Q_2 \colon t_s = t, \\ \Phi_{t_r} & \text{else.} \end{cases}$	where $t_r \in Q_1, t_r = t$
Identity:	$t \in T$	$\Phi_t = \epsilon_i$	where ϵ_i is the event that $t_i \in T$

Table 3.2.: The lineage construction rules for the individual relational basic operators

Person

	DEI	name	age	p
t_1	p1	J.Doe	27	0.8
t_2	p1	J.Doe	28	0.2
t_3	p2	K.Smith	32	1.0
t_4	p3	J.Doe	28	0.6
t_5	p3	J.Ho	29	0.3

Read

	DEI	person	title	year	p
t_6	r1	p1	The Stranger	1998	0.6
t_7	r2	p2	Pale Fire	2003	1.0
t_8	r3	p2	The Stranger	1997	0.4
t_9	r4	p3	White Nights	2002	0.8
t_{10}	r5	p3	Homo Faber	2001	1.0

(a) Sample BID-database

SELECT p.name
FROM Person p, Read r
WHERE p.DEI = r.person
AND r.year $>$ 2000

(b) Sample query Q_5

	name	lineage	p
t_{11}	K.Smith	$(\epsilon_3 \wedge \epsilon_7)$	1.0
t_{12}	J.Doe	$(\epsilon_4 \wedge \epsilon_9) \vee (\epsilon_4 \wedge \epsilon_{10})$	0.6
t_{13}	J.Ho	$(\epsilon_5 \wedge \epsilon_9) \vee (\epsilon_5 \wedge \epsilon_{10})$	0.3

(c) Possible query answers

Figure 3.14.: Sample for a BID-database, a query on this database, and its set of possible query answers

Note that the operators selection and cross product does not eliminate duplicate entries and hence we always assume that the input tuple t_r (or the input tuples t_r and t_s respectively) are exactly known and unique for each output tuple t. Moreover, recall that the event ϵ_i can be represented by another propositional formula, i.e. the condition of tuple t_i, if T is a pc-table.

Example 35 *For demonstration*[6]*, we consider the BID-database from Figure 3.14(a). This database contains the BID-table* 'Person' *that we already know from Figure 3.8(a) and in addition contains a second BID-table* 'Read' *(note, due to the tuples in* 'Read' *are mutually independent it is even a TI-table). For simplification, we do not model foreign keys in this database (see discussion on referential integrity in Section 3.6.1). Moreover, we consider the query Q_5 from Figure 3.14(b) and pose it to the given BID-database. This query requests for the names of the persons that have read a book since* 2001*. Its set of possible answers is presented in Figure 3.14(c) and contains three output tuples. In order to illustrate the construction of the lineage formulas of the output tuples, we consider tuple t_{12} as an example. This tuple is a query answer if tuple t_4 is joined with tuple t_9 or if t_4 is joined with tuple t_{10}. Consequently, its lineage formula results in $\Phi_{t_{12}} = (\epsilon_4 \wedge \epsilon_9) \vee (\epsilon_4 \wedge \epsilon_{10})$ where the logical 'AND's are constructed from the join operator and the logical 'OR' is constructed by the final projection.*

The probabilities of the three possible query answers are presented in Figure 3.14(c) and can be explained as follows: Tuple t_{11} result from joining two certain tuples and therefore is certain by itself. The probability of tuple t_{12} is exactly the marginal probability of tuple t_4. This is correlation can be in turn simply explained. Due to tuple t_{10} is certain and belongs to all possible input worlds, its existence includes the existence of tuple t_9. Thus, the lineage condition $(\epsilon_4 \wedge \epsilon_{10})$ covers the condition $(\epsilon_4 \wedge \epsilon_9)$ and we therefore can ignore the latter condition for computing the probability of condition $\Phi_{t_{12}}$. As a consequence, the tuple $t' = t_4 \bowtie t_{10}$ (and hence the tuple t_{12}) results from each input world that contains t_4 and hence has the same probability than t_4. The same principle holds for tuple t_{13}. Because

[6]This example is based on a sample given by Dalvi et al. in [DRS09].

the existence of t_{10} includes the existence of tuple t_9, we only need to consider the condition $(\epsilon_5 \wedge \epsilon_{10})$ to compute the probability of t_{13}. Since this condition is true if tuple t_5 exists, the probability of t_{13} is equal to the probability of t_5.

3.5.2.2. Extensional Query Evaluation

The advantage of the extensional query semantics is that we can leverage standard optimization techniques of conventional database systems to speed up query evaluation in probabilistic databases. For this purpose, probability computation is incorporated into the operators of the relational algebra[7].

For instance, let t be a tuple that results from joining the two tuples t_r and t_s, i.e. $t = t_r \bowtie t_s$. Following the possible worlds semantics, t belongs to all possible worlds of the join result that are produced from any input world containing both t_r and t_s, i.e. $p(t) = p_\wedge(\{t_r, t_s\})$. In the simplest case, all input tuples are independent and the accumulative probability of all input worlds that produces any output world that contains t can be computed by the product of the marginal probabilities of t_r and t_s, i.e. $p(t) = p(t_r) \times p(t_s)$. In contrast, it holds that $p(t) = p(t_r)$ if the presence of t_r in any of the possible worlds includes the presence of t_s in the same world and results in $p(t) = 0$ if t_r and t_s are exclusive, i.e. they do not coexist in any of the possible worlds. In summary, the probability computed by an implementation of the join operator depends on its assumed tuple dependency and can contain one of the three computation rules:

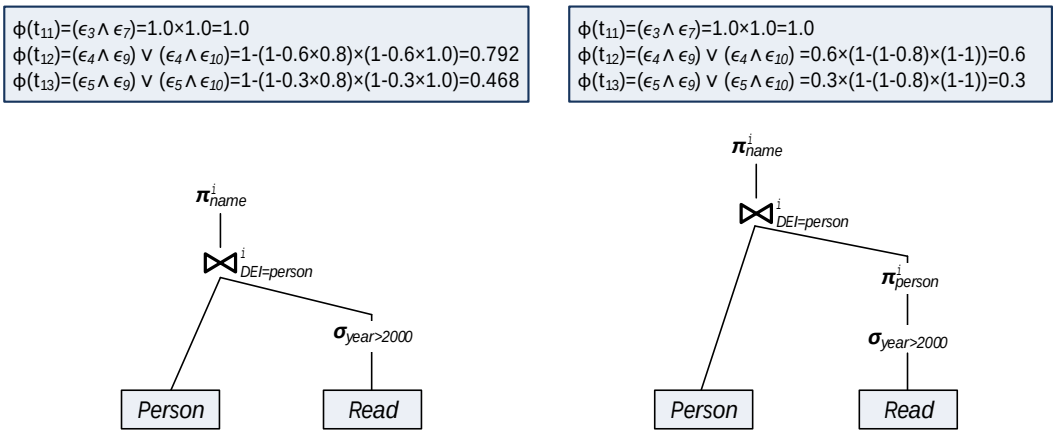
$$p(t) = \begin{cases} p(t_r) \times p(t_s), & \text{if } t_r \text{ and } t_s \text{ are independent,} \\ p(t_r), & \text{if } t_r \text{ includes } t_s, \\ 0, & \text{if } t_r \text{ and } t_s \text{ are exclusive,} \end{cases} \tag{3.10}$$

Similar computation rules can be inferred for the other relational algebra operators by assuming a specific kind of tuple dependency. For instance, the projection operator eliminates duplicate entries. Thus, let t be a tuple that results from projecting any of the tuples $t_1, \ldots, t_k$ on an attribute set $\mathcal{A}$, i.e. $t = \pi_{\mathcal{A}}(t_i)$ for each $i \in \{1, \ldots, k\}$. Tuple t only belongs to a possible world of the projection result if any of the k input tuples belong to a corresponding world of the input database, i.e. $p(t) = p_\vee(\{t_1, \ldots, t_k\})$. Consequently, the probability of t is computed as $p(t) = 1 - \prod_{i=1}^{k}(1 - p(t_i))$ if all these input tuples are mutually independent and results in $p(t) = \sum_{i=1}^{k} p(t_i)$ if all these tuples are mutually exclusive. Therefore, an implementation of the projection operator can perform one of the following rules:

$$p(t) = \begin{cases} 1 - \prod_{i=1}^{k}(1 - p(t_i)), & \text{if } t_1, \ldots, t_k \text{ are mutually independent,} \\ p(t_s), & \text{if } t_s \in \{t_1, \ldots, t_k\} \text{ is included by } t_1, \ldots, t_k, \\ \sum_{i=1}^{k} p(t_i), & \text{if } t_1, \ldots, t_k \text{ are mutually exclusive,} \end{cases} \tag{3.11}$$

Note that the operator cannot assume different kinds of dependencies for different input sets, but has to assume the same kind of dependency for all possible inputs. As a consequence, for each kind of assumed dependency we need another implementation per operator of the relational algebra.

[7] In an alternative approach, the relational operators are left unchanged and probability computation is hard coded into the query [DRS09].



(a) Unsafe query plan (b) Safe query plan

Figure 3.15.: An unsafe plan of sample query Q_5 and a safe plan of of sample query Q_5

Obviously, the computed probabilities are only correct if the actual dependencies between the input tuples correspond to the dependencies that are assumed by the used operator implementation. A query plan that returns a correct probability is called to be a *safe plan* [DS07b]. The concept of safe plans is a difference to query optimization in conventional databases, because not all query plans that can be constructed for a given query are safe and the query optimizer has to check safety during optimization, i.e. it has to find the cheapest plan that is safe.

Example 36 *For demonstration, we reconsider query Q_5 from Figure 3.14(b). Two of its plausible query plans are presented in Figure 3.15(a) and Figure 3.15(b). If all relational operators assume independence between the input tuples (denoted by $\bowtie^i$ and π^i), the first plan is not safe, but the second is. This can be explained as follows. In the first plan we first select all readings since* 2001 *and then join these readings with persons on the condition person $=$ DEI. Finally, we project on the name of these persons. The two tuples t_9 and t_{10} are both joined with the maybe-tuple t_4. Since the join results $t' = t_4 \bowtie t_9$ and $t^* = t_4 \bowtie t_{10}$ only belong to an output world if t_4 belongs to the input world, they are not independent*[8]*. Nevertheless, both tuples have the same value in the attribute* 'name' *and consequently are combined to tuple t_{12} by the finally performed projection operator. Thus, by assuming independence the finally computed probability of t_{12} incorrectly results in* 0.792*. Note that this incorrectness can be simple illustrated by the fact that the computed probability of t_{12} is greater than the probability of t_4, but as we have explained above t_{12} can only result from worlds that contain t_4 and hence $p(t_{12})$ can be at most as great as $p(t_4)$. The same holds for the output tuple t_{13} because its existence strongly depends on the existence of the input tuple t_5.*

In contrast, if we first project all readings since 2001 *on the attribute* 'person' *before performing the join with the table* 'Person'*, there is only one tuple, i.e. $t^\circ = \pi_{person}(t_9) = \pi_{person}(t_{10})$, that is joined with t_4 and the finally performed projection operator does not combine correlated tuples under an incorrect assumption. The independence assumption on the first projection, however, is valid because it combines*

[8] Note, the tuples t' and t^* are also neither exclusive nor include one the other, but they are correlated in a more complex way. Thus, even using another implementation of the projection operator cannot change the unsafe plan into a safe one.

only tuples from the table 'Read' *and all these tuples are mutually independent. As a consequence, by using this query plan the probability of tuple* t_{12} *correctly results in* 0.6.

The problem with this approach is that tuple dependencies can become really complex in intermediate query results even if the queried database is modeled in a simple representation system with only less dependencies. However, it is impossible to cover any kind of dependence by an own operator implementation, because the number of different kinds of dependencies can become infinite. For that reason, extensional query evaluation approaches usually restrict their implementations of the algebra operators to general kinds of dependencies as independence, exclusion, or inclusion. Moreover, an operator implementation needs to assume the same dependency for all its possible inputs. As a consequence, for some queries we cannot construct a safe plan. A query is called safe (and tractable) if it has a safe plan and is called unsafe (and intractable) otherwise [DS07b]. Algorithms to find safe plans for safe queries are presented in [DS07b, SORK11].

Example 37 *A simple example of a query that is unsafe with respect to BID-databases, is query* Q_4 *from Figure 3.12(a). The fact that this query is unsafe can be demonstrated by the possible answer* t_6 *(see Figure 3.13). This tuple results from joining the tuples* t_1 *and* t_3*, results from joining the tuples* t_2 *and* t_3*, and results from joining the tuples* t_4 *and* t_3*. Tuple* t_3 *is independent to all other tuples. The two tuples* t_1 *and* t_2 *are exclusive (they belong to the same block), but tuple* t_4 *is independent to both (belongs to another block). Therefore, a projection operator would not compute a correct probability neither by assuming independence* ($p(t_6) \neq 1 - (1 - p(t_1) \times p(t_3)) \times (1 - p(t_2) \times p(t_3)) \times (1 - p(t_4) \times p(t_3))$) *nor by assuming exclusion* ($p(t_6) \neq p(t_1) \times p(t_3) + p(t_2) \times p(t_3) + p(t_4) \times p(t_3)$).

Note that this query becomes safe (with respect to BID-databases) if we include the world keys of both join members into the finally projected set of attributes because all tuples that have the same values in these attributes result from joining different tuples of the same blocks and hence are mutually exclusive. Thus, probabilities can be correctly computed by using an implementation of the projection operator that assumes a mutual exclusion between all input tuples. The corresponding query Q_6 *is presented in Figure 3.16(a).*

Interestingly, this modified query becomes in turn unsafe (with respect to BID-databases), if we only join on the attribute 'age' *and remove the comparison on the non-equivalence of the world key from the join condition (see query* Q_7 *in Figure 3.16(b)), because in this case we join tuples with themselves and such queries are per se unsafe [DS07b]. In general, as Dalvi and Suciu have figured out, queries with selfjoins are rarely safe [DS07b].*

Whether or not a query is safe does not only depend on the query itself, but also depends on the correlations between the queried tuples and therefore essentially depends on the considered representation system. As a consequence, queries that contain joins and/or projections are rarely safe with respect to pc-databases, because in this case the tuple correlations are not predefined by the representation system, but can be individually defined for each tuple set instead.

By only considering data complexity[9] all queries that are written in a specific query language can be distinct into two complexity classes [DS07b]. Whereas the evaluation of the queries of the first class

[9] Data complexity refers to the complexity of evaluating a fixed query depending on the size of the input data [Var82].

SELECT	t.WK, t.name, u.WK, u.name
FROM	Person t, Person u
WHERE	t.WK ! = u.WK
AND	t.age $<$ u.age

(a) Safe sample query Q_6

SELECT	t.WK, t.name, u.WK, u.name
FROM	Person t, Person u
WHERE	t.age $<$ u.age

(b) Unsafe sample query Q_7

Figure 3.16.: Sample of a safe selfjoin query and a sample of an unsafe selfjoin query

is always in PTIME, the evaluation of the queries of the second class is always #P-hard. According to Dalvi et al. [DS07b] independent from the considered language the first complexity class corresponds to the class of safe queries and the second complexity class corresponds to the class of unsafe queries. Dalvi et al. present such a *Dichotomy* for conjunctive queries without self joins and with respect to TI-databases in [DS07b, DS07a] and detect that in this case the set of safe queries corresponds to the class of hierarchical queries (a further dichotomy is proven in [DS12]). Interestingly, if the complexity of approximations is additionally taken into account, some query classes even form a trichotomy [RS09]. The advantage of this observation is that if this dichotomy (or trichotomy respectively) is known for the used query language and the used representation system, safe queries can be detected with manageable effort (usually in polynomial time [DS07b]).

Due to the high complexity of probabilistic inference, it is desirable to use extensional query evaluation as often as possible. Nevertheless, if we require for exact probabilities in the case of an unsafe query, we usually have no other choice than using intensional query evaluation and to compute marginal probabilities by a conventional probabilistic inference algorithm instead. However, as presented by Olteanu et al.[OHK10, SORK11] extensional query evaluation can be used even for unsafe queries. By doing so, the resultant probabilities are indeed incorrect, but can be used as approximations of the correct probabilities. Obviously, the accuracy of the resultant approximations depends on the considered query. For instance, in the case of unsafe UCQ queries without self joins, an extensional plan that computes tuple probabilities by the independence assumption returns an upper bound for the correct probabilities of the possible query answers [OHK10]. This upper bound in turn can be suitable to remove a tuple from the final query result if only the k most probable query answers are requested.

In recent time several research has been done to blur the hard line between both data complexity classes. One approach is to use materialized views [RS07, DRS11]. In that case frequently used unsafe subqueries are persisted in the database and queries using these subqueries are reformulated in order to use the corresponding views instead. Another approach is to decompose a query plan into sub-plans and to evaluate as many sub-plans by using extensional querying mechanisms as possible [DS07b]. Finally, functional dependencies, e.g. keys, can help to transform an unsafe query into an semantically equivalent safe query [DS07b, OHK09].

3.5.3. Approximation based on Sampling Possible Worlds

In a later section, we will make use of an approximation technique that is based on the Monte-Carlo Simulation [KLM89] and that is the fundamental query mechanism in Monte-Carlo databases (short

MCDBs) [JXW+08, AJP+10, JXW+11]. The underlying semantics of this approximation technique is not to evaluate the considered query in all possible worlds, but to rather evaluate it in a set of N sample worlds where N is the number of Monte-Carlo iterations specified. MCDBs model uncertainty by assigning variable generation functions (discrete or continuous), as for example the normal distribution, to single attribute values, to sets of correlated attribute values, or to sets of correlated tuples. Therefore, sampling one world is done by selecting a value from each variable generation function randomly.

The probability of a possible query answer is finally approximated as the proportion of sample worlds in which this tuple is a query answer. Thus, let Q be the considered query that is posed to a probabilistic database $pdb = (\mathbf{W}, Pr)$, let $\mathbf{W}_N \subseteq \mathbf{W}$ be the set of the N sample worlds, and let $M \leq N$ be the number of all sample worlds in which tuple t is a query answer, the marginal probability that t is an answer of $Q(pdb)$ is approximated as:

$$p(t) = \frac{|\{W \in \mathbf{W}_N \mid t \in Q(W)\}|}{|\mathbf{W}_N|} = \frac{M}{N} \tag{3.12}$$

The drawback of this naive approach is that each world can be large in size and that the number of Monte-Carlo iterations N usually need to be between 10 and 1000 to guarantee an appropriate accuracy of the approximated probabilities [JXW+11]. Moreover, many queries only return few tuples, i.e. the most tuples are filtered out by the queries' conditions. This leads to the problem that in many cases the most samples will not contain a single possible query answer and hence we expend much effort in sampling and querying tuples to almost no value. To solve this problem, sampling is often processed after applying the query to all input tuples. By doing so each tuple needs to be queried only once. In addition, MCDBs use some model-specific optimization methods to decrease runtime further on.

In general, the same approximation principle can be applied to any of the representation systems we have presented in Section 3.3. In that case, all tuples are collectively queried at a stroke and a lineage function is build for each of them. Since the lineage functions refer to the original representations of uncertainty (e.g. probabilities or variable assignments), sampling can be performed afterwards. Of course, compared to MCDBs another sampling method need to be used, e.g. by selecting a variable assignment randomly (pc-databases) or by selecting a tuple per block randomly (BID-databases).

3.5.4. Evaluation of Aggregate Queries in Probabilistic Databases

Evaluating aggregate queries over uncertain data has been considered in several works [GMSS09, MIW11, CCT96, MSS01, LSV02, CKP03, JKV07, ACN08, JMMV08, MIW11, YWCK11, FHO12, BKOZ13, CD08]. Obviously, from each possible world another aggregation can result. Thus, in the worst case the number of possible aggregations is close to the number of possible worlds and therefore can be prohibitively large. As a consequence, aggregation results often need to be presented in a more compact way. Gal et al. [GMSS09] define three semantics for evaluating aggregate queries over probabilistic databases.

- **Range Semantics:** In the range semantics the aggregation result is presented by a lower and an upper bound, i.e. let $pdb = (\mathbf{W}, Pr)$ be a probabilistic database and let Q be an aggregate query, the result of $Q(pdb)$ is a pair $[l, u]$ with $l = \mathbf{min}_{W \in \mathbf{W}} Q(W)$ and $u = \mathbf{max}_{W \in \mathbf{W}} Q(W)$.

- **Distribution Semantics:** In the distribution semantics every possible aggregation result is listed along with its probability, i.e. let $pdb = (\mathbf{W}, Pr)$ be a probabilistic database and let Q be an aggregate query, the result of $Q(pdb)$ is a set of pairs (v, p) where $v \in \{Q(W) \mid W \in \mathbf{W}\}$ and $p = \sum_{W \in \mathbf{W}, Q(W)=v} Pr(W)$.

- **Expected Value Semantics:** In the expected semantics the aggregation result is presented by its expected value, i.e. let $pdb = (\mathbf{W}, Pr)$ be a probabilistic database and let Q be an aggregate query, the result of $Q(pdb)$ is a the single value $exp = \sum_{W \in \mathbf{W}} Pr(W) \times Q(W)$.

It is simple to see that the distribution semantics corresponds to the possible worlds semantics and hence has the aforementioned complexity of the number of possible worlds. Of course, the result of the first semantics and the result of the third semantics can be derived from the result of second semantics. Nevertheless, in most cases the computation of these semantics is much more efficient if we compute them directly on the input data instead of computing the result of the distribution semantics first. Obviously, let Q be an arbitrary aggregate query, let *pdb* a probabilistic database, let $[l, u]$ be the aggregation result of $Q(pdb)$ under the range semantics and let be exp the aggregation result of $Q(pdb)$ under the expected value semantics, it generally holds that $l \leq exp \leq u$.

Whether or not an aggregate query can be exactly computed in an efficient way depends on the used aggregate operator, the used aggregation semantics, and the considered representation system. Gal et al. [GMSS09] consider aggregate queries under the *by-tuple* semantics in the presence of uncertain schema mappings. Evaluating an aggregate query under the *by-tuple* semantics in the presence of uncertain schema mappings corresponds to evaluating the same aggregate query on a BID-table. Thus, we can adopt the results provided by Gal et al. from the by-tuple semantics to BID-tables. Queries using the aggregate operator COUNT can be evaluated in polynomial time under each of the three aggregation semantics. In the case of the aggregate operator SUM, Gal et al. found a method for computing the aggregation result in polynomial time only for the range semantics and the expected semantics, but have not found such a method for the distribution semantics. In the case of the aggregate functions MIN, MAX, and AVG, they found a polynomial time computation method only for the range semantics.

Ross et al. [RSG02, RSG05] analyze aggregate queries for FP-Relations under the distribution semantics, propose an exact and generic (but usually intractable) approach for aggregate computation, and present an algorithm that approximate the aggregation result in polynomial time by processing only some of the possible worlds. Obviously, the accuracy of the approximated result increases with the number of processed worlds, but it has been shown that in most cases a manageable number of worlds is sufficient to produce a result of acceptable quality. Since FP-relations are generalizations of BID-tables, i.e. each BID-table can be represented as a FP-relation by setting the upper bound and the lower bound to the same probability value, these approximation algorithm can be used for querying a BID-table, too.

Murthy et al. [MW07, MIW11] describe how aggregate queries are evaluated in the Trio system [Wid09]. In general, Trio supports all three aggregation semantics, but restricts the evaluation of aggregate queries to base-tables that do not contain lineage and therefore correspond to BID-tables. They present methods to compute COUNT, MIN, MAX, and SUM exactly under all semantics, present methods to compute AVG exactly under the range semantics and the distribution semantics, and propose an algorithm to approximate AVG under the expected value semantics. In the case of the distribution seman-

tics, aggregation results are computed by enumerating all possible worlds and hence are only tractable for small world spaces. In contrast, in the range semantics and the expected value semantics, Trio uses its encoding scheme to rewrite a TriQL query, i.e. a query written in the query language of Trio, into a conventional SQL query or uses a stored procedure that is composed by a sequence of conventional SQL queries and thus is able to compute aggregate queries under this semantics in an efficient way. The only exception is the computation of the expected value of the AVG operator. For this operator such a rewriting approach is currently not known, but its result need to be approximated instead. In Trio, approximation is done by dividing the expected SUM aggregation by the expected COUNT aggregation.

In summary, we have a powerful repertoire of exact algorithms and approximate algorithms to compute an aggregate query on a BID-database. In contrast, an evaluation of aggregate queries in pc-databases is more complex and has been only less considered in the research community so far. Lechtenbörger et al. [LSV02] consider an evaluation of the distribution semantics on c-tables and illustrate the exponentially complexity of the size of the aggregation result (recall, the number of possible aggregation results can be equal to the number of possible worlds). Moreover, they present algorithms to approximate a probability distribution on possible aggregation results of a reasonable size. Fink et al. [FHO12] introduce a specific representation system called *probabilistic value-conditioned tables* (short *pvc-tables*) in order to represent the enormous complexity of the aggregation result computed with the distribution semantics in a more compact way. Koch et al. [Koc09] incorporate the range semantics and the expected value semantics into the MayBMS system. However, they do not provide details on this incorporation so that we do not know whether these values are exactly computed or only approximated.

3.6. Further Remarks

In this section, we conclude our presentation of uncertain databases by discussing some further interesting aspects of uncertain data

3.6.1. Referential Integrity

To the best of our knowledge, referential integrity in uncertain databases has not been considered in database literature so far. For that reason, we present some of our own thoughts in this section and discuss what kinds of restrictions are required in the individual representation systems in order to ensure referential integrity.

An integrity constraint is satisfied by an uncertain database if it is satisfied in every possible world of this database. Thus, to satisfy referential integrity in an uncertain database, a referencing tuple can only belong to a possible world if its referenced tuple belongs to this world as well.

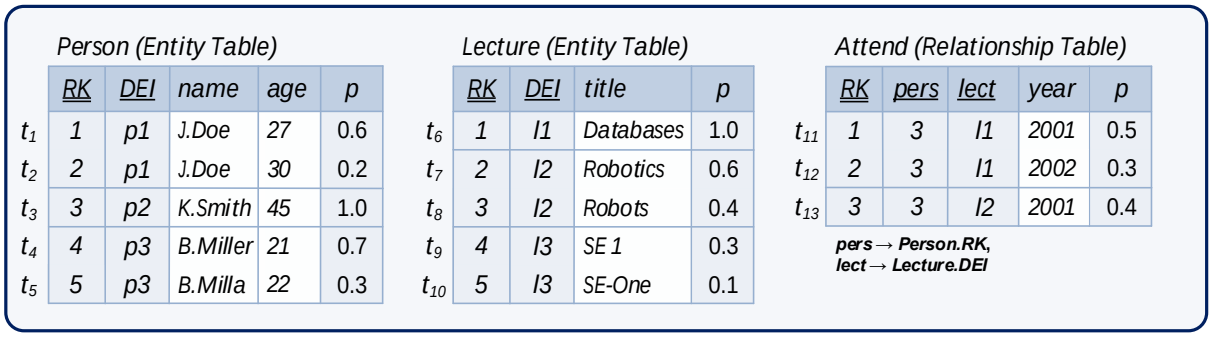
In general, foreign keys can reference to every attribute (or combination of attributes) that is unique in every possible world. In this thesis, we restrict our consideration to references to the world key and references to the representation key that are both unique in every possible world per definition.

Recall, in TI-databases, AOR-databases, and AOR?-databases the world key corresponds to the representation key so that we do not need to distinguish between both cases. In AOR-databases all tuples are certain and key attributes have to be certain. Consequently, referential integrity in AOR-databases corresponds to referential integrity in conventional databases, i.e. it is satisfied if every foreign key value refers

to an existing primary key value of the referenced table. In contrast, TI-databases and AOR?-databases can contain (A-)tuples that are only maybe. Moreover, all (A-)tuples are mutually independent. Thus, by constructing the possible world space as defined in Section 3.3.1 (or as defined in Section 3.3.3 respectively), referential integrity can only be ensured if all referenced (A-)tuples are certain. In theory, this requirement can be relaxed by including a check for referential integrity into the world space construction algorithm, i.e. worlds are only added to the resultant world space if they satisfy referential integrity. In this case, the referencing (A-)tuple is always at most as probable as its referenced (A-)tuple, because the existence of the first includes the existence of the latter. This approach, however, does not only introduce dependencies between the referencing (A-)tuple and the referenced (A-)tuple, but also introduce dependencies between other (A-)tuples. For instance, if two (A-)tuples t_r and t_s both reference to a third (A-)tuple t_u, the (A-)tuples t_r and t_s are not independent anymore. Moreover, they do not necessarily include or exclude each other, but can be correlated in a much more complex way. Nevertheless, by using intensional or extensional query mechanisms the possible world space is never constructed. Thus, worlds cannot be directly checked for satisfying referential integrity and the aforementioned correlations need to be stored in the database instead. Since this is not the purpose of TI-databases or AOR?-databases and because ignoring these correlations can cause incorrect query answers, adapting the world space construction algorithm for checking referential integrity is only an option if queries are answered by using an approximative approach that samples possible worlds such as the Monte-Carlo Simulation.

In BID-databases and pc-databases, we have to distinguish between references to world keys and references to representation keys. Whereas a foreign key that references to a representation key always corresponds to a reference to a single tuple, a foreign key that references to a world key corresponds to a collective reference to a block of tuples, i.e. all tuples that share the same world key value. Under the entity-based interpretation the second corresponds to a reference to an entity and the first corresponds to a reference to a possible instance of an entity, i.e. it restricts the existence of the modeled relationship to a specific instance of the referenced entity. For example, a car can belong to a person regardless of its residence (reference to an entity), but a car can also belong to a person only if she lives in Hamburg (reference to a possible instance of an entity). Thus, by defining a foreign key as a reference to a world key, the value uncertainty that is modeled in the referencing table is independent from the value uncertainty that is modeled in the referenced table. In contrast, a reference to a representation key can introduce correlations between the value uncertainties that are modeled in both tables.

Since the representation key is not part of the world schema, the referenced representation key value need to be replaced by its corresponding world key value, i.e. the *DEI* of the referenced entity, for constructing a possible world (recall, the world key value can be deterministically derived from the representation key value). For the same reason, the system needs to rewrite queries that are written based on the world schema and therefore aim to select tuples or aim to join tuples based on the referenced world key value. We demonstrate such a rewriting in Example 39. Another option is to define the foreign key not only on the representation key, but to define it on the combination of the representation key and the world key instead, i.e. a foreign key that references to the table T is composed by the two attributes A_1 and A_2 and is of the form $(A_1, A_2) \rightarrow T.(RK, WK)$. In this case, the world key value of the referenced tuple is additionally stored in the referencing tuple and queries do not need to be rewritten and possible worlds do not need to be manipulated. Due to space reasons, however, we abstain from



Person (Entity Table)

	RK	DEI	name	age	p
t_1	1	p1	J.Doe	27	0.6
t_2	2	p1	J.Doe	30	0.2
t_3	3	p2	K.Smith	45	1.0
t_4	4	p3	B.Miller	21	0.7
t_5	5	p3	B.Milla	22	0.3

Lecture (Entity Table)

	RK	DEI	title	p
t_6	1	l1	Databases	1.0
t_7	2	l2	Robotics	0.6
t_8	3	l2	Robots	0.4
t_9	4	l3	SE 1	0.3
t_{10}	5	l3	SE-One	0.1

Attend (Relationship Table)

	RK	pers	lect	year	p
t_{11}	1	3	l1	2001	0.5
t_{12}	2	3	l1	2002	0.3
t_{13}	3	3	l2	2001	0.4

pers* → *Person.RK*, *lect* → *Lecture.DEI

Figure 3.17.: Sample of foreign keys within BID-databases

using this solution in modeling ordinary relationships and use it only for modeling *is-a*-relationships within inheritance hierarchies, because otherwise entity tables that represent subtypes would not contain the *DEI* attribute which complicates an entity-based interpretation of probabilistic databases.

In BID-databases, tuples from different blocks and thus tuples from different tables are independent (recall, in the entity-based interpretation of BID-databases, we consider that all tuples of one block represent the same database entity and therefore the *DEI* (i.e. the world key) can be used as block number). Due to the aforementioned independencies, referential integrity is only satisfied if all the referenced key values are certain. Thus, if a referenced value is a representation key value, we need to constrain that its corresponding tuple is certain and hence need to constrain that its corresponding entity has a single possible instance in the referenced table. In contrast, if a referenced value is a world key value we need to constrain that its corresponding block is certain and thus need to constrain that the membership of its corresponding entity to the considered table is certain.

Note that it makes only less sense to define a foreign key on the representation key of a database table, i.e. the table's representation key references to the representation key (or world key respectively) of another table, because in this case the table cannot model any uncertainty, i.e. each tuple of the referenced table can only be referenced once in the referencing table.

The world key of a relationship table results in the combination of its foreign keys and maybe some additional attributes. Since all tuples of one block need to share the same world key value, in BID-databases we can only model uncertainty on the attributes that do not belong to the world key. Thus, if we want to model uncertainty on the referenced entities that fill the individual roles, we need to add a surrogate key attribute that serves as world key instead.

Example 38 *For illustration we consider the BID-database presented in Figure 3.17. This database contains three tables, two entity tables (*'Person' *and* 'Lecture'*) and one relationship table (*'Attend'*) where the latter table connects the first two tables by foreign keys. In a certain relational database, the table* 'Attend' *would contain two foreign keys. One that references to the world key (the DEI) of the table* 'Person' *and one that references to the world key (the DEI) of the table* 'Lecture'. *Of course, several possible instances per relationship are enabled by representing this table as a BID-table and therefore by adding the representation key attribute to the table's schema. If we additionally want to correlate a particular possible instance of a relationship with a particular possible instance of the entities that are referenced by this relationship, we need to replace the foreign key that references to the world key*

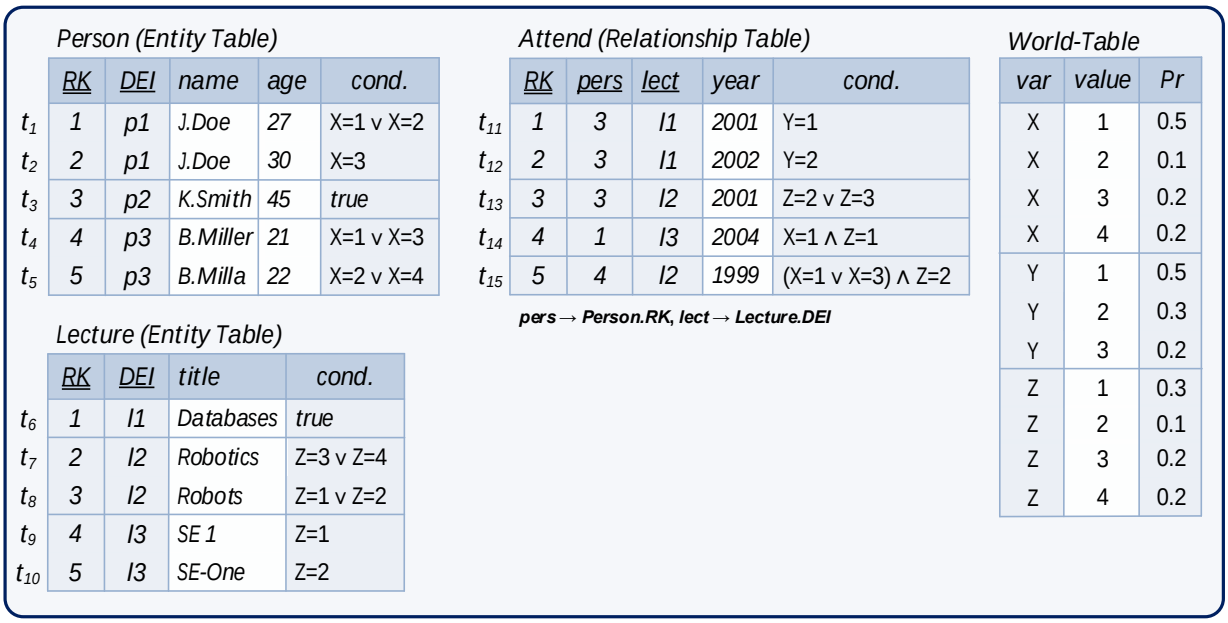


Figure 3.18.: Sample of foreign keys within pc-databases

by a foreign key that references to the corresponding representation key. In our example, we perform such a replacement for the table 'Person'*, i.e. the attribute* 'pers' *references to the representation key of table* 'Person' *instead of its world key, but do not make such a replacement for the table* 'Lecture'*. As a consequence, we can define a relationship of the type* 'Attend' *depending on a particular possible instances of a person, but cannot define it depending on a particular possible instances of a lecture.*

The world key of the relationship table 'Attend' *corresponds to its primary key in the possible worlds representation and therefore is composed by its foreign key attributes, i.e. the two attributes* 'pers' *and* 'lect'*.*

Since we use the world key as block number, tuples in relationship tables that share the same world key value represent mutual exclusive alternative instances of the same relationship. For demonstration, whereas the first two tuples of the table 'Attend' *model two alternative instances of a possible attendance of person p_2 at the lecture l_1, the third tuple models the single alternative instance of a possible attendance of person p_2 at the lecture l_2.*

Recall, in BID-databases all referenced tuples and all referenced blocks need to be certain. Consequently, tuple t_3 (and hence person p_2) is the only tuple from table 'Person' *that is allowed to be referenced in table* 'Attend'*. In contrast, lecture l_3 is the only lecture than is not allowed to be referenced in table* 'Attend'*, because it is the only lecture that is not represented by a certain block, i.e. this entity is maybe not a lecture.*

The given example shows that using foreign keys in BID-databases is very restrictive, because we cannot model a lecture attendance for some persons and cannot model the attended persons for some lectures if we connect the individual tables by foreign keys.

In pc-databases, we are able to model any kind of dependency and hence do not need to restrict to cases where foreign keys reference to certain representation key values (single tuples) or certain world key values (tuple blocks) respectively. Nevertheless, we need to ensure that the referencing tuple only

belongs to worlds that contain the referenced tuple (or the referenced block respectively) as well. For that reason, the condition of the referencing tuples is only allowed to be satisfied if the condition of the referenced tuple (or block respectively) is satisfied. Therefore, let t_r be a tuple that references to tuple t_s, referential integrity is only satisfied if $\varkappa(\Phi_{t_r}) \subseteq \varkappa(\Phi_{t_s})$[10]. Moreover, let t_r be a tuple that references to a tuple block B_s, referential integrity is only satisfied if $\varkappa(\Phi_{t_r}) \subseteq \bigvee_{t_s \in B_s} \varkappa(\Phi_{t_s})$. Note as for referential integrity in certain databases, these constraints do not need to be considered during query evaluation or possible world space construction, but need to be validated during data insertion, data updating, or data deletion instead.

Example 39 *For illustration we consider the same world schema as in the previous example, but now assume that the database is represented as a pc-database, i.e. each tuple stores a condition instead of a probability and a separate world-table stores the assignments of all variables that are used in the tuples' conditions. The correspdoning pc-database is presented in Figure 3.18. The first three tuples of the relationship table* 'Attend' *are the same as in the previous example. The tuples t_{11} and t_{12} each references only to certain tuples and therefore only require a condition that models the given uncertainty on the attribute* 'year' *by using the variable Y. The third tuple references to a certain tuple (table* 'Person'*) and a certain block (table* 'Lecture'*) respectively. Since its condition is covered by the combined conditions of the tuples that represent lecture l_2, the reference to this lecture is valid.*

In contrast to BID-databases, in pc-databases we are not restricted to references to certain tuples and certain blocks. This is illustrated by the last two tuples in table 'Attend'*. The fourth tuple references to the uncertain tuple t_1 and references to the uncertain block l_3, but because its condition is covered by the condition of tuple t_1, i.e. $\varkappa(X = 1 \wedge Z = 1) \subseteq \varkappa(X = 1 \vee X = 2)$, and is covered by the combined condition of the tuples t_9 and t_{10}, i.e. $\varkappa(X = 1 \wedge Z = 1) \subseteq \varkappa(Z = 1 \vee Z = 2)$, referential integrity is satisfied. Similar holds for the condition of tuple t_{15}, i.e. $\varkappa((X = 1 \vee X = 3) \wedge Z = 2) \subseteq \varkappa(X = 1 \vee X = 3)$ and $\varkappa((X = 1 \vee X = 3) \wedge Z = 2) \subseteq \varkappa(true)$.*

Note, a reference to a particular entity instance is demonstrated by tuple t_{14}. Since this tuple references to tuple t_1 and there is no other tuple that references to the other possible instances of person p_1, we condition the attendance of p_1 at lecture l_3 to the case where p_1 is 27 years old.

A sample for query rewriting in the presence of foreign keys that references to representation keys is presented in Figure 3.19. The original query is defined on the world schema and therefore aim to select all lectures (including annual details) that have been attended by person p_1. Since the DEI of the person is not stored in the table 'Attend'*, we need to rewrite the query into one that first joins the tables* 'Attend' *and* 'Person' *on the defined foreign key and then selects all tuples with the wanted DEI.*

The decision whether a foreign key should reference to a representation key or should reference to a world key must be made on schema level and consequently is made for all tuples that are stored in the designed table collectively. Thus, references to representation keys can implicate an overload in cases of independencies, because each possible combination must be explicitly modeled by an own tuple in the referencing table. Such a waste of storage can be avoided if we define the foreign key on the aforementioned combination of the world key and the representation key, and then set the representation key value to null if the modeled relationship is valid for all possible instances of the referenced entity. Of

[10] Recall $\varkappa(\Phi)$ is the set of all variable assignments that satisfy condition Φ).

SELECT	a.lect, a.year		**SELECT**	a.lect, a.year
FROM	Attend a	*rewrite*	**FROM**	Attend a, Person p
WHERE	a.pers = 'p1'	$\xrightarrow{\quad}$	**WHERE**	a.pers = p.RK
			AND	p.*DEI* = 'p1'

Figure 3.19.: A sample query and its rewritten equivalent defined on the database schema from Figure 3.18

course, such an adaptation possibly requires a rewriting of queries that directly address the representation key.

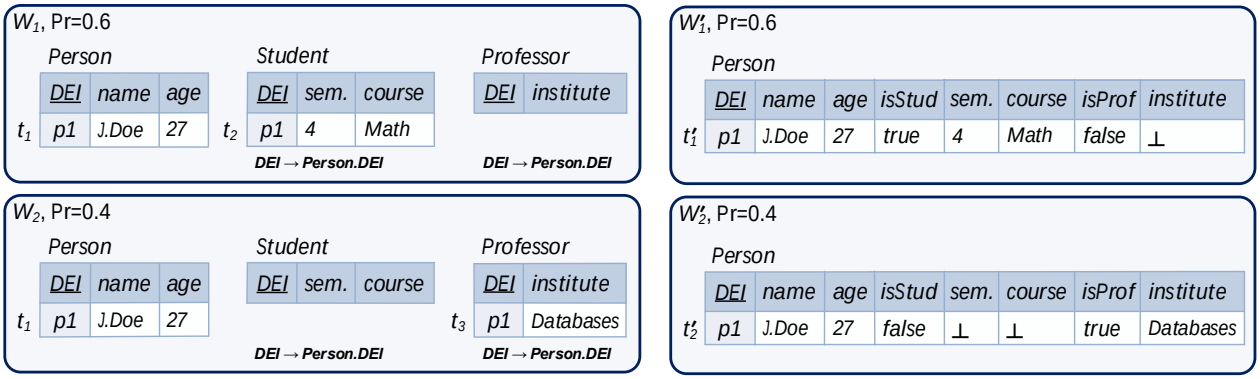
3.6.2. Modeling Inheritance Hierarchies in Probabilistic Representation Systems

As referential integrity, a modeling of inheritance hierarchies within a probabilistic representation system has not been considered in existing literature so far. In general, inheritance is an entity-specific concept. For that reason, we directly interpret it in an entity-based way.

In Section 2.1.2, we present two approaches for modeling an inheritance hierarchy within the relational data model. One by storing all attributes as long as some specific membership attributes in a single entity table (Single Table Approach) and one that creates one entity table per involved entity type and that connects these tables with foreign keys (Vertical Partitioning Approach). In certain relational data, the main difference between both approaches are the space requirements and the complexity of query answering, but each of them is applicable for every possible scenario and it is often only less important which of them is actually used in relational schema design. With respect to probabilistic representation systems, however, these approaches have some essential differences that we will present in this section.

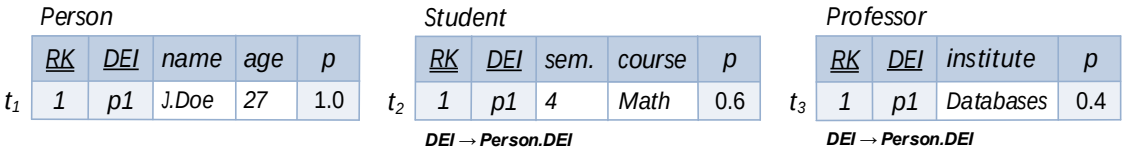
Of course, the concept of pc-databases is a complete representation system and therefore for designing a pc-database the Single Table Approach and the Vertical Partitioning Approach can be used abundantly. In contrast, there are several scenarios in which a possible world space cannot be represented by a BID-database if its schema has been designed by using the Vertical Partitioning Approach, but an information equivalent world space can be represented by a BID-database if its schema has been designed by using the Single Table Approach. For instance, uncertainty on whether a specific person is a student or a professor cannot be stored in a BID-database by using the Vertical Partitioning Approach, because both memberships are mutual exclusive and this exclusion cannot be modeled between tuples from different tables. In contrast, we can model this membership exclusion in a BID-database if we use the Single Table Approach, because in this case we can utilize the system's inherent mutual exclusion between the tuples of one block to model the exclusion between the aforementioned memberships. Altogether, we observe four cases in which an inheritance hierarchy cannot be modeled within a BID-database if it has been designed by using the Vertical Partitioning Approach, but can be modeled within a BID-database if it has been designed by using the Single Table Approach. Such a case occur, if

- the inheritance hierarchy contains a disjoint set of specializations,
- the inheritance hierarchy contains a total set of specializations,
- the membership to a supertype can be uncertain, or

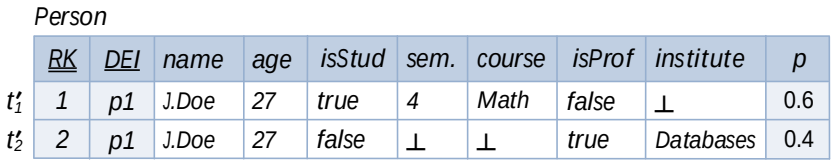


(a) Possible worlds representation of pdb_1

(b) Possible worlds representation of pdb'_1



(c) BID-database pdb_1^*



(d) BID-database representation of pdb'_1

Figure 3.20.: Modeling a set of disjoint specializations in BID-databases

- for some entities the memberships to some subtypes are correlated.

This curiosity can be explained by the fact that in BID-databases all the tuples from different tables are always independent and whereas the Vertical Partitioning Approach models inheritance by the use of different tables (and hence different tuples) the Single Table Approach does not. Therefore, the latter can introduce correlations that cannot be introduced by the first.

We start with the aforementioned situation where two or more subtypes of the same supertype are membership exclusive, i.e. these types form a disjoint set of specializations to the same supertype. Note, modeling such a situation within a BID-database can become crucial in many uncertain database applications because it is often not known to which subtype an entity actually belongs to.

Example 40 *For illustration, we consider an inheritance hierarchy that includes the three entity types* 'Person', 'Student', *and* 'Professor'. *Obviously, the first is a supertype of the latter two. Furthermore, the extensions of* 'Student' *and* 'Professor' *are disjoint. Moreover, we consider the uncertain instance of a database entity* p_1. *In the first instance* p_1 *is a person and a student and in the second instance* p_1 *is a person and a professor. As a consequence,* p_1 *is certainly a person, but it is unknown whether she is a student or a professor.*

Now, we consider the possible worlds representations of two probabilistic databases pdb_1 and pdb'_1 that are depicted in . Figure 3.20(a) and Figure 3.20(b) respectively. The possible worlds of the first database are defined on a relational schema that results from transforming the aforementioned inheritance hierarchy into the relational data model by using the Vertical Partitioning Approach. In contrast, the schema of the possible worlds of the second database results from transforming the same inheritance hierarchy into the relational data model by using the Single Table Approach. Note that the attribute 'isStud' *models the membership to the subtype* 'student' *and the attribute* 'isProf' *models the membership to the subtype* 'professor'. *Recall, in certain databases (and possible worlds are certain databases) both transformation approaches produce semantically equivalent database schemas. Moreover, as depicted in Figure 3.20(a) and Figure 3.20(b) respectively both possible worlds representations model the same instance data, i.e. the two possible instances of p_1. Thus, despite of the differences in their schemas, both possible worlds representations model the same information.*

The possible worlds representation of pdb_1, however, cannot be modeled within a BID-database, because it contains dependencies between tuples from different tables. For demonstration, we consider the BID-database pdb_1^* presented in Figure 3.20(c). This database has all three tuples of the considered possible worlds and indeed assigns each tuple with its correct marginal probability. The mutual exclusion between tuple t_2 and tuple t_3, however, is not captured in this database. As a consequence, the possible worlds representation of pdb_1^* is not the one from Figure 3.20(a), but is one that has four possible worlds (one with only t_1, one with t_1 and t_2, one with t_1 and t_3, and one with all three tuples).

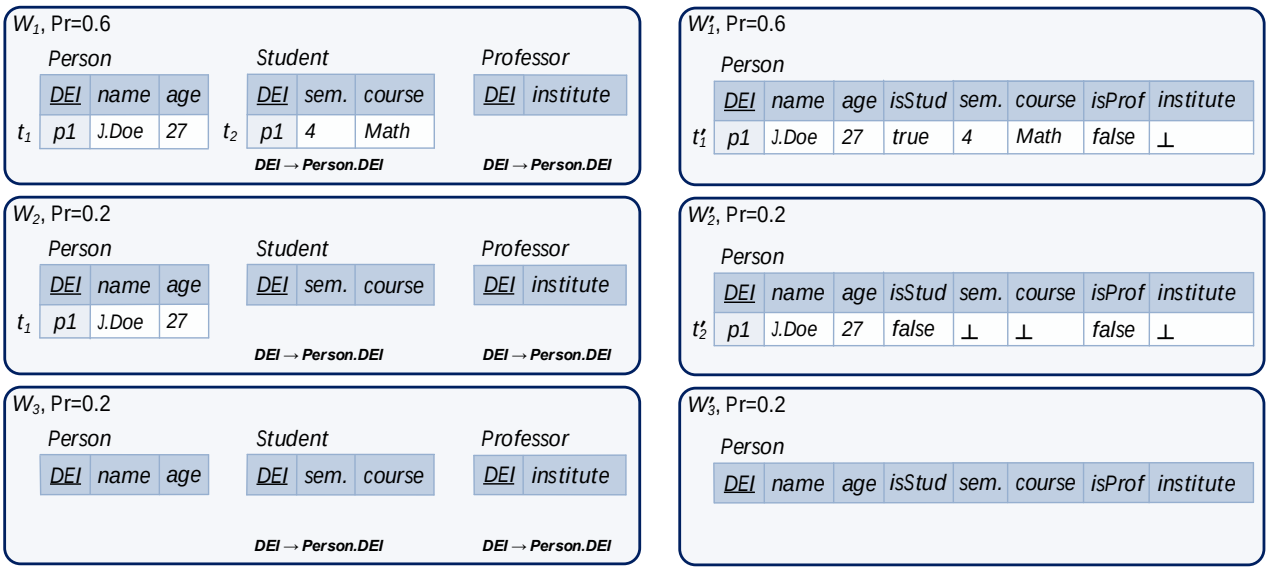
Interestingly, the possible worlds representation of pdb'_1 can be represented by a BID-database as presented in Figure 3.20(d). Since both tuples of this BID-database belong to the same block (i.e. they share the same DEI), the mutual exclusion between the membership to the subtype 'student' *and the subtype* 'professor' *is modeled by using the system-specific mutual exclusion between tuples from the same block.*

This example demonstrates that we can represent a set of disjoint specializations within a BID-database if we use the Single Table Approach for schema design, but cannot represent such a case if we design the schema by using the Vertical Partitioning Approach. For the first, we simply require a constraint that guarantees that a tuple cannot have the value 'true' in two membership attributes that represent membership exclusive subtypes.

A similar observation can be made if a set of specializations is total, because the correlation that the entity must belong to any of the subtypes if it belongs to the supertype cannot be modeled within a BID-database if the entity is represented by tuples in different tables. Nevertheless, we can model it by the use of integrity constraints if all membership information of one entity is modeled within the same tuple.

Example 41 *For illustration, we re-consider Example 40 and assume that each person need to be a student or a professor. This additional requirement can be simply incorporated into the BID-database presented in Figure 3.20(d) by adding the constraint that a tuple is not allowed to have a null value in both of the membership attributes* 'isStud' *and* 'isProf'.

Thus, for modeling a total set of specializations, we simply require a constraint that ensures that each tuple has the value 'true' in at least one of the corresponding membership attributes.



(a) Possible worlds representation of pdb_2

(b) Possible worlds representation of pdb'_2

Person

	RK	DEI	name	age	isStud	sem.	course	isProf	institute	p
t'_1	1	p1	J.Doe	27	true	4	Math	false	⊥	0.6
t'_2	2	p1	J.Doe	27	false	⊥	⊥	false	⊥	0.2

(c) BID-database representation of pdb'_2

Figure 3.21.: Modeling uncertain memberships to supertypes in BID-databases

A third difference can be observed if the membership of an entity to a supertype is uncertain. Due to in BID-databases, foreign keys are only allowed to reference to certain blocks (or certain tuples respectively), we cannot represent such a situation in a BID-database if its schema is designed by using the Vertical Partitioning Approach. Nevertheless, it can be modeled by using the Single Table Approach because in that case we do not need foreign keys and can model the uncertainty of a supertype by tuple uncertainty or block uncertainty instead.

Example 42 *For illustration we consider the possible worlds representation of the two probabilistic databases pdb_2 and pdb'_2 that are presented in Figure 3.21(a) and Figure 3.21(b) respectively. The world schemas of both databases are defined as in the previous example, i.e. the world schema of the first database has been designed by using the Vertical Partitioning Approach and the world schema of the second database has been designed by using the Single Table Approach. Both databases are information equivalent, because they model both the information that entity $p1$ is either a person and a student, is only a person but not a student, or is not a person at all. The possible worlds representation of pdb_2 cannot be represented by a BID-database, because we cannot model the correlation that the existence of tuple t_2 requires the existence of tuple t_1 (recall foreign key references to maybe blocks are not allowed in BID-databases). In contrast, the possible worlds representation of pdb'_2 can be represented by the*

BID-database that is presented in Figure 3.21(c). Note that the uncertain membership of p_1 *to the table* 'Person' *is modeled by the fact that the corresponding block is only maybe.*

As a consequence, if the database schema has been designed by using the Single Table Approach, the uncertain membership to a supertype can be modeled by setting the value 'false' in the corresponding membership attribute if the supertype is a subtype of another supertype or can be modeled by a maybe block if the supertype is the root of the considered inheritance hierarchy.

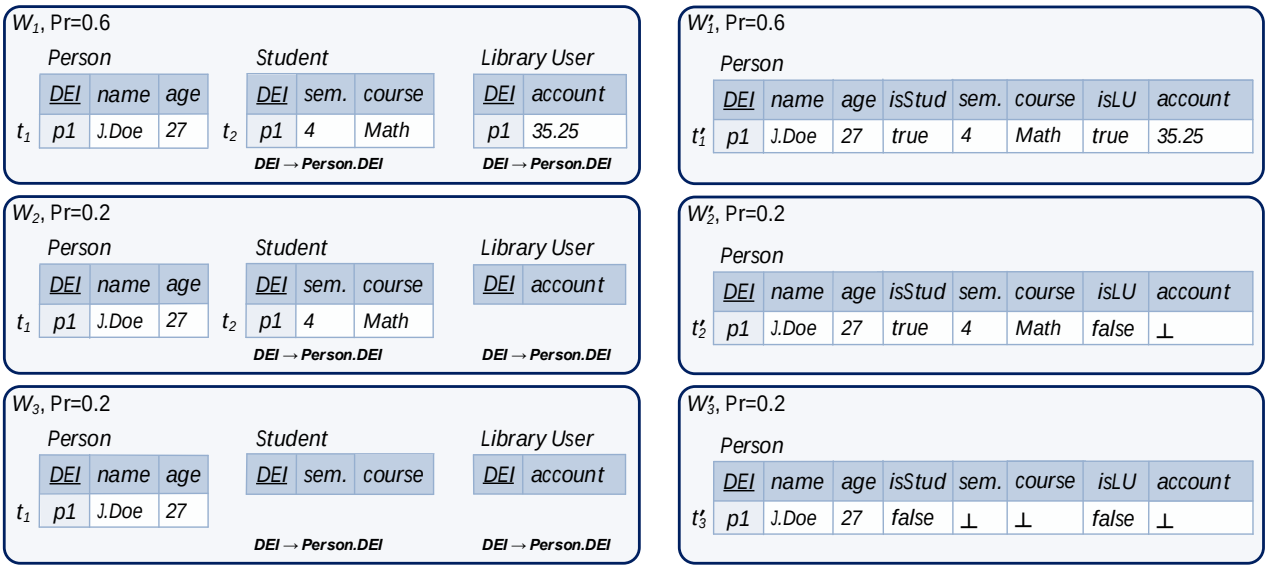
The last difference appears if we consider a situation where the membership of an entity to one subtype is correlated to another subtype as for example a person whose membership to the group of students and whose membership to the group of library users is unknown, but we know that her membership to the library users implicates a membership to the group of students. Note that such a correlation must not be valid for all entities, but can be specific for an individual entity and therefore cannot be modeled by the schema itself. If we use the Vertical Partitioning Approach for schema design, both memberships are represented by tuples from different tables. As a consequence, we are not able to introduce the given correlation into the BID-database. In contrast, if we consider a schema that is designed by using the Single Table Approach, the given inclusion can be modeled by correlations between the attribute values of the same tuple.

Example 43 *For illustration we consider a concrete instance of the situation described above. For this purpose, we consider the the possible worlds representations of the two probabilistic databases* pdb_3 *and* pdb_3' *that are presented in Figure 3.22(a) and Figure 3.22(b) respectively. Note, the* 'isLU' *models the membership to the entity type* 'Library User'*). As it can be simply seen, the world schema of the first database has been designed by using the Vertical Partitioning Approach and the world schema of the second database has been designed by using the Single Table Approach. Whereas* pdb_3 *cannot be represented as a BID-database, a possible BID-database representation of* pdb_3' *is presented in Figure 3.22(c).*

Notice, by using the Single Table Approach even a representation of more complex membership correlations, e.g. given by a joint probability distribution, are possible. Actually, we are even able to introduce correlations between the attributes from different subtypes of the modeled inheritance hierarchy.

Modeling an inheritance hierarchy within a TI-database is only possible to a limited extent because we cannot introduce any kind of correlation. Even if we use the Single Table Approach we require a mutual exclusion between tuples to model uncertain memberships to subtypes. As a consequence, the only situation in which in an inheritance hierarchy can be modeled within a TI-database is if the memberships to all subtypes are automatically implicated by the membership to the supertype and we only require a single tuple to model all the uncertainty on the memberships to the individual types of the considered hierarchy.

Similar holds for AOR-databases and AOR?-databases. Of course, both systems are able to represent uncertainty of attribute values, but they are not able to model correlations between the values of several uncertain attributes. Such correlations, however, are required if the underlying schema is designed by using the Single Table Approach because the value of an attribute that originates from a specific subtype depends on the value of the type's corresponding membership attribute. For instance, in the above considered examples, the value in the attribute *'course'* always depends on the value in the attribute *'isStud'*.



(a) Possible worlds representation of pdb_3

(b) Possible worlds representation of pdb'_3

Person

	RK	DEI	name	age	isStud	sem.	course	isLU	account	p
t'_1	1	p1	J.Doe	27	true	4	Math	true	35.25	0.6
t'_2	2	p1	J.Doe	27	true	4	Math	false	⊥	0.2
t'_3	3	p1	J.Doe	27	false	⊥	⊥	false	⊥	0.2

(c) BID-database representation of pdb'_3

Figure 3.22.: Modeling correlations between the memberships to different subtypes of the same supertype in BID-databases

In conclusion, modeling an inheritance hierarchy in a probabilistic database essentially depends on the used representation system and depends on the used transformation approach. Whereas in pc-databases all transformation approaches can be used abundantly, a BID-database is more powerful if its world schema is designed by using the Single Table Approach than by using the Vertical Partitioning Approach.

Despite of the discussed benefits of the Single Table Approach compared to the Vertical Partitioning Approach, we think the latter is more suitable for graphical presentations. For that reason, we will always use the Vertical Partitioning Approach in the rest of this thesis and assume that in cases in which we consider a BID-database the concrete schema of this database is actually designed by using the Single Table Approach. In order to improve the graphical presentation of inheritance hierarchies further on we use the graphical notion of *is-a*-relationships to connect the entity tables that represent the subtypes with the entity table that represents the supertype. For instance, Figure 3.23 shows the BID-database from Figure 3.20(c) by using this enhanced graphical notion.

3.6.3. Sources of Probabilities

Probabilities are usually used to describe the likelihood that a particular event will occur in the future, or to describe the likelihood that a particular unknown event has already occurred in the past. Both cases

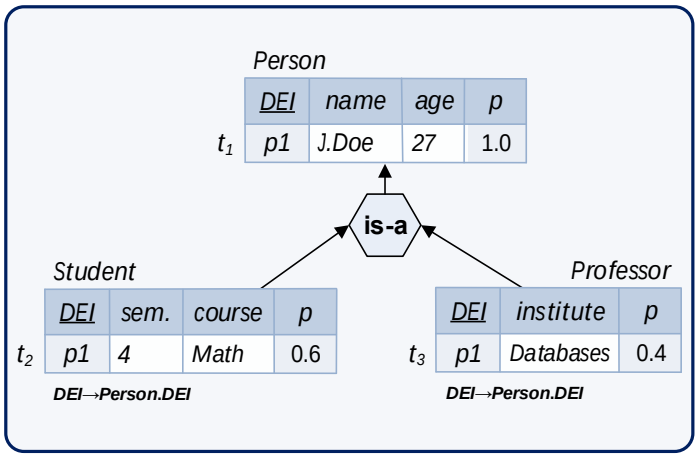


Figure 3.23.: Enhanced graphical notion of the BID-database pdb_1^* from Figure 3.20(c)

can be illustrated by rolling a dice. In the first case we want to role a dice and try to predict the result beforehand. In the second case, the dice has already been rolled, but its result is still unknown to us (maybe the dice still lies below the dice box) and we want to estimate it.

In both cases, the probabilities can be estimated in several ways. The way of estimation in turn depends on the known context information. In general, the less information is available the more vague and subjective the defined probabilities become.

- **Mathematical Probabilities (mathematical models):** In the most conventional meaning, probabilities are related to mathematical models and hence can be derived from them. In this case, a probability is called a *mathematical probability* [Eis69] or a *theoretical probability* [Por94, LeB04, Ros11]. and is typically defined as the number of ways the considered event can occur divided by the to the total number of ways any event can occur.

  Example 44 *For illustration, we assume a dice with six sides that is well-shaped (all sides are equally large and each edge is rectangular) and where each side presents another number between 1 and 6. Under this condition, the probability that casting the dice results in a six can be exactly computed to* $1/6$*, because each of the six numbers is verifiable equally likely.*

  Although a mathematical model enables a computation of exact probabilities, the accuracy of these probabilities can be vague because the used model often does not perfectly fit to the considered situation or it is based on a simplified situation. Moreover, this way of probability computation is not always applicable, because in many situations a corresponding model is not known or simply does not exist.

- **Empirical Probabilities (statistical analysis):** Another way to estimate probabilities is to consult the past (the known) to make predictions on the future (the unknown). This way is usually used if no mathematical model is available and the considered future (unknown) event is closely related to a set of past (known) events. Obviously, the closer the relations between the individual events and the more related past (known) events are available, the more exact become the made predictions and hence the more exact become the estimated probabilities.

Example 45 *For illustration, the chances of recovery of a cancer patient is usually predicated by consulting statistics on patients with similar properties as gender or age that have been suffered by the same type of cancer. In this case, the likelihood that the patient will recovery is estimated as the proportion of similar cancer patients that have been successfully cured from their diseases.*

This way of probability computation corresponds to an empirical observation that is made on the possible outcomes of an experiment by performing a number of trials. In this case, a probability is typically called an *empirical probability* [LeB04, Ros11] or an *a-posteriori probability* [AM74].

- **Subjective Probabilities (degrees of belief):**

  A third way of estimating probabilities is based on the intuition and experience of peoples. Since every person has another intuition and every person has another experience, probabilities that are estimated in this way are usually denoted as *subjective probabilities* [Hub06].

  Example 46 *For demonstration, let us assume that we have to estimate the likelihood that a stranger man is older than forty years. Although we do not know any mathematical model that can be used to derive this likelihood from the properties of his outer appearances as hair color, number of skin folds, or style of clothing, and also we are not able to perform several trials on highly similar events at hand, we can make an estimation of that likelihood because in our lifetime we have already seen many peoples in different ages with different clothing styles and we are able to conclude correlations between the different properties and a person's age by ourselves.*

  Actually, this way of estimating probabilities is a mix of using a mathematical model and of using known results of past related events, because experience is nothing else than information on past events and logical reasoning is nothing else than applying a mathematical model. Nevertheless, because each person has another experience and each person has another way of thinking, the accuracy of the estimated probabilities becomes more vague and becomes more subjective than by using an exact mathematical model or by objectively performing an empirical study on a large set of similar events.

Obviously, the meaning of probabilities rather depends on the way these values have been produced than on the way they will be used. For this reason, a data consuming application should always be aware of the sources from which the processed probabilities originate in order to be able to verify the data's suitability for its consuming purpose. Nonetheless, the possible query answers are usually ranked by their probabilities in decreasing order and in this case the order of these answers is often more informative to the user than their actual probabilities.

Since the meanings of probability values depend on the way these values have been produced, it is important to note that if probabilities do not originate from the same model or if probabilities do not originate from the same empirical study, these probabilities can be conflicting, i.e. the sum of these probabilities can be larger than one although they describe mutually exclusive events, they possibly do not sum up to one even they describe jointly exhaustive events, or the probability of an event is lower than the probability of a second event that implicates the first.

Example 47 *For illustration, at the beginning of the basketball season we ask people from different fan sets about the likelihood that their team will win the season's championship. Due to the evaluations are made independently, it can happen that the fans of the Miami Heat will estimate the likelihood to become the champion to* 60%*, but the fans of the Los Angeles Lakers will do the same. However, since only one team can become the champion, these probabilities are in a conflict.*

In duplicate detection, decisions on individual entity pairs are usually made independently before combining them to a single clustering. In reality, however, the decision on one pair can influence the decision on another pair because real-world identity is a transitive relation. As a consequence, if we compute the probabilities for the duplicate decisions in a pairwise fashion, these probabilities can be conflicting and we have to resolve these conflicts in a later detection phase.

3.6.4. Point Probabilities vs. Interval Probabilities

In the previous sections of this chapter, we considered probability values as certain information and hence represent them by a single numerical value. Probabilities of this kind are called point probabilities [SD09]. However, many sources of probabilities are rather vague than certain and modeling uncertainty in the probabilities itself seems to be the logical consequence. To model such kinds of uncertainty, some probabilistic data approaches [LLRS97, ELW01, RSG05, MM08a] use interval probabilities instead of point probabilities. By doing so each probability is represented by a lower bound and an upper bound instead of a single value.

Barbara et al. [BGMP92] extent their probabilistic data model by specific '*' values in order to represent missing probabilities, i.e. situations where the probability distribution is only partially known or cannot be exactly specified, inside the database. An incorporation of uncertainty on probabilities into an uncertain database is also discussed in [AW10]. Nevertheless, using interval probabilities (and hence using any model for uncertain probabilities in general) makes processing a probabilistic database much more complex and sophisticated. For this reason, we abstain from using interval probabilities in this thesis and always assume that concrete probabilities are available.

3.6.5. Continuous Probabilistic Data

In this thesis we restrict to probabilistic databases that each represents a finite number of possible worlds. Consequently, we only consider discrete probability distributions. In many use cases, however, the resultant probability distributions are not discrete, but continuous. For that reason, several research on probabilistic data focus on the modeling and processing of continuous probabilistic data [SMM+08a, KK10, JXW+11, CKP04, FRC11, ACK+11]. In these models continuous distributions are often restricted to single attribute values as for example the speed of a car or the position of a moving object. Nonetheless, the most research on probabilistic databases consider finite sets of possible worlds. Since these databases already pose a variety of challenges, we adopt this restriction in this thesis and therefore only consider probabilistic databases that represent finite sets of possible worlds.

3.6.6. Meaning of Null Values in Uncertain Databases

Over decades several semantics have been defined for null values. The three most commonly used semantics are *'inapplicable'*, *'applicable, but unknown'*, and *'unknown whether applicable or not'* (also known as *'no information'* semantics) [Kle01, LN06]. Note, whereas the last two semantics represent incomplete information, the first does not. In theory, we do not need null value semantics that represent any kind of incomplete information in an uncertain database, because we can explicitly represent that information by the concept of possible worlds. In reality, however, we need null values with incomplete information semantics even in uncertain databases, because:

- the number of possible values that is required to represent incomplete information on a particular attribute value can be extremely large in domains that are only less restricted and in cases where we only have less information available. For instance, it is unclear which set of possible values should be used to model the unknown name of a person, because her name can theoretically be any possible string. If the considered domain is not discrete, the number of possible worlds would become even infinite and a modeling of infinite world spaces is not provided by the most probabilistic representation systems.

- incomplete information can implicate missing information on probability distributions. Thus, some lacks of information cannot be represented by point probabilities, but actually require any concept to represent missing or imprecise probabilities like interval probabilities instead.

  For demonstration, to model the semantics *'applicable, but unknown'* in a categorical attribute A that has the domain $dom(A)$, we actually need to associate each of the possible values $v \in dom(A)$ with an interval probability $[0, 1]$, because we do not know which of them is how likely. As a consequence, if the used representation systems is restricted to point probabilities, we have to assume some information that is actually not available. The closest semantics to *'applicable, but unknown'* that can be represented by point probabilities is to use an uniform probability distribution on all domain elements and hence to assign each possible value with the probability $1/|dom(A)|$. This assignment, however, can lead to incorrect conclusions, because the query evaluation mechanisms assume that these probabilities are precise.

For both reasons null values are usually used in probabilistic databases to express incomplete information that cannot be represented by a simple probability mass function. Moreover, in uncertain databases null values are typically handled as in conventional databases. This means that they are not resolved for constructing a possible world, but instead remain in the possible worlds as null values. According to the possible worlds semantics, a set of possible worlds is queried by querying each of these worlds separately. In addition, each possible world is queried like a conventional database.

As a consequence, let Q be a query that selects all tuples that have the value 'x' in attribute *'A'*, and let t be a tuple that has the null value in attribute *'A'* in every possible world. By using the three-valued logic of relational databases the condition $A =$ 'x' is evaluated to UNKNOWN for t in each of these worlds. Thus t is not selected in any of these worlds and therefore is not a possible query answer although it cannot be excluded for sure that this tuple actually has the value 'x' in the selected attribute. Obviously, this approach is a contradiction to the idea of returning all possible query answers, because t possibly

satisfies the query condition. However, it is not possible to compute the probability to which t satisfies the query condition. For this reason, probabilistic databases usually query null values in the same way as conventional databases do (in most research on probabilistic databases, the challenges that result from using null values are actually ignored).

3.6.7. Further Models for Uncertain Data

Although we consider uncertain databases to be incomplete or probabilistic in this thesis, other concepts have been used to model uncertainty within a database. A common way for modeling imperfect information is to use possibility theory instead of probability theory and hence to utilize the concept of fuzzy sets [Zad65, Zad78, PG98] to model uncertainty, imprecision, or vagueness inside a database [ZK84, SM90, UF94, MPM94, Pet96, GUP06]. The use of fuzzy terms, e.g. possibility distributions that are represented by words like 'young' or 'large', enables a direct incorporation of imprecision that is inherent in natural languages into the data.

Lehrack et al. [Leh09, DLKS10, LSS12] uses quantum logic to query uncertain databases in a natural imprecise way. Gatterbauer et al. [GBKS09] introduce the concept of *belief databases* in order to capture individual degrees of believe on the stored information by annotating the operational data (or belief annotations) with belief statements. The Dempster-Shafer theory [Sha76] is another concept that has been used to model uncertain information inside a database [LSS96b, AW10].

In this thesis, we restrict our consideration to probabilistic extensions of the relational data model. Nevertheless, other data models can be adapted to a representation of uncertainty as well. For instance, probabilistic representation systems that are based on XML have been considered in several works [NJ02, HGS03, HGS07, vKdKA05, dKvK08, BKOS10, KS13]. Senellart et al. [KNS10, ACK+11, SS13] extensively study querying and updating probabilistic XML data. Hollander and van Keulen [HvK10] as well as Amarilli and Senellart [AS13] build connections between relational probabilistic representation systems and XML based probabilistic representation systems. Probabilistic deductive databases have been elaborated by Kiessling et al. [KTG92] and by Lakshmanan and Sadri [LS94a, LS94b, LS97, LS01]. Eiter et al. [ELLS01] research on probabilistic object bases. Probabilistic temporal databases have been studied by Dyreson and Snodgrass [DS98] and Subrahmanian et al. [DRS01, Sub09]. A straightforward way to incorporate probability theory into a database is to use a well-studied probabilistic graphical model [Pea88, CDLS99] as Bayesian Networks [BG07a] (directed) or Markov Networks [KS80] (undirected). Sen and Deshpande [SD07, DGS09] propose a probabilistic database approach that is based on such graphical models.

3.6.8. Managing Uncertain Data

In recent time a large amount of concepts have been adapted from certain relational databases to uncertain relational databases in order to enable an appropriate uncertain data management. The meaning of functional dependencies in probabilistic databases has been examined in [DK10a]. Schema design, i.e. normalization with respect to functional dependencies, has been considered by Sarma et al. [SUW09]. Approaches for indexing uncertain databases has been proposed by several parties [CXP+04, TCX+05, LS07, PSS09, KMZ10]. Techniques for efficiently evaluating joins on dis-

crete probabilistic databases as well as continuous probabilistic databases are surveyed by Kriegel et al. [KBRZ09]. The meaning and evaluation of skyline queries [PJLY07, BFO+09, KML10], nearest-neighbor queries [RCK+10, NZE+13], similarity joins [KKPR06, LC11b], similarity searches [XZTY12], or ranking [HPZL08, LSD09, CLY09, IS11] has been adapted to uncertain databases in several works. Materialized views have been adapted to probabilistic databases by Dalvi et al. [DRS11]. Managing probabilistic data streams is considered in [CG07b, AY08a, RLBS08, Vee09, ZC13].

Several concepts of data warehousing and data mining has been adapted to probabilistic data, too. Warehousing of uncertain data has been considered by Burdick et al. [BDJ+07, BDJ+06] and Mokhtar [Mok11]. Moreover, approaches for frequent item set mining [CKH07, ALWW09, WCC+12, BCC+13], clustering [KP05, NKC+06, Agg09, JPTL13], classification [QXPT09, SSRS10], or outlier detection [AY08b, JP11] in probabilistic databases have been developed in recent years.

3.6.9. Uncertain Database Management Systems

Besides MayBMS[11] [HAKO09, Koc09] and Trio[12] [Wid09, ABS+06, MTdK+07] several incomplete/probabilistic database management systems (with discrete probability distributions as well as continuous probability distributions) have been developed in recent years. Whereas MystiQ[13] [BDM+05, RS08], PossDB[14] [GOT13b, GOT13a], Sprout[15] [OHK09, FHOR11], BayesStore[16] [WMGH08], and PrDB[17] [SD07] are based on discrete probability distributions, Orion[18] [SMM+08b, SMM+08a] (originally called as U-DBMS [CSP05]), Pip[19] [KK10], and MCDB [JXW+08, AJP+10, JXW+11] are also designed for managing uncertainty that is modeled by probability density functions.

To represent correlations between tuples, PrDB and BayesStore use probabilistic graphical models that are very popular in the statistics and machine learning communities. PossDB is a native implementation of c-tables. MystiQ is able to evaluate uncertain query predicates on uncertain databases. ProbView [LLRS97] was an early probabilistic database system that works on a finite number of possible worlds, but associates interval probabilities to these worlds instead of point probabilities. ProQua[20] [LSW13, LSS12] is a probabilistic data management system that couples the concept of uncertain databases with quantum logic.

[11] http://maybms.sourceforge.net/
[12] http://infolab.stanford.edu/trio/
[13] http://homes.cs.washington.edu/~suciu/project-mystiq.html
[14] http://triptych.encs.concordia.ca/PossDB
[15] http://www.cs.ox.ac.uk/projects/SPROUT/
[16] http://www.eecs.berkeley.edu/Research/Projects/Data/102060.html/
[17] http://www.cs.umd.edu/~amol/PrDB/
[18] http://orion.cs.purdue.edu/
[19] http://maybms.sourceforge.net/pip/index.html
[20] http://dbis.informatik.tu-cottbus.de/ProQua/

Chapter 4

Duplicate Elimination

This chapter gives an overview on the principles of duplicate elimination in general and duplicate detection as well as duplicate merging in particular. It especially shows the variety of existing approaches for duplicate detection in certain databases and therefore is intended to illustrate the great challenge of finding an appropriate configuration for a duplicate detection process with respect to a specific application scenario. We start with a formal discussion on all three concepts in Section 4.1 and shortly present the application fields of data cleaning and data integration in Section 4.2. Then we describe the problem of detecting duplicates in certain relational databases, examine the individual phases of a conventional duplicate detection approach in more detail, and present some information on related aspects in Section 4.3. Finally, we conclude this chapter with a short consideration on duplicate merging in Section 4.4.

4.1. Problem Description

A database can be considered as a digital image of the real world. (or a part of the real world respectively). Due to errors in data acquisition and data maintenance, there are usually some divergences between the real world and the database's image of this world. One type of divergence appear if a single real-world entity is mistakenly presented in the database for multiple times

As illustrated in Figure 4.1, a database maps real-world entities to database entities.

Definition 21 (Real World Mapping): *Let $\mathfrak{W}$ be the set of all real-world entities and let db be a database with the extension Ext(db), the mapping ω: Ext(db) $\rightarrow 2^{\mathfrak{W}}$ maps each entity of db to some entities of $\mathfrak{W}$.*

In a clean database the mapping ω is 1:1, i.e. each real-world entity is mapped to exact one database entity and vice versa. However, due to the above mentioned errors in data acquisition and data maintenance, a real-world entity can be mapped to several database entities and in turn a database entity can mistakenly combine information that actually belongs to different real-world entities. A resolution of the latter is a very difficult task and is not in the focus of this thesis. For that reason, in the rest of this thesis we assume that each database entity maps to exact one real-world entity. A database is *duplicate-free*, iff the mapping ω is injective w.r.t. the database's extension, i.e. all entities of this database are mapped

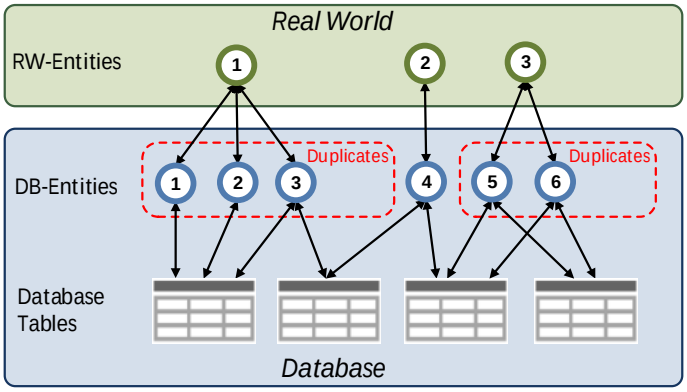


Figure 4.1.: Mapping from database entities to real-world entities

to different real-world entities. Two database entities e_r and e_s are called duplicates and are called to be real-world equivalent (notation $e_r =_{\text{id}} e_s$), iff $\omega(e_r) = \omega(e_s)$.

Duplicate elimination [LLL01, SB04] (also known as deduplication [Chr11, SB02, ARS09] or entity resolution [Chr12, Tal11, BG07b, BGMM+09, BBKL12, GM12, GM13]) is the process of removing duplicate entities from a database so that the database becomes duplicate-free. A duplicate elimination process is usually decomposed into two subphases: *duplicate detection* and *duplicate merging*.

Duplicate detection [EIV07, NH10] (also known as record linkage [NK62, FS69, Jar89, Win02, BG04, HSW07], object matching [DLLH03, ZSC10], or data matching [Chr12]) is the process of identifying entities in a database that refer to the same real-world entity. Since we consider the concept of indeterministic duplicate detection in this thesis, we will refer to conventional duplicate detection approaches as *deterministic* ones. Since identity is a transitive relation, a deterministic duplicate detection process is a partitioning of the database entities into clusters (equivalence classes) such that all database entities of one cluster are assumed to refer to the same real-world entity.

Definition 22 (Deterministic Duplicate Detection): *Formally, a deterministic duplicate detection process is a function δ_{det} that maps a set of database entities $\mathfrak{E} = \{e_1, \ldots, e_m\}$ to a disjoint clustering $\mathcal{C} = \{C_1, \ldots, C_n\}$ such that $\bigcup \mathcal{C} = \mathfrak{E}$ (each entity is assigned to a cluster) and $(\forall C_p, C_q \in \mathcal{C})\colon C_p \cap C_q = \emptyset$ (the clusters are disjoint). The duplicate detection process is considered to be* perfect, *iff:*

- $\forall C \in \mathcal{C}\colon \forall e_r, e_s \in C\colon \omega(e_r) = \omega(e_s)$, *i.e. all database entities from one cluster represent the same real-world entity (all detected duplicates are true duplicates and hence the number of false positives is zero).*
- $\forall C_p, C_q \in \mathcal{C}\colon \forall e_r \in C_p\colon \forall e_s \in C_q\colon C_p \neq C_q \Rightarrow \omega(e_r) \neq \omega(e_s)$, *i.e. all database entities from different clusters represent different real-world entities (all true duplicates are detected and hence the number of false negatives is zero).*

Duplicate detection can become extremely difficult, if the considered database is corrupted by many errors, is corrupted by serious errors, or if the database is highly incomplete, i.e. many attribute values are missing. We will extensively discuss the challenges of duplicate detection as well as existing solutions to these challenges in Section 4.3.

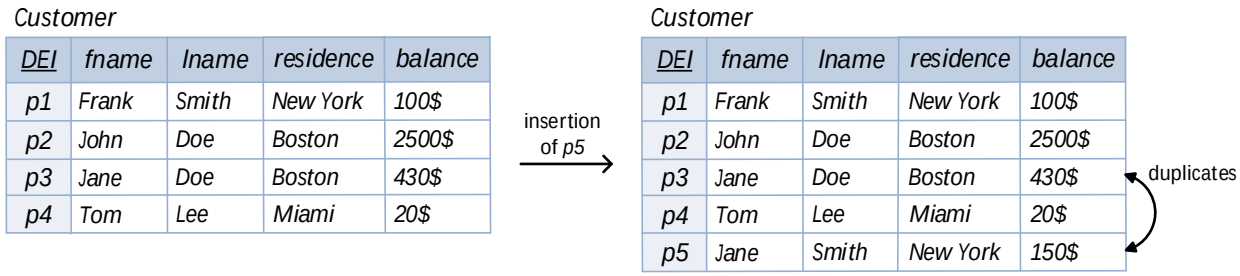


Figure 4.2.: Inserting a duplicate into a customer database

In the duplicate merging phase, all the database entities of one duplicate cluster are merged to a single representation by using a merge function f_μ that maps a set of database entities $\mathfrak{E} = \{e_1, \ldots, e_m\}$ to a single database entity e.

Definition 23 (Duplicate Merging): *Formally, a duplicate merging process is a function μ that maps a clustering of database entities $\mathcal{C}$ to a set of database entities $\mathfrak{E}$ by using a merge function f_μ that merges the entities of each cluster in $\mathcal{C}$ to a single entity, i.e. $\mu(\mathcal{C}) = \{f_\mu(C) \mid C \in \mathcal{C}\}$.*

Duplicate merging becomes especially a challenge, if the attribute values of the detected duplicates contradict each other and it is unclear which of them represents reality best. Detailed information on duplicate merging is presented in Section 4.4.

By using Definition 22 and Definition 23 a deterministic duplicate elimination process can be defined as a composition of a duplicate detection process and a duplicate merging process.

Definition 24 (Deterministic Duplicate Elimination): *Formally, a deterministic duplicate elimination process is a function ϵ_{det} that maps a set of database entities $\mathfrak{E}_1 = \{e_1, \ldots, e_m\}$ to a second set of database entities $\mathfrak{E}_2 = \{e'_1, \ldots, e'_n\}$ with $n \leq m$. Let δ_{det} be a deterministic duplicate detection process and let μ be a duplicate merging process that uses the merge function f_μ, the deterministic duplicate elimination process $\epsilon_{det} = \mu \circ \delta_{det}$ on a set of database entities $\mathfrak{E}$ is defined as: $\epsilon_{det}(\mathfrak{E}) = \mu \circ \delta_{det}(\mathfrak{E}) = \{f_\mu(C) \mid C \in \delta_{det}(\mathfrak{E})\}$.*

4.2. Application Fields

Duplicate elimination is primarily used in two contexts: data cleaning and data integration.

4.2.1. Data Cleaning

The purpose of data cleaning [GFS+01, LN06, Chr12, GS13] is to eliminate errors from the database and hence to increase the database's quality. A variety of cleaning tasks focus on the correction of small errors as typos or adapt the data values to standards and conventions. Since the most of these tasks also improve the effectiveness of a duplicate detection process we will discuss them in more detail in Section 4.3.3. In a single database, duplicate database entities are primarily caused by mistakenly inserting a representation of the same real-world entity for several times.

Example 48 *A simplified example is illustrated in Figure 4.2. A customer database initially contains the account information of four different customers of an e-commerce company (for space reasons we assume that the account number corresponds to the DEI). After some time, one of these customers, Jane Doe, want to make a new order. Due to several reasons, she does not know her account number anymore or forget that she already has an account for that company. Moreover in the meantime she has married and therefore changed her last name and her residence. As a consequence, the system does not recognize that this customer is already contained in the database and inserts a new database entity p_5 that is a duplicate to the existing entity p_3.*

In the presented scenario of a Customer Relationship Management*, the risk of duplicate entities is especially high because customers often forget that they already have an account on the corresponding system and therefore often create a new customer account when they want to make a new order. Moreover, having multiple accounts can be deliberated by the customer as for example for criminal reasons or privacy reasons.*

This example perfectly illustrates the challenge in duplicate detection because duplicate database entities such as p_3 and p_5 can more dissimilar than non-duplicate database entities such as p_2 and p_3. To reduce the number of duplicates that result from erroneous insertions into the database, an identity management system can be used (see Section 4.3.10.3). This system is able to store information on past detection results and is able to capture changes of data values such as those changes that are caused by marriage or are caused by house moving.

4.2.2. Data Integration

The second field of applications that require duplicate elimination techniques is the field of data integration [Len02, LN06, HRO06, BJ07, HDI12]. Data is often stored decentralized and is not completely managed by a single owner, but is distributed around the world and managed autonomously by different owners instead. Since the number of data sources that are connect by the internet increases from year to year, it becomes more and more worthwhile to meaningfully combine data that originate from different sources. This integration can be done physically or can be realized virtually. Physical data integration means that the data are extracted from the sources and are physically stored in a single target database system. By doing so, any kind of data heterogeneity has to be resolved by the extraction process and database queries can then be posed to the target system in a conventional manner. Popular examples for such data integration systems are data warehouses [Leh03, KE99, UGMW02, Vos08] that are designed to analyze historical data extracted from several sources. In contrast, virtual data integration means that the data remain at the sources (and hence are still managed autonomously) and are only integrated into the target system at query time. Of course, this makes query evaluation more challenging than in a physical integration system. However, all the data that is used for answering a query are up to date.

In data integration, duplicate database entities appear because of the autonomy of the integrated sources. Moreover, duplicate detection and duplicate merging can become more challenging, due to the different conventions of the individual sources.

Since the source data can be represented in different schemas or can be represented even in different data models, before integrating these data a schema matching need to be performed [RB01, HMH01,

CustomerS1

DEI	fname	lname	residence	balance
p1	Frank	Smith	New York	100$
p2	John	Doe	Boston	2500$
p3	Jane	Doe	Boston	430$
p4	Tom	Lee	Miami	20$

(a) Data source S_1

CustomerS2

DEI	lname	fname	residence
p5	Smith	Frakn	Albany
p6	Brown	Lea	Paris
p7	Lee	Tom	Florida
p8	Li	Wu	Seattle

(b) Data source S_2

CustomerS3

DEI	name	balance
p6	Lea Brown	€ 0
p9	Tom Leee	€ 30
p8	Wu Li	€ 1200
p10	Doe, Janet	€ 380

(c) Data source S_3

CustomerResult

DEI	fname	lname	residence	balance	lineage
p1	Frank	Smith	New York	100$	S1
p2	John	Doe	Boston	2500$	S1
p3	Jane	Doe	Boston	430$	S1
p4	Tom	Lee	Miami	20$	S1
p5	Frakn	Smith	Albany	⊥	S2 × ⊥
p6	Lea	Brown	Paris	€ 0	S2 × S3
p7	Tom	Lee	Florida	⊥	S2 × ⊥
p8	Wu	Li	Seattle	€ 1200	S2 × S3
p9	Tom	Leee	⊥	€ 30	⊥ × S3
p10	Doe	Janet	⊥	€ 380	⊥ × S3

(d) Integration result without duplicate elimination

DEI	fname	lname	residence	balance
p1	Frank	Smith	New York	100$
p5	Frakn	Smith	Albany	⊥
merge:	?	Smith	?	?

DEI	fname	lname	residence	balance
p3	Jane	Doe	Boston	430$
p10	Doe	Janet	⊥	€ 380
merge:	?	?	?	?

DEI	fname	lname	residence	balance
p4	Tom	Lee	Miami	20$
p7	Tom	Lee	Florida	⊥
p9	Tom	Leee	⊥	€ 30
merge:	Tom	?	?	?

(e) Duplicate database entities

Figure 4.3.: Example for the emerge of duplicates in the integration of three data sources

Gal06b, BBR11]. Furthermore, in a complex integration system queries do not only combine single tables by using the set union operator, but can be decomposed into subqueries that are each posed to another source and then their results are combined to the result of the original query by joins or even more complex data operations. For that reason, there are two aspects where detecting real-world equivalence becomes crucial: (a) in combining subquery results that originate from different data sources that do not share a common database identifier, and (b) in cleaning the final query result from duplicate answers. For combining/joining tables from different sources during query evaluation instead of an expensive of duplicate detection process usually a cheaper similarity join technique is used (see Section 4.3.10.1). A complete duplicate elimination process is then only applied to the final result. Obviously, besides the duplicates that are caused by combining the entities from the different sources, each of the integrated sources can contain duplicate entities by itself.

Example 49 *An integration example with three sources is illustrated in Figure 4.3. Let us assume an integration query that returns the names of a customer, his residence, and his balance. Source S_1 (Figure 4.3(a)) contains all of these data, but source S_2 (Figure 4.3(b)) and source S_3 (Figure 4.3(c)) contain only a part of it. One option is to simple query S_1. This result, however, would not be as complete as possible, because the other two sources contain customers that are not provided by source S_1. Therefore,*

it can be useful to pose a more complex query that first performs a join between the sources S_2 and S_3 and then combine the join result with the data of source S_1 by using the set union operator. Since source S_3 has a different schema, before joining it with S_2 we need to split the attribute 'name' *into the two attributes* 'fname' *and* 'lname'. *The query therefore results in:* $Q = S_1 \cup (S_2 \rtimes split_{name \rightarrow fname,lname}(S_3))$ *where* $split_{name \rightarrow fname,lname}$ *is the aforementioned split operation and* $\rtimes$ *is a full-outer join. The query result is depicted in Figure 4.3(d) and contains ten database entities (tuples). Note that we assume that all databases use the same database identifier, namely the DEI, and hence joining entities from different sources does not require a kind of similarity join in this example. Nevertheless, although all sources use the DEI to identify their customers, some customers have been modeled by different DEIs in the different sources and the final result is corrupted by duplicates. These duplicates are listed in Figure 4.3(e). The challenge of duplicate detection is to identify these duplicates despite of data errors like misspelled names (e.g. 'Frank' vs. 'Frakn') or value transpositions (e.g. the values first name and the last name of entity* p_{10} *are transposed), despite of data heterogeneity that results from using distinct aggregation levels (e.g. 'Miami' vs. 'Florida') or that results from using distinct units of measurement (e.g.* \$ *vs.*)€*, and despite of missing data (e.g. 'Miami' vs.* $\perp$*). The challenge of duplicate merging is to select a value for each attribute despite the contradictions provided by the duplicate database entities. For instance, it is not clear which value should be used for the attributes* 'residence' *in the merge of the three database entities* p_4*,* p_7*, and* p_8*. Moreover, sometimes an appropriate merge depends on the considered semantics. For example, it is unclear if the values in the attribute* 'balance' *that are provided by the different sources represent the same set of transactions (or at least overlapping sets) or if they represent values that result from different sets of transactions. Whereas in the first case the merge result should be close to at least one of the provided values, the merge result should be computed by their sum in the second case.*

More information on data integration, especially on resolving schema heterogeneity and on query evaluation in virtual integration systems, can be found in the books of Leser and Naumann [LN06] and Halevy et al. [HDI12].

4.3. Duplicate Detection

In this section, we survey existing research in duplicate detection in relational databases. For that purpose, we start with the problem description in Section 4.3.1, give an overview on the whole detection process in Section 5.2.3, and then consider each of the detection phases in more detail in Section 4.3.3 to Section 4.3.7. Finally, we discuss measures to evaluate the quality of duplicate detection results in Section 4.3.8, existing research on duplicate detection in multi-table databases in Section 4.3.9, and consider several topis that are closely related to duplicate detection in Section 4.3.10.

4.3.1. Problem Description

Designing an appropriate duplicate detection process has two goals:

(i) **Effectiveness:** The resultant duplicate clustering should be as close to the perfect clustering as possible.

(ii) **Efficiency:** The detection process should have a low runtime and a low storage requirement.

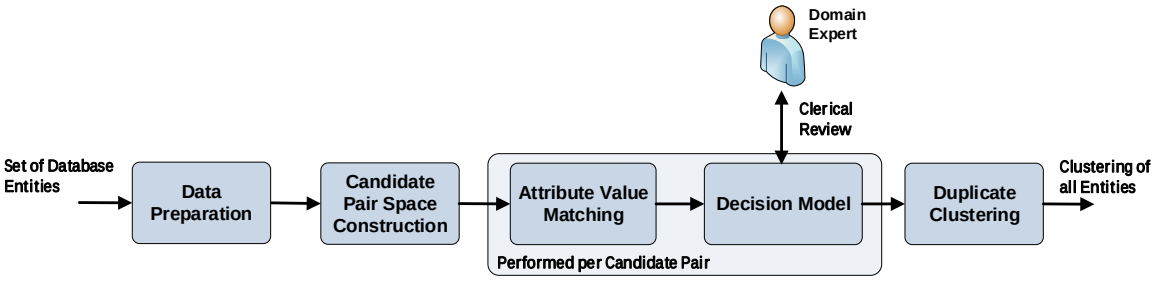


Figure 4.4.: The main phases of a process for matching database entities

It can be easily seen that these two goals are contradictory, because a higher effectiveness usually implies a larger runtime and more storage required. Thus, the suitability of a duplicate detection process is always a trade off between these two goals and mainly depends on the considered application scenario. For example, an online performed duplicate detection process as it is used in a virtual integration process must be efficient in terms of time and storage, because no user is willing to wait several hours for her query answer and a lot of queries are answered simultaneously. In contrast, in a physical integration process, e.g. a one time fusion of the databases of multiple companies, or in a data cleaning process it is usually not a problem to perform duplicate detection offline in several hours or even several days. Moreover, a lot of storage can be explicitly reserved for this process so that it does not negatively influence other running applications and processes. Since the requirements of efficiency are less restrictive in such integration processes, more effective duplicate detection processes can be used or are even required.

4.3.2. Detection Process

Most duplicate detection approaches only consider a detection of duplicates within a single database table. For that reason, we will first present the state of the art on detecting entities in a single entity table where each database entity is represented by a single tuple. For that reason, we consider the concepts of database entities and tuples interchangeably in this section. Existing extensions to multi-table databases are then discussed in Section 4.3.9.

Conventional approaches for duplicate detection [NH10, Chr12] are based on subsequently performed pairwise comparisons of database entities and therefore are called to be *iterative* [NH10, BG04, LF05b]. Iterative duplicate detection approaches consist of an extraction phase and the five main phases that are presented in Figure 4.4. The input to the extraction phase is a database from which a set of database entities is extracted. The output of the last phase is a partitioning of this entity set. We now give a short overview about the individual phases and then, exept the trivial extraction phase, consider each of these phase in more detail in the following sections.

1. **Data Extraction:** Obviously, some attributes are more suitable to detect duplicates than others. Moreover, comparing attribute values is one of the most expensive procedure in duplicate detection. For that reason, duplicate detection processes usually do not use the complete tuples as detection input but only extract some of their attributes. Naumann et al. [NH10] construct an *object description* from the extracted attribute values and then match the database entities by matching

their descriptions. We will present such a description-based detection approach in more detail in Chapter 5 and consider each database entity to be represented by an ordinary tuple by then.

2. **Data Preparation/Data Cleaning:** After extraction, the extracted values are standardized (e.g., unification of conventions and units) and cleaned (elimination of easy to recognize errors) to obtain a homogeneous representation of all database entities.

3. **Candidate Pair Space Construction:** Since a comparison of all pairwise combinations of entities is mostly too inefficient, the search space is usually reduced by using heuristic methods such as the Sorted Neighborhood Method or any other blocking technique [Chr11]. The result of this phase is the *Candidate Pair Space* that is a set containing all entity pairs that are assumed to be duplicates. Note that all entity pairs that do not belong to this space are not matched and hence are automatically considered to be non-duplicates if the final clustering phase does not change them into MATCHES because of consistency reasons.

4. **Attribute Value Matching:** The similarity between two entities depends on the similarity between their attribute values. Despite data preparation, syntactic as well as semantic irregularities remain. Thus, the similarity between two valid elements of an attribute domain is quantified by syntactic (e.g. edit-based measures, token-based measures or hybrid measures [EIV07, NH10]) and/or semantic (e.g. glossaries or ontologies) means. From comparing two entities, we obtain a *comparison vector* $\vec{c} = \langle c_1, \ldots, c_n \rangle$, where c_i represents the similarity of the values from the i^{th} attribute.

5. **Decision Model:** The comaprison vector is then used as input for a decision model [GB06, EIV07] that classifies the considered entity pair into the set of MATCHES (duplicate pair) or the set of UNMATCHES (non-duplicate pair).

   The decision model can be internally divided into two (or three respectively) subphases. In the first subphase (*Similarity Computation*) a single similarity score per candidate pair is computed on the basis of the comparison vector. In the second subphase (*Classification*) a threshold is used to classify each candidate pair as a MATCH or an UNMATCH based on its similarity score. In semi-automated approaches, a third set of POSSIBLE MATCHES is introduced by using two thresholds instead of one. In this case, each candidate pair that has been classified as a POSSIBLE MATCH by the automatic detection process is then manually classified as a MATCH or an UNMATCH by domain experts. The process of manual assignment is typically called a *clerical review*.

6. **Duplicate Clustering:** Based on the decisions made for the individual candidate pairs, a globally consistent result is finally achieved by using a duplicate clustering technique, as for example the technique of connected components that computes the transitive closure of all detected MATCHES [NH10].

1.-5. **Evaluation/Verification:** In cases where the gold standard is available, e.g. in test runs that are used to identify an adequate process configuration, the effectiveness of the applied duplicate detection process can be evaluated in terms of recall, precision, F_1-measure, or any other evaluation

Person

DEI	name	DoB	city	country	phone	email
e1	Dejohn W. Tucker	1974-05-03	U.S.A.	Manhattan	0345233848	abc@me.cmo
e2	John Bill Tacker	5 march 1974	New York	U.S.	(0345)233848	tucker@xy.com
e3	Bill Tacker	1974-05-03	Albany	United States	⊥	tacker@xy.com
e4	John William Tucker	03.05.74	New York City	United States of America	0345-233884	⊥
e5	Tacker, Bill Thomas	03 may 1974	Albany	USA	233884(0335)	123@ab.com
e6	Bill T. Tacker	03.05.1975	NY-Albany	⊥	(0335)233884	btacker@xyz.com
e7	Lilou Zoe Lefebvre	23.09.1976	Paris	France	567312	lilou@me.fr
e8	Lilu Lefevre	23.09.1967	Paris-Sarcelles	French Republic	0221-567312	lileff@home.fr
e9	Phil T. Tacker	⊥	New York	U.S.A.	(0543)848233	ptacker@xyz.com
e10	Philip Tiberius Tacker	1965-05-06	NYC	U.S.A.	(0543)848233	ptt@ny.com

Figure 4.5.: Unprepared sample table

measure (see Section 4.3.8). If the computed effectiveness is not satisfactory, duplicate detection is repeated with other, better suitable thresholds or methods.

To illustrate the individual phases of a duplicate detection process we will use the sample table *'Person'* that is depicted in Figure 4.5. This table contains ten tuples, each representing another database entity. Besides the entity identifier, the table has six attributes. The values of the different entities are formatted in different standards and are corrupted in different ways.

4.3.3. Data Preparation/Data Cleaning

Databases are usually corrupted by errors and their values are often non-standardized which makes duplicate detection a though challenge. However, standardization can be obtained and many of existing data errors can be erased by simple preparation activities. Thus, the effectiveness of a duplicate detection process can be dramatically increased by preparing data first.

Some often used preparation activities are [LN06]:

- **Upper Case/Lower Case:** To enable a more effective matching of attribute values, letters are transformed into upper cases or lower cases respectively.
- **Discarding Stop Words and Stop Symbols:** To get a unified representation of attribute values, stop words (e.g. 'and', 'or', 'the') and stop symbols (e.g. '-','.',';') are discarded.
- **Spell Checking:** Errors in text data that consist of words from natural language (e.g. movie titles) can be removed by applying a spell checker to the data.
- **Removing Abbreviations:** Well known abbreviations are replaced by their full spelling (e.g. 'Av.' is transformed to 'Avenue', 'NY' is transformed to 'New York', or 'U.S.A.' is transformed to 'United States of America').
- **Format Standardization:** Formats are unified by choosing a standard representation form. This is especially useable for phone numbers or dates of birth (see Example 50).

- **Unification of Units:** Often data values are described in different units of measurement as for example some values of an attribute speed are defined in *kilometer per hour* and some other values are defined in *miles per hour*. In such cases, one of the units is selected and all data values that are modeled in another unit are recomputed to the selected unit.

- **Unification of Accuracy:** Besides different units, data can be also described in different degrees of accuracy, as for example some time values are accurate to two decimal places and some other values are accurate to five decimal places. In such cases, a re-computation to the lowest accuracy degree can be useful.

- **Unification of the Aggregation Level:** Values of one attribute can be defined on different aggregation levels. For example, an attribute *'residence'* can contain the values 'Hollywood', 'Los Angeles' and 'California'. Whereas 'Hollywood' refers to a district, 'Los Angeles' refers to a city and 'California' refers to a state. Such inconsistencies can be reduced by first choosing an adequate level of aggregation (e.g. city) and then by transforming all values from a lower aggregation level to the selected level (e.g. district to city). Remaining inconsistencies can be handled during attribute value matching by taking the different aggregation levels into account. For example, in matching a person living in Los Angeles with a person living in California, we can use the information that Los Angeles is a city in California. Of course, the aggregation level that is selected for unification should not be too coarse, because in that case a lot of information can be lost. For instance, if we select the level state as the aggregation level and thus transform all values that describe cities to their corresponding states, the difference that one person lives in Los Angeles and a second person lives in San Francisco is lost by the transformation.

- **Value Transpositions:** By comparing attribute values with other values of the same attribute transpositions between different attributes, for example *city*='France' and *country*='Paris', can be detected and repaired.

- **Reference List based Value Checking:** Reference lists can be used to prove the consistency of the given data values and to repair them if necessary. A simple example is a lists of all countries of the world. If a value of the attribute *'country'* is not contained in this list and if this value is not detected to be part of an attribute transposition, it can be assumed that this value is incorrect. Since many reasons for inconsistency are typos, the country list mostly contains an element that is very similar to the considered value and the given inconsistency can be repaired by replacing the incorrect value with the list's element that is most similar to that value.

- **Attribute Splitting/Merging:** In many cases, it is useful to split an attribute in a set of subattributes or to merge a set of subattributes to a single attribute. Splitting is often used for attributes of names where first names and last names are written in the same value (often separated by a delimiter) or for attributes of addresses where street names, zip-codes and cities are collectively stored in a single attribute, but matching becomes much more effective if they are stored in different attributes. Sometimes it is known that there exists a high number of transpositions between two attributes, but the concrete cases of transpositions are unknown. In that case it can be valuable to

Person

DEI	fname	lname	DoB	city	country	phone	email
e1	dejohn w.	tucker	1974-05-03	new york city	usa	(0345)233848	abc@me.com
e2	john bill	tacker	1974-03-05	new york city	usa	(0345)233848	tucker@xy.com
e3	bill	tacker	1974-05-03	albany	usa	⊥	tacker@xy.com
e4	john william	tucker	1974-05-03	new york city	usa	(0345)233884	⊥
e5	bill thomas	tacker	1974-05-03	albany	usa	(0335)233884	123@ab.com
e6	bill t.	tacker	1975-05-03	albany	⊥	(0335)233884	btacker@xyz.com
e7	lilou zoe	lefebvre	1976-09-23	paris	france	)567312	lilou@me.fr
e8	lilu	levefre	1967-09-23	paris	france	(0221)567312	lileff@home.fr
e9	phil t.	tacker	⊥	new york city	usa	(0543)848233	ptacker@xyz.com
e10	philip tiberius	tacker	1965-05-06	new york city	usa	(0543)848233	ptt@ny.com

Figure 4.6.: Prepared sample table

merge both attributes and then to use a set-based similarity measure for attribute value comparison (see Section 4.3.5.1).

Example 50 *For preparing the illustrating sample table from Figure 4.5, we convert all string values into lower case. Furthermore, we split the attribute* 'name' *into the two attributes* 'fname' *(first name) and* 'lname' *(last name). We standardized the data on the date of birth and the phone number by using a unique representation style for both attributes. In the attribute* 'city' *we replaced district names by their corresponding city names ('Manhattan' → 'New York' and 'Paris - Sarcelles' → 'Paris'). In the attribute* 'country' *we replace the different spellings of the United States of America by a single one (here we choose 'usa' because of its short length, but in real scenarios the value 'united states of america' could be more useful). Finally, we correct the value transposition between the attributes* 'city' *and* 'country' *for entity* e_1 *and correct some typos (e.g. '.cmo' → '.com'). The cleaned database table is presented in Figure 4.6.*

Of course, despite of data preparation, the database contains still some unresolved errors. This especially concerns the attributes 'fname', 'lname', 'DoB', 'phone', *and* 'email', *because the values of these attributes are not predefined by some external list*[1] *and reference data can only be used to a limited extent. Moreover, a conflict of aggregation level maybe exists in the attribute* 'city', *because 'New York' can also refer to a state.*

Further information on data preparation and data cleaning (also known as data cleansing or data scrubbing) can be found in [GS13, GFS+01, Chr12, HDI12, LN06].

4.3.4. Candidate Pair Space Construction

Iterative duplicate detection is based on pairwise comparisons of database entities. Without any reduction, the candidate pair space of an entity set $\mathfrak{E} = \{e_1, e_2, \ldots, e_m\}$ is principally the set of all comparable

[1] Of course, the values of the attribute *'DoB'* are predefined, but typos are hard to detect because for each incorrect date we have a set of similar dates, e.g. '1974-12-41' can be transformed into several valid dates, and even incorrect dates can correspond to a correct element of the domain, e.g. '1974-12-31' is a valid date although it contains a typo and actually has to be the date '1974-12-21' instead.

DEI	bkv	code	block
e1	tucde	t230	B1
e2	tacjo	t220	B2
e3	tacbi	t210	B3
e4	tucjo	t220	B2
e5	tacbi	t210	B3
e6	tacbi	t210	B3
e7	lefli	l140	B4
e8	lefli	l140	B4
e9	tacph	t210	B3
e10	tacph	t210	B3

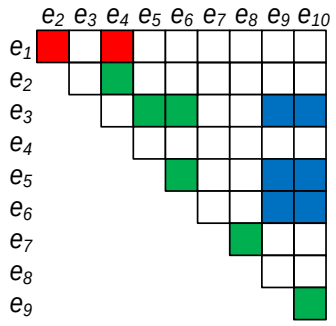
(a) Standard Blocking

DEI	bkv	windows
e8	lefl7	{W1}
e7	lefl6	{W1,W2}
e3	tacb4	{W1,W2,W3}
e5	tacb4	{W2,W3,W4}
e6	tacb5	{W3,W4,W5}
e2	tacj4	{W4,W5,W6}
e9	tacp5	{W5,W6,W7}
e10	tacp5	{W6,W7,W8}
e1	tucd4	{W7,W8}
e4	tucj4	{W8}

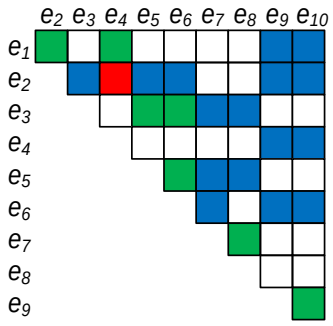
(b) SNM

DEI	bkv	suffixe	blocks
e1	edcut	{edcut,dcut,cut}	{B1,B2,B3}
e2	ojcat	{ojcat,jcat,cat}	{B4,B5,~~B6~~}
e3	ibcat	{ibcat,bcat,cat}	{B7,B8,~~B6~~}
e4	ojcut	{ojcut,jcut,cut}	{B9,B10,B3}
e5	ibcat	{ibcat,bcat,cat}	{B7,B8,~~B6~~}
e6	ibcat	{ibcat,bcat,cat}	{B7,B8,~~B6~~}
e7	ilfel	{ilfel,lfel,fel}	**{B11,B12,B13}**
e8	ilfel	{ilfel,lfel,fel}	{B11,B12,B13}
e9	hpcat	{hpcat,pcat,cat}	{B14,B15,~~B6~~}
e10	hpcat	{hpcat,pcat,cat}	{B14,B15,~~B6~~}

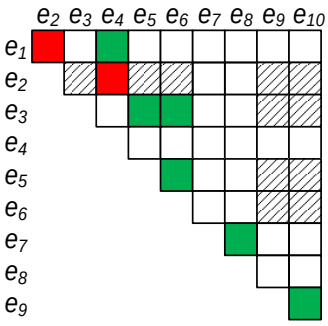
(c) Suffix-Array Blocking



(d) CPS Standard Blocking



(e) CPS SNM



(f) CPS Suffix-Array Blocking

Figure 4.7.: Illustration of several techniques for candidate pair space construction

entity pairs:

$$CPS_{naive} = \{\{e_r, e_s\} \mid e_r, e_s \in \mathfrak{E}, , e_r \neq e_s, e_r \stackrel{\circ}{=} e_s\} \tag{4.1}$$

where $e_r \stackrel{\circ}{=} e_s$ is a boolean condition that is true if e_r and e_s are comparable and false otherwise. Since two entities only need to be compared once and an entity does not need to be compared with itself, the naive candidate pair space consists of $\frac{|\mathfrak{E}| \times (|\mathfrak{E}|-1)}{2} = \frac{|\mathfrak{E}|^2 - |\mathfrak{E}|}{2}$ entity pairs (complexity $\mathcal{O}(|\mathfrak{E}|^2)$), if all entities of $\mathfrak{E}$ are pairwise comparable. For large databases with millions or more database entities, the size of the naive candidate pair space explodes and hence duplicate detection becomes impractical. For that reason, the candidate pair space need to be initially reduced before comparing entities in detail. Reduction is realized by rejecting pairs of entities being no duplicates for sure and adding them to the set of UNMATCHES without any detailed comparison. The construction of such a reduced candidate pair space $CPS \subset CPS_{naive}$ is usually performed by heuristic approaches that are commonly denoted as *blocking* [BCC03]. Besides blocking, entity pairs can be sometimes rejected by the use of additional context information. For instance, if we consider the result of integrating several sources and if we know that one source is duplicate-free, we can reject all entity pairs whose entities originate from the duplicate-free source from the candidate pair space because these entities cannot be duplicates. This kind of reduction is typically denoted as *pruning* [BS06].

Candidate pair space reduction is effective, if the number of rejected entity pairs is high, but is the more accurate, the less true duplicate pairs are rejected. In general, blocking is based on cheap comparisons and hence is known to cause two kinds of errors: a *false acceptance*, i.e. leaving an actual UNMATCH in the candidate pair space, and – even worse – a *false rejection*, i.e. removing an actual MATCH from the candidate pair space by assigning it to the set of UNMATCHES. False rejection is worse than false acceptance, because an actual MATCH that is rejected from the space is not considered again and therefore changes the duplicate detection result for the worse, whereas a false acceptance is eventually corrected during the following phases and hence has no impact on the effectiveness of the duplicate detection process at all, but only on its efficiency (each false rejection leads to a false negative, but not each false acceptance leads to a false positive).

Since the goals of effectiveness and accuracy[2] are contradictory (i.e. the more pairs are rejected, the smaller the candidate pair space, but the higher is the likelihood that an actual MATCH is rejected mistakenly), an appropriate blocking technique has to deal with an adequate trade-off between both requirements.

To illustrate the principle of candidate pair space construction we consider three simple blocking techniques, namely the *Standard Blocking*, the *Sorted Neighborhood Method*, and the *Suffix-Array Blocking*.. All these techniques are based on the use of a *blocking key* (short *bk*) that is a function that extracts a string value from a set of attribute values by concatenating some proportions of them. Blocking keys are usually defined on schema level so that a single blocking key can be applied to the set of all entities sharing the same schema. An example of a blocking key defined for a schema having the three attributes *'fname'*, *'lname'*, and *'DoB'* is to take the first four letters of the last name, the first two letters of the first name, the last two digits of the date of birth and to concatenate them in exact this order. The string that result from applying a blocking key to the attribute values of a concrete database entity is called a *blocking key value* (short *bkv*).

- **Standard Blocking** [Jar85, LF05a] is one of the oldest and simplest blocking techniques that groups all entities with the same *bkv* together in the same block and then constructs the candidate pair space by pairing all entities of one block. Standard blocking can be considered as hashing where the *bkv* serves as hash code. Since only entities with the same *bkv* are paired, this technique is very sensitive to errors in the *bkv*s. To reduce this effect a phonetic encoding like Soundex (see Section 4.3.5.3) can be applied to the *bkv*s before grouping them. This encoding transforms the *bkv*s in a way that values with similar sounding patterns become equal.

  The effectiveness of standard blocking essentially depends on the size of the largest block. Thus, the goal is to uniformly distribute the entities on an adequate number of blocks. This makes the selection of the blocking key always a trade off between accuracy and effectiveness. Long keys lead to small blocks, but typically cause many false rejections. In contrast, short keys reduce the risk of false rejections, but tend to produce large blocks.

[2] Recall, the purpose of candidate pair space reduction is to reduce the number of pairwise entity comparisons. Thus, a candidate pair space reduction is effective if it increases the efficiency of the considered duplicate detection process and is accurate if it does not decrease the effectiveness of this detection process.

- The **Sorted Neighborhood Method** (short SNM) [HS95, HS98] constructs the candidate pair space from the *bkv*s in two steps. First the entities are sorted - usually lexicographically - by their respective *bkv*s. Second a window of fixed size w slides sequentially over the sorted entities. All entities being within the window at the same time are paired with each other and added to the candidate pair space. Due to the fixed window size, each entity is compared with at most $2w - 2$ entities from its immediate neighborhood. The underlying assumption of the SNM is that duplicate entities have similar *bkv*s and hence are sorted close together. The problem of a fixed window size is that a high number of entities can have the same *bkv* (for example a *bkv* that is derived from the last name 'Smith'), but do not fit together in the window at the same time. One solution to that problem is to only sorts the *bkv*s and to annotate each *bkv* with its corresponding a set of entities [Chr11]. To solve the problem of many similar *bkv*s an extended version using a variable window size has been proposed by Yan et al. [YLKG07].

  By assuming a data set with n entities and a window of a fixed size w, the number of candidate pair that result from using the SNM with a single pass is $\mathcal{O}(wn)$ [HS95].

- **Suffix-Array Blocking** [AO05] is an extension of standard blocking that enables the use of short keys (the shortest suffix allowed) without risking to produce large blocks. First a key of median size is selected and for each entity a corresponding *bkv* is extracted. Then all suffixes of the initial *bkv*s that are longer than a minimal length are built and each suffix serves then as an additional *bkv* (optionally a phonetic encoding can be applied to the suffixes before). Due to each entity has now multiple *bkv*s, it is inserted into several blocks. To avoid large blocks that would result from common suffixes (common *bkv*s of short keys), blocks with more than a specific number of entries are removed from further considerations. Finally, the candidate pair space is constructed by pairing all entities being in at least one same block. In conclusion, this technique always uses the shortest key per entity that does not produce a large block[3].

  An extended version that is additionally robust against typos in the suffixes has been proposed by Vries et al. [dVKCC09, dVKCC11].

Example 51 *For illustration, we consider the cleaned sampled table from Figure 4.6. Each of these three techniques has different requirements to the characteristics of the blocking key. For that reason, we use a different blocking key for each technique. For standard blocking we select a key that concatenates the first three letters of the last name with the first two letters of the first name. Since we encode the bkvs with using Soundex, the first letter of the key must be the most identifying. The bkvs and the Soundex codes are presented in Figure 4.7(a).*

The key that is used for suffix-array blocking is exact the inverse of the key for standard blocking, because here the most identifying letters have to be at the end of the bkvs. In our example, we set the

[3]The complexity of standard blocking depends on the size of the largest block. Thus, using short *bkv*s can be bad, because it risks the appearance of a large block. In contrast, if the *bkv*s are too large, many true duplicates will not be added to the candidate pair space. If two pairs have the same *bkv*, they will also agree in any substring of this *bkv*. Thus, if two pairs agree in a long suffix they will also agree in a short suffix. Consequently, by utilizing suffix-array blocking we do not use a universal key length for all entities, but use an individual key length per entity, i.e. the shortest *bkv* that does not produce a large block.

minimal suffix length to three and set the maximal block size to five (see Figure 4.7(c)). For that reason, for each entity three bkvs are created and block 'B6' is finally removed.

For the SNM we use a key that concatenates the first three letters of the last name with the first letter of the first name and the last digit of the year of birth. Moreover, we use a window of size three. Figure 4.7(b) presents the sorted list of entities along with their bkvs and the different window positions they belong to.

The candidate pair spaces that result from these techniques are presented by the matrices in Figure 4.7(d) (standard blocking), Figure 4.7(e) (SNM), and Figure 4.7(f) (suffix-array blocking). All candidate pairs that are accepted correctly, i.e. all true acceptances, are colored green. As you can see, the SNM produce a lot of false acceptance (blue colored), but because of the high number of candidate pairs it produces only one false rejection (red colored). Standard blocking produces a smaller candidate pair space, but produces one false rejection more. Suffix-array blocking has no false acceptance, but has as many false rejections as standard blocking. To illustrate the effect of the block removal that is performed in the suffix-array blocking, we mark all candidate pairs that have been avoided by removing block 'B6' with a stripped pattern. As it can be seen, without bock removal the number of false acceptance increases by eleven, but the number of true acceptance does increase by a single pair. This demonstrates that the removal of the large blocks cans increase the effectiveness of blocking to a large extent.

Obviously the definition of the blocking key is a crucial point for all techniques. A poor blocking key will always lead to a poor candidate pair space. For that reason, keys should be designed in a way that duplicate tuples do have similar *bkv*s. Moreover, less error-prone attributes should be used. To reduce this bottleneck further on Hernandez et al. [HS95] propose a multi-pass approach where a single blocking technique is performed for several times and at each time another blocking key is used. The final candidate pair space contains then all candidate pairs that at least belong to the space produced by one pass, or is determined by a voting strategy. It is obvious that the resultant candidate pair space is much larger than the space that results from a single pass approach. However, the risk of choosing a poor blocking key is lowered and the result is usually more accurate.

Besides the three techniques presented above a variety of further techniques has been proposed during the last decades. Sorted Blocks [DN11] is a technique that aims to avoid the problem of a fixed window size in the SNM by combining the ideas of standard blocking and the SNM. Q-gram Indexing [BCC03] is based on the idea that the number of false rejections that are caused by typos in the *bkv*s can be reduced by using combinations of their q-grams for block building instead the *bkv*s themselves. A very effective and accurate technique is the RAR-algorithm proposed by Sung et al. [SLP02]. This algorithm combines the SNM and the TI-similarity that is a computational efficient measure for string similarity. Although this technique is originally designed with having the SNM in mind, the concept of using TI-similarity can be simply combined with any other blocking technique.

Further blocking techniques are K-way Sorting [FC00], Similarity-Aware Inverted Indexing [CG08], String Map based Indexing [JLM03], Priority Queue [ME97], Adaptive Filtering [GB04], Locality-Sensitive Hashing [sKL10], Fuzzy Blocking [NT07], Canopy Clustering [MNU00, CR02, FGPP10], Spectral Neighborhood Blocking [SCXM11], CBLOCK [SJMB11], and blocking with MFIBlocks [KG09]. Kolbe et al. [KTR10, KTR12] consider a parallelization of blocking using the MapReduce programming model. Approaches for blocking based on semantic relationships between data items

are proposed in [NMMMBLP07b, NMMMBLP07a] (tuple relationships given by foreign keys) and [PWN06] (hierarchical relationships in XML documents). In [PINF11] blocking data items with heterogeneous data structures is considered. Further interesting and useful work on blocking can be found in [BKM06, MK06, Chr07, WMK+09, PIN+11]. Recently, Christen wrote a survey on some of the techniques mentioned above [Chr11].

4.3.5. Attribute Value Matching

In the phase of attribute value matching, the attribute values of two database entities are compared by the use of syntactic and semantic similarity measures. Since the subsequently made duplicate decisions are based on the similarity scores that are computed in this phase, attribute value matching is the fundamental phase in a duplicate detection process and a poor selection of similarity measures immediately leads to a poor detection result.

Due to typos and other data errors, a test of equivalence is usually not sufficient for similarity computation but attribute values need to be compared by more complex similarity measures instead. In many contexts, however, the meaning of similarity is not clear and therefore a variety of similarity measures has been developed during the last decades from which no one can be considered as an ultimate solution. In contrast, each of these similarity measures has been designed with some specific input data and with some specific error pattern in mind. Data types can be strings, numbers, or dates and they can be defined on single values or sets/bags of values. Principally, existing similarity measures can be distinguished into three classes: measures that aim to score syntactic similarity, measures that aim to score semantic similarity, and measures that aim to score phonetic similarity. Because scoring semantic similarity is only possible with additional context information such as Thesauri or ontologies available (see Section 4.3.5.4), the most similarity measures focus on syntactic similarity.

Most often only values of the same attribute are compared. However, it can be also valuable to match values across attributes. For instance, the email address of a person often contains the name of the person and hence matching email addresses with names can produce some further evidence for the following decision model.

We start with a presentation of syntactic similarity measures for sets and strings. Then we continue with a discussion on phonetic similarity measures as well as semantic similarity measures of strings, and consider the similarity of numbers and dates. Finally, we present some hybrid measures that are able to incorporate some of the previously presented measures into a measure for set similarity.

4.3.5.1. Set Similarity

Intuitively, two sets are the more similar, the more elements they have in common. Although the underlying idea is quite simple, measuring set similarity is not straightforward as we will illustrate by the following four measures:

- The **Common Neighbor Score** [BG07b] scores the overlap between two sets and normalize it by a large constant. Due to all overlaps are normalized by the same value, the resultant scores can become unrepresentative, if the considered sets vary a lot in their size, because an overlap of size

two obviously implies a greater similarity for two sets each having three elements than for two sets each having hundred elements.

- For that reason, the **Jaccard Coefficient** [NH10] presented in Equation 4.2 computes the amount of elements two sets S_m and S_n have in common and hence normalizes the overlap by the number of elements that belong to any of both sets instead by a constant:

$$Jacc(S_m, S_n) = \frac{|S_m \cap S_n|}{|S_m \cup S_n|} \tag{4.2}$$

  The Jaccard Coefficient can be generalized to the Tversky Similarity [Tve77] that allows to define penalizations for missing elements.

  The shortcoming of the Jaccard Coefficient is twofold: (i) First, all elements are considered equally, but the set membership of an element that only belongs to few sets is much more distinguishing than an element that belongs to many sets. (ii) Second, the coefficient only compares on equality and does not consider cases, where set elements are unequal but very similar as it can result from typos or different standardizations that has not been resolved during the data preparation phase.

- A generic version of the **Adamic/Adar Similarity** [AA03] that has been proposed by Bhattacharya et al. [BG07b] aims to neglect the first shortcoming of the Jaccard Coefficient by taking the uniqueness of each element into account. Let $u(e)$ be the uniqueness of element e, the Adamic/Adar Similarity of two sets S_m and S_n is defined as:

$$Adar(S_m, S_n) = \frac{\sum_{e \in S_m \cap S_n} u(e)}{\sum_{e \in S_m \cup S_n} u(e)} \tag{4.3}$$

  A well known measure for uniqueness is the *term frequency - inverse document frequency* (short *TF-IDF*) [BYRN99] that comes from the area of information retrieval and defines the uniqueness of an element based on the number of sets (documents) in the considered database this element belongs to. The term frequency $tf(t, d)$ scores how often the term t appears in the document d and the document frequency $df(t)$ scores the number of documents in which the term t appears. Due to a term is more unique, the less often it appears in all documents, the inverse document frequency $idf(t) = n/df(t)$, where n is the total number of documents, is used as a measurement of uniqueness.

  As described by Halevy et al. [HDI12] term frequency and inverse document frequency can be combined in different ways. The most common variant of *TF-IDF* is defined as:

$$TF\text{-}IDF(t, d) = \log(tf(t, d) + 1) \times \log(idf(t)) \tag{4.4}$$

- The **Cosine Similarity** [NH10] is another weighting based approach for measuring set similarity[4]. It first builds the two vectors V_{S_m} and V_{S_n} from the two compared sets S_m and S_n and then

[4] Actually the Cosine Similarity is the similarity measure that is used for matching the sets' corresponding vectors, but its name has been adopted to the presented measure of set similarity, i.e., $CosineSim(S_m, S_n) = CosineSim(V_{S_m}, V_{S_n})$.

computes the final similarity score as the angle between these vectors:

$$\textit{CosineSim}(S_m, S_n) = \frac{V_{S_m} \cdot V_{S_n}}{||V_{S_m}|| \cdot ||V_{S_n}||} = \frac{\sum_{i=1}^{k} V_{S_m}(i) \times V_{S_n}(i)}{\sqrt{\sum_{i=1}^{k} (V_{S_m}(i))^2} \times \sqrt{\sum_{i=1}^{k} (V_{S_n}(i))^2}} \tag{4.5}$$

The vector space contains one dimension per possible set element and the value $V_S(i)$ of a vector V_S in the i-th dimension is the weight of the corresponding element to the set S. As for the Adamic/Adar Similarity, the *TF-IDF* is typically used for weighting the individual elements.

Whereas the Adamic/Adar Similarity and the Cosine Similarity aim to solve the first shortcoming of the Jaccard Coefficient, the second shortcoming is still present in all measures, but can be resolved by using a hybrid similarity measures as described in Section 4.3.5.6,

Measures for set similarity can be applied to any kind of set. However, because most attribute values are strings, in the context of duplicate detection, measures for set similarity are typically used to compare two string values by initially decomposing the strings into their sets of tokens or by decomposing the strings into their sets of q-grams [NH10].

Example 52 *For illustration we consider the two name values v_1 = 'Tacker, Bill Thomas' and v_2 = 'John Bill Tacker' from the sample table from Figure 4.5 (here we assume that data preparation has not been applied to the data). A tokenization of both names leads to the following two token sets: S_1 = tokenize(v_1) = {'Tacker', 'Bill', 'Thomas'} and S_2 = tokenize(v_2) = {'John', 'Bill', 'Tacker'}. The Jaccard Coefficient of both sets is Jacc$(S_1, S_2) = 2/4 = 0.5$. The term frequency, document frequency and the TF-IDF-score of the individual tokens are listed in Table 4.1. Consequently, the Adamic/Adar Similarity of both sets results in Adar$(S_1, S_2) = (0.0668 + 0.1198)/(0.0668 + 0.1198 + 0.2104 + 0.301) = 0.1812$ and the Cosine Similarity results in CosineSim$(S_1, S_2) = 0.2262$ (for illustration, see Figure 4.1). Due to the latter two measures consider the fact that the shared tokens are very common in the considered database, but the non shared tokens are not, the similarity score that results from the Adamic/Adar Similarity and the similarity score that results from the Cosine Similarity are likely more representative then the similarity score that results from the Jaccard Coefficient.*

As a second example, we consider the two strings v_3 = 'John' and v_4 = 'Dejohn'. Both string only consist of one token so that performing a tokenization as we have used in the first example does not bring any advantage and the Jaccard Coefficient as well as the Adamic/Adar Similarity produce a similarity score of zero. It is obvious that this score does not represent the actual similarity of both strings. To improve representativeness, we can decompose both string into two sets of q-grams. From using $q = 3$, the two values v_3 and v_4 can be transformed into the sets S_3 and S_4 where # is a padding character:

$$\begin{aligned} S_3 &= \{\text{'\#\#J'}, \text{'\#Jo'}, \text{'Joh'}, \text{'ohn'}, \text{'hn\#'}, \text{'n\#\#'}\} \\ S_4 &= \{\text{'\#\#D'}, \text{'\#De'}, \text{'Dej'}, \text{'ejo'}, \text{'joh'}, \text{'ohn'}, \text{'hn\#'}, \text{'n\#\#'}\} \end{aligned}$$

The Jaccard Coefficient of both sets results in Jacc$(S_3, S_4) = 4/10 = 0.4$. Thus, by using q-gram decomposition, the set similarity measure is able to capture the significant resemblance of both string value, although they are not identical.

	$t =$ 'Tacker'	$t =$ 'Bill'	$t =$ 'Thomas'	$t =$ 'John'
$df(t)$	6	4	1	2
$tf(t, S_1)$	1	1	1	0
$tf(t, S_2)$	1	1	0	1
$TF\text{-}IDF(t, S_1)$	0.0668	0.1198	0.301	0
$TF\text{-}IDF(t, S_2)$	0.0668	0.1198	0	0.2104

	$dim1$	$dim2$	$dim3$	$dim4$
V_{S_1}	0.0668	0.1198	0.301	0
V_{S_2}	0.0668	0.1198	0	0.2104

$\Rightarrow V_{S_1} \cdot V_{S_2} = 0.0188$
$\Rightarrow ||V_{S_1}|| \cdot ||V_{S_2}|| = 0.3308 \cdot 0.2512 = 0.0831$
$\Rightarrow CosineSim(S_1, S_2) = 0.2262$

Table 4.1.: The *df*, the *tf*, and the *TF-IDF* of four sample tokens and the Cosine Similarity of two token sets

4.3.5.2. String Similarity

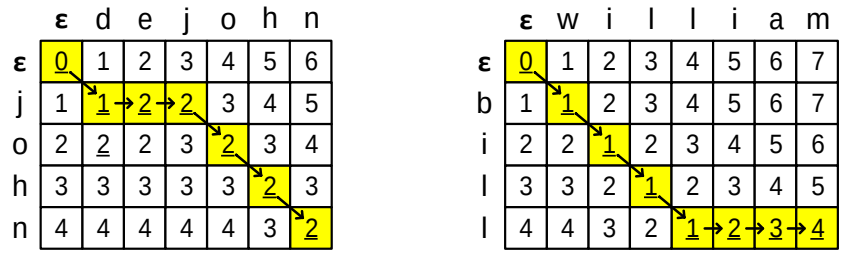
The purpose of a measure for string similarity is to compute a score that reflects the syntactic similarity of two strings as good as possible despite the presence of typos or letter transpositions. We will illustrate the variety of different solutions by presenting two simple approaches.

- The **Levenshtein Distance** [Lev66] is the simplest variant of a family of measures for string similarity that is well known as *Edit-Distance*. The underlying idea of Edit-Distance measures is that the distance between two strings can be defined by the lowest cost that is required to transform one of these strings into the other by the use of a specific set of edit operations where each edit operation can be associated with an individual cost. The Levenshtein Distance uses the three edit operations *letter insert*, *letter delete*, and *letter replace* each associated with the same cost of one. Further edit-distance measures are the *Needleman-Wunch Distance* [NW70], the *Affine Gap Distance* [WSB76], the *Smith-Waterman Distance* [SW81], and the *Typewriter Distance* [BL13]. They all differ in their set of edit operations and/or cost assignments.

  The Edit-Distance between two string values v_m and v_n can be computed based on the method of *dynamic programming* [Nav01, NH10], because a transformation between two strings can always be broke down into transformations of their substrings. Let i, j be two variables with $0 \leq i \leq |v_m|$ and $0 \leq j \leq |v_n|$ and let $v_{k,l}$ be the l-th letter of string v_k, a corresponding algorithm that computes the Levenshtein Distance between v_m and v_n is based on filling a $|v_m| \times |v_n|$ matrix M by the following computation rules:

$$
\begin{aligned}
M_{i,0} &= i \\
M_{0,j} &= j \\
M_{i,j} &= \begin{cases} M_{i-1,j-1}, & \text{if } v_{m,i} = v_{n,j}, \\ \mathbf{min}(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) + 1, & \text{otherwise.} \end{cases}
\end{aligned}
$$

  The algorithm starts at $M_{0,0}$ and computes all positions of the matrix from top left to bottom right. The last computed value $M_{|v_m|,|v_n|}$ corresponds to the Levenshtein Distance of v_m and v_n. Due to



(a) Sample 1: 'john' vs. 'dejohn' (b) Sample 2: 'bill' vs. 'william'

Figure 4.8.: Two samples of computing the Levenshtein Distance with dynamic programming

all cells of the matrix need to be computed once, the algorithm has the computational complexity $\mathcal{O}(|v_m| \times |v_n|)$. Other variants of the Edit-Distance differ to the Levenshtein Distance in their computation rules.

The Levenshtein Distance between two strings can be normalized by the length of the longer string, because in the worst case where the two compared strings v_m and v_n does not share a single letter, the cheapest transformation is made by first replacing each letter of the shorter string v_m by one letter of v_n and then by inserting the remaining letters of v_n. As a consequence, we maximally need $|v_m|$ letter replaces and $|v_n| - |v_m|$ letter inserts to transform v_m into v_n. The normalized Levenshtein Similarity is defined as the difference of the normalized distance to one:

$$\mathit{LevSim}(v_m, v_n) = 1 - \frac{\mathit{Levenshtein\ Distance}(v_m, v_n)}{\mathbf{max}(|v_m|, |v_n)|)} \tag{4.6}$$

- **Jaro Similarity** is a measure for string similarity that in contrast to the measures of the Edit-Distance family cannot be computed by dynamic programming. The Jaro Similarity has been primarily designed for matching short strings [Jar89] and it aims to overcome syntactical dissimilarities that result from letter transpositions. A transposition is considered to be a swap of a common letter, where a letter is only considered common, if its occurrence in both strings is close to each other.

  Let S be the set of letters that the two strings v_m and v_n have in common, i.e. their position distance is less than $0.5 \times \mathbf{min}(|v_m|, |v_n|)$, and let t be the number of transpositions, the Jaro Similarity of v_m and v_n is defined as:

$$\mathit{JaroSim}(v_m, v_n) = \frac{1}{3} \times \left(\frac{|S|}{|v_m|} + \frac{|S|}{|v_n|} + \frac{|S| - 0.5t}{|S|} \right) \tag{4.7}$$

  An extension of the Jaro Similarity is the *Jaro-Winkler Similarity* [WT91] that is tailor made for domains where a common prefix is sometimes followed by an additional suffix (e.g. the domain of firstnames: 'Bill' vs. 'Bill Thomas').

Example 53 *To demonstrate the dynamic programming based algorithm of the Levenshtein Distance, we compare two pairs of string values from the sample table from Figure 4.6. The matrix of v_1 = 'john' and v_2 = 'dejohn' is presented in Figure 4.8(a) and the matrix of v_3 = 'bill' and v_4 = 'william'*

is presented in Figure 4.8(b) (the symbol ε represents an empty string). In each case, one of several possible sequences of edit operations that lead to a transformation with minimal costs is colored yellow and emphasized by arrows. Going a cell to the right means that a new letter is inserted, going a cell down means that a letter is deleted and going a cell diagonal down means that either nothing has been edited or a letter is replaced. 'john' can be transformed into 'dejohn' by inserting two letters at the beginning. In contrast, 'bill' can be transformed into 'william' by replacing the first letter and by appending the letters 'i','a', and 'm' at the end. Since 'dejohn' has six letters and 'william' has seven letters, the normalized scores of the Levenshtein Similarity result in LevSim$(v_1, v_2) = 2/3$ and LevSim$(v_3, v_4) = 3/7$.

The maximal distance that is allowed for letters of 'dejohn' and 'john' in order to be denoted as common is $0.5 \times \mathbf{min}(|v_1|, |v_2|) = 2$. Thus, the set of common letters of v_1 and v_2 is {'j', 'o', 'h', 'n'}. In this case, the number of transpositions is zero and the Jaro Similarity of v_1 and v_2 results in 0.89. Let us compare v_1 with another string v_5 = 'nojha'. The maximal distance is still 2, but the set of common letters is only {'j', 'o', 'h'}, because the distance of the appearances of letter 'n' is too large. The common letter sequence in v_1 is 'joh', but is 'ojh' for v_5. Thus, we have one transposition and the Jaro Similarity of v_1 and v_5 results in 0.73.

4.3.5.3. Phonetic String Similarity

The similarity measures discussed in the previous section consider the syntactic similarity of strings. Nevertheless, many string errors can result from mishearing as for example in a *Customer Relationship Management* scenario where information is often exchanged by phone. Similar sounding strings must not be syntactical similar. Thus, phonetic encodings have been developed as another type of similarity measure in order to neglect such errors in string matching.

The idea of a phonetic encoding is that similar sounding strings get similar (or in the optimal case even identical) codes. From these codes a similarity score is derived. In the simplest case, the similarity of two strings is set to 1 if they have the same code and is set to 0 otherwise.

For illustration, we present the *Soundex Code* [**?**, Rus22] which is one of the most used phonetic encoding algorithm and which we have already used to illustrate Standard Blocking in Section 4.3.4. The Soundex Code is designed for the English language and transforms a string value v in its code by deleting or replacing letters in four steps:

1. All occurrences of 'a', 'e', 'i', 'o', 'u', 'y', 'h', and 'w' except the first letter are deleted from s.
2. All remaining letters of s except the first letter are replaced by digits following the mapping presented in Table 4.2, e.g. the letters 'b' and 'f' are replaced by the digit '1'.
3. All consecutive appearances of the same digit are replaced by a single appearance of this digit.
4. Finally, the code is truncated to length four if it is longer than four or filled up to a length of four by appending '0's if it is shorter than four.

Example 54 *For demonstration, we consider the four strings 'Tukker', 'Tacker', 'William', and 'Bill' from our sample table and transform them to their soundex code. The individual steps of transformation*

digit	letter
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

Table 4.2.: Soundex letter-digit map

	'Tukker'	'Tacker'	'William'	'Bill'
Step 1	'Tkkr'	'Tckr'	'Wllm'	'Bll'
Step 2	'T226'	'T226'	'W445'	'B44'
Step 3	'T26'	'T26'	'W45'	'B4'
Step 4	**'T260'**	**'T260'**	**'W450'**	**'B400'**

Table 4.3.: Four Samples Soundex Encoding

are presented in Table 4.3. As you can see, the two similar sounding strings 'Tukker' and 'Tacker' are both transformed into the same code. In contrast, the somewhat similar sounding strings 'William' and 'Bill' are not, because the soundex algorithm lets the first letter unchanged.

Further phonetic similarity measures are *Metaphone*, *Double Metaphone*, *Metaphone 3* [Phi90, Phi00], *Caverphone* [PLSM09], the *NYSIIS Algorithm* [Taf70], *Daitch-Mokotoff Soundex*[5] (jewish and east european names), *Kölner Phonetik* (german language) [Pos69], *Match Rating Approach* [MK77], *Phonix* [Gad98], and the *Oxford Name Compression Algorithm* [Gil97].

The measures *Editex* [ZD96] and *Syllable Alignment Distance* [GC06] combines several of the aforementioned techniques in order to rate syntactic similarity and phonetic similarity within a single score.

4.3.5.4. Semantic Similarity

Although two strings are syntactically dissimilar and phonetically dissimilar, they can have some semantical resemblance if they are closely related by any meaning. To score semantic similarity additional context information is required. Such information is usually represented by ontologies that range from informal ones that are primarily designed for human usage to formal ones from which knowledge can be automatically derived by the use of inference [LN06].

Thesauri are examples of informal ontologies. A Thesaurus is a collection of words that have a similar meaning [LN06]. It usually includes but is not limited to synonyms, hyponyms, umbrella terms, and sometimes antonyms. A sample Thesaurus for the German language is *OpenThesaurus*[6]. Another Thesaurus that is specialized on names is *Onomastik*[7]. Figure 4.9 lists some names that Onomastik proposes to have some resemblance to the name 'Alexander' along with their Levenshtein Distance. An interesting example is the name 'Sascha'. Syntactically both names have nearly nothing in common, but in Russian 'Sascha' is used as a nickname for 'Alexander'. Another commonly used synonym is word 'Down Under' that refers to Australia. An approach that incorporates information on such synonyms into the detection process has been proposed by Arasu et al. [ACGK08].

Thesauri are also often used to model hierarchies of terms. Such hierarchies can be useful for detecting duplicates, because they help the system to detect and handle cases where two values are defined on

[5] http://www.jewishgen.org/InfoFiles/soundex.html
[6] http://www.openthesaurus.de/
[7] http://www.onomastik.com/

name	***LevSim***
Alejandro	0.67
Aleksandr	0.67
Alessandro	0.6
Alexandros	0.8
Alexis	0.44
Leszek	0.33
Sascha	0.11
Xander	0.67

Figure 4.9.: Names related to 'Alexander'

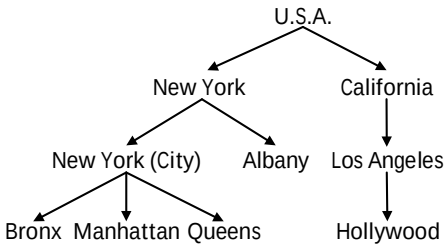


Figure 4.10.: Hierarchy on districts, cities, and states within the U.S.A.

different aggregation levels. A part of an ontology that models the geographical hierarchy of the United States of America is depicted in Figure 4.10.

Example 55 *In the sample table from Figure 4.6, a Thesaurus can help us to detect that the name 'bill' is often used as a short version of the name 'william' and hence they are semantically similar. Moreover, it can improve the matching result by providing the information that New York is not only a city, but is also a state and that Albany is a city (even the capital) of this state. Thus, the semantic similarity between the two values 'Albany' and 'New York' is much higher than its syntactic similarity.*

4.3.5.5. Number & Date Similarity

Values of the domains of numbers or the domains of dates are difficult to compare. On one hand they are as sensitive against typos as string values (e.g. '12.05.2012' vs. '05.12.2012'), but on the other hand their domains are ordered and hence are associated with a natural meaning of distance (e.g. '31.12.2012' vs. '01.01.2013'). As a consequence, values of both domains should be matched by a combination of a measure for string similarity that is resistant against typos and a second measure that scores the natural distance between the compared numbers (or date values respectively).

Another aspect that need to be considered is that the meaning of the natural distance depends on the considered use case. For example, a distance of $1cm$ is not much for the size of persons, but can be worlds in micro biology. More information on matching values from numerical domains like date, age, time can be found in [Chr12].

4.3.5.6. Hybrid Similarity Measures

A major shortcoming of the set similarity measures presented in Section 4.3.5.1 is that they only distinguish between equal elements and non-equal elements. On the other hand, the string similarity measures presented in Section 4.3.5.2 are extremely sensitive to the token order. Thus, it is only a logical consequence to combine both concepts. Hybrid similarity measures integrate domain-specific measures for element similarity into measures for set similarity to overcome the shortcomings mentioned above. In general, the integrated measure for element similarity can be of any nature. It can be a measure for the

similarity of strings, numbers, or dates, it can be a measure for semantic similarity, it can be a measure for phonetic similarity, or it can be a combination of several measures.

We illustrate the concept of integration by two straightforward approaches.

- As its name implies, the **Extended Jaccard Coefficient** [NH10] integrates a measure for element similarity into the Jaccard Coefficient that we have presented in Section 4.3.5.1. Given two sets S_m and S_n, a measure for element similarity sim_e, and a threshold $\theta \in [0, 1]$, the extended Jaccard Coefficient initially computes the set of all elements shared by S_m and S_n, the set of elements of S_m that are unique, and the set of elements of S_n that are unique:

$$\textit{Shared}(S_m, S_n) = \{(e_r, e_s) \mid e_r \in S_m, e_s \in S_n, sim_e(e_r, e_s) > \theta\} \tag{4.8}$$

$$\textit{Unique}(S_m) = \{e_r \mid e_r \in S_m, \nexists e_s \in S_n \colon (e_r, e_s) \in \textit{Shared}(S_m, S_n)\} \tag{4.9}$$

$$\textit{Unique}(S_n) = \{e_s \mid e_s \in S_n, \nexists e_r \in S_m \colon (e_r, e_s) \in \textit{Shared}(S_m, S_n)\} \tag{4.10}$$

  Using these three sets the Extended Jaccard Coefficient is then computed as:

$$\textit{ExtJacc}(S_m, S_n) = \frac{|\textit{Shared}(S_m, S_n)|}{|\textit{Shared}(S_m, S_n)| + |\textit{Unique}(S_m)| + |\textit{Unique}(S_n)|} \tag{4.11}$$

  Another extension of the Jaccard Coefficient has been proposed by Naumann et al. [NH10]. This approach first assigns a weight to each pair of shared elements and assigns a weight to each unique element and then aggregates these weights to compute a score for each of the three sets. Finally, the scores are used instead of the sets' sizes to compute the final result. An intuitive setting of this approach is to weight a pair by its similarity, to weight a single element by one and to use the sum operator for aggregation.

  A third version that is described in [HDI12] uses a maximum-weight matching between the shared elements.

- The **Monge-Elkan Similarity** [ME96] computes the average maximal similarity the elements of a first set have to the elements of a second set. Thus, given the two sets S_m and S_n as well as a measure for element similarity sim_e, the Monge-Elkan Similarity from S_m to S_n results in:

$$\textit{MongeElkanSim}(S_m, S_n) = \frac{1}{|S_m|} \sum_{i=1}^{|S_m|} \mathbf{max}_{j=1}^{|S_n|} sim_e(e_i, e_j) \tag{4.12}$$

  The problem with the Monge-Elkan Similarity is that this measure is non-symmetric, i.e. $\textit{MongeElkanSim}(S_m, S_n) \neq \textit{MongeElkanSim}(S_n, S_m)$. A possible solution to this shortcoming is to use the average (or the minimum/maximum respectively) of both directions as a final similarity score.

An hybrid version of the Cosine Similarity is described in [NH10, HDI12].

Example 56 *As an example we consider the two token sets* $S_1 = \{$*'John', 'Bill', 'Tacker'*$\}$ *and* $S_2 = \{$*'John', 'William', 'Tucker'*$\}$. *As computed in Example 52, the Jaccard Coefficient of them is only* $Jacc(S_1, S_2) = 0.2$. *The Levenshtein Similarities of the token pairs are presented in Figure 4.4.*

	'John'	'Bill'	'Tacker'
'John'	1	0	0
'William'	0	3/7	0
'Tucker'	0	0	5/6

Table 4.4.: The Levenshtein Similarity between the tokens of the sample sets S_1 and S_2

Using these similarities and by using a threshold $\theta = 0.5$*, the sets Shared*(S_1, S_2)*, Unique*(S_1)*, and Unique*(S_2) *are computed as follows:*

$$\begin{aligned} \textit{Shared}(S_1, S_2) &= \{(\textit{'John'}, \textit{'John'}), (\textit{'Tucker'}, \textit{'Tacker'})\} \\ \textit{Unique}(S_1) &= \{\textit{'William'}\} \\ \textit{Unique}(S_2) &= \{\textit{'Bill'}\} \end{aligned}$$

Consequently, the Extended Jaccard Coefficient as defined in Equation 4.11 results in $\textit{ExtJacc}(S_1, S_2) = 0.5$*. In contrast to the standard variant of the Jaccard Coefficient the similarity is increased from* 0.2 *to* 0.5 *because the two tokens 'Tucker' and 'Tacker' are now considered to be identical. Note, the Monge-Elkan Similarities between* S_1 *and* S_2 *even results in* $\textit{MongeElkanSim}(S_1, S_2) = \textit{MongeElkanSim}(S_2, S_1) = 0.754$*.*

A shortcoming of all the approaches presented above (except the extended version of the Jaccard Coefficient presented in [HDI12]) is that they map the elements of one set to the elements of the other set, but do not take care whether the resultant mapping is 1:1 or not. In the extreme case, all elements of one set are mapped to the same element of the second set which in turn can lead to a similarity score that is extremely unrepresentative for the actual similarity between both sets. Nevertheless, a 1:1 mapping as it can be achieved by an algorithm that solves the *Stable Marriage Problem* [GI89] is computationally much more expensive and therefore is usually avoided within the application of matching attribute values for duplicate detection.

4.3.5.7. Similarity of Null Values

A less considered, but important challenge, especially for databases that are corrupted by many null values, is to rate the similarity between two attribute values if at least one of them is a null value. One standard approach is to consider the similarity between any value and a null value as zero, but this approach can create many false negatives, especially if null values occur in important attributes. Another approach is to use a fix standard similarity, but here we have a similar problem. If the chosen similarity is too low, the risk of producing false negatives increases, but if the chosen similarity is too high, producing false positives becomes more likely.

4.3.6. Decision Model

The input to the decision model is a candidate pair and the comparison vector that has been computed for this pair in the previous phase of attribute value matching. The purpose of the decision model is to

decide whether the considered candidate pair is a duplicate or not based on the values provided by the comparison vector. All pairs classified as duplicates are collected in the set of MATCHES and all pairs classified as non-duplicates are collected in the set of UNMATCHES. In semi-automatic approaches, a third set of POSSIBLE MATCHES is intermediately introduced. In that case, each candidate pair that has been classified as a POSSIBLE MATCH is later manually assigned to the set of MATCHES or the set of UNMATCHES by a domain expert. For space reasons, we sometimes refer to the three sets also by M (MATCHES), U (UNMATCHES), and P (POSSIBLE MATCHES).

Existing approaches for decision models can be classified into five classes: *Distance-based Approaches*, *Rule-based Approaches*, *Probabilistic Approaches*, *Learning-based Approaches*, and *Clustering-based Approaches*. In order to give an impression on the large diversity of decision models, we will present the individual classes in more detail in the remainder of this section.

Besides the classification results, many approaches of the finally performed duplicate clustering phase (see Section 4.3.7) additionally require a similarity score for each candidate pair as input. For that reason, we consider a duplicate decision as an ordered pair whose first element is a matching class, i.e. the classification result MATCH, UNMATCH, or POSSIBLE MATCH, and whose second element is a similarity score. By using distance-based decision models, the computed distances can be transformed into similarity scores. In the most machine learning techniques a confidence value can be computed for the classification result, for example in Support Vector Machines the distance to the hyperplane can be used as confidence. From these confidence values a similarity score can be derived. The same holds for rule-based decision models that work with confidence values. For decision models that do not produce a distance score or a confidence value, a separate function (e.g. a distance-based decision model) for similarity score computation need to be utilized.

Note, each position i of the comparison vector represents a pair of attributes $\{A_k, A_l\}$. For that reason, we sometimes use the notion $\vec{c}(A_k, A_l)$ instead of $\vec{c}[i]$ when it is more illustrative for the reader.

4.3.6.1. Distance-based Decision Models

Distance-based approaches for decision making can be decomposed into two steps. First, a distance (or a similarity) score between the two compared entities is computed on the basis of the comparison vector. Finally, the candidate pair is classified as a MATCH, a POSSIBLE MATCH, or an UNMATCH by comparing the computed distance (or similarity) score with two thresholds.

Let dst be a normalized distance measure, let $sim = 1 - dst$ be the corresponding similarity measure, and let $\theta_{M/P}, \theta_{P/U} \in [0, 1]$ be two thresholds with $\theta_{M/P} \geq \theta_{P/U}$, the classification of two entities e_r and e_s that is made by a distance-based decision model that utilizes dst, $\theta_{P/M}$, and $\theta_{U/P}$ is computed as:

$$\textit{classify}(e_r, e_s) = \begin{cases} \text{MATCH}, & \text{if } sim(e_r, e_s) \geq \theta_{P/M}, \\ \text{POSSIBLE MATCH}, & \text{if } \theta_{U/P} < sim(e_r, e_s) < \theta_{P/M}, \\ \text{UNMATCH}, & \text{if } sim(e_r, e_s) < \theta_{U/P}. \end{cases} \tag{4.13}$$

This principle of thresholding is graphically illustrated in Figure 4.11. Note, in the case of an automatic decision making process (no clerical review), both thresholds are identical, i.e. $\theta_{P/M} = \theta_{U/P}$.

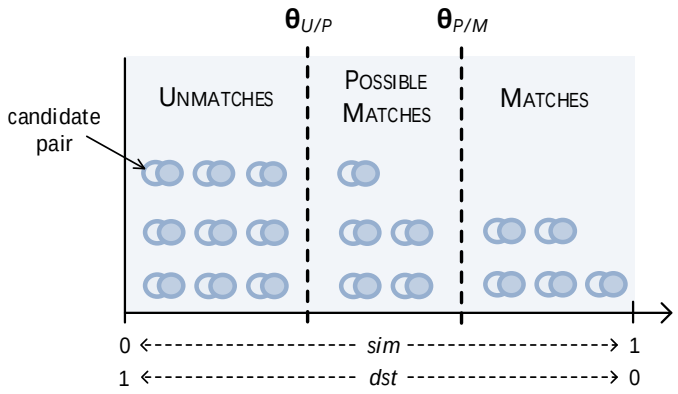


Figure 4.11.: The principle of thresholding in duplicate decision making

The simplest approach for distance computation is to concatenate all attribute values per entity to a single large string, and then to define the distance between two entities as the distance between their strings. For computing the distance between the two strings, any of the similarity measures (or distance measures respectively) that we have presented in Section 4.3.5 can be used. By doing so a separate phase of attribute value matching is not required, because the comparison vector is not used and similarity is directly computed in the decision model instead. Nevertheless, this approach is usually not very effective and a more complex distance computation is required instead.

Two interesting approaches for distance computation that are based on the comparison vector are:

- The **Average Attribute Distance** defines the distance between two entities as the average distance of their attribute values. Since some attributes are more suitable for distinguishing one entity from another than other attributes, decision quality is increased by weighting the individual attributes and hence to compute the weighted average instead of the simple average. Let $W = \{w_1, \ldots, w_{|\vec{c}|}\}$ be a weighting scheme, the average attribute distance of the two entities e_r and e_s having the comparison vector $\vec{c}$ is defined as:

$$AAD(e_r, e_s, W) = \frac{\sum_{i=1}^{|\vec{c}|} w_i \times (1 - \vec{c}(i))}{\sum_{i=1}^{|\vec{c}|} w_i} \tag{4.14}$$

  A property of the average attribute distance is that the entity distance increases and decreases linearly with the similarity of any of their attributes. In many scenarios, however, it is desirable that only some of the given similarity scores need to be high (e.g. a combination of name and phone, or a combination of name and address, or a combination of phone and address) to produce a high score for entity similarity. Dey [Dey08] and Halevy et al. [HDI12] describe an extension of the average attribute distance that aims to overcome this shortcoming by the use of logistic regression.

- The **Vector Space Distance** [DSD98] considers the normalized comparison vector $\vec{c}$ as a point in an $|\vec{c}|$ dimensional vector space. The distance of two entities is then defined as the distance of $\vec{c}$

to the point $\mathbf{1}_{|\vec{c}|} = (1, 1, \ldots, 1)$ that represents the situation of absolute similarity. Because we require a normalized distance score, the computed distance is finally divided by the greatest possible distance in that space, i.e. the distance between the point $\mathbf{1}_{|\vec{c}|}$ and the point $\mathbf{0}_{|\vec{c}|} = (0, 0, \ldots, 0)$ that represents the situation of absolute dissimilarity. The incorporation of a weighting scheme can be additional useful to take the different significance of the individual attributes into account. Let dst be a distance measure, let $W = \{w_1, \ldots, w_{|\vec{c}|}\}$ be a weighting scheme, the vector space distance of the two entities e_r and e_s having the comparison vector $\vec{c}$ is defined as:

$$\text{VSD}(e_r, e_s, W) = \frac{dst(\vec{c}, \mathbf{1}_{|\vec{c}|}, W)}{dst(\mathbf{0}_{|\vec{c}|}, \mathbf{1}_{|\vec{c}|}, W)} \tag{4.15}$$

For distance computation any vector-based distance measure as for example the *Euclidean Distance* [DD09, JH11, SB12] can be used. A weighted version of the euclidean distance is defined as:

$$dst_E(x, y, W) = \sqrt{\sum_{i=1}^{|x|} w_i \times (x - y)^2} \tag{4.16}$$

Interestingly, by using the weighted version of the *Manhattan Distance* [JH11] (also known as *City-Block Distance* [SB12]), i.e. $dst_M(x, y, W) = \sum_{i=1}^{|x|} w_i \times |x - y|$, to compute the distance between the vectors, the vector space distance corresponds to the average attribute distance that we have defined in Equation 4.14.

Example 57 *For demonstration, we consider the two entities e_1 and e_2 from the prepared sample table from Figure 4.6. Both entities as well as their comparison vector and the used weighting scheme are presented in Figure 4.12. The weighted average attribute distance results in AAD$(e_1, e_2, W) = 0.292$, and the vector space distance results in VSD$(e_1, e_2, W) = 0.208$. Consequently, the similarity computed by the first approach results in $1 - 0.292 = 0.708$ and the similarity computed by the second approach results in $1 - 0.208 = 0.782$.*

The advantage of distance-based decision models is that they are simple to understand and hence more meaningful to the user than some of the models we will present in the remainder of this section. This makes adaptations to the domain and reconfigurations as a reaction to poor detection results much more intuitive. The problem with the weighting based approaches is that the model's quality essentially depends on the used weighting scheme, but the goodness of a weighting scheme always depends on the considered domain. This makes finding an adequate weighting scheme a non-trivial but crucial task.

4.3.6.2. Rule-based Decision Models

Rule-based decision models have first been proposed by Wang and Madnick [WM89]. The underlying idea of these models is to map domain knowledge into a set of *identification rules* and then to make duplicate decisions based on these rules.

Although they are often designed in a more informal way, identification rules can be typically transformed into logical expressions of the form [NH10, Chr12]:

$$\textit{Premise} \Rightarrow \textit{Conclusion} \tag{4.17}$$

DEI	fname	lname	DoB	city	country	phone	email
e1	dejohn w.	tucker	1974-05-03	new york city	usa	(0345)233848	abc@me.com
e2	john bill	tacker	1974-03-05	new york city	usa	(0345)233848	tucker@xy.com

(a) Two sample database entities

	fname	lname	DoB	city	country	phone	email
weighting scheme W	0.4	0.4	0.2	0.1	0.1	0.2	0.2
comparison vector $\vec{c}$	0.5	5/6	1.0	1.0	1.0	1.0	0.0

(b) Weighting scheme and comparison vector

Figure 4.12.: A sample candidate pair, a weighting scheme and the pair's comparison vector

where the premise is a predicate written in the conjunctive normal form:

$$Premise = (term_{1,1} \vee term_{1,2} \vee \ldots) \wedge \ldots \wedge (term_{n,1} \vee term_{n,2} \vee \ldots) \tag{4.18}$$

and the conclusion is either MATCH, POSSIBLE MATCH, or UNMATCH. The terms used in the premise are usually atomic predicates that compares a value of the comparison vector with a constant, e.g. $\vec{c}[1] \geq 0.7$, but can also be predicates that work on single attribute values, e.g. $e_1[A_1] = X \vee e_2[A_2] = Y$.

One of the first rule-based approach has been proposed by Hernandez and Stolfo and is known as *Equational Theory* [HS95, HS98]. In Equational Theory, only positive rules, i.e. rules of the form *Premise* $\rightarrow$ MATCH, are considered. A candidate pair is classified as a MATCH, if at least one of the given rules is triggered and is classified as an UNMATCH otherwise.

Profile-based approaches [DLLH03, WNJ+08] make additionally use of negative rules, i.e. rules of the form *Premise* $\rightarrow$ UNMATCH that directly classify a candidate pair as an UNMATCH, and sometimes use maybe rules (*Premise* $\rightarrow$ POSSIBLE MATCH), i.e. rules that directly classify a candidate pair as a POSSIBLE MATCH. Whereas in Equational Theory the order in which the rules are checked is irrelevant, the checking order is significant in profile-based approaches if negative rules and positive rules are allowed to have overlapping premises, i.e. there can be cases where rules of both kinds are triggered by the same input. Moreover, by using negative rules, an UNMATCH is not implicated by the case where none of the rules is triggered and the set of rules either has to be complete, i.e. they cover all possible values of the comparison vector, or a default classification result has to be defined (e.g. *True* $\Rightarrow$ POSSIBLE MATCH or *True* $\Rightarrow$ UNMATCH).

Doan et al. [DLLH03, SLD05] propose a rule-based approach that additionally use (negative) rules for modeling constraints between the values of different attributes, as for example a rule that specifies that a six year old kid cannot have an income of $100,000$\$. In this case, the rules are not applied to the comparison vector, but directly work on the attribute values of the compared entities.

Defining an identification rule is always a trade off between accuracy and coverage. On one hand a rule should only trigger if its conclusion can be made with an high confidence (high accuracy), but on the other hand it should trigger for as many candidate pairs as possible (high coverage) in order to held the number of rules small and manageable.

Rules are typically considered to be hard, i.e. they make decisions with absolute certainty. Nonetheless, rules can be also defined to be soft by attaching a confidence score to each of them (confidences can be also assigned to set of rules). After checking all rules (negatives as well as positives) a final confidence is derived from the confidences of the triggered rules. The final confidence is then used as input to a threshold-based classification as it is known from the distance-based decision models.

Rules can be learned by a sequential covering algorithm [HKP06] or by decision trees (see Section 4.3.6.4). The problem with a rule-based decision model is that in cases where these rules cannot be automatically learned, they need to be manually designed by domain experts which is expensive in terms of time and money. Moreover, the modeled knowledge is most often only domain-specific and hence cannot be used across different applications.

Further rule-based approaches are described by Galhardas et al. [GFSS00, GFS+01], Lim et al. [LSPR93], and Low et al. [LLL01]. A similar approach that is based on knowledge pattern has been proposed by Schewe and Wang [SW12].

Example 58 *To illustrate the concept of identification rules, we consider the sample data table from Figure 4.6 and design a profile of hard identification rules that is shown in Figure 4.13. The first rule classifies a candidate pair as a* MATCH *if both entities have the same birthday, have a first name that is more similar than* 0.6*, and have a last name that is more similar than* 0.8*. The rules* R_2 *and* R_3 *are additionally defined for cases where first names and last names are mistakenly swapped. The fourth rule classifies two entities as an* UNMATCH*, if they live in different countries and their last names are less similar than* 0.5*. The fifth rule handles the case where the first name and the last name of the first person are both included in the email address of the second person. Finally, the default rule classifies all candidate pairs for which none of the previous rules has been triggered as* UNMATCHES.

$\boldsymbol{R}_1$:	$(\vec{c}(\text{fname}, \text{fname}) > 0.6) \wedge (\vec{c}(\text{lname}, \text{lname}) > 0.8) \wedge (\vec{c}(\text{DoB}, \text{DoB}) = 1.0)$	$\Rightarrow$ MATCH
$\boldsymbol{R}_2$:	$(\vec{c}(\text{fname}, \text{lname}) > 0.6) \wedge (\vec{c}(\text{lname}, \text{fname}) > 0.8) \wedge (\vec{c}(\text{DoB}, \text{DoB}) = 1.0)$	$\Rightarrow$ MATCH
$\boldsymbol{R}_3$:	$(\vec{c}(\text{lname}, \text{fname}) > 0.6) \wedge (\vec{c}(\text{fname}, \text{lname}) > 0.8) \wedge (\vec{c}(\text{DoB}, \text{DoB}) = 1.0)$	$\Rightarrow$ MATCH
$\boldsymbol{R}_4$:	$(\vec{c}(\text{lname}, \text{lname}) \leq 0.5) \wedge (\vec{c}(\text{country}, \text{country}) \neq 1.0)$	$\Rightarrow$ UNMATCH
$\boldsymbol{R}_5$:	$(\vec{c}(\text{lname}, \text{email}) > 0.5) \wedge (\vec{c}(\text{fname}, \text{email}) > 0.5)$	$\Rightarrow$ MATCH
$\boldsymbol{R}_{\text{default}}$:	True	$\Rightarrow$ UNMATCH

Figure 4.13.: Sample profile that consists of five hard identification rules

4.3.6.3. Probabilistic Decision Models

The statistical concept of probabilistic decision models was initially introduced by Newcombe et al. [NKAJ59, NK62] and has first been formalized by Fellegi and Sunter [FS69]. The underlying idea of probabilistic decision models is to classify a candidate pair based on the probability whether its comparison vector is typical for a MATCH or is typical for an UNMATCH.

Let $Prob(C \mid \vec{c})$ be the conditional probability that an entity pair with a comparison vector is identical to $\vec{c}$ belongs to the matching class $C \in \{\text{MATCH}, \text{UNMATCH}\}$. A candidate pair $\{e_r, e_s\}$ is classified by the following rule:

$$\textit{classify}(e_r, e_s) = \begin{cases} \text{MATCH}, & \text{if } Prob(\text{MATCH} \mid \vec{c}) \leq Prob(\text{UNMATCH} \mid \vec{c}), \\ \text{UNMATCH}, & \text{otherwise .} \end{cases} \tag{4.19}$$

Using Bayes Theorem [CA03] and the binary logarithm to keep the ranges of the computed ratios small, this rule can be transformed into:

$$\textit{classify}(e_i, e_j) = \begin{cases} \text{MATCH}, & \text{if } \log_2\left(\frac{Prob(\vec{c}|\text{MATCH})}{Prob(\vec{c}|\text{UNMATCH})}\right) \leq \log_2\left(\frac{Prob(\text{UNMATCH})}{Prob(\text{MATCH})}\right), \\ \text{UNMATCH}, & \text{otherwise .} \end{cases} \tag{4.20}$$

The ratio $W_{\vec{c}} = \log_2(\frac{Prob(\vec{c}|\text{MATCH})}{Prob(\vec{c}|\text{UNMATCH})})$ is called the *matching weight* (also known as *likelihood ratio* [EIV07]) of this comparison vector and the ratio $\log_2(\frac{Prob(\text{UNMATCH})}{Prob(\text{MATCH})})$ corresponds to the threshold that leads to the minimal classification error [EIV07].

The problem with this statistical approach is that the probabilities $Prob(\text{MATCH})$ and $Prob(\text{UNMATCH})$ as well as the conditional probabilities $Prob(\vec{x} \mid \text{MATCH})$ and $Prob(\vec{x} \mid \text{UNMATCH})$ have to be known for any possible comparison vector $\vec{x}$. Since these values are usually not available, they need to be estimated using a set of labeled training data.

A common way to reduce computational complexity and to reduce the dependency to large training data sets is to assume conditional independence between the values from different vector positions. By doing so, the conditional probability $Prob(\vec{c} \mid C)$ of a comparison vector $\vec{c}$ and a matching class C can be computed as:

$$Prob(\vec{c} \mid C) = \prod_{i=1}^{|\vec{c}|} Prob(\vec{c}[i] \mid C) \tag{4.21}$$

It is obvious that the number of probabilities that need to be computed (or estimated respectively) essentially depends on the number of possible comparison vectors. Since this number is unmanageable in cases where the similarity scores of the comparison vector are real-valued, complexity is usually further reduced by discretizing these scores. A discretized comparison vector is also called a probabilistic matching pattern [Tal11]. After discretizing the comparison vectors, the matching patterns are used as input for the decision model instead of the original comparison vectors.

In the extreme case, the real-valued scores are discretized into the binary domain $\{0, 1\}$ where the value 0 represent the case of disagreement and the value 1 represents the case of agreement. Let $\theta \in [0, 1]$ be a threshold, two attribute values are considered to agree if their similarity is above θ and are considered to be disagree otherwise, i.e. the i-th position of the probabilistic matching pattern *PMP* is defined as:

$$\textit{PMP}[i] = \begin{cases} 1, & \text{if } \vec{c}[i] \geq \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{4.22}$$

Discretization makes probabilistic decision model useable in practice. However, the distinction into 0 (disagreement) and 1 (agreement) is not exactly enough for many use cases. Moreover, the number of

possible patterns is only restricted to $2^{|\vec{c}|}$ which is still a lot for large vectors. Thus, on the one hand, the stronger the scores are discretized, the less accurate the resultant patterns become because the more entity pairs are represented by the same pattern. On the other hand, the number of possible patterns grows exponentially to the size of the domains into which the scores are discretized. Thus, the strength of discretization is always a trade-off between accuracy and complexity.

For discretization a variety of functions can be used and it makes sometimes sense to use different functions for different positions of the comparison vector, because the scores of some positions are more discriminating than others. For example, the similarity of persons' last names is more important for decision making than the similarity of their residences. Thus, the similarity scores of the former are discretized into a larger domain than the similarity scores of the latter.

In the case of a binary discretization, Naive Bayes can be further simplified by defining m_i as the probability that two duplicates agree at pattern position i and by defining u_i as the probability that two non-duplicates agree at pattern position i. Based on these probabilities, the matching weight for the pattern position i is defined as:

$$W_{\vec{c}}(i) = \begin{cases} \log_2\left(\frac{m_i}{u_i}\right), & \text{if } \mathit{PMP}[i] = 1, \\ \log_2\left(\frac{(1-m_i)}{(1-u_i)}\right), & \text{if } \mathit{PMP}[i] = 0. \end{cases} \tag{4.23}$$

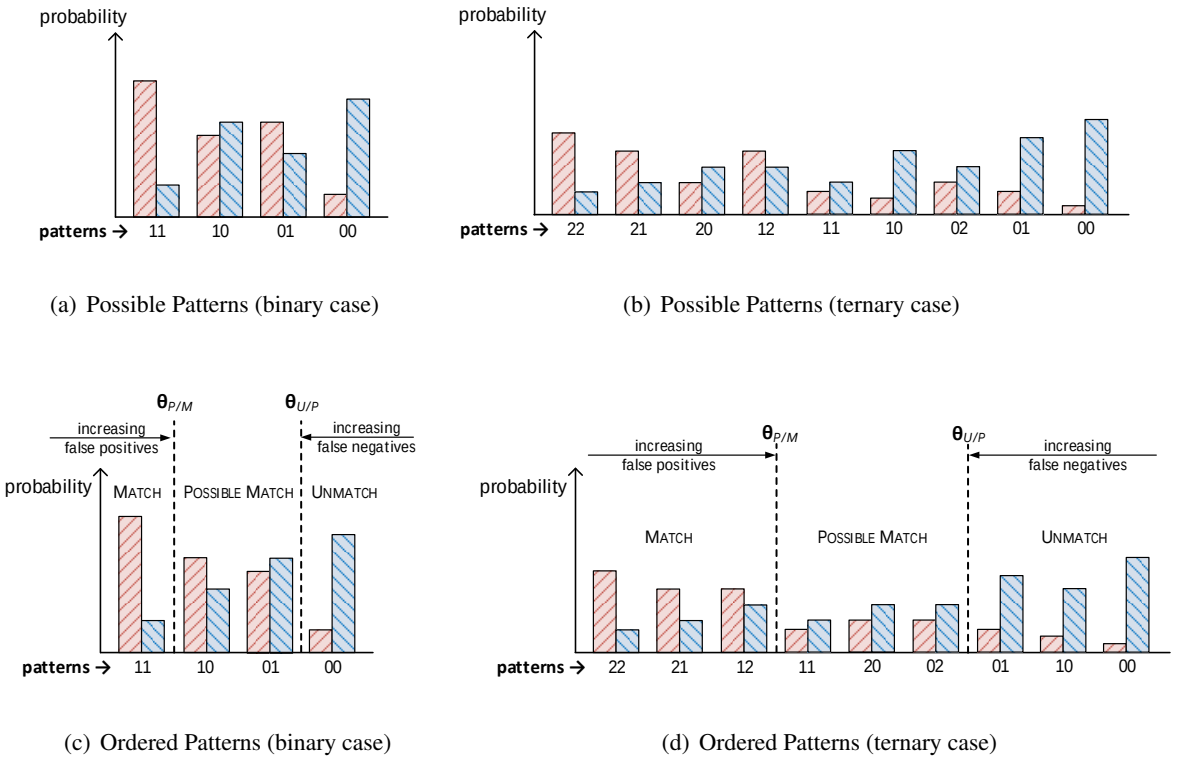
The matching weight of the whole pattern can then be computed as $W_{\vec{c}} = \sum_{i=0}^{|\vec{c}|} W_{\vec{c}}(i)$.

Example 59 *For illustration, we consider the sample table schema from Figure 4.6. We use a comparison vector that consists of two similarity scores. The first score represents the similarity between the entities' first names and the second score represents the similarity between the entities' last names.*

In the first example, we use a binary discretization. Two first names agree if their similarity score is greater than $\theta_{fn} = 0.7$ *and two last names agree if their similarity score is greater than* $\theta_{ln} = 0.6$*. As presented in Figure 4.14(a), the domain of possible patterns is of size four, namely '11', '10', '01', and '00'.*

Let us assume that we have made a statistical analysis on labeled training data. The resultant probabilities for all four possible patterns are depicted in Figure 4.14(a) by two bars where the first bar represents the probability that two duplicate entities have that pattern and the second bar represents the probability that two non-duplicates have that pattern. In the second step, we order the patterns by their matching weight (see Figure 4.14(c)) and use the two thresholds $\theta_{M/P} \geq \log_2(\frac{Prob(\text{UNMATCH})}{Prob(\text{MATCH})})$ *and* $\theta_{P/U} < \log_2(\frac{Prob(\text{UNMATCH})}{Prob(\text{MATCH})})$ *to separate the patterns into the sets of* MATCHES, POSSIBLE MATCHES, *and* UNMATCHES. *The closer a pattern is to one of the two thresholds the higher is the risk to produce a false positive or the higher is the risk to produce a false negative respectively. Since we partition the set of possible patterns by their matching weight, each pattern is associated with a matching class and a new candidate pair can now be simply classified by computing its pattern.*

To demonstrate the differences that result from increasing the domain of possible patterns, we consider a second example that is presented in Figure 4.14(b) and Figure 4.14(d), where we use a ternary discretization, i.e. each similarity score is discretized to 0, 1, *or* 2*. The similarity of two first names is represented by* 2 *if it is greater than* $\theta_{fn1} = 0.7$*, is represented by* 0 *if it is lower than* $\theta_{fn2} = 0.3$*, and*



(a) Possible Patterns (binary case)

(b) Possible Patterns (ternary case)

(c) Ordered Patterns (binary case)

(d) Ordered Patterns (ternary case)

Figure 4.14.: The concept of probabilistic decision model illustrated by binary and ternary patterns

is represented by 1 *otherwise. The similarity scores for last name are discretized accordingly by using the thresholds* $\theta_{ln1} = 0.8$ *and* $\theta_{ln2} = 0.4$*. We present five candidate pairs along with their comparison vectors in Table 4.5. All pairs have different comparison vectors that are not even similar to each other. Whereas the five candidate pairs are only assigned to two different patterns in the binary case, each of them is assigned to another pattern in the ternary case. Thus, the patterns resulting from the ternary discretization represent the original comparison vectors much better than the patterns that result from the binary discretization. On the other hand this example illustrates the dramatic increase of possible patterns (*9 *instead of* 4*), although we only consider comparison vectors of size two. For each pattern two conditional probabilities are required. Thus, the number of probabilities that need to be computed increases from* 8 *to* 18*. If we assume a conditional independence between the similarities of first name and last name, the number of required probabilities can be decreased from* $18 = 2 \times 3^2$ *to* $12 = 2 \times 3 \times 2$*, but is still one and a half as much as in the binary case.*

Fellegi and Sunter [FS69] as well as Jaro [Jar89] primarily consider the case of a binary discretization. Porter and Winkler [PWCC97, Win90, WT91] extend these considerations to larger domains of patterns. For the case of Naive Bayes, Dempster et al. [DLR77] suggests to use the *expectation maximization* algorithm to estimate the conditional probabilities of the individual pattern positions. To relax this assumption of attribute independence, Winkler et al. present an unsupervised estimation model that is based on *general expectation maximization* [Win93] and propose a semi-supervised model that com-

Candidate Pair	Comparison Vector	Binary Pattern	Ternary Pattern
$\{e_1, e_2\}$	$\langle 0.4, 1.0\rangle$	$\langle 0, 1\rangle$	$\langle 1, 2\rangle$
$\{e_1, e_3\}$	$\langle 0.7, 0.7\rangle$	$\langle 1, 1\rangle$	$\langle 2, 1\rangle$
$\{e_1, e_4\}$	$\langle 0.9, 1.0\rangle$	$\langle 1, 1\rangle$	$\langle 2, 2\rangle$
$\{e_1, e_5\}$	$\langle 0.1, 0.6\rangle$	$\langle 0, 1\rangle$	$\langle 0, 1\rangle$
$\{e_1, e_6\}$	$\langle 0.6, 0.6\rangle$	$\langle 0, 1\rangle$	$\langle 1, 1\rangle$

Table 4.5.: Sample candidate pairs along with their comparison vectors and their probabilistic matching patterns

bines the Fellig-Sunter Method with Bayesian Networks [Win02]. Similar combination approaches have been proposed by Fortini et al. [FLNS01] and Herzog et al. [HSW07].

Since agreements on less frequent attribute values are usually better indications for duplicates than agreements on high frequent values, Winkler and Thibaudeau [WT91] as well as Herzog et al [HSW07] consider the incorporation of attribute value frequency into weight estimation. An adaptation that includes a special handling of null values has been made by du Bois [NDB69]. A cost-based extension of the Fellegi-Sunter Method that enables an assignment of specific costs to the four error classes *true positives*, *false positives*, *true negatives*, and *false negatives* was firstly considered by Tepping [Tep68] and has been later formalized by Verykios et al. [VME03, VM04].

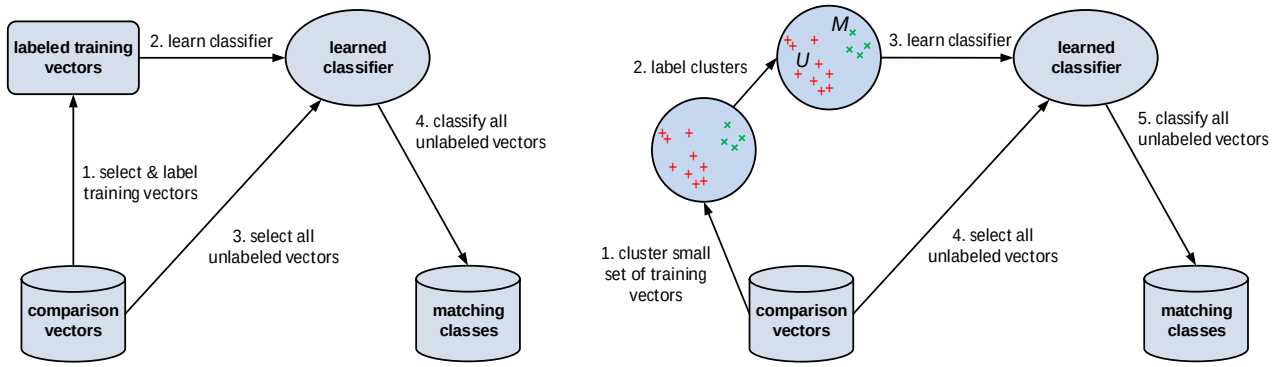
An extensive discussion on probabilistic decision models is given in [Mag08]. A mathematically less detailed, but very illustrative presentation can be found in the book published by Talburt [Tal11].

4.3.6.4. Learning-based Decision Models

The idea of learning-based decision models is to construct a classifier using a conventional learning algorithm.

Unsupervised Learning-based Decision Models Unsupervised learning [HTF11, BB10] is characterized by the fact that all the processed data is unlabeled, and so the learning algorithm has to directly infer the desired properties from the input data without an external prompting by a supervisor. The most intuitive approach for an unsupervised learning based decision model is to cluster the candidate pairs by their comparison vectors into two clusters as it can be realized by using the k-means algorithm [HW79]. Finally, one of the resultant clusters is marked as the set of MATCHES and the other is marked as the set of UNMATCHES where marking can be done based on the closeness of the clusters to the two points $\mathbf{0}_{|\vec{c}|} = (0, 0, \ldots, 0)$ and $\mathbf{1}_{|\vec{c}|} = (1, 1, \ldots, 1)$. Learning three clusters (an additional one for the POSSIBLE MATCHES) has shown to be less effective, because POSSIBLE MATCHES can be very dissimilar [GB06]. Instead Gu and Baxter recommend to build a class of POSSIBLE MATCHES afterwards by selecting the candidate pairs whose comparison vectors have similar distances to the centroids of both clusters.

An unsupervised approach that has been implemented in the TAILOR system [EEV02] clusters all candidate pairs by their comparison vector using an unconstrained clustering algorithm, i.e. an algorithm where the number of clusters is not predefined. Then one candidate pair of each cluster is manually classified by a domain expert and the resultant matching class is adopted to all candidate pairs of the same



(a) Decision model based on supervised learning

(b) Decision model based on the hybrid learning approach

Figure 4.15.: Principle Concepts of supervised learning decision models and hybrid learning decision models

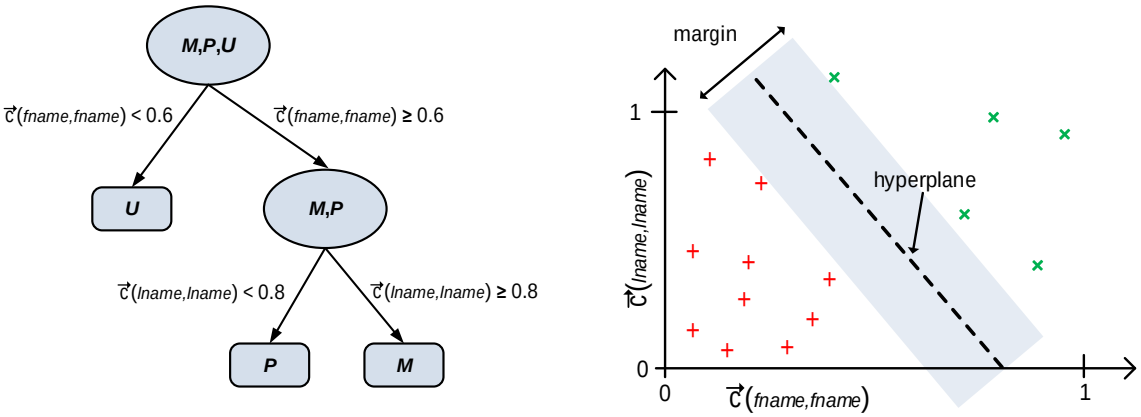
cluster. The advantage of this approach is that the number of clusters can be larger than in the approach presented above and thus the boundaries between the clusters is usually more clear. Its shortcoming is that the approach is only semi-automatic and requires the contribution of domain experts.

It is important to note that by using an unsupervised learning approach, the decision model is not applied to each candidate pair separately, but to the whole candidate pair space at a stroke.

Supervised Learning based Decision Models In contrast to unsupervised learning techniques, supervised learning techniques require the existence of labeled training data. This training data set is used to learn a classifier that can then be utilized to classify unlabeled data of the same domain. The principle of a decision model that is based on a supervised learning algorithm is presented in Figure 4.15(a). First a sample set of candidate pairs along with their comparison vectors is selected and manually labeled (optionally an already labeled set of training data can be used). Second, the supervised learning algorithm is used to learn a classifier based on the labeled data. Then the remaining entity pairs are classified using the learned classifier.

Two supervised learning algorithms that are primarily used in the context of duplicate detection are *Decision Trees* and *Support Vector Machines*.

- A **Decision Tree** learning algorithm starts with the set of all labeled training elements and splits this set step by step into disjoint subsets until each of these subsets only contains equally labeled elements. Usually, the information gain is used as splitting criterion. Popular decision tree learning algorithms are CART [BFOS84], ID3 [Qui83], and its successor C4.5 [Qui93]. The result of such a learning algorithm is a tree structure where the inner nodes represent sets of possible classification results, the leaf nodes represent single classification results, and the edges represents boolean conditions that are based on the comparison vectors. Since the tree should be able to classify each possible input element, the conditions that are represented by the outgoing edges of one node must cover the complete range of possible inputs, i.e. for each possible comparison vector the condition of one of these edges must be satisfied. Figure 4.16(a) shows a sample decision tree that has been produced by the CART algorithm.



(a) Sample decision tree

(b) Sample support vector machine (linear kernel)

Figure 4.16.: Samples of a Decision Tree and a Support Vector Machine

In principle, each path from the root to any leaf in a decision tree can be considered as an own identification rule. All root-leaf paths together form a profile. Since a decision tree is deterministic, i.e. for each input exact one path is chosen, the profile's rules have disjoint conditions and hence represent a profile with an irrelevant checking order.

- **Support Vector Machines** [SS02] represent a class of algorithms for learning binary classifiers. The algorithm considers the numerical training elements as points in a multidimensional vector space and constructs a hyperplane that separates the training elements into two groups so that all elements in the same group are equally labeled and the distance of the nearest element to the hyperplane (the so called margin) is maximized.

  Since a hyperplane is always linear and the most element sets cannot be linearly separated, the Support Vector Machine uses the kernel trick. The kernel trick is based on the idea that transforming the vector space (and with it the training elements) into a higher dimensional space which makes a linear separation possible. The shortcoming of this approach is twofold. First the transformation is computational expensive and second the re-transformation most often produces a function that cannot be simply represented. For that reason, the transformation is realized by specific kernel functions that guarantee that the re-transformed function is less complex.

  A set of training elements as well as the hyperplane that has been produced by a linear kernel are graphically presented in Figure 4.16(b).

  For making duplicate decisions, the Support Vector Machine was firstly used by Bilenko et al. [BM03]. Note, the set of POSSIBLE MATCHES can be introduced afterwards by defining an uncertainty region around the hyperplane.

Of course, other supervised learning algorithm such as Bayesian Networks [Was04, NJ09], Nearest Neighbor Classifier [Chr08a, HTF11], or Neural Networks [HTF11, Kan11] can be (and already have been) used for learning classifiers as well.

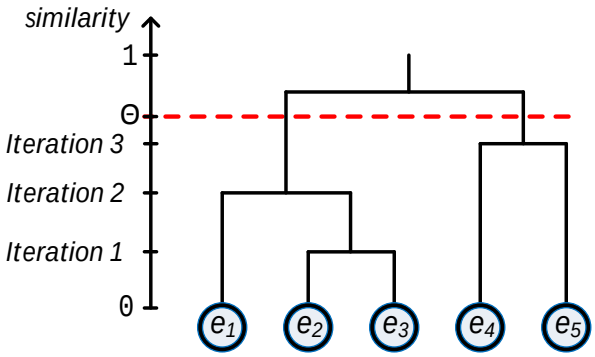


Figure 4.17.: Sample of an agglomerative hierarchical clustering on five entities

The accuracy of a classifier trained by a supervised learning algorithm essentially depends on the used training data. Sarawagi and Bhamidipaty [SB02, SBKM02] propose an active learning technique that iteratively selects a small set of candidate pairs for manual labeling that are considered to be most ambiguous and hence their labeling is expected to improve the accuracy of the preliminary learned classifier best.

Hybrid Decision Model The idea of an hybrid decision model is to combine a supervised learning algorithm with an unsupervised learning algorithm [Chr08a] or to combine a supervised learning algorithm with a distance-based decision models [CKLS01, Chr08a]. The principle of a hybrid decision model is depicted in Figure 4.15(b). First the unsupervised learning algorithm (or the distance-based decision model respectively) is used to label a small set of candidate pairs. Then the labeled pairs are used as training data for the the supervised learning algorithm. Christen [Chr08a, Chr08b] analyzes the effectiveness of different combinations by evaluating them on real-world databases and synthetic databases.

4.3.6.5. Clustering-based Decision Models

The underlying idea of a clustering-based decision model is to directly partition the input set of entities (not comparison vectors!) into a set of clusters [HDI12]. A popular representative of such decision models is based on the agglomerative hierarchical clustering [ZKF05] and is performed in an iterative fashion. At the beginning, each entity is considered as an own cluster. Then, in each iteration all clusters are pairwise matched and the most similar cluster pair is merged to a single cluster. The process stops if a predefined number of iterations is reached or the highest similarity score that has been computed in the current iteration is smaller than a predefined threshold. In the first iteration, the similarity between two clusters corresponds to the similarity between two entities. Thus any of the distance computation approaches that have been presented in Section 4.3.6.1 can be used. The similarity between two larger clusters can be defined in different ways. Two of the most interesting methods are:

- The **Average Link** method defines the similarity between two clusters as the average similarity between two of their entities. Let C_r, C_s be two clusters and let sim be a measure for entity

similarity, the average link similarity between C_r and C_s is defined as:

$$ALSim(C_r, C_s) = \frac{\sum_{e_r \in C_r} \sum_{e_s \in C_s} sim(e_r, e_s)}{|C_r| \times |C_s|} \tag{4.24}$$

- The **Canonical Entity** method creates an entity that represents all the entities of one cluster and defines the similarity between two clusters as the similarity between their canonical entities. A canonical entity is usually created by merging the clusters' entities on attribute basis, i.e. per attribute a new value is created by either aggregating the values of all entities or by selecting one of them. Selection can be done by a voting strategy that takes the most often occurring non-null value. Aggregation can be done by string operations (e.g. the name values 'Bill Tucker' and 'John B. Tucker' are aggregated to the value 'John Bill Tucker') or by computing the average value for numbers. Thus, this method incorporates the process of duplicate merging (see Section 4.4) into the detection process in order to increase the process's accuracy.

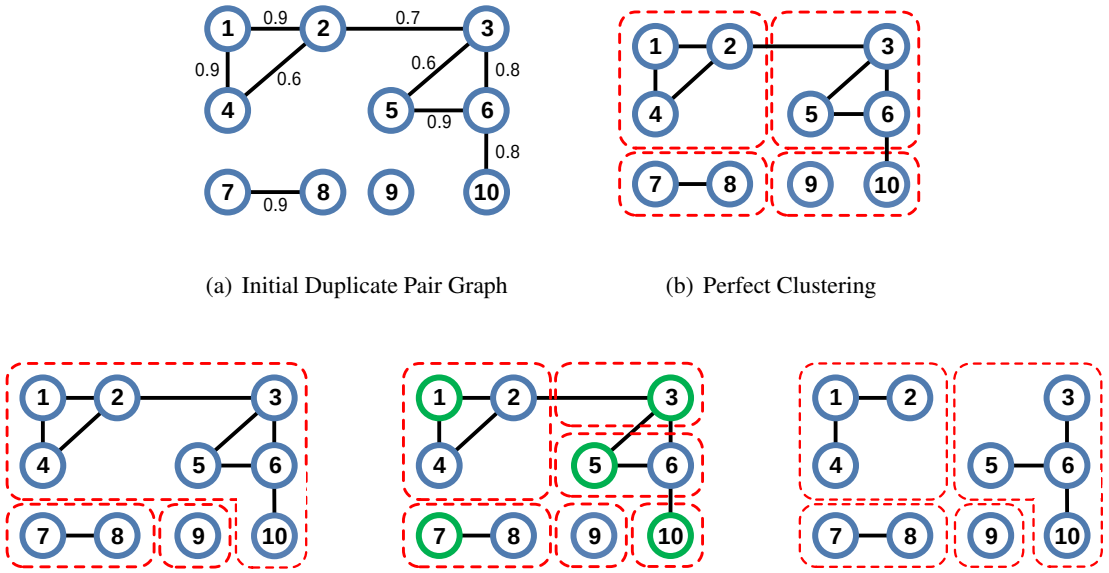
Two other similarity computation methods that are listed in [HDI12] are the *Single Link* method and the *Complete Link* method. The first defines the similarity between two clusters as the highest similarity between two of their entities and the second defines the similarity between two clusters as the lowest similarity between two of their entities. However, both methods are identical with a combination of a distance-based decision model and a specific approach for the phase of duplicate clustering (see Section 4.3.7). The single link method corresponds to applying the *Partitioning based on Connected Components* to the duplicate pair graph and the complete link method is identical to applying a duplicate clustering approach that divides the duplicate pair graph into its completely connected components.

4.3.7. Duplicate Clustering

In the final phase of the duplicate detection process, the pairwise made duplicate decisions need to be combined to a global result, i.e. the cluster-disjoint clustering of the database entities. Thus, the input to this phase is a set of database entities and a set of pairwise duplicate decisions for some entity pairs. The output is a cluster-disjoint clustering of all database entities. Notice, the clustering algorithm used for this phase needs to be unconstrained because we do not know the number of resultant clusters in advance.

We will illustrate this phase by three simple approaches: the *Partitioning based on Connected Components*, the *Partitioning based on Centers*, and the approach of *Edge Removal*. All three approaches only take those entity pairs into account that have been classified as a MATCH and use them to initially build a so-called *duplicate pair graph* [NH10]. The duplicate pair graph is a weighted graph that has a node per database entity. Two nodes are connected by an edge, if the corresponding entities have been classified as a MATCH. The weight of an edge is the similarity of the corresponding entities. A sample duplicate pair graph is illustrated in Figure 4.18(a). The perfect clustering of this example has four clusters and is depicted in Figure 4.18(b).

Partitioning based on Connected Components [HS98] is the simplest clustering approach, because it just considers each connected component of the duplicate pair graph as an own cluster. This approach corresponds to an algorithm that forms the transitive closure of all proposed MATCHES and is used in many duplicate detection approaches.



(a) Initial Duplicate Pair Graph (b) Perfect Clustering

(c) Result: Connected Components (d) Result: Center (e) Result: Edge Removal

Figure 4.18.: Sample scenario for duplicate clustering along with three illustrating results

From applying this approach to our sample duplicate pair graph, a clustering with three clusters results (see Figure 4.18(c)). Building the transitive closure can group sets of entities together in the same cluster even if the evidence that they represent the same real-world entity is rather low. For instance, consider the two entity sets $\{e_1, e_2, e_4\}$ and $\{e_3, e_5, e_6, e_9\}$ from the given sample graph. Across these two sets only the entities e_2 and e_3 have been classified as a MATCH. The other pairs (11 in numbers) either have not been classified because of the candidate pair space reduction or have been classified as an UNMATCH. Thus, using this approach implies a high risk of producing false positives (15 in our example). To reduce that risk, this approach should be combined with a restrictive decision model that classifies an entity pair only as a MATCH if it is somewhat confident in this decision.

Since the approach based on connected components tends to produce too many false positives, other clustering approaches aim to divide the connected components into smaller clusters.

- **Partitioning based on Centers** [HGI00] is an approach that selects for each connected component some nodes as centers and then puts each node into the cluster of its closest center. One strategy for center selection is to sort all edges by their similarity in descending order and then to scan the sorted list from the top to the bottom. Each scanned edge falls in one of the following four cases: (a) both nodes are already assigned to a cluster, (b) only one node is already assigned to a cluster, but is not a center, (c) only one node is already assigned to a cluster and has been selected as a center, or (d) both nodes have not been assigned to a cluster. In the first case, nothing is done because both nodes are already assigned to a cluster and if assigned once the cluster of a node never changes. In the second case, the node that has not been assigned to a cluster so far is selected as a new center. In the third case, the node that has not been assigned to a cluster so far is added to the cluster of the other node. In the last case, one of the two nodes is selected as a new

center and the other node is added to this center's cluster. It can be easily seen, that this strategy of center selection and hence the final clustering is affected by the strategy that sorts edges with equivalent similarities and is affected by the strategy that selects one node as a new center when both nodes have not been assigned to a cluster so far. Since for both strategies often a random function is used, this effect can lead to non-deterministic results, i.e., the same process applied to the same duplicate pair graph can lead to different clusterings. Moreover, experimental studies showed that the approach of Partitioning based on Centers often produces many small clusters and thus tends to produce many false negatives. An extension that diminishes both effects by finally merging clusters with similar centers has been proposed by Hassanzadeh et al. [HM09, HCML09].

- The approach of **Edge Removal** [NH10] iteratively removes the edge with the lowest weight from the initial duplicate pair graph until a stop criterion is satisfied and then builds the connected components of the modified graph. A sample stop criterion could be that the graph does not contain a transitivity path (the shortest path between two nodes) being longer than a certain number of edges. Since paths are restricted to components, edge removal can be restricted to components as well.

Example 60 *For illustration, we reconsider the duplicate pair graph from Figure 4.18(a). If we always select the node with the lower index as a center in the fourth scanning case, independent from the sorting strategy the Partitioning based on Centers always produces the clustering that is depicted in Figure 4.18(d). This clustering has six clusters (the center nodes are colored green). Due to the clusters being smaller than the clusters resulting from the approach of Partitioning based on Connected Components, the number of false positives has been reduced to zero. On the other hand three false negatives have been produced.*

Using the stop criterion that no transitivity path is longer than two edges, the approach of edge removal modifies the duplicate pair graph from Figure 4.18(a) by removing the edges $\{e_3, e_5\}$*,* $\{e_2, e_4\}$*, and* $\{e_2, e_3\}$ *in exact this order. The final clustering depicted in Figure 4.18(e) has three clusters. Compared to the Partitioning based on Connected Components, the number of false positives has been reduced from fifteen to three without producing any additional false negative.*

Partitioning based on Connected Components and Partitioning based on Centers belong to the class of *single-path clustering* algorithms that perform the final clustering by iterating over the edges of the duplicate pair graph only once. Thus, the complexity of these approaches is linear to the number of proposed MATCHES and therefore scales even for large sets of database entities (always provided that the number of proposed MATCHES does not escalate). The complexity of the approach of edge removal essentially depends on the stop criterion. However, if the computation of the stop criterion is less complex, the total complexity of this approach is similar to the one of the single-path algorithms.

Further approaches that can be utilized for duplicate clustering are the *Star Clustering* algorithm [APR04], the algorithms of the *Ricochet family* [WB09], *Correlation Clustering* [BBC04, DEFI06, ACN08, ARS09], *Markov Clustering* [vD09], *Cut Clustering* [FTT03], and *Articulation Point Clustering* [THCS90, BCKT07]. Experimental evaluations of some of these clustering algorithms can be found in [HCML09, Har12]. Note, these approaches are no single-path algorithms and therefore are computationally more expensive than the three approaches presented above. Moreover, some of these approaches

(star-clustering, articulation point clustering, and some algorithms of the ricochet family) can produce non-cluster-disjoint clusterings. In such cases, appearing overlaps must be eliminated afterwards either by clerical reviews or by an automatic solution.

An interesting approach has been proposed by Chaudhuri et al. [CGM05]. In contrast to the duplicate pair graph, the approach only considers the similarities between two entities and not their matching class. The final clustering is then created based on the two properties *sparse neighborhood* and *compact set* so that all entities of one cluster are high similar (compact set) and all entities of different clusters are less similar (sparse neighborhood). By doing so this approach does not use the same two thresholds to classify all candidate pairs based on their similarities, but uses two individual thresholds per candidate pair instead.

Note, since a meaningful clustering approach would never assign two entities to the same cluster if the duplicate-pair graph does not contain a path between them, all meaningful duplicate clustering solutions are always dominated by the clustering that result from computing the connected components of the duplicate pair graph.

4.3.8. Quality Evaluation

As mentioned before, designing an appropriate duplicate detection process is always a trade-off between effectiveness and efficiency. The efficiency of a duplicate detection process can be simply rated by its processing time and the size of the required storage. The effectiveness of a duplicate detection process is evaluated by comparing its resultant clustering with a benchmark clustering (called as gold standard $\mathcal{C}_{\text{gold}}$) that is considered to be perfect. Usually, the effectiveness of a duplicate detection process is considered as its quality. The meaning of quality is not universal and depends on the considered use case, because sometimes a not detected duplicate is much worse than an incorrectly classified duplicate and sometimes the contrary is the case. Moreover, even if both errors are considered to be equally poor it is generally not completely clear which quality measure represents the intended meaning of quality best, because the similarity to the gold standard can be defined in several ways.

Existing quality measures for deterministic duplicate detection [MWGM10, Tal11, Chr12, MNHG12] can be classified into pair-based evaluation measures and cluster-based evaluation measures. Measures of the first class treat duplicate detection as a binary classification problem, i.e. each entity pair is classified as a MATCH or an UNMATCH, and hence define quality on the pairwise duplicate decisions made. In contrast, measures of the second class rate detection quality as the similarity between two clusterings, i.e. the detection result and the gold standard.

4.3.8.1. Pair-based Evaluation

Iterative approaches for duplicate detection [NH10] are based on matching entities in a pairwise fashion. For that reason, the quality of an iterative duplicate detection process is usually measured by the quality of the pairwise duplicate decisions that have been made by this process.

The set of all entity pairs that are modeled in a clustering $\mathcal{C}$ is defined as:

$$\mathit{Pairs}(\mathcal{C}) = \{\{e_r, e_s\} \mid e_r, e_s \in rng(\mathcal{C}), e_r \neq e_s\} \tag{4.25}$$

The set of pairwise MATCHES (short $M(\mathcal{C})$) and the set of pairwise UNMATCHES (short $U(\mathcal{C})$) that are implicated by a cluster-disjoint clustering $\mathcal{C}$ are defined as:

$$M(\mathcal{C}) = \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\mathcal{C}), \exists C \in \mathcal{C} \colon \{e_r, e_s\} \subseteq C\} \tag{4.26}$$

$$= \bigcup\nolimits_{C \in \mathcal{C}} \textit{Pairs}(\{C\}) \tag{4.27}$$

$$U(\mathcal{C}) = \{\{e_r, e_s\} \mid \{e_r, e_s\} \notin M(\mathcal{C})\} \tag{4.28}$$

$$= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\mathcal{C}), \nexists C \in \mathcal{C} \colon \{e_r, e_s\} \subseteq C\} \tag{4.29}$$

$$= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\mathcal{C}), \exists C_p, C_q \in \mathcal{C} \colon e_r \in C_p, e_s \in C_q, C_p \neq C_q\} \tag{4.30}$$

It is important to note that each entity pair is either a MATCH or an UNMATCH and hence it holds that $\textit{Pairs}(\mathcal{C}) = M(\mathcal{C}) \cup U(\mathcal{C})$. In standard duplicate detection scenarios, the number of UNMATCHES exceeds the number of MATCHES by far. Thus, computing the exact set of UNMATCHES can be computational very expensive and hence need to be avoided. For that reason, an entity pair $\{e_r, e_s\} \in \textit{Pairs}(\mathcal{C})$ is usually only logically assigned to $U(\mathcal{C})$ by not being in $M(\mathcal{C})$, i.e. by using the equation $U(\mathcal{C}) = \textit{Pairs}(\mathcal{C}) - M(\mathcal{C})$.

To shorten the formulas, we sometimes denote a quality score $\textit{QM}(\mathcal{C}, \mathcal{C}_{\text{gold}})$ that result from evaluating the clustering $\mathcal{C}$ on the clustering $\mathcal{C}_{\text{gold}}$ by using the quality measure *QM* simply as *QM* as long as $\mathcal{C}$ and $\mathcal{C}_{\text{gold}}$ are clear from context.

Measures for pair-based quality evaluation are based on the set of *Pair True Positives* (short pTP), the set of *Pair False Positives* (short pFP), and the set of *Pair False Negatives* (short pFN):

$$p\text{TP}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = M(\mathcal{C}) \cap M(\mathcal{C}_{\text{gold}}) \tag{4.31}$$

$$p\text{FP}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = M(\mathcal{C}) \cap U(\mathcal{C}_{\text{gold}}) = M(\mathcal{C}) - M(\mathcal{C}_{\text{gold}}) \tag{4.32}$$

$$p\text{FN}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = U(\mathcal{C}) \cap M(\mathcal{C}_{\text{gold}}) = M(\mathcal{C}_{\text{gold}}) - M(\mathcal{C}) \tag{4.33}$$

The most commonly used pair-based quality measures are:

- The **Number of False Decisions** is the size of the set of *Pair False Decisions* (short pFDec) and hence is the total number of entity pairs that are incorrectly classified as an UNMATCH or are incorrectly classified as a MATCH respectively:

$$p\text{FDec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = p\text{FP}(\mathcal{C}, \mathcal{C}_{\text{gold}}) \cup p\text{FN}(\mathcal{C}, \mathcal{C}_{\text{gold}}) \tag{4.34}$$

$$\Rightarrow |p\text{FDec}(\mathcal{C}, \mathcal{C}_{\text{gold}})| = |p\text{FP}(\mathcal{C}, \mathcal{C}_{\text{gold}})| + |p\text{FN}(\mathcal{C}, \mathcal{C}_{\text{gold}})| \tag{4.35}$$

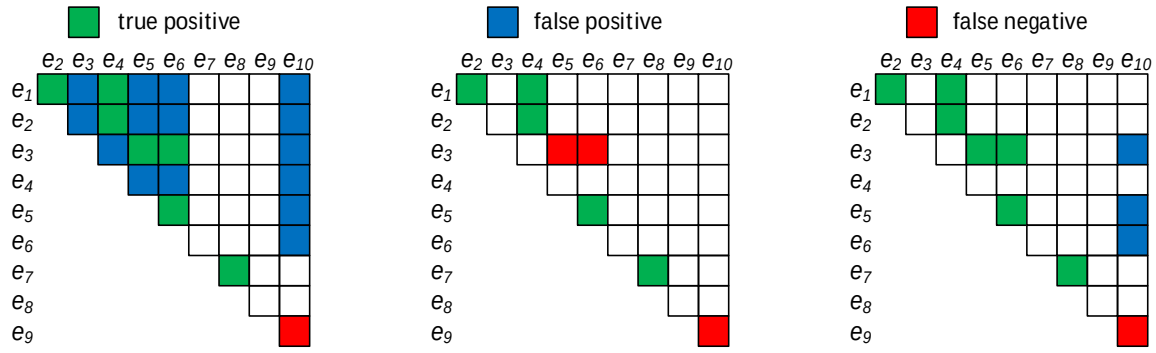
- The **Pair Recall** (short pRec) is the amount of true duplicates that are correctly classified as a MATCH:

$$p\text{Rec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{|M(\mathcal{C}) \cap M(\mathcal{C}_{\text{gold}})|}{|M(\mathcal{C}_{\text{gold}})|} \tag{4.36}$$

$$= \frac{|p\text{TP}|}{|p\text{TP}| + |p\text{FN}|} \tag{4.37}$$

- The **Pair Precision** (short pPrec) is the amount of all MATCHES that are correctly classified as a MATCH:

$$p\text{Prec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{|M(\mathcal{C}) \cap M(\mathcal{C}_{\text{gold}})|}{|M(\mathcal{C})|} = \frac{|p\text{TP}|}{|p\text{TP}| + |p\text{FP}|} \tag{4.38}$$



(a) Clustering $\mathcal{C}_1$ (b) Clustering $\mathcal{C}_2$ (c) Clustering $\mathcal{C}_3$

Figure 4.19.: The *Pair True Positives*, *Pair False Positives*, and *Pair False Negatives* of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$

Quality Measure:	**Clustering $\mathcal{C}_1$**	**Clustering $\mathcal{C}_2$**	**Clustering $\mathcal{C}_3$**
Number of pTP	7	5	7
Number of pFP	15	0	3
Number of pFN	1	3	1
Number of pFDec	16	3	4
pRec	7/8=0.875	5/8=0.625	7/8=0.875
pPrec	7/22=0.318	5/5=1.0	7/11=0.636
pF_1	7/15=0.467	10/13=0.769	7/9=0.778

Table 4.6.: Quality scores of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ for the discussed pair-based evaluation measures

- The **Pair F_1-score** (short pF_1) is the harmonic mean of Pair Recall and Pair Precision:

$$pF_1(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{2 \times p\text{Rec} \times p\text{Prec}}{p\text{Rec} + p\text{Prec}} = \frac{2 \times |p\text{TP}|}{2 \times |p\text{TP}| + |p\text{FP}| + |p\text{FN}|} \tag{4.39}$$

The number of Pair False Decisions is an element of the domain $\mathbb{N}_0$. In contrast, the quality score of Pair Recall, Pair Precision, and Pair F_1-score is a normalized value in the range $[0, 1]$.

Because the most entity pairs are no duplicates and are correctly classified as UNMATCHES, the set of *Pair True Negatives* $p\text{TN} = U(\mathcal{C}) \cap U(\mathcal{C}_{\text{gold}})$ clearly dominates the classification problem, i.e. $|p\text{TN}| = |CPS_{naive}| - |p\text{TP}| - |p\text{FP}| - |p\text{FN}| \ggg |p\text{TP}| + |p\text{FP}| + |p\text{FN}|$. For that reason, measures for classification quality that are based on the number of Pair True Negatives, e.g. *Accuracy* [Chr12], are most often less suitable for duplicate detection, because the trivial and unacceptable solution of classifying all entity pairs as UNMATCHES would produce a result of high quality.

Example 61 *For illustration, we consider the three clusterings from Figure 4.18. The clustering that results from the partitioning based on connected components (see Figure 4.18(c)) is denoted as $\mathcal{C}_1$, the clustering that results from the partitioning based on centers (see Figure 4.18(d)) is denoted as $\mathcal{C}_2$, and the clustering that results from the approach of edge removal (see Figure 4.18(e)) is denoted as $\mathcal{C}_3$. The Pair True Positives, the Pair False Positives, and the Pair False Negatives of these clusterings are*

graphically presented by the three matrices in Figure 4.19. The size of these three sets as well as the quality scores of the other pairwise evaluation measures for these three clusterings are listed in Table 4.6.

Whereas clustering $\mathcal{C}_2$ has a perfect Pair Precision, clustering $\mathcal{C}_3$ has the best Pair Recall and F_1-score. In contrast, clustering $\mathcal{C}_1$ has by far the worst quality in Pair Precision and hence a low F_1-score as well. By considering these quality measures, clustering $\mathcal{C}_1$ is clearly dominated by clustering $\mathcal{C}_3$ because the latter has the same quality score as the first one or has a better quality score than the first one for every measure. In contrast, the decision whether $\mathcal{C}_2$ is better than $\mathcal{C}_3$ or $\mathcal{C}_3$ is better than $\mathcal{C}_2$ depends on the quality requirements of the considered application.

A less often used (but also little-known) pair-based quality measure is the *Efficacy* [BT06] that computes a recall value and a precision value per duplicate cluster and finally aggregates these values by computing the average F_1-score of all clusters. To illustrate the robustness of a duplicate detection approach against variations in parameter settings a graphical representation as the *Recall-Precision Graph* or the *ROC-curve* is often used [Chr12].

In the context of candidate pair space construction, Pair Recall is typically denoted as *Pairs Completeness* and Pair Precision is typically denoted as *Pairs Quality* [Chr11] where in both measures the candidate pair space, i.e. the set of all acceptances, is considered as the set of MATCHES and the set of all rejections is considered as the set of UNMATCHES. As a consequence, the set of true acceptances corresponds to the set of Pair True Positives, the set of false acceptance corresponds to the set of Pair False Positives, and the set of false rejections corresponds to the set of Pair False Negatives. Another typical measure that is used to rate the effectiveness of a blocking method is the *Reduction Ratio* [Chr11] that computes the amount of rejected entity pairs among all possible entity pairs and hence reflects the percentage of entity pairs that are removed from the search space. Note, the Reduction Ratio is not affected by the gold standard. Let CPS be the constructed candidate pair space and let CPS_{naive} be its corresponding naive candidate pair space, the Reduction Ratio is computed as:

$$RRatio(CPS, CPS_{naive}) = \frac{CPS_{naive} - CPS}{CPS_{naive}} = 1 - \frac{CPS}{CPS_{naive}} \tag{4.40}$$

4.3.8.2. Cluster-based Evaluation

In measures for cluster-based evaluation the quality of a duplicate detection process is rated by the similarity of its final clustering $\mathcal{C}$ to the perfect clustering $\mathcal{C}_{\text{gold}}$. The more similar both clusterings are, the better is the process's quality.

The simplest cluster-based measures are *Cluster True Positives*, *Cluster False Positives*, *Cluster False Negatives*, *Cluster Recall*, *Cluster Precision*, and *Cluster F_1-score* [MWGM10] that are defined accordingly to their pairwise equivalents, but computes set intersections and set exclusions of clusters instead of pairs. The problem with these measures is that a cluster is already classified as a false positive, if it differs from its corresponding cluster of the gold standard only by a single entity. This usually leads to a low number of true positives even if many clusters of both clusterings are very similar. As a consequence, these measures are only less suitable for quality evaluation and more sophisticated measures are typically used instead.

More suitable cluster-based quality measures are:

- The **Closest Cluster F_1-score** (short ccF$_1$) [MWGM10, MNHG12] eliminates the drawback of the Cluster F_1-score by also taking similar clusters into account. It is defined as the harmonic mean of the *Closest Cluster Precision* and the *Closest Cluster Recall*. The Closest Cluster Precision (short ccPrec) is defined as:

$$\text{ccPrec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{\sum_{C_p \in \mathcal{C}} \mathbf{max}_{C_q \in \mathcal{C}_{\text{gold}}} sim(C_p, C_q)}{|\mathcal{C}|} \tag{4.41}$$

where $sim(C_p, C_q)$ is a set-based similarity function (e.g. the Jaccard Coefficient $\frac{|C_p \cap C_q|}{|C_p \cup C_q|}$). Thus, the closest cluster precision is the average maximal similarity the clusters of $\mathcal{C}$ have to a cluster of the gold standard. Similarly, the Closest Cluster Recall (short ccRec) is defined as:

$$\text{ccRec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{\sum_{C_q \in \mathcal{C}_{\text{gold}}} \mathbf{max}_{C_p \in \mathcal{C}} sim(C_p, C_q)}{|\mathcal{C}_{\text{gold}}|} \tag{4.42}$$

As a consequence, the Closest Cluster F_1-score is defined as:

$$\text{ccF}_1(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{2 \times \text{ccPrec} \times \text{ccRec}}{\text{ccPrec} + \text{ccRec}} \tag{4.43}$$

The score domain of all three measures is the range $[0, 1]$.

A quality measure which is based on a similar idea is the K-measure [CGL07] that uses the *Average Cluster Purity* and the *Average Author Purity* to compute a representative quality score.

- The **Variation of Information** [Mei03, Mei07] (short VI) measures the 'information' that is lost or gained by converting the clustering $\mathcal{C}$ to the gold standard $\mathcal{C}_{\text{gold}}$ and is defined as follows:

$$\text{VI}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = H(\mathcal{C}) + H(\mathcal{C}_{\text{gold}}) - 2I(\mathcal{C}, \mathcal{C}_{\text{gold}}) \tag{4.44}$$

The function H represents the the total entropy of the individual clusters and the function I represents the mutual information between $\mathcal{C}$ and $\mathcal{C}_{\text{gold}}$:

$$H(\mathcal{C}) = -\sum_{C \in \mathcal{C}} \frac{|C|}{|rng(\mathcal{C})|} \times \log\left(\frac{|C|}{|rng(\mathcal{C})|}\right) \tag{4.45}$$

$$I(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \sum_{C_p \in \mathcal{C}} \sum_{C_q \in \mathcal{C}_{\text{gold}}} \frac{|C_p \cap C_q|}{|rng(\mathcal{C})|} \times \log\left(\frac{|C_p \cap C_q| \times |rng(\mathcal{C})|}{|C_p| \times |C_q|}\right) \tag{4.46}$$

By comparing two clusterings $\mathcal{C}$ and $\mathcal{C}'$ with the same range $r = rng(\mathcal{C}) = rng(\mathcal{C}')$, the Variation of Information produces a quality score that is a normalized value in the range $[0, \log(r)]$ [Mei03], where 0 means that there is no distance between the compared clusterings, i.e. $\mathcal{C} = \mathcal{C}'$, and $\log(r)$ means that one of the clusterings contains a single cluster and the other clustering contains r clusters.

Extensions of the Variation of Information are the *LogN Normalized Variation of Information* [Mei07] and the *Normalized Variation of Information* [RR09]. Two other entropy based measures are the *V-measure* that has been introduced by Rosenberg et al. [RH07], and the *Normalized Mutual Information* proposed by Strehl [Str02, SG03].

- The **Basic Merge Distance** (short BMD) [AKE04] has been inspired by the Edit-Distance. Recall, the Edit-Distance between two strings is the minimal cost that is required to transform the first string into the second one by using letter operations such as insert, delete, update, or swap. In the domain of clusterings, the fundamental 'edit' operations are *cluster split* and *cluster merge*.

  A cluster split is an operation $C \rightarrow C_p, C_q$ that partitions a cluster C into the two disjoint non-empty clusters C_p and C_q, i.e. $C_p \cap C_q = \emptyset, C_p \cup C_q = C, C_p, C_q \neq \emptyset$. Applying a split $C \rightarrow C_p, C_q$ to a clustering $\mathcal{C}$ results in the clustering $(\mathcal{C} - \{C\}) \cup \{C_p, C_q\}$ and is only a valid operation on $\mathcal{C}$ if $C \in \mathcal{C}$.

  In contrast, a cluster merge is an operation $C_p, C_q \rightarrow C$ that merges two clusters to a single one, i.e. $C = C_p \cup C_q$. Applying a merge $C_p, C_q \rightarrow C$ to a clustering $\mathcal{C}$ results in the clustering $(\mathcal{C} - \{C_p, C_q\}) \cup \{C\}$ and is only a valid operation on $\mathcal{C}$ if $C_p, C_q \in \mathcal{C}$.

  To avoid trivial and less meaningful transformations such as to initially merge all cluster of $\mathcal{C}$ to a single one and then to split this single cluster into the clusters of $\mathcal{C}_{\text{gold}}$, Al-Kamha et al. [AKE04] additionally introduce the condition of path legality that defines a path (sequence) of merge and split operations from the clustering $\mathcal{C}$ to the clustering $\mathcal{C}'$ to be legal, if each cluster that result from any of the path's merge operations is the subset of any cluster in $\mathcal{C}'$.

  The BMD of $\mathcal{C}$ and $\mathcal{C}_{\text{gold}}$ is then defined as the number of operations in the shortest legal path from $\mathcal{C}$ to $\mathcal{C}_{\text{gold}}$. A quality score resulting from using the BMD is a natural number or zero if $\mathcal{C} = \mathcal{C}_{\text{gold}}$. A normalized version of the BMD has been proposed by Menestrina et al. [MWGM10].

- Menestrina et al. [MWGM10] extend the Basic Merge Distance by two cost functions f_m and f_s that penalize merge and split operations depending on the sizes of the clusters that are merged or split respectively. Thus, the **Generalized Merge Distance** (short GMD) from a clustering $\mathcal{C}$ to the gold standard $\mathcal{C}_{\text{gold}}$ is the minimum cost of a legal path from $\mathcal{C}$ to $\mathcal{C}_{\text{gold}}$, where:

  - the cost of a merge operation $C_p, C_q \rightarrow C$ is $f_m(|C_p|, |C_q|)$,
  - the cost of a split operation $C \rightarrow C_p, C_q$ is $f_s(|C_p|, |C_q|)$.

  Both cost functions cannot produce negative costs, are symmetric, i.e. $f(x, y) = f(y, x)$, and monotonically increase with their parameters, i.e. $f(x, y) \leq f(x + j, y + k)$ for non-negative j, k. Obviously, the GDM can be specialized to the BMD using $f_m(x, y) = f_s(x, y) = 1$. Interestingly, Menestrina et al. show that the GMD can be specialized to Pair Recall, Pair Precision, and the Variation of Information using specific configurations of the two costs functions f_m and f_s.

  A quality score resulting from using the GMD is a positive real number or zero if $\mathcal{C} = \mathcal{C}_{\text{gold}}$.

A commonly used measure for computing clustering similarity is the *Rand Index* [Ran71] that is equivalent to the accuracy of a binary classification problem. Because accuracy is dominated by the high number of Pair True Positives, the same holds for the Rand Index, the *Adjusted Rand Index* [HA85] and the recently developed *Talburt-Wang Index* [Tal11] which is a lighter version of the Rand Index that aims to simulate the same result with lower computation effort. As a consequence, all these measures are only less suitable for evaluating the quality of a duplicate detection process.

$\downarrow$ **cluster of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$** $\quad \mathcal{C}_{\text{gold}} \rightarrow$	$\langle e_1, e_2, e_4 \rangle$	$\langle e_3, e_5, e_6 \rangle$	$\langle e_7, e_8 \rangle$	$\langle e_9, e_{10} \rangle$
$C_1 = \langle e_1, e_2, e_3, e_4, e_5, e_6, e_{10} \rangle$	$3/7$	$3/7$	0	$1/7$
$C_2 = \langle e_7, e_8 \rangle$	0	0	1	0
$C_3 = \langle e_9 \rangle$	0	0	0	$1/2$
$C_4 = \langle e_1, e_2, e_4 \rangle$	1	0	0	0
$C_5 = \langle e_3 \rangle$	0	$1/3$	0	0
$C_6 = \langle e_5, e_6 \rangle$	0	$2/3$	0	0
$C_7 = \langle e_{10} \rangle$	0	0	0	$1/2$
$C_8 = \langle e_3, e_5, e_6, e_{10} \rangle$	0	$3/4$	0	$1/5$

Table 4.7.: The Jaccard Coefficient between the individual clusters of $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and the clusters of the gold standard $\mathcal{C}_{\text{gold}}$

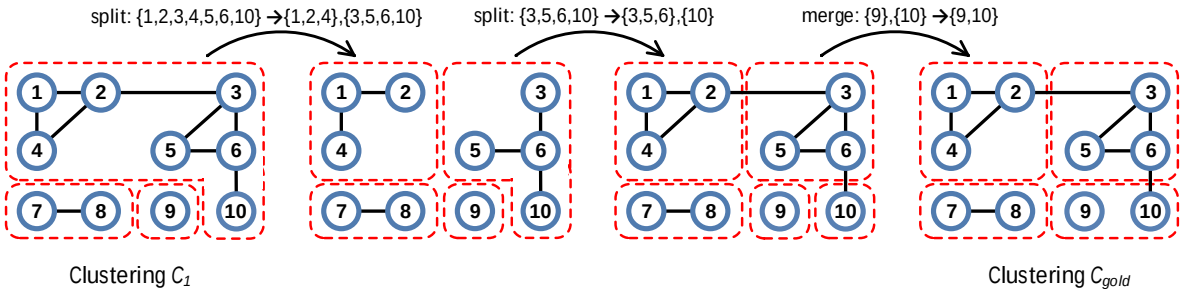


Figure 4.20.: The shortest legal split / merge path from $\mathcal{C}_1$ to $\mathcal{C}_{\text{gold}}$

Further cluster-based quality measures are the *MUC*-F_1-score [VBA+95], the B^3-F_1-score [BB98], the *Automatic Content Extraction (ACE) evaluation score* [DMP+04], and the *Constrained Entity-Alignment F-score* [Luo05]. All these measures are described and compared in the survey written by Maidasani et al. [MNHG12].

Example 62 *To illustrate the Closest Cluster F_1-score, we first compute the Jaccard Coefficient between all clusters in $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ to all clusters of $\mathcal{C}_{gold}$. These values are presented in Table 4.7. From these similarities, the Closest Cluster Recall and the Closest Cluster Precision can be computed from which in turn the Closest Cluster F_1-score can be derived. The respective results as well as the quality score for the Variation of Information are presented in Table 4.8. Note that the Variation of Information is a distance measure (best value is 0.0) whereas the Closest Cluster F_1-score is a similarity measure (best value is 1.0).*

To illustrate the Merge Distance, we choose clustering $\mathcal{C}_1$ and transform it into the gold standard by two split operations and one merge operation. The transformation is graphically presented in Figure 4.20. The BMD of all clusterings are presented in Table 4.8.

As you can see in this example, the BMD is often not as selective as the Closest Cluster F_1-score, because $\mathcal{C}_2$ and $\mathcal{C}_3$ have both the same distance, but have different Closest Cluster F_1-scores.

More detailed comparisons of the individual evaluation measures can be found in [MWGM10, MNHG12, Rem12].

Quality Measure:	$\mathcal{C}_1 = \{C_1, C_2, C_3\}$	$\mathcal{C}_2 = \{C_2, C_3, C_4, C_5, C_6, C_7\}$	$\mathcal{C}_3 = \{C_2, C_3, C_4, C_8\}$
ccRec	0.589	0.792	0.8125
ccPrec	0.643	0.667	0.8125
ccF$_1$	0.615	0.724	0.8125
VI	0.36553	0.1427	0.09764
BMD	3	2	2

Table 4.8.: Quality scores for $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ for the discussed cluster-based evaluation measures

4.3.9. Duplicate Detection in Multi-Table Databases

In complex databases, tables are connected by foreign keys and thus tuples from different tables are related with each other. Although database entities can theoretically belong to the extensions of several entity tables, such a multi-table membership not been considered in duplicate detection approaches so far and is firstly discussed in Section 5.3 of this thesis. Therefore, existing approaches for duplicate detection in multi-table databases still consider each entity to be represented by a single tuple, but in contrast to single-table approaches they additionally take relationships between these tuples (and hence their corresponding entities) into account. Such relationship information can be used to improve the effectiveness (and sometimes even the efficiency) of the detection process.

Relationship information is usually described by a set of related entities. Thus, let e be a database entity, Naumann and Herschel [NH10] define the *relationship description* of e (denoted as $RD(e)$) as the set of all database entities that are descriptive of e.

Example 63 *For demonstration we consider the database presented in Figure 4.21. This database contains the three tables* 'Person'*,* 'Book'*, and* 'Purchase'*. Whereas the first two tables are entity tables, the third table is a relationship table. The extension of the first table contains two entities (persons) and the extension of the second table contains four entities (books). Person p_1 bought the three books b_1, b_2, and b_3, and person p_2 purchased the three books b_2, b_3, and b_4. As a consequence, if we consider all directly related entities to be descriptive, the relationship descriptions of the individual database entities are:*

$$RD(p_1) = \{b_1, b_2, b_3\} \qquad RD(b_1) = \{p_1\}$$
$$RD(p_2) = \{b_2, b_3, b_4\} \qquad RD(b_2) = RD(b_3) = \{p_1, p_2\}$$
$$RD(b_4) = \{p_2\}$$

Of course, the meaning of related entities is not necessarily restricted to relationships that are directly modeled in the database, but can be extended to indirectly modeled relationships. For instance, we could also say that book b_1 is related to book b_2 because they have been bought by the same person. In general, which relationships are included into the relationship description depends on the respective use case. Sometimes it can be even beneficial to include implicit relationships into the description and to exclude explicit relationships from the description if the information of the first one is useful to identify duplicate database entities while the information of the second one is not.

Measures that rate the similarity between two relationship descriptions can be classified into two classes: (a) measures that use the identities of the related entities to compute a similarity between the

Person (Entity Table)

DEI	name
p1	John Doe
p2	Joan Doe

Purchase (Relationship Table)

person	book
p1	b1
p1	b2
p1	b3
p2	b2
p2	b3
p2	b4

person → Person.DEI,
book → Book.DEI

Book (Entity Table)

DEI	title	author
b1	Stranger, The	A.Kamu
b2	Pale Fire	V.Nabokov
b3	Mrs Dalloway	V.Woolf
b4	The Stranger	A.Camus

Figure 4.21.: Multi-table database with two entity tables and one relationship table

compared entity sets and (b) measures that use the similarities of the related entities to compute the similarity between the compared entity sets [BG07b].

- **Identity-based Entity Set Matching:** Identity-based measures rate the similarity between two relationship descriptions based on the number of shared entities, i.e. the more entities both descriptions share, the more similar they are. As a consequence, we can use a set similarity measure like the Jaccard Coefficient as an identity-based measure to compare two relationship descriptions.

- **Similarity-based Entity Set Matching:** In contrast to measures of the first class, similarity-based measures do not use the identities of the related entities to compute the similarity between two entity sets, but use the similarities between these entities instead. By doing so, two entity sets are considered to be similar, if their entities are similar. Nevertheless, a high similarity between two entity sets is not restricted to cases where all their elements are pairwise similar to each other, but rather have to satisfy the condition that a set S is always maximally similar to itself, i.e. $sim(S, S) = 1$. For that reason, the similarity between two relationship descriptions should not be computed as the average similarity between their elements, but should be computed by using a hybrid similarity measures like the Monge-Elkan Similarity. Recall, hybrid measures use an internal measure for element similarity. In the case of entity sets, the similarity between two elements corresponds to the similarity between two entities. As a consequence, the internal similarity measure needs to be a method for matching whole entities as we have described in the previous sections.

Example 64 *To illustrate the difference between both matching approaches, we reconsider the two relationship descriptions* $RD(p_1) = \{b_1, b_2, b_3\}$ *and* $RD(p_2) = \{b_2, b_3, b_4\}$ *from Example 63. If we use the Jaccard Coefficient to compute the identity-based similarity of these descriptions, we receive the similarity score* $sim(RD(p_1), RD(p_2)) = 2/4 = 0.5$. *In contrast, if we assume the following entity similarities* $sim(b_1, b_2) = 0.1$, $sim(b_1, b_3) = 0.0$, $sim(b_1, b_4) = 0.5$, $sim(b_2, b_3) = 0.2$, $sim(b_2, b_4) = 0.1$, *and* $sim(b_3, b_4) = 0.2$, *the Monge-Elkan Similarity between* $RD(p_1)$ *and* $RD(p_2)$ *results in* $sim(RD(p_1), RD(p_2)) = sim(RD(p_2), RD(p_1)) = (0.5 + 1 + 1)/3 = 5/6$.

4.3.9.1. Collective Duplicate Detection

Naive approaches for relationship-based duplicate detection [BG07b] incorporate the similarity of their relationship information into the decision process, but still make their duplicate decisions independently, i.e. the decision on one pair only depends on the initially given data and is not affected by the decisions on the other candidate pairs. Note that in this case each candidate pair is only matched once and if a similarity-based measure is used for matching relationship descriptions, the similarity between related entities is then only computed on the basis of the similarities between the entities' attribute values.

In contrast, approaches for collective duplicate detection make all the pairwise duplicate decisions in a collective way, i.e. for making the decision on one candidate pair all decisions on the other candidate pairs are taken into account. For that reason, the effectiveness of collective detection approaches is usually improved compared to naive approaches.

Collective duplicate detection approaches can be further categorized into two classes. Approaches of the first class [DHM05, BG07b, BG07c, HNST12] consider the fact that the result of matching one entity pair can affect the relationship similarity of other entity pairs. In the case of an identity-based measure, relationship similarities are affected by a detected MATCH because two previously distinguished entities become equal. In contrast, if a similarity-based measure is used, the similarities of the relationship descriptions are directly affected by the newly computed entity similarities. Since the increase of relationship similarity can in turn increase the final similarity between two entities, their rematch can cause new affections. In order to incorporate these affections into the detection process, collective duplicate detection approaches perform multiple iterations on the matching phases and rematch an entity pair if it has been affected by a newly computed duplicate decision. To reduce the number of required iterations, several approaches, e.g. [WN06b, HNST12], use a queue that contains all candidate pairs that still need to be matched and update the queue for each newly detected affection, i.e. all newly affected pairs that are not already in the queue are added to the queue. Changing the queue order can save a lot of computation time, because it can reduce the number of newly detected affections (and hence can reduce the total number of required matchings). For that reason, several algorithms for optimizing the queue order have been proposed [WN06b]. The iteration process stops if all pairs have been matched, i.e. the queue is empty. To avoid endless iterations in cases where a similarity-based measure is used to rate relationship similarity, affections are typically only considered if its underlying change in the corresponding entity similarity is above a given threshold.

Approaches of the second class of collective duplicate detection are based on the use of probabilistic inference [MW04, PD04, PMM+02, SD06, BG06, HSM08] and hence make duplicate decisions in a truly collective manner. Rastogi et al. [RDG11] present message based strategies that enable a parallelization of both class of collective detection approaches.

4.3.9.2. Negative Information

Another way to use information on entity relationships in order to increase the effectiveness of a duplicate detection process is to extract negative information from the database. Negative information means evidence that two entities must be an UNMATCH. For instance, if we detect that one person has been the coauthor of a second person in several papers, the likelihood that both persons are duplicates is

minimized, because otherwise the database must be heavily corrupted. Thus, we can certainly classify both persons as an UNMATCH without matching their attribute values or relationship information first. Consequently, negative information does not only help to increase the confidence of the made duplicate decisions, but can also save matching time. An approach that extract negative information from the database by using specific rules is Dedupalog which has been developed by Arasu et al. [ARS09].

4.3.10. Related Topics

The presented approach for iterative duplicate detection is closely related to some topics that we have not discussed in this chapter so far.

4.3.10.1. Similarity Joins

A closely related area of research focuses on the efficient computation of similarity joins. Similarity joins are join operators that do not only join tuples on equivalent values, but join two tuples if their similarity is above a given threshold.

From a computational viewpoint, a similarity join can be considered as a light version of a duplicate detection process that need to be computed in at most a few seconds if a low response time is required. From a semantical viewpoint, however, there is an essential difference between both concepts. Whereas duplicate detection only uses the property of data similarity to infer real-world equivalence, the goal of a similarity join is to compute all pairs of similar tuples. Thus, duplicate detection is a heuristic process and its exact result is usually not known. Moreover, the quality of its result essentially depends on the suitability of the selected similarity measures for indicating real-world equivalence. In contrast, the result of a similarity join is clearly defined by the used similarity measure and can be exactly computed. Nevertheless, many techniques from duplicate detection can be adopted for similarity join computation and vice versa. Recently a survey on similarity joins has been published by Augsten and Böhlen [AB13].

4.3.10.2. Stanford Entity Resolution Framework

The *Stanford Entity Resolution Framework*[8] (short SERF) [BGMK+06] is a well known framework for duplicate elimination that incorporates duplicate merging into the detection process in order to increase detection effectiveness. Thus, in contrast to the general approach for duplicate elimination that we have presented in this chapter, SERF does not separate the phases of duplicate detection and duplicate merging, but perform both phases in an interleaved manner. Once a MATCH is detected, the corresponding entities are immediately merged and replaced by the merged entity so that subsequently performed matching processes will use the values from the merged entity instead of the values from the original entities. In general, this approach is quite similar to a clustering-based decision model that uses the canonical entity method to compute the similarity between two clusters (see Section 4.3.6.5).

In SERF, the matching process as well as the merging process are considered as abstract functions that can be implemented in several ways, but always need to satisfy some fundamental requirements. To date several duplicate elimination approaches that are conform to SERF have been proposed. Implementations that have been developed within the SERF project are *G-Swoosh* [BGMM+09], *R-Swoosh* [BGMM+09],

[8] http://infolab.stanford.edu/serf/

F-Swoosh [BGMM+09], *D-Swoosh* [BGMG+07], and *P-Swoosh* [HHMO+06]. All these approaches differ in the way they match and merge entities. Koosh [MBGM06] is an implementation that take confidence values of the input data, e.g. reliability of the source or accuracy of the value, into account. Whang et al. [WBGM09] published an extension of SERF to negative rules, i.e. rules that restrict the space of possible merge outputs. Blocking approaches for scaling up duplicate elimination with SERF has been proposed by Whang et al. [WMK+09] and Kim et al. [sKL10]. More details on SERF can be found in the papers that are referenced in this subsection or can be found in the book written by Talburt [Tal11].

4.3.10.3. Identity Management

Talburt et al. [Tal11] developed a system called *Oyster*[9] that is not only designed to eliminate duplicates in a database, but rather to manage database entities over a life cycle.

Conventional duplicate elimination approaches get a database as input and produce a database as output. An application that works on the output database has no information on the origins of the processed database entities and therefore can never check and repair a false positive afterwards. Moreover, a conventional database system only stores the current value of an attribute.

In contrast, the Oyster system manages all database entities along with all the changes that have been applied to their data values, e.g. a change in the address of a person. This information can help to improve duplicate detection quality because the system automatically provides context data to overcome value conflicts that result from outdated data. Moreover, the system does not only store the current instance of the database, but also stores any of its old entities, i.e. entities that have been removed from the actual database. For instance, if a duplicate cluster is merged, the old entities are kept in the system. Thus, the system is able to check and repair false positives when new evidence becomes available. Moreover, the system helps to avoid duplicates that results from duplicative insertions into the database. Each time a new entity is inserted into the database, the existing entities (current as well as old ones) are checked for a duplicate by considering current values as well as old values. If the new database entity is detected to be a duplicate of an already existing one, it is not inserted into the database, but the data on the existing entity is changed instead.

4.3.10.4. Non-relational Approaches

Naturally, research on duplicate detection has not been restricted to relational databases. Several approaches for detecting duplicates in XML documents have been proposed by Herschel et al. [WN05, WN06a, PWN06, Wei08] and Leitao et al. [LCW07, CHL10, LCH13b, LC13]. Volz et al. [VBGK09b, VBGK09a] and Jentzsch et al. [JIB10] consider the detection of duplicates in RDF data.

4.3.10.5. Data Coupling and Entity Identification

Two specific problems that are similar to the one of duplicate detection are *data coupling* [vK12] and *entity identification* [TWPH07] (also denoted as *entity search* [NH10]). In data coupling the entities of two duplicate-free databases are matched with each other in order to create an 1:1 mapping between the

[9] http://sourceforge.net/projects/oysterer/

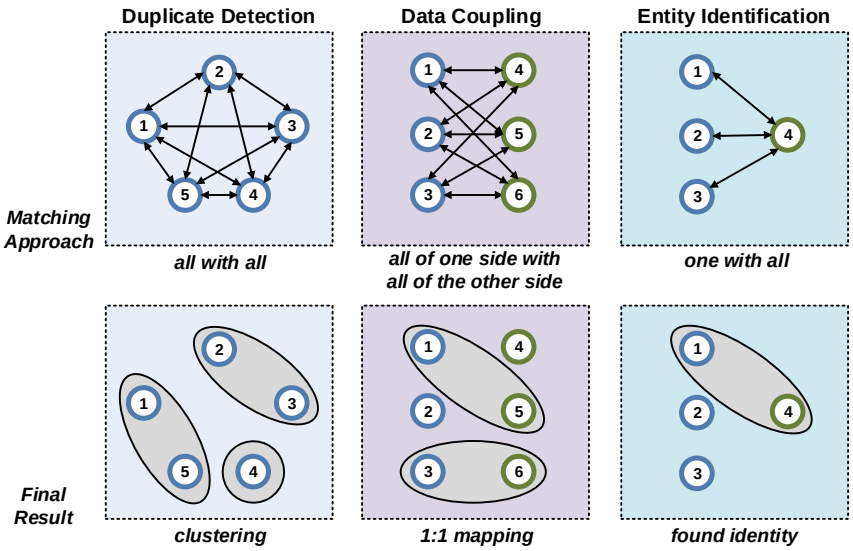


Figure 4.22.: Differences of duplicate detection, data coupling, and entity identification

entities of both databases. This scenario often appears in the integration of autonomic, but clean sources. In entity identification, the identity of a given database entity needs to be detected by investigating the entities in a database. This scenario often appears to avoid duplicates at the time of data insertion by matching each newly inserted database entity with all existing database entities (see discussion on identity management above) and can be also used in applications where a specific entity from the database is searched based on some of its remarkable characteristics.

The differences to the general consideration of duplicate detection are illustrated in Figure 4.22. Whereas in duplicate detection each database entity is theoretically matched with each other database entity and the result of the detection is a clustering of the entities, in data coupling the database entities of one source are only matched with database entities of the other source. The result is an 1:1 mapping, where it can happen that a database entity of one source does not have a partner in the other source. As a consequence, in data coupling we have only pairs of matching entities instead of clusters. In entity identification the single inserted entity (or searched entity respectively) is matched with all entities of the database and the result is a single database entity that corresponds to the searched entity or is the empty set if the database does not contain a corresponding entity.

Nevertheless, as it can be seen easily, both problems are special cases of the problem of duplicate detection. For instance, duplicate detection can be specialized to data coupling by restricting the search space to only candidate pairs that originate from different sources and to classify all entities from the same source as hard UNMATCHES in order to avoid that an entity of one source is classified as a MATCH with several entities of the other source.

Some methods that are considered in duplicate detection literature have been originally designed for data coupling (e.g. the ranking based decision model presented in [GKMS04]) or entity identification (e.g. some indexing techniques by Christen [Chr11]), but they can be usually adopted to duplicate detection with only marginal changes.

DEI	fname	lname	DoB	city	country	phone	email
e1	dejohn w.	tucker	1974-05-03	newyork city	usa	(0345)233848	abc@me.com
e2	john bill	tacker	1974-03-05	newyork city	usa	(0345)233848	tucker@xy.com
e4	john william	tucker	1974-05-03	newyork city	usa	(0345)233884	⊥

(a) Duplicate cluster with three entities

DEI	fname	lname	DoB	city	country	phone	email
e*	dejohn william	tucker	1974-05-03	newyork city	usa	(0345)233848	tucker@xy.com

(b) Merging with conflict resolution

RK	DEI	fname	lname	DoB	city	country	phone	email	p
1	e*	dejohn w.	tucker	1974-05-03	newyork city	usa	(0345)233848	abc@me.com	1/3
2	e*	john bill	tacker	1974-03-05	newyork city	usa	(0345)233848	tucker@xy.com	1/3
3	e*	john william	tucker	1974-05-03	newyork city	usa	(0345)233884	⊥	1/3

(c) Merging with conflict repairing (tuple level)

DEI	fname		lname		DoB		city		...	email	
e*	dejohn w.	:1/3	tucker	:2/3	1974-05-03	:2/3	newyork city	:1.0	...	abc@me.com	:0.5
	john william	:2/3	tacker	:1/3	1974-03-05	:1/3				tucker@xy.com	:0.5

(d) Merging with conflict repairing (attribute value level)

Figure 4.23.: Sample for merging three duplicate entities with different merging functions

4.4. Duplicate Merging

After detecting duplicate database entities, these duplicates need to be eliminated by merging all the entities of one duplicate cluster to a single entity. Since database entities are only logical constructs, their underlying data need to be merged instead. Existing research on duplicate merging [WBGM09, LSS96b] (also known as data fusion [MA06, BBB+05, BN08, DN09, ZH11]) in relational data focuses on the merge of database entities that corresponds to single database tuples. As already presented in Example 49, merging a set of duplicate tuples is a challenge because the values of the individual tuples are usually conflicting and these conflicts need to be handled by the merging approach.

Approaches for duplicate merging can be divided into two classes; approaches that completely resolve conflicts by condensing all duplicate tuples to a certain database tuple and approaches that do not completely resolve conflicts, but instead model all plausible alternatives in an uncertain database. In the rest of this thesis, we will refer to the first class as *conflict resolution* and will refer to the second class as *conflict repairing*.

1. **Conflict Resolution:** Approaches of conflict resolution resolve all conflicts for individual attributes by preferring non-null values to null values and by utilizing aggregation functions that either select one of the conflicting values (deciding strategy) or aggregate them to a new value (mediating strategy). Conflict resolution approaches always produce a certain output and hence

are closed under the relational data model, i.e. the merging result can always be represented by an ordinary database tuple. Resolution approaches have been considered by Bleiholder et al. [BN06] and Talburt et al. [Tal11, ZKT12].

2. **Conflict Repairing:** In contrast to approaches of conflict resolution, conflict repairing approaches do not necessarily resolve every conflict, but instead produce a set of possible repairs if the given conflict cannot be resolved in an appropriate way. Repairing approaches usually produce an uncertain output and hence are not closed under the relational data model, i.e. an uncertain database is produced by the merging process. Existing approaches either model conflicts on tuple level [AFM06, HM09] or model conflicts on attribute value level [DeM89, T+92, TCY93]. Whereas the first approaches produce an x-tuple per duplicate cluster, the latter produce an A-tuple per duplicate cluster.

We will illustrate the fundamental differences of these two merging concepts by the following example.

Example 65 *For that purpose, we consider the three database entities e_1, e_2, and e_4 from the running example (see Figure 4.23(a)). These three entities form a duplicate cluster and therefore need to be merged to a single entity that we identify as e^* in this example.*

First, we consider a conflict resolution approach. The corresponding merge function, merges the conflicts in the individual attributes by means of aggregation functions. In this example, aggregation is performed by choosing the most often occurring non null-value per attribute (voting strategy). For instance, the last name 'tucker' is selected in attribute 'lname'. *If no most often occurring value exists, aggregation has to be performed in another way. Since the name 'dejohn' contains the name 'john' and because 'w.' and 'bill' are typical shortcuts for the name 'william', the most plausible aggregation of the three conflicting first names is the name 'dejohn william'. In the case of the email address, mediating the given values does not make any sense because the chance that the aggregation result is an existing email address is very low. Actually, the most plausible solution is to keep both addresses because persons can own several email addresses. Nevertheless, this approach would imply a modification of the schema, because in this case the resolution result is a multi-valued attribute instead of a single-valued attribute. If schema modifications should be avoided, one of the conflicting values needs to be selected. In our example, we choose the value 'tucker@xy.com' because it contains the last name of the considered person and therefore seems most reliable. The resultant database tuple is presented in Figure 4.23(b).*

Second, we consider a conflict repairing approach that repairs conflicts on tuple level. In this case, the merging result is an x-tuple that simply contains each of the duplicate database tuples as one of its alternatives. The resultant x-tuple is presented in Figure 4.23(c). In this example, we simply assign a uniform probability distribution to the alternatives. Nevertheless, more sophisticated approaches for probability computation are possible. For instance, the approach proposed by Andritsos et al. [AFM06] computes a vector that serves as a mean of all considered tuples and then defines the probability of each alternative based on its distance to this mean.

Finally, we consider a conflict repairing approach that repairs conflicts on attribute value level. In this case, the result is the tuple of an attribute-OR database that is presented in Figure 4.23(d). In general, we use the conflicting input values as the alternative values of the output tuple in the corresponding

attribute, but sometimes restrict on the most plausible values. For example, in the first name we omit the value 'john bill' because 'bill' is a nickname of 'william' and it looks quite confident that this value correspond to 'john william'. Again, probabilities are computed by assuming a uniform distribution on the input set, i.e. the probability of an alternative value is the amount of all input tuples that have this value. For instance the probability that the last name is 'tucker' is set to $2/3$ *and the probability that the last name is 'tucker' is set to* $1/3$*.*

Obviously, by modeling uncertainty on attribute level instead of tuple level the number of possible instances of the output entity increases because we neglect the originally given dependencies between the input values and therefore enable possible instances that do not correspond to any of the input tuples. For example, an instance that contains the combination of the attribute values 'tucker' and '1974-03-05' is possible for the tuple of the attribute-OR database, but is impossible for the x-tuple from Figure 4.23(c).

Merging database entities in a complex database goes far beyond simply merging a set of duplicate tuples that originate from a single entity table, but appearances in multiple, maybe semantically exclusive, entity tables need to be considered instead. Moreover, values of multi-valued attributes need to be merged by modifying the corresponding tuples in multi-value tables and relationship information need to be combined by modifying the corresponding tuples in relationship tables. A simple method to merge multi-valued attributes or relationship information is to compute the set union. In that case, only all foreign key values referencing to one of the duplicate entities need to be replaced by the identifier of the merged entity. Afterwards, we possibly need to remove some tuples from these tables in order to avoid identical tuples and to avoid a violation of primary key constraints. In 1:n relationships, however, such a simple approach cannot be used and a more complex approach is required instead.

Duplicate merging is out of the scope of this thesis. Thus, in the rest of this thesis we will abstract from a particular merging approach and will assume a merge function that is associative and idempotent. Therefore, let $\mathfrak{I}$ be the space of all entity instances that are conform to the considered target schema and let f_μ be the considered merge function, it holds that: $\forall I \in \mathfrak{I}\colon f_\mu(\{I\}) = I$ and it holds that: $\forall S_t, S_u \subseteq \mathfrak{I}\colon S_t \cap S_u = \emptyset \Rightarrow f_\mu(\{f_\mu(S_t)\} \cup S_u) = f_\mu(\{f_\mu(S_u)\} \cup S_t)$.

Chapter 5

HaDDeF - A Duplicate Detection Framework

In this chapter we present a framework for duplicate detection in certain databases. This framework extends the five-phase model for duplicate detection that we have presented in Section 4.3 in different ways. First we present methods to extract and match complex entity descriptions in Section 5.1 and Section 5.2. Then, we extend the detection process by some helpful ideas such as the Belief Map *(Section 5.2.1) and the concept of* Impact Values *(Section 5.2.2). Finally, we extend the presented framework to a consideration of multi-table memberships in Section 5.3, i.e. we adapt the approaches for describing and matching entities from databases where each entity is only represented in exact entity table to database where each entity can be represented by tuples in several entity tables.*

As presented in Section 4.3, a variety of duplicate detection approaches has been developed in the last decades. Nonetheless, the most of these approaches focus on detecting duplicates in single entity tables or only consider multi-table databases where each entity belong to a single entity table. In complex databases, however, it can happen that single database entities belong to several entity tables, e.g. an entity is a person, a doctor, and a patient at the same time. Moreover, membership uncertainty is an important aspect in probabilistic databases. For that reason, we extend existing duplicate detection approaches by a method for handling multi-table memberships. In addition, we propose the concept of impact values which is intended to increase the effectiveness of a conventional duplicate detection approach, but can be also very useful in the detection of duplicates in probabilistic data. Finally, in order to enable a matching of abstract database entities we consider a description-based matching approach. For consolidating all these conceptions within a single model, we introduce a duplicate detection framework in this chapter that we call HaDDef (HAmburg Duplicate DEtection Framework). This framework serves as a fundamental baseline for our consideration of duplicate detection in probabilistic data that we will propose in the remaining chapters of this thesis.

The rest of this chapter is constructed as follows. First we present our approach of describing database entities in Section 5.1, then we introduce the belief map and the concept of impact values in Section 5.2.1 and Section 5.2.2. Then, we present the individual phases of HaDDeF for description-based entity matching in Section 5.2.3 and extend our description-based matching approach to multi-table memberships in Section 5.3.

5.1. Describing Database Entities

A database entity is a logical concept for encapsulating information that can be represented by several data tuples. For that reason, in multi-table databases, database entities cannot be directly compared, but first for each database entity a description need to be generated from the underlying database instead. Obviously, this description can contain information on the entity's attribute values as well as information on its relationships to other entities. In the optimal case, it only contains information that is useful for identifying duplicates.

For describing a database entity we adopt the concepts of the *object description* and the *relationship description* that has been described by Naumann and Herschel in [NH10] (recall that we have described the concept of a relationship description already in Section 4.3.9). Whereas an object description describes the attribute values of an entity, the relationship description describes its relationships to other database entities by listing a set of related entities. In HaDDeF we use this description approach, but make some extensions. First, we extend the modeling concept of relationship information by using a more complex, but also more informative description concept. Second, we incorporate membership information into the description approach. Finally, because we aim to create a single description per database entity, we encapsulate both, the object description and the relationship description, in a single concept, namely the entity description. To avoid semantical confusions and to make the naming more meaningful, we rename the object description into attribute description in this thesis.

Let f_{ED} be a function that is used to extract entity descriptions from a database, i.e. $f_{ED}(e, db)$ is the entity description of the database entity e that is extracted from database *db* by using extraction function f_{ED}. For incorporating multi-table membership into the description approach, we extract an entity description for the whole database by extracting entity descriptions for some of its tables. For that reason, we generalize the aforementioned notation by considering a database to be a set of tables, e.g. $f_{ED}(e, \{T_1, T_2\})$ is the entity description of e that is extracted from the two tables T_1 and T_2. If the considered database is clear from context, we sometimes shorten this notation by omitting the database variable, that is, $f_{ED}(e, db)$ is represented by $f_{ED}(e)$. Since the description of an entity can change during the detection process, e.g. in the phase of data preparation, we use the function $\mathfrak{d}_{ED}$ to map to the current description of an entity.

Since entity descriptions only contain information that is useful for duplicate detection (no surrogate keys) and because data preparation methods can replace different values by the same value, two database entities can have equivalent descriptions, i.e. $\forall e_r, e_s \in Ext(db)\colon e_r \neq e_s \not\Rightarrow \mathfrak{d}_{ED}(e_r) = \mathfrak{d}_{ED}(e_s)$. The underlying idea of the description-based matching approach is that duplicate decisions on entity pairs are made by matching their descriptions. That means we consider a pair of database entities to be a MATCH if they have similar descriptions, i.e. $\mathfrak{d}_{ED}(e_r) \simeq \mathfrak{d}_{ED}(e_s) \Rightarrow \{e_r, e_s\} \in$ MATCH. Thus, in the optimal case, the descriptions are real-world unique, i.e. $\mathfrak{d}_{ED}(e_t) = \mathfrak{d}_{ED}(e_s) \Rightarrow \omega(e_r) = \omega(e_s)$, because entities with same descriptions are automatically classified as a MATCH if no external knowledge is considered by the matching process. Of course, to avoid a high number of mismatches the descriptions need to be well designed.

The role of an entity description in the duplicate detection process is illustrated in Figure 5.1: First for each database entity a description is extracted from the database, and then the entities are pairwise

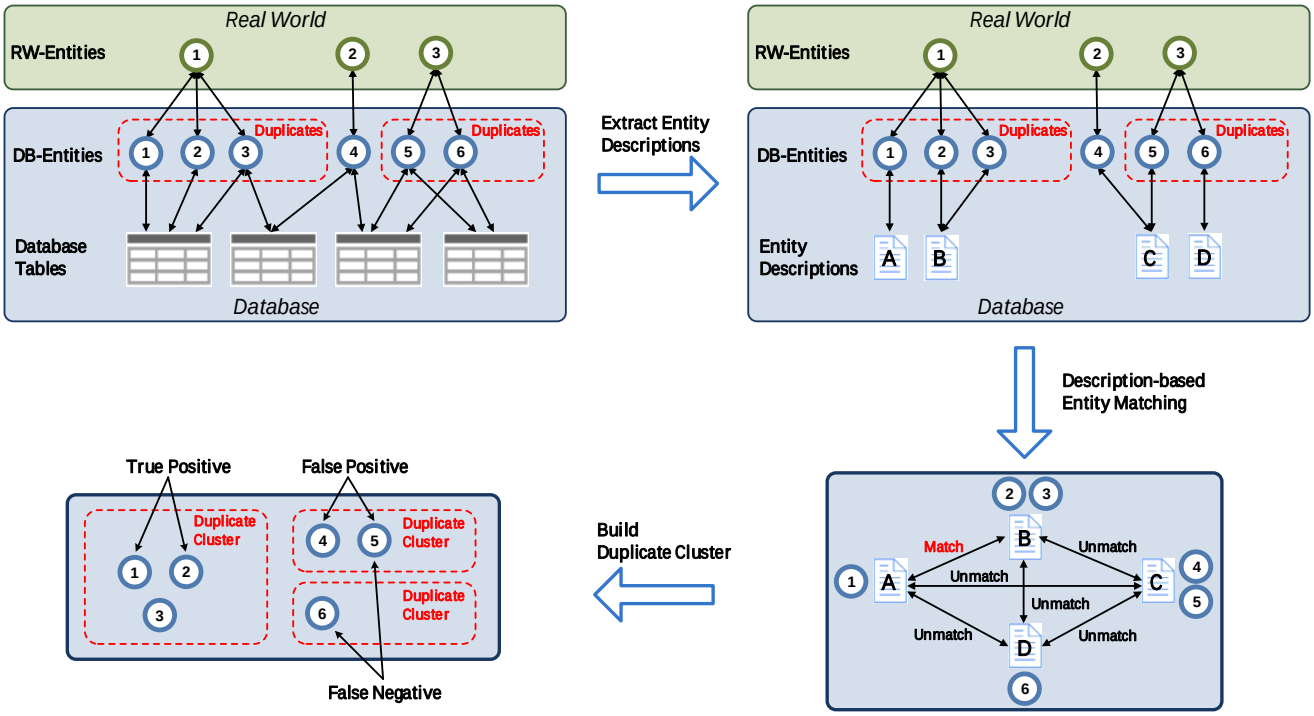


Figure 5.1.: Role of entity descriptions in a description-based matching process

matched by matching their descriptions. Finally from the pairwise matching results, the clustering of the database entities is derived.

Example 66 *In the sample of Figure 5.1, we have six database entities that map to three different real-world entities and are modeled in the database by tuples of several tables. In the first step we extract the descriptions from the database. Since some database entities have the same description, instead of six only four different entity description result. If entity matching does not use any external knowledge, i.e. it is only based on the descriptions, a pairwise matching of all entities can be reduced to a pairwise matching of all descriptions. In the considered example, we assume such a simple matching approach. Thus, the entity pairs* $\{e_2, e_3\}$ *and* $\{e_4, e_5\}$ *are automatically classified as* MATCHES. *In the third step, all four descriptions are pairwise matched (for matching details see Section 4.3). The matching process assigns each description pair (or entity pair respectively) to the set of* MATCHES *or the set of* UNMATCHES, *where two descriptions match if they are so similar that it can be implied with a high confidence that their corresponding entities are duplicates.*

In the given example, we assume that only the descriptions A *and* B *are classified as a* MATCH *and therefore the entity pairs* $\{e_1, e_2\}$ *and* $\{e_1, e_3\}$ *are indirectly classified as* MATCHES. *From the set of* MATCHES *and* UNMATCHES *the final duplicate clustering is derived. In the considered example, three duplicate cluster result: One with the database entities* e_1, e_2, *and* e_3, *one with the database entities* e_4 *and* e_5, *and one with only the database entity* e_6.

Before going into description details, we have to distinguish between a concrete description of a specific database entity and a description type that is a rule that determines how the description of a

database entity looks like. Thus, the description type can be considered as a schema that models all duplicate detection relevant characteristics of a set of database entities.

Definition 25 ***(Entity Description:)*** *An entity description d is a pair $d = (d_A, d_R)$, where d_A is an attribute description and d_R is a relationship description.*

Recall, the function $f_{ED}(e, db)$ extracts the entity description of a database entity e from the considered database *db* and the function $\mathfrak{d}_{ED}(e, db)$ maps e to its entity description that is currently used for detecting duplicates in database *db*.

An entity description is denoted to be empty, if its attribute description and its relationship description are empty.

Definition 26 ***(Description Type:)*** *A description type $\mathfrak{T}$ is a pair $\mathfrak{T} = (\mathfrak{T}^{Attr}, \mathfrak{T}^{Rel})$, where $\mathfrak{T}^{Attr}$ is an attribute description type and $\mathfrak{T}^{Rel}$ is a relationship description type.*

5.1.1. Describing Attribute Information

The attribute description of a database entity contains all the entity's attribute values that are useful for detecting duplicates. Since surrogate keys typically do not represent a property of a real-world entity, it is usually excluded from the entity description. In this thesis, we restrict to single-valued attributes. Nevertheless, the description approach can be simply extended to multi-valued attributes. Recall, in turn to single-valued attribute, multi-valed attributes are modeled by tuples in multi-value tables and therefore need to be extracted by specific queries.

An attribute description and an attribute description type are defined as follows.

Definition 27 ***(Attribute Description:)*** *An attribute description d_A is a set of attribute-value pairs $\{(A_1, v_1), \ldots, (A_k, v_k)\}$, where each value $v_{i,1 \leq i \leq k}$ is an element of its corresponding attribute domain $dom(A_i)$ or the null value $\perp$. Since the value for each attribute should be unique, each attribute-value pair refers to another attribute, i.e.: $\forall (A_i, v_i), (A_j, v_j) \in d_A \colon A_i = A_j \Rightarrow v_i = v_j$.*

An attribute description is denoted to be empty, if it is the empty set or if all attributes are paired with the null value.

Definition 28 ***(Attribute Description Type:)*** *An attribute description type $\mathfrak{T}^{Attr}$ is a set of attributes $\{A_1, \ldots, A_k\}$. An attribute description d_A is conform to an attribute description type $\mathfrak{T}^{Attr}$ (denoted as $d_A \vdash \mathfrak{T}^{Attr}$), if it is defined on the same set of attributes, i.e.: $\bigcup_{(A,v) \in d_A} \{A\} = \mathfrak{T}^{Attr}$.*

In the rest of this thesis, we sometimes directly address the attribute description of a database entity. For that purpose, we introduce the two functions $f_{AD}(\cdot)$ and $\mathfrak{d}_{AD}(\cdot)$ The function $f_{AD}(e, db)$ extracts the attribute description of database entity e from the considered database *db* and the function $\mathfrak{d}_{AD}(e, db)$ maps e to its attribute description that is currently used for detecting duplicates in database *db*.

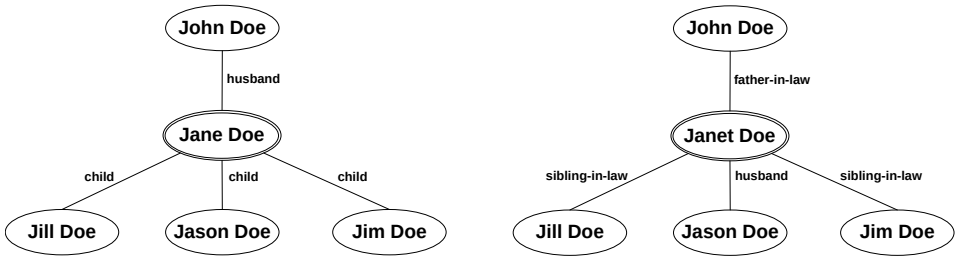


Figure 5.2.: Example for illustrating the disadvantages of the simple approach for describing relationship information

5.1.2. Describing Relationship Information

The relationship description of a database entity should describe the interactions of this entity to other database entities. Obviously, we have to distinguish between a type of relationship (e.g. *is brother of*) and a specific instance of a relationship (e.g. *Tim is brother of Max*). In general, a database entity can strike up multiple instances of the same type (e.g. a person can be the *brother* of several other persons). In addition, relationship types can be equipped with attributes. Moreover, the database entities that are involved in a relationship are associated with role names (e.g. *wife* and *husband* in a relationship *marriage*) and an entity can fill in different roles in different instances (or even the same instance) of the same relationship type. Finally, a database entity can be in different relationships with another database entity (e.g. the *brother* of *Tim* is also his *neighbor*). In summary, the relationships of a database entity model complex information.

Due to this complexity, a complete description of all relationship information is usually not desired and an incomplete, but a simpler description method is used instead. Recall, Naumann and Herschel [NH10] simply describe the relationship information of a database entity by the set of all other entities that are related with the considered entity by any mean. Thus, this description approach does not distinguish between the semantics, i.e. types, of the relationships nor take the role names of the related entities nor attribute values into account. Consequently, this approach produces very simple descriptions that are fast and easy to match, but can loss some useful information.

The disadvantage of this simple description approach should be illustrated by the following example.

Example 67 *Let us consider the scenario pictured in Figure 5.2 where we have to describe the relationship information of two persons p_1 and p_2, the first is named 'Jane Doe' and the second is named 'Janet Doe'. Jane Doe has a husband who is named 'John Doe' (p_3) and three children which are named 'Jill Doe' (p_4), 'James Doe' (p_5), and 'Jim Doe' (p_6). Janet Doe is married with James Doe. Consequently, John Doe is her father-in-law and Jill Doe as well as Jim Doe are her siblings-in-law. If the information that 'Jane Doe' is the mother-in-law of 'Janet Doe' is unknown or if we would ignore the given information in the extraction of the relationship description for whatever reason, the relationship descriptions of both persons would result in the same* related entity set*, i.e.: $RD(p_1) = RD(p_2) = \{p_3, p_4, p_5, p_6\}$. As a consequence, the risk to mistakenly classify both persons as a* MATCH *is very high, because they have similar names and have the same relationship description. On the contrary, if we additionally take the respective relationship types into account and compute a set of related entities per type instead of computing a single set, the descriptions of both persons would result in:*

	husband	***child***	***father-in-law***	***sibling-in-law***
p_1	$\{p_3\}$	$\{p_4, p_5, p_6\}$	$\emptyset$	$\emptyset$
p_2	$\{p_5\}$	$\emptyset$	$\{p_3\}$	$\{p_4, p_6\}$

Thus, although p_1 and p_2 are related to the same set of persons, the descriptions are now so discriminative that we can clearly distinguish between both persons.

In general, the more information is considered for describing relationship information, the more discriminative the resultant descriptions become. Since we still neglect information on role names and attributes in the previous example, more sophisticated but also more complex description approaches are theoretically possible. Complex approaches can be especially useful in cases where duplicates can be most often only detected by their relationships as for example in social networks where, because of the use of alias names, the relationships to other persons, or marked items (e.g. favorite movies or books) is the most identifying kind of information.

A complete discussion about describing relationship information is out of the scope of this thesis. Nevertheless, our formalism should be as generic as possible so that more detailed description approaches can be incorporated easily. For that reason, we define a relationship description in a flexible way.

Definition 29 ***(Relationship Description:)*** *A relationship description d_R is a set of label-set pairs $\{(L_1, S_1), \ldots, (L_k, S_k)\}$ where each label serves as a semantical key for a specific kind of relationship information and each set is a collection of description items. Since the set of description items should be unique for each label, each label-set pair refers to another label, i.e.: $\forall (L_i, S_i), (L_j, S_j) \in d_R \colon L_i = L_j \Rightarrow S_i = S_j$.*

A relationship description is denoted to be empty, if it is the empty set of if all sets of description items are empty, i.e. if for each $(R, S) \in d_R$ holds: $S = \emptyset$.

Each label represents a specific kind of relationship information and therefore need to be connected with a query that extracts this information from the database. In the simplest case, we only consider a single query that selects all entities which are related with the considered entity by any mean and hence produce a single set of database entities. By doing so, relationship information is described by a single label-set pair where the description items are database entities and the relationship descriptions is conform to the original definition made by Naumann and Herschel [NH10].

Nevertheless, this description approach is not restricted to the simple case, but multiple complex queries can be used instead. One option is to define a query per relationship type that is explicitly modeled in the database either by a relationship table (many-to-many) or by a foreign key (one-to-one or one-to-many). If the queries return all the entities that are involved in such a relationship with the considered entity, the relationship description contains a set of label-set pairs where each label corresponds to a relationship type and each set corresponds to a set of related entities. Queries, however, can also extract entities that are implicitly related or can extract more complex description items that include the correlations of roles and/or attributes.

Example 68 *We want to illustrate the flexibility of this description approach by four sample queries that are defined on a database schema that contains the relationship table* 'Medicates'. *This table is represented in Figure 5.3(e) and models the relationship information which doctor medicates which patient*

```
SELECT m.patient AS Entity
FROM Medicates m
WHERE m.doctor = 'p1'
UNION
SELECT m.doctor AS Entity
FROM Medicates m
WHERE m.patient = 'p1'
```

(a) Query returning all related persons

```
SELECT m.patient AS Entity
FROM Medicates m
WHERE m.doctor = 'p1'
```

(b) Query returning all patients

```
SELECT m2.patient AS Entity
FROM Medicates m1, Medicates m2
WHERE m1.doctor = m2.doctor
AND m1.patient = 'p1'
AND m2.patient ≠ 'p1'
```

(c) Query returning implicit relationships

```
SELECT m.patient, m.medicine, m.date
FROM Medicates m
WHERE m.doctor = 'p1'
```

(d) Query returning complex relationship information

Medicates (Relationship Table)

doctor	patient	medicine	date	dose
p1	p3	m1	12.11.07	3ml
p2	p3	m1	08.10.06	4ml
p2	p1	m2	21.05.07	10g
p1	p5	m1	05.06.10	3ml
p1	p5	m2	17.01.09	10g

doctor → MedStaff.DEI, patient → Patient.DEI, medicine → Medicine.DEI

(e) Relationship table *'Medicates'*

Figure 5.3.: Four sample queries for extracting relationship information from a database

by giving her which dose of which medicine at which date. Thus, this table has five attributes. Whereas the first three attributes 'doctor', 'patient', *and* 'medicine' *are foreign keys that reference to entity tables, the last two attributes* 'date' *and* 'dose' *are ordinary attributes. The first query (Figure 5.3(d)) returns all persons that are related to the considered person* p_1 *by a relationship of the type* medicates *whereby we do not distinguish between the individual roles, i.e. the query returns all persons that have been medicated by* p_1 *or that have medicate* p_1*. In the given sample instance, the result is the entity set* $\{p_2, p_3, p_5\}$.

In contrast, the query from Figure 5.3(b) restrict to a specific role and only returns person that have been medicated by p_1*. The query result is the entity set* $\{p_3, p_5\}$.

The third query (Figure 5.3(c)) returns all persons that have been medicated by the same doctor than p_1 *and therefore returns persons that are related by an implicitly modeled relationship, i.e. it returns the entity set* $\{p_3\}$.

Finally, the last query presented in Figure 5.3(a) does not return a set of entities, but returns a set of triples where the first two elements are entities, i.e. persons and medicines, that are associated with roles and the last element is an attribute. The query result is therefore a set with the three description items (('patient', 'p_3'), ('medicine', 'm_1'), ('date', '12.11.07')), (('patient', 'p_5'), ('medicine', 'm_1'), ('date', '05.06.10')), *and* (('patient', 'p_5'), ('medicine', 'm_2'), ('date', '17.01.09')).

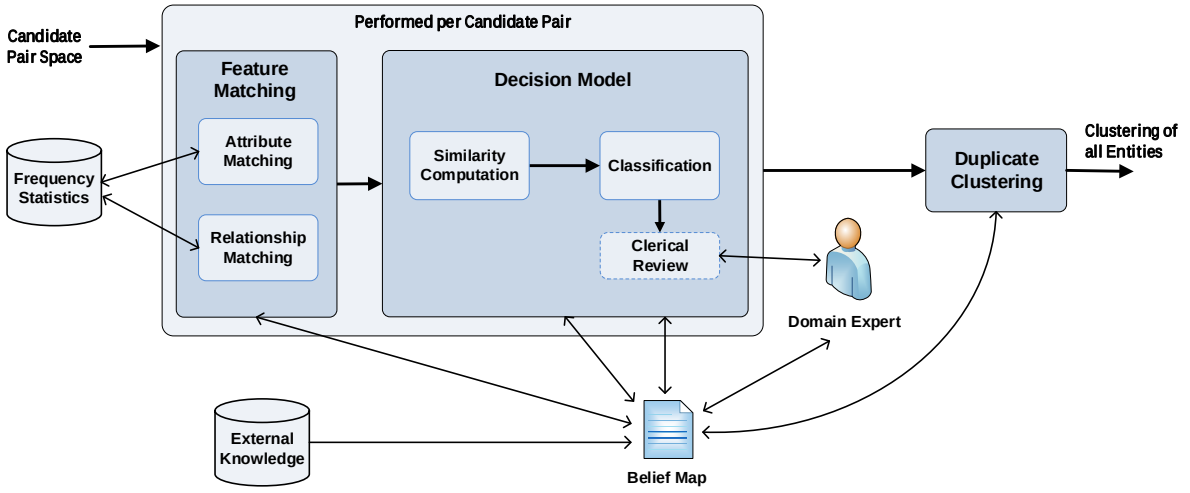


Figure 5.4.: Execution model of the pairwise matching phases with HaDDeF

Definition 30 ***(Relationship Description Type:)*** *A relationship description type* $\mathfrak{T}^{Rel}$ *is a set of label-domain pairs* $\{(L_1, dom_1), \ldots, (L_k, dom_k)\}$ *where each label is unique, i.e.* $\forall (L_i, dom_i), (L_j, dom_j) \in \mathfrak{T}^{Rel} \colon L_i = L_j \Rightarrow dom_i = dom_j$*. A relationship description* d_R *is conform to a relationship description type* $\mathfrak{T}^{Rel}$ *(denoted as* $d_R \vdash \mathfrak{T}^{Rel}$*), if it is defined on the same set of labels and if for each label its set of description items is a subset of the domain that is assigned to this label by the description type, i.e.:* $\bigcup_{(L,S)\in d_R}\{L\} = \bigcup_{(L,dom)\in\mathfrak{T}^{Rel}}\{L\} \wedge \forall (L,S) \in d_R \colon \exists (L, dom) \in \mathfrak{T}^{Rel} \colon S \subseteq dom$.

In this thesis, we will restrict our considerations of description items to the domain of database entities. Nevertheless, the approaches presented in this thesis can be easily extended to more complex domains if measures for computing the similarity between two elements of these domains are given.

Function $f_{RD}(e, db)$ extracts a relationship description of database entity e from the considered database db and function $\mathfrak{d}_{RD}(e, db)$ maps e to its relationship description that is currently used for detecting duplicates in database db.

5.1.2.1. Comparability

Two entity descriptions are comparable, if their description types are comparable. Whether or not two description types $\mathfrak{T}_A$ and $\mathfrak{T}_B$ are comparable needs to be defined by a domain expert. Usually only entity descriptions that are conform to the same description type are matched. Sometimes, however, it makes sense to compare descriptions of non-equal types. For example, if pets are allowed to have human names and it cannot be excluded that a family's pet is confused with one of the family's kids and it can make sense to compare pets with peoples. For example in some circumstances it can be not completely clear, if the attribute description $\{(\textit{'firstname'}, \text{'Bob'}), (\textit{'lastname'}, \text{'Smith'}), (\textit{'DoB'}, \text{'13.12.2003'})\}$ describes a kid or a pet. For simplification we always assume a comparison of entity descriptions that are conform to the same description type.

5.2. Description-based Entity Matching

As mentioned above database entities are matched by comparing their entity descriptions. Principally, HaDDeF is based on the five phase model that we have introduced in Section 4.3, but we divide the decision model phase into two subphases: The *similarity computation phase* and the *classification phase*. Whereas the first computes a single similarity score (called *entity similarity*) per candidate pair based on the similarities of their attribute descriptions and relationship descriptions, the classification phase uses one or two thresholds to derive a matching class based on the given score of entity similarity. Note, the most of the decision models presented in Section 4.3.6 can be used to compute a similarity score and therefore can be used as an implementation of the similarity computation phase in HaDDeF.

Due to the consideration of relationship information, we need to analyze the attribute description and the relationship description and therefore do not consider attribute value matching as an own phase anymore, but consider it as a part of a larger phase that we call the *Feature Matching Phase*. Moreover we extend some phases to the use of two concepts, the *belief map* and the *impact values*, making the detection process more effective. Figure 5.4 presents the execution model of the pairwise matching phases of HaDDeF and illustrates the interactions between the belief map and the individual matching phases.

We start with introducing the new concepts that extent the matching process in Section 5.2.1 (belief map) and Section 5.2.2 (impact values). Then we present an overview over the whole matching process in Section 5.2.3 and describe in which way the individual phases are adapted to the use of relationship information, the belief map, and the impact values. Finally, we present a method for incorporating information on multi-table memberships into our approach for describing and matching entities.

5.2.1. Belief Map

To enable an incorporation of external knowledge on concrete entity pairs into the detection process, we propose the *Belief Map*. The belief map is managed separately from the individual phases and stores a *belief vector* $\vec{v}_B \in [0,1] \times [0,1]$ per entity pair. The first element of the belief vector is the confidence that both entities are a MATCH and the second element is the confidence that both entities are an UNMATCH. It is important to note that we consider confidences instead of probabilities and a confidence of 0 for a MATCH does not implicate a confidence of 1 for an UNMATCH. In contrast, the standard belief vector (no knowledge is incorporated) is $\vec{v}_B = \langle 0, 0 \rangle$. Theoretically, the belief map contains a belief vector per entity pair. Nevertheless, since for most entity pairs no external knowledge is available, the vector is actually only stored for pairs with $\vec{v}_B \neq \langle 0, 0 \rangle$. A sample instance of a belief map is shown in Table 5.1.

The purpose of the belief map is manifold (see Figure 5.4). First, it enables the domain expert to incorporate his expertise or any external knowledge permanently into the detection process at any matching phase (e.g. by clerical reviews). Second, unnecessary computations can be avoided. For example, if the confidence that a candidate pair is a MATCH or an UNMATCH is equal to 1, the pair's decision is obvious without performing any of the matching phases. At least, it unburdens domain experts in clerical reviews because it enables them to also incorporate their own degree of belief rather than a certain knowledge.

Example 69 *For illustration we consider a scenario in which duplicate database entities from several different sources need to be identified and it is known for sure that one of the sources is duplicate-free,*

candidate pair	*confidence* MATCH	*confidence* UNMATCH
$\{e_1,e_2\}$	*0.9*	*0.0*
$\{e_1,e_5\}$	*0.2*	*0.7*
$\{e_3,e_7\}$	*0.4*	*0.0*
$\{e_4,e_5\}$	*0.0*	*1.0*

Table 5.1.: Sample instance of the belief map

i.e. two entities of this source cannot be duplicates. Without using the belief map, we can incorporate this knowledge only by omitting such pairs from the candidate pair space and hence by initially classifying them as UNMATCHES. *Nevertheless, some approaches for duplicate clustering change* UNMATCHES *to* MATCHES *in order to satisfy the transitivity of real-world equivalence. Changing two entities from the duplicate-free source to a* MATCH*, however, contradict to the external knowledge. As a consequence, simply omitting these pairs from the candidate pair space is not sufficient to ensure that the detection result is conform with this knowledge. However, if we store this knowledge into the belief map, we can incorporate it at any matching phase and hence enable the clustering approach to distinguish between hard* UNMATCHES *(certainly no* MATCH*) and soft* UNMATCHES *(likely no* MATCH*).*

In another integration scenario we may know that some entities are certainly duplicates, because they are explicitly linked to be real-world equivalent (see concept of Linked Data *[HB11]). In such a case, the belief map can be used to ensure that these entities will end up in the same duplicate cluster.*

The belief map can be of an additional use to store certain decisions that result from evaluating non-identity constraints on the database, e.g. if two database entities are detected to be father and son they are certainly no duplicates. Moreover, it can be used in collective duplicate detection approach in order to propagate decisions from one iteration to another.

5.2.2. Impact Values

We have observed that the identification power of two attribute values does not only depend on their similarity and the importance of the attributes these values belong to, but is also affected by some further characteristics of these values as their *content size* and their *value rarity*[1]. As a consequence, in many use cases, the effectiveness of a decision model could be improved, if its decision making is not only based on the similarity scores of the comparison vector and the importance of the attributes, but also takes the significance of the computed similarity scores into account. For that purpose we introduce the concept of *impact values*. An impact value is a real number between 0 and 1 and models the significance of its corresponding similarity score. The impact values of all similarity scores of the comparison vector form a so called *impact vector*. A similar approach can be used for the similarity scores that result from comparing two relationship descriptions.

For duplicate detection in probabilistic data, impact values can be even useful for the results of later matching phases. For that purpose, we compute an impact value for the result of each matching phase

[1]Notice, similar characteristics have been listed by Leitao and Calado [LC11a] in order to measure the identification power of attributes. However, they consider these characteristics on the level of attributes and not on the level of attribute values. Therefore, the observations that have been made by these authors cannot be adopted to our purpose.

where the impact value of one phase is computed based on the impact values of its previous phase, i.e. the impact value of an entity similarity is based on the impact values of the attribute similarities and the relationship similarities the entity similarity is computed from.

Actually, an impact value refers to a similarity score. Nevertheless, if clear from context we sometimes simply write *'impact (value) of the two attribute values v_r and v_s'* or *'impact (value) of the attribute A'* instead of writing *'impact value of the similarity score that has been computed for the two values v_r and v_s from the attribute A'*. Note, referring to a single attribute is only possible if we consider a matching of values that belong to the same attribute. In the case of matching values across attributes we need to write *'impact (value) of the attribute pair (A_i, A_j)'* instead. Nevertheless, to simplify representation we usually abstain from matching values across attributes in this section. The impact of the values from two database entities e_r and e_s in attribute A, i.e. $e_r[A]$ and $e_s[A]$, is denoted as $imp(e_r[A], e_s[A])$. Impact values within an impact vector $\vec{i}$ can be also addressed as $\vec{i}(i)$ (the ith value in this vector) or as $\vec{i}(A_i, A_j)$ (the impact value in this vector that represents the attribute pair (A_i, A_j)).

The remainder of this section is structured as follows. First we present several factors that determine the significance of a score that represents the similarity between two attribute values. Then we discuss strategies to incorporate the impact vectors into the different types of decision models. Finally, we shortly discuss methods to compute the impacts of the later matching phases.

5.2.2.1. Factors that describes the Significance of the Similarity between two Values

A similarity scores gives an idea how similar two values are, but does not necessarily describe the likelihood that these values actually refer to the same thing of the real-world. To which extent a similarity score can be used to conclude on real-world equivalence depends on several factors, four of them are:

- **Content Size:** The first factor of significance is the content size of the compared values (e.g. length of strings or cardinality of sets). The greater the content size of two values, the more significant is the similarity between them, because the less this similarity has been affected by small errors (e.g. typographical errors in strings).

  For illustration, we consider the normalized Levenshtein Similarity between two strings of length three and two strings of length twelve. First, we assume that each pair of strings represents the same thing of the real-world, e.g. the same first name. Although the two short strings are actually equivalent, one typo is sufficient to reduce the similarity from 1 to $2/3$ and a swap of letters already decreases the similarity from 1 to $1/3$. In contrast, a reduction of similarity between two strings of length twelve from 1 to $2/3$ requires at least four typos and a reduction from 1 to $1/3$ would implicate at least eight typos. Thus, the likelihood that two dissimilar long strings actually refer to the same real-world thing is much less than for two dissimilar short strings.

  This problem is also present in the other direction where strings that do not refer to the same thing of the real-world become very similar or even equivalent by chance or by accident. For instance, it requires only two typos to increase the similarity from $1/3$ to 1, if the strings have only three letters. In contrast, an increase of similarity from $1/3$ to 1 requires at least eight typos for strings of length twelve. As a consequence, a high similarity between two long strings is an higher indication for real-world equivalence than a high similarity between two short strings.

Example 70 *We want to demonstrate it by a concrete example. The two strings 'Jim' and 'Jon' have only one letter in common and hence have a normalized Levenshtein Similarity of* $1/3$*. Indeed, on a first look it seems to be high likely that they actually refer to different names. On the other hand, the letters 'i' and 'o' as well as 'm' and 'n' are neighbored on a typical keyboard and hence make a sequence of two typos also possible. In contrast, the similarity of* $1/3$ *between the two strings 'Christian' and 'Fabian' is an high indication that both strings refer to different names and a sequence of typos can be nearly excluded for sure.*

Another illustrative case of application is a matching of middle names that are often only given by their first letter. The similarity between the two middle names 'M' and 'M' is 1.0, but is only a low indication for a duplicate whereas the similarity between the two middle names 'Marcus' and 'Markus' is an higher indication for a duplicate although it is only $5/6$ (Levenshtein Similarity) and thus lower than the first. The same holds in the inverse direction. The similarity between the middle names 'W' and 'B' is zero, but is only a low indication for a non-duplicate because they could represent the highly related names 'William' and 'Bill' or the difference between them could be caused by a mistake that is based on the phonetic similarity between 'W' and 'B'. In contrast, the Levenshtein Similarity between the middle names 'Werner' and 'Bernd' is a much higher indication for a non-duplicate despite it is 0.5 and thus greater than the Levenshtein Similarity between 'W' and 'B'. This example moreover demonstrates that a utilization of impact values enables a better use of secondary attributes, i.e. attributes that are generally suitable to discriminate MATCHES from UNMATCHES, but its values are often missing or are only filled partially, to classify candidate pairs.

The same problem holds for the similarity of sets as they result from tokenizing string values or as they appear in a consideration of relationship descriptions. The larger the compared sets the more significant is the similarity between them. For instance, the Jaccard Coefficient of the two sets $S_1 = \{e_1\}$ and $S_1 = \{e_1, e_2\}$ is 0.5 even only a single element is missing in S_1. In contrast, for large sets with around 1000 elements, a Jaccard Coefficient of 0.5 requires a high number of non-common elements.

- **Value Rarity:** Besides content size, the frequencies of the compared values have an impact on the significance of their similarity, because the more rare a value is, the less likely is the case that two different entities share that value. For example, an equal last name 'Smith' (in the year 2000, this last name occurs in the United States $2,376,206$ times[2]) is a much less indication for a MATCH than an equal last name 'Ribak' (in the year 2000, this last name occurs in the United States 106 times[3]).

  Of course, the significance of a similarity score does not only depend on the frequency of the compared values, but also on their contexts. As an example we consider an attribute description with the tree attribute-attribute value pairs (*'first name'*, 'Maximilian'), (*'last name'*, 'Cheng'), and (*'residence'*, 'Berlin'). 'Max' is a common first name, 'Cheng' is a common last name and

[2] http://names.mongabay.com/data/sm/SMITH.html
[3] http://names.mongabay.com/data/ri/RIBAK.html

around three million people live in Berlin. Nevertheless, whereas 'Cheng' is a common last name in Beijing, it is less common in Berlin. Moreover, whereas 'Max' and 'Cheng' are both frequent names, the combination is less frequent. Thus, for computing the significance of the similarity between two attribute values, it can be useful to compute the values' frequencies in combination with the values of some related attributes.

Similar holds for sets, especially sets of related entities, because an often related entity is not as discriminating than a less often related entity. For instance, the information that two person both like the relatively unknown singer Kat Frankie ($220,522$ '*scrobbels*' on lastfm[4]) is an higher indication for a MATCH than the information that they both like Madonna (around $117,117,854$ '*scrobbels*' on lastfm[5]).

As presented in Section 4.3.5, some set similarity measures take the frequency of values into account, e.g. by using *TF-IDF*, but restrict this consideration to weight the individual elements in computing the similarity of the compared sets. In the end, however, two sets with many common elements will be associated with a high similarity score, even if all their elements are frequent (in that case, the weighting compensate itself). This characteristic is a necessity for these measures as well as a disadvantage at the same time. On one hand, two equivalent sets must produce a similarity score of 1, because otherwise it could finally happen that identical entities are not detected as MATCHES, but on the other hand a high similarity between two sets with rare elements is much more significant than a high similarity between two sets with frequent elements. Due to the first, the latter cannot be handled by the similarity measure itself and thus must be considered in another way as we do with the impact values.

- **Measure Significance:** The third factor is the suitability of the measure that has been used for computing the similarity score. Obviously, this factor is primarily relevant if values of the same pair of attributes are matched by different measures. This is especially the case in the presence of null values, because conventional similarity measures are inapplicable to match them and a fixed similarity score is used instead. Nonetheless, as discussed in Section 4.3.5.7, using a fixed similarity score implicates a trade-off between running the risk of producing a false positive (high score for null values) or running the risk of producing a false negative (low score for null values). By assigning low significances to similarity scores that result from comparing one or two null values, we can simply avoid this trade-off, because it enables the decision model to prefer the similarity scores of non-null value pairs to the similarity scores of null-value pairs for making its final decision.

  This factor is also useful, if we need to compare values from a domain for which we do not know a suitable similarity measure, but instead have to use a conventional measure without having any information on its suitability. In that case, we can inform the subsequent phases about this ambiguity by decreasing the impact of the produced similarity scores.

  Finally, for reasons of efficiency values are sometimes initially matched by using a simple similarity measures and then are only re-matched by using a more complex similarity measure if the

[4] http://www.lastfm.de/music/Kat+Frankie,date:01.05.2014.
[5] http://www.lastfm.de/music/Madonna,date:01.05.2014.

score that was computed by using the simple measure is above a given threshold. Obviously, the result of the complex measure is more reliable than the result of the simple measure. Nevertheless, by doing so some of the finally used scores are computed by using the simple measure and some of the finally used scores are computed by using the complex measure. As a consequence, these scores are only less comparable.

Example 71 *For illustration, we consider a matching approach that compares two values by first computing their simple similarity and then computes their complex similarity only if the simple similarity is above the threshold* $\theta = 0.3$*. Now we assume two value pairs. The first pair has a simple similarity of* 0.4 *and the second pair has a simple similarity of* 0.2*. As a consequence, the first pair is additionally matched by using the complex similarity measure, but the second is not. Now let us assume that the complex similarity of the first pair is only* 0.1*. Thus, in the final comparison vector, the first pair is represented by a similarity of* 0.1 *and the second pair is represented by the a similarity of* 0.2*. These similarity scores, however, are somewhat misleading because according to the simple measure, the similarity of the first pair is greater than the similarity of the second pair.*

To solve this conflict, we can assign a low impact value to all similarity scores that have been computed by using the simple measure and can assign a high impact value to all similarity scores that have been computed by the complex measure.

- **Range Significance:** In some cases the significance of a similarity score also depends on the score itself, because it is only informative, if it is inside a specific range.

  Example 72 *For instance, two equal or high similar email addresses are an high indication for a* MATCH*, but two dissimilar email addresses are not an high indication for an* UNMATCH*, because it is not unusual that a person has several email addresses. For that reason, the resultant similarity score is only significant if it is close to one and non-significant else.*

  Whereas most decision models can handle that problem, decision-based models as described in Section 4.3.6.1 can not, because assigning such an attribute with a high weight implicates that two dissimilar values are an high indication for an UNMATCH and assigning it with a low weight implicates that two similar values are not an high indication for a MATCH. This shortcoming, however, can be solved by using different impact values for different ranges.

5.2.2.2. Computing Impact Values

In the previous section, we present several factors that influence the significance of a similarity score. The next challenge is to rate these influences by numbers and to combine them to a single impact value.

- **Content Size:** We start with the content size. Intuitively, we can use the maximal content size of the database (or any other available statistic) to normalize the size of the individual value pairs. Thus, let A be the considered attribute, let $|v|$ be the size of value v, and let s_{max} be the maximal

size that appears in the values' domain, the content size of the two values v_r and v_s in A can be computed as:

$$\textit{ContSize}(v_r, v_s, A) = \frac{\textbf{agg}(|v_r|, |v_s|)}{s_{max}} \tag{5.1}$$

where **agg** is an aggregation function that is most suitable for the considered domain.

However, we think that the likelihood that two strings each with three letters are accidentally high similar because of a sequence of typographical errors is independent from the maximal length the strings of the corresponding domain can have. Moreover, as we have observed in our experiments, this computation is often too strict because the most string values have a length less than 50% of the maximal length, but are still long enough to produce significant similarity scores. For that reason, we use a functions that does not increase linear, but polynomial and therefore compute the content size of the values v_r and v_s in attribute A as:

$$\textit{ContSize}(v_r, v_s, A) = 1 - \frac{1}{(1-c) + c \times \textbf{agg}(|v_r|, |v_s|)} \tag{5.2}$$

where $c \in \mathbb{R}^+$ is a constant that can be used to adjust the function's slope to the considered use case.

It is important to note that we abstract from the used aggregation function because in some domains suing the maximal size seems to be most suitable, but in other domains sing the minimal size seems to be the better option. For illustration, in the domain of first names that are usually written-out, the maximal string length seems to be most appropriate because it determines the number of typos that are required to transform one string into the other. In contrast, in the domain of middle names, names are often abbreviated by a single letter and using the maximal length to compute the impact value would implicate that the low similarity between the two values 'W' and 'William' is highly significant, but it is obvious that this is actually not the case because the first can be an abbreviation of the second.

- **Value Rarity:** The most intuitive measures for quantifying the rarity of two values is based on their aggregated frequency. Let A be the considered attribute, let $\textit{freq}(v, A)$ be the frequency of value v in A, and let m be a value with $\textbf{max}_{v \in A}\textit{freq}_A(v) \leq m \leq |A|$, the value rarity factor of the two values v_r and v_s in A can be computed as:

$$\textit{ValRarity}(v_r, v_s, A) = \frac{\textbf{agg}(\textit{freq}_A(v_r), \textit{freq}_A(v_s))}{m} \tag{5.3}$$

where **agg** is an aggregation function that is most suitable for the considered domain.

Of course, more complex measures are conceivable and maybe more suitable. For instance, as mentioned before it can be useful to consider the frequency of the individual values in combination with the values of some related attributes.

- **Measure Significance:** In the case of measure significance we restrict our consideration to the special case of null values and set measure significance to 0 if at least one of the compared values is null.

- **Range Significance:** Range significance depends on the semantics of the considered attributes. Thus, we cannot define a general computation method for this factor.

A similarity score is already less significant if one of its factors is low. For that reason, we compute the impact value for a similarity score by the product of its content size, its value rarity, its measure significance, and its range significance.

5.2.2.3. Incorporating Impact Values into Similarity Computation

After analyzing the factors that affect significance, we discuss methods to incorporate the produced impact values into the decision model.

- **Distance-based Decision Models:** First we consider distance-based decision models as presented in Section 4.3.6.1. The idea here is that we do not use the same weighting scheme for all candidate pairs, but use the impact values to compute an individual weighting scheme per pair. Let $\{e_i, e_j\}$ be an candidate pair with the comparison vector $\vec{c} = \langle c_1, c_2, \ldots, c_n \rangle$ and the impact vector $\vec{i} = \langle v_1, v_2, \ldots, v_n \rangle$, and let $W = \{w_1, w_2, \ldots, w_n\}$ be the weighting scheme that is used for all candidate pairs whose descriptions agree with a specific description type, the individual weighting scheme of $\{e_r, e_s\}$ is computed as $W_{rs} = \{w_1 \times v_1, w_2 \times v_2, \ldots, w_n \times v_n\}$.

- **Rule-based Decision Models:** In rule-based decision models, the values of the impact vector can be included into the rules' predicates. For example, the first rule from Figure 4.13 can be adapted to the following two rules by demanding that for high impact values a lower similarity score is required than for smaller impact values.

$$\boldsymbol{R}_{1a}: \quad (\vec{c}(\text{fname}, \text{fname}) > 0.6) \wedge (\vec{i}(\text{fname}, \text{fname}) > 0.9) \wedge \ldots \quad \Rightarrow \textsc{Match}$$

$$\boldsymbol{R}_{1b}: \quad (\vec{c}(\text{fname}, \text{lname}) > 0.8) \wedge (\vec{i}(\text{fname}, \text{fname}) > 0.7) \wedge \ldots \quad \Rightarrow \textsc{Match}$$

- **Learning-based Decision Models:** For learning-based decision model the impact vector can be appended to the comparison vector so that the learning algorithm use the similarity scores as well as the impact values to train the classifier.

For demonstrating an incorporation of impact values into distance-based decision models and to illustrate the positive effects that result from using impact values, we consider the following example.

Example 73 *For illustration, we consider the five entities from Figure 5.5(a) and consider the three candidate pairs* $\{e_1, e_2\}$, $\{e_3, e_4\}$, *and* $\{e_4, e_5\}$. *The comparison vectors and the impact vectors of these pairs are presented in Figure 5.5(b) and Figure 5.5(c) respectively. The values of the impact vectors result as follows: Due to the names 'tim' and 'tom' are both short and frequent, their impact is relatively small. The middle names 'marcus' and 'markus' have an average length and have an average rarity. Therefore, we set their impact to* 0.6. *The last names 'ribak' are rather short but less frequent and hence they get an impact of* 0.7. *The phone numbers of* e_1 *and* e_2 *are complete and equal. Therefore, they are a good indication for a* MATCH *and we assign a high impact value (range significance) to their similarity. In contrast, the email addresses are long, less frequent and dissimilar. Therefore, because of the range*

DEI	fname	mname	lname	DoB	phone	email
e1	tim	marcus	ribak	1982-07-23	0345-233848	abc@me.com
e2	tom	markus	ribak	⊥	0345-233848	tribak@you.uk
e3	chandrashekhar	george	tucker	1974-03-11	⊥	ctacker@me.com
e4	chandrasehkar	g	tacker	1974-03-11	0542-624678	ctacker@me.com
e5	charles	g	tacker	1974-11-03	0542-639999	chtacker@you.com

(a) Sample database table

	fname	mname	lname	DoB	phone	email
$\vec{c}(e_1,e_2)$	2/3	5/6	1.0	0.0	1.0	0.0
$\vec{c}(e_3,e_4)$	11/14	1/6	5/6	1.0	0.0	1.0
$\vec{c}(e_4,e_5)$	5/13	1.0	1.0	0.8	0.2	0.8

(b) Sample attribute comparison vector

	fname	mname	lname	DoB	phone	email
$\vec{i}(e_1,e_2)$	0.4	0.6	0.7	0.0	1.0	0.0
$\vec{i}(e_3,e_4)$	0.9	0.1	0.4	1.0	0.0	1.0
$\vec{i}(e_4,e_5)$	0.9	0.1	0.4	1.0	0.2	0.2

(c) Sample attribute impact vector

Figure 5.5.: Sample for illustrating the incorporation of impact values into a distance-based decision model

significance (only equal email addresses are useful) their impact is set to a low value. For the candidate pair $\{e_3, e_4\}$*, we use a high impact value for their first names because they are both rare and long, but use a lower impact value for their last names because both names are frequent and of an average length. The similarity of their middle names is associated with a low impact, because of the length of the shorter one. The birthdays are equal and complete and hence get the maximal impact (here we assume that each date is equally frequent in the database). The email addresses are equal and complete. Thus, we set their impact to* 1.0*. The first names 'charles' and 'chandrasehkar' are average frequent ('charles') or less frequent ('chandrasehkar') respectively. Moreover, the longest of these names has an above-average length. Therefore, the corresponding impact is set to a high value. The middle names of* e_4 *and* e_5 *only have a single letter and therefore have a low impact. The last names of both entities are frequent and of an average length. For that reason, we assign them with an impact value of an average size. The phone numbers and email addresses of* e_4 *and* e_5 *are similar, but not high similar and hence they are only associated with a low impact value. Finally, all value pairs with at least one null value get an impact value of zero, because we cannot derive any useful information from them.*

Now, let us assume that the two entities e_1 *and* e_2 *as well as the two entities* e_3 *and* e_4 *are duplicates, but the entities* e_4 *and* e_5 *are not.*

For decision making we use a distance-based decision model that computes the average attribute distance with the weighting scheme $W = \{0.2, 0.1, 0.2, 0.1, 0.2, 0.2\}$*. First we compute the weighted average similarities without using the impact vectors. These similarities result in:*

$$AAD(e_1, e_2, W) = 0.2 \times 2/3 + 0.1 \times 5/6 + 0.2 \times 1 + 0.1 \times 0 + 0.2 \times 1 + 0.2 \times 0 \quad = \underline{0.617}$$
$$AAD(e_3, e_4, W) = 0.2 \times 11/14 + 0.1 \times 1/6 + 0.2 \times 5/6 + 0.1 \times 1 + 0.2 \times 0 + 0.2 \times 1 \quad = \underline{0.641}$$
$$AAD(e_4, e_5, W) = 0.2 \times 5/13 + 0.1 \times 1.0 + 0.2 \times 1.0 + 0.1 \times 0.8 + 0.2 \times 0.2 + 0.2 \times 0.7 \quad = \underline{0.637}$$

All similarities are between 0.6 *and* 0.7 *and it is impossible to classify the two true duplicate pairs as a* MATCH *without classifying the true non-duplicate pair* $\{e_4, e_5\}$ *as a* MATCH *as well. As a consequence, without using the impact vectors the true* MATCHES *cannot be clearly distinguished from the true* UNMATCH.

Now, we use the impact vectors to compute an individual weighting scheme per candidate pair:

$$W_{12} = \{0.2 \times 0.4, 0.1 \times 0.6, 0.2 \times 0.7, 0.1 \times 0, 0.2 \times 1, 0.2 \times 0\} = \{0.08, 0.06, 0.14, 0, 0.2, 0\}$$
$$W_{34} = \{0.2 \times 0.9, 0.1 \times 0.1, 0.2 \times 0.4, 0.1 \times 1.0, 0.2 \times 0, 0.2 \times 1\} = \{0.18, 0.01, 0.08, 0.1, 0, 0.2\}$$
$$W_{45} = \{0.2 \times 0.9, 0.1 \times 0.1, 0.2 \times 0.4, 0.1 \times 1.0, 0.2 \times 0.2, 0.2 \times 0.2\} = \{0.18, 0.01, 0.08, 0.1, 0.04, 0.04\}$$

If we use these schemes to compute the weighted average similarities, the following scores result:

$$\begin{aligned} AAD(e_1, e_2, W_{12}) &= (0.08 \times 2/3 + 0.06 \times 5/6 + 0.14 \times 1 + 0 \times 0 + 0.2 \times 1 + 0 \times 0)/0.48 \\ &= \underline{0.924} \\ AAD(e_3, e_4, W_{34}) &= (0.18 \times 11/14 + 0.01 \times 1/6 + 0.08 \times 5/6 + 0.1 \times 1 + 0 \times 0 + 0.2 \times 1)/0.57 \\ &= \underline{0.894} \\ AAD(e_4, e_5, W_{45}) &= (0.18 \times 5/13 + 0.01 \times 1.0 + 0.08 \times 1.0 + 0.1 \times 0.8 + 0.04 \times 0.2 + 0.04 \times 0.7)/0.45 \\ &= \underline{0.617} \end{aligned}$$

For the first candidate pair, the phone number and the last name have the greatest impact on the final score. In contrast, for the second candidate pair, the first name, the date of birth, and the email address have the greatest impact on the computed similarity score. In the case of the third candidate pair, the first name and the date of birth have the greatest impact on the resultant similarity.

As we can see, the similarities of the two duplicate pairs are now around 0.9 *and the similarity of the non-duplicate pair is still around* 0.6 *and hence the true* MATCHES *can be clearly demarcate from the true* UNMATCH*. The reason is simple. The similarity score of the first pair was kept down because the average similarity of the first names and the zero similarity scores of the date of birth and the email address. Whereas the second similarity was zero because of null values and hence missing information, the third one was zero because of completely different email addresses. By using the impact vectors, these malfunctions become erased by assigning an average impact value to the attribute* 'fname' *(content size) and by assigning an impact value of zero to the attributes* 'DoB' *and* 'email' *(measure significance and range significance respectively). In the case of the second candidate pair, the similarity score was initially decreased because of the missing phone number and the dissimilar middle names. The impact values, however, neglect the first effect due to the significance of the used measure and neglect the second effect because of the short lengths of the names. The main reason that the similarity score of the third candidate pair* $\{e_4, e_5\}$ *decreases slightly by using the impact values is the fact that the similarity of the equal last names and the similarity of the equal middle names are both counted into the average similarity computation with lower weights. This in turn is caused by the low rarity of that last names and the short length of the middle names.*

For all three candidate pairs, the impact values of the middle names improve the result much, because the high similarity score between the middle names of e_4 *and* e_5 *as well as the low similarity score between the middle names of* e_3 *and* e_4 *are relativized, i.e. they do not have much impact on the aggregation result, and the high similarity score between the middle names of* e_1 *and* e_2 *is not relativized, i.e. it still has a high impact on the aggregation result.*

In summary, the impact vectors enable the decision model to base its decisions on the information that is most distinguishing for each of the individual candidate pairs and hence can improve the accuracy of the duplicate detection process to a large extent.

5.2.2.4. Computing Impact Values for Subsequent Matching Phases

As we will discuss in Section 7.2.1, for detecting duplicates in probabilistic data, it can be useful to also compute impact values for the results of the later detection phases. Intuitively, the impact of a duplicate decisions corresponds to the impact of the entity similarity it is derived from. Thus, regarding pairwise matching results we only need a method to derive an impact of the entity similarity score from the impact values that are computed for the value pairs that are used for computing the considered entity similarity. As we think, the similarity between two entities (or their descriptions respectively) is the more reliable the more significant the similarities between their attribute descriptions and their relationship descriptions are. Obviously, some attribute value similarities (or relationship similarities) have a greater impact on the similarity between the entities than others. In distance-based decision models, we can derive this impact from the individually computed weighting schemes. On a first glance, an attribute value similarity with zero impact should not affect the impact value of the corresponding entity similarity because it did not contribute to that score. On the other hand, the larger the number of significant attribute value similarities that contribute to the computation of the entity similarity, the more reliable is this computed score. By considering the first argument, we would assign the maximal impact value $imp(e_r, e_s) = 1$ to an entity pair $\{e_r, e_s\}$ if only one of its attribute value similarities has the maximal impact and all the other attribute value similarities have the impact zero. It is obvious that this is in conflict with the second argument. For that reason, we compute the impact an entity similarity score by the difference of the individually computed weighting scheme and the maximal possible weighting scheme, i.e. the scheme that result if all impact values are one. Note, the latter is identical to the original weighting scheme. Thus, let e_r and e_s be two database entities, let $W = \{w_1, \ldots, w_n\}$ be a weighting scheme and let $W_{rs} = \{w_1^{rs}, \ldots, w_k^{rs}\}$ be the individual weighting scheme that has been computed for $\{e_r, e_s\}$, the impact value of the entity similarity between e_r and e_s is therefore computed as:

$$imp(e_r, e_s) = 1 - \frac{\sum_{i=1}^{n} |w_i - w_i^{rs}|}{\sum_{i=1}^{k} w_i} \tag{5.4}$$

A probabilistic database can be represented by a set of possible worlds. Thus, we sometimes even need an impact value for a whole database or the database's duplicate clustering respectively. Let *db* be a database, we denote the impact value of this database as $imp(db)$.

Its seems intuitive to compute the impact of a clustering as the average impact of its entity pairs. By doing so, however, we ignore the fact that we do not have an impact value for each entity pair because most of them do not belong to the candidate pair space. Moreover, some possible worlds can contain more entities than others and a duplicate clustering is the more informative the more entities it contains. By simply computing the average impact of all entity pairs, however, we can come into a situation where a clustering with less entities gets a larger impact than a clustering with more entities. For both reasons, computing the impact of a clustering as the average impact of all its pairwise duplicate decisions should be treat with caution.

Another conceivable approach is to use the attribute features proposed by Leitao and Calado [LC11a]. By doing so we can compute the *uniqueness*, the *content length*, and the *absence* for each attribute as described in [LC11a], then combine all these values to a single normalized value that describes the information content of the whole database, and to use this value as the database's impact. Of course,

it seems useful to weight the individual attributes with their importance for the detection process. The shortcoming of this approach, however, is that it does not consider the individual impact values that we have computed for the individual value pairs. As a consequence, it does not make a difference whether or not we use these impact values in the used decision model.

In summary, computing suitable impact values for whole databases (or whole duplicate clusterings respectively) is not straightforward. For that reason, we encourage the user in Section 7.2.1 to use impact values of pairwise matching results instead of using impact values of whole databases.

5.2.3. Matching Process: An Overview

This section gives an overview of the six detection phases. We present each of these phases in detail and discuss which adaptation are possible or even necessary because of the relationship descriptions, the belief map, and the impact values.

5.2.3.1. Data Extraction

In the first phase, the entity descriptions need to be extracted from the databases. This is realized by user defined queries that either select useful attributes from an entity table (attribute description) or extract sets of related entities from entity tables or relationship tables (relationship description).

5.2.3.2. Data Preparation

The most preparation activities are only performed for increasing the effectiveness of the duplicate detection process and are not intended to increase the quality of the original data. Thus, we prepare the entity descriptions instead of the original data. We call a sequence of preparation activities as a preparation process. Formally, a preparation process is a function that maps an entity description to another entity description by applying all its preparation activities to the input description in the predefined order. Since a preparation process can include activities that change the structure of the data, e.g. merges or splits of attributes, the description type of the output description can differ from the description type of the input description.

Usually for each description type another preparation process can be defined. Thus, we usually work with a set of preparation processes (one for each description type of the input).

5.2.3.3. Candidate Pair Space Construction

The input to the candidate pair space construction phase is a set of database entities along with their entity descriptions. If relationship descriptions do not contain attributes, blocking key values are only extracted from the attribute description.

In HaDDeF the candidate pair space is constructed as described in Section 4.3.4. Sometimes, however, it is useful to expand the candidate pair space dynamically during one of the following phases by some specific entity pairs. Thus, the only difference to the standard execution model is that we allow a post-fetching of candidate pairs.

For illustration we consider a duplicate clustering phase in which the two entity pairs $\{e_r, e_s\}$ and $\{e_r, e_t\}$ are classified as a MATCH, but the pair $\{e_s, e_t\}$ is not part of the candidate pair space. Due to

the transitivity of identity, the pair $\{e_s, e_t\}$ must be a duplicate as well. One option is to classify it as a MATCH without further analysis as it is done by the approach of partitioning based on components. However, the confidence of the made decisions can be increased by matching the missing entity pair subsequently and hence by post-fetching it. Another useful example is a case where the similarity between two entities is required for matching the relationship descriptions of two other entities.

5.2.3.4. Feature Matching

The feature matching phase gets a candidate pair $\{e_r, e_s\}$, the entity descriptions $\mathfrak{d}_{ED}(e_r)$ and $\mathfrak{d}_{ED}(e_s)$, and the belief map as input. Moreover, if needed for impact value computation it collects data statistics from the database. The purpose of the feature matching phase is to extract all the information from the two descriptions that is required by the decision model to classify $\{e_r, e_s\}$ as a MATCH, a POSSIBLE MATCH, or an UNMATCH. The output of this phase is a feature score $fs(e_r, e_s) = (\vec{c}_A, \vec{i}_A, \vec{c}_R, \vec{i}_R)$ that encapsulates the results of matching and analyzing the attribute descriptions and the relationship descriptions of both entities as described below:

- Attribute descriptions are matched by using some of the similarity measures that we have described in Section 4.3.5. Since attribute values can be transposed (e.g. the last name and the first name of a person), we also allow a matching of values that belong to different attributes. Such an attribute across matching can be also valuable, if values of different attributes are often very similar or one is often encoded in the other. A good example for the latter case is the name of a person and its email address, because the email address often contains the person's name.

  From comparing two attribute descriptions d_1 and d_2 where d_1 is conform to type $\mathfrak{T}_1^{Attr}$ and d_2 is conform to type $\mathfrak{T}_2^{Attr}$, we obtain the *attribute comparison vector* $\vec{c}_A = \langle c_1, \ldots, c_m \rangle$ and the *attribute impact vector* $\vec{i}_A = \langle v_1, \ldots, v_m \rangle$, where each c_i represents the similarity of the values from the ith pair of attributes $(A_1, A_2) \in \mathfrak{T}_1^{Attr} \times \mathfrak{T}_2^{Attr}$ and each v_i represents the impact of the ith similarity score.

- Recall, in the case of relationship descriptions, we restrict our consideration on pairs of labels and entity sets. Two sets of related entities are matched as described in Section 4.3.9. Relationship information can be transposed as well. For example, a database tuple is inserted into the relationship-table *reviewer* instead of the relationship-table *author*. For that reason, we also allow a matching of sets from different labels.

  From comparing two relationship descriptions d_1 and d_2 where d_1 is conform to type $\mathfrak{T}_1^{Rel}$ and d_2 is conform to type $\mathfrak{T}_2^{Rel}$, we obtain the *relationship comparison vector* $\vec{c}_R = \langle c_1, \ldots, c_n \rangle$ and the *relationship impact vector* $\vec{i}_R = \langle v_1, \ldots, v_n \rangle$, where each c_i represents the similarity of the entity sets from the ith pair of labels $(L_1, L_2) \in \mathfrak{T}_1^{Rel} \times \mathfrak{T}_2^{Rel}$ and each v_i represents the impact of the ith similarity score.

The configuration of the feature matching phase generally depends on the description types the considered entity descriptions are conform with.

5.2.3.5. Similarity Computation

The similarity computation phase gets an entity pair $\{e_r, e_s\}$, the feature score $fs(e_r, e_s)$, and the belief map as input and produces a pair $(sim(e_r, e_s), imp(e_r, e_s))$ as output where $sim(e_r, e_s)$ is the normalized similarity between the entities e_r and e_s and $imp(e_r, e_s)$ is the corresponding impact value. In general, any of the decision models that we have presented in Section 4.3.6 can be used to compute a single similarity score from the feature score. An incorporation of impact vectors into these decision models has been already discussed in Section 5.2.2.

Existing approaches that incorporate relationship similarity into decision models typically compute the similarity between the attribute value information and compute the similarity between the relationship information separately and finally aggregate both similarity scores by calculating the weighted average. For example Bhattacharya and Getoor [BG07b] use the following aggregation function:

$$sim(e_r, e_s) = (1 - \alpha) \times sim_A(e_r, e_s) + \alpha \times sim_R(e_r, e_s) \tag{5.5}$$

where sim_A is the entity similarity derived from the attribute information, sim_R is the entity similarity derived from the relationship information, and α is the weighting factor.

Since we construct one comparison vector for the attribute description and construct one comparison vector for the relationship description, the similarities sim_A and sim_R can be computed by different decision models (e.g. distance-based, learning-based, rule-based with confidences) or the final entity similarity $sim(e_r, e_s)$ can be directly computed by concatenating both comparison vectors to a single one.

5.2.3.6. Classification

The classification phase gets an entity pair $\{e_r, e_s\}$, the entity similarity $sim(e_r, e_s)$, and the impact value $imp(e_r, e_s)$ as input and then uses two thresholds $\theta_{P/M}$ and $\theta_{U/P}$ in order to classify $\{e_r, e_s\}$ as a MATCH, POSSIBLE MATCH, or UNMATCH by comparing $sim(e_r, e_s)$ with these thresholds. The output of the classification phase is a duplicate decision that we formally define as the triple $dec(e_r, e_s) = (C, sim(e_r, e_s), imp(e_r, e_s))$ where $C \in \{\text{MATCH}, \text{POSSIBLE MATCH}, \text{UNMATCH}\}$ is the computed matching class.

If the similarity score that has been computed for an entity pair has a low impact value, it means that this score is likely not very representative for the considered entity pair. As a consequence, it makes sense to classify it as a POSSIBLE MATCH. In conclusion, we do not only classify pairs with an average similarity as POSSIBLE MATCHES, but also classify pairs whose similarity has a low impact as POSSIBLE MATCHES.

The purpose of the belief map is to store external knowledge on the decisions of individual candidate pairs. Thus, if the belief map stores the information that two entities are certainly a MATCH or are certainly an UNMATCH, the pair's decision can be automatically set to $(\text{MATCH}, 1.0, 1.0)$ or $(\text{UNMATCH}, 0.0, 1.0)$ without invoking the decision model. Thus, if a domain expert manually determines a candidate pair as a MATCH or UNMATCH once, this decision will never change except this expert or another expert revokes this determination afterwards. In cases, where the belief vector does not contain a value of absolute confidence, but instead only contains an assumption the belief map has

to be used in another way. One option is to use the belief values as an additional input to the decision model (similarity computation phase), for example by considering them as two extra input parameters of a learning algorithm, or as two extra values for a distance-based decision models where we automatically use an impact value of 0 if the confidence value is 0 (no external knowledge was available) or 1 otherwise. Another option is to use the confidence values to boost the computed similarity afterwards negatively (confidence of UNMATCH > 0) or positively (confidence of MATCH > 0).

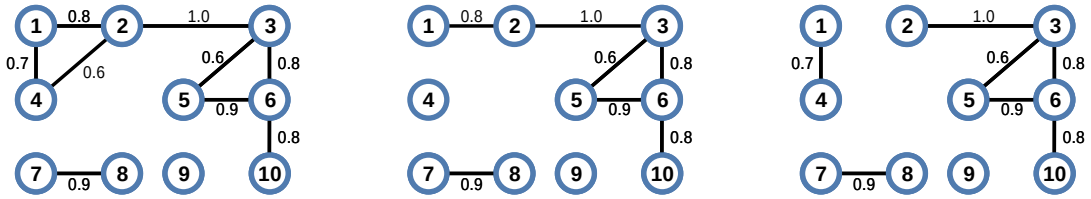
5.2.3.7. Duplicate Clustering

Principally, the duplicate clustering phase is applied as usual and therefore gets all entities and all computed duplicate decisions as input. If required, an impact value for the whole duplicate detection result, i.e. its reliability, is computed as described in Section 5.2.2.4.

If we use the belief map, we have to distinguish between soft decisions, i.e. decisions that have been made by the decision model, and hard decisions, i.e. decisions that have been manually made by domain experts or have been derived from external knowledge. Since the individually taken decisions can be contradictory, the clustering algorithm often need to revoke some of them. Nevertheless, in the presence of hard decisions we want to avoid a revocation of any of them. In general, there are two ways to guarantee that every hard decision is not revoked by the duplicate clustering algorithm. The first way is to use a clustering algorithm that is able to differentiate between hard decisions and soft decisions as for example the extended version of the correlation clustering algorithm that has been proposed by Arasu et al. [ARS09]. The other way is to prepare the set of given decisions so that two entities that have been declared as an hard UNMATCH can never become a MATCH. For example, if the clustering algorithm is working on the Duplicate-Pair Graph, we need to transform the initial graph in a way that each two entities that have been classified as an hard UNMATCH belong to different components without shifting the entities of any hard MATCH into different components.

Example 74 *A simple example of such a preparation is illustrated in Figure 5.6. Let us assume that the two entities e_3 and e_4 are a hard* UNMATCH *and the two entities e_2 and e_3 are a hard* MATCH*. To guarantee that any clustering algorithm will not group e_3 and e_4 together in the same cluster, we transform the duplicate pair graph so that both entities belong to different components without revoking the hard* MATCH *between e_2 and e_3. In this case, this transformations can be done either by revoking the two* MATCHES *$\{e_1, e_4\}$ and $\{e_2, e_4\}$ (see Figure 5.6(b)) or by revoking the two* MATCHES *$\{e_1, e_2\}$ and $\{e_2, e_4\}$ (see Figure 5.6(c)). Since the similarity between e_1 and e_4 is smaller than the similarity between e_1 and e_2, the first transformation can be considered as the cost optimal one.*

As a simple preparation algorithm, the approach of the edge-removal (see Section 4.3.7) can be used. The edge-removal algorithm removes edges with the lowest weights from the graph until the entities of all hard UNMATCHES belong to different components. The solution computed by this approach, however, can be far away from being cost optimal (cost in terms of the accumulative similarity of all revoked MATCHES) in complex graphs. Nevertheless, depending on the number of hard decisions a cost optimal preparation algorithm can be computational expensive.



(a) Initial Duplicate Pair Graph (b) Transformation 1 (c) Transformation 2

Figure 5.6.: Sample transformations of a duplicate pair graph that guarante the hard UNMATCH between e_3 and e_4

If different hard decisions contradict each other, guaranteeing that no hard decisions is revoked is de facto impossible and existing contradictions are either resolve automatically by minimizing the number of revoked hard decisions or are reported to the user.

5.3. Modeling and Matching Database Entities with Multi-Table Memberships

Until now, we restrict our considerations to duplicate detection in databases in which each database entity can only belong to the extension of one entity table. In many real-world scenarios, however, a database entity can belong to different collections and hence can belong to the extension of several entity tables. To the best of our knowledge, none of the existing duplicate detection approaches takes multi-table memberships into account. For that reason, we extend the description-based detection approach that we have presented in the previous section to an handling of such multi-table memberships.

The remainder of this section is structured as follows: First, we give an example to motivate the need for dealing with multi-table memberships. Then we present an approach for incorporating information on multi-table memberships into entity descriptions and finally presents methods to match such descriptions.

5.3.1. Motivating Example

As a motivating example, we consider the information system of an hospital. Figure 5.7 presents the entity-relationship schema of this system. The schema has seven entity types that are connected by several specializations and one ternary relationship type. The basic entity type is *'Person'* having the three attributes *DEI*, *'name'* and *'DoB'* (date of birth). A person can be specialized to a member of the hospital's staff (entity type *'Staff'*) or can be specialized to a patient (entity type *'Patient'*). Staff members can be in turn specialized to members of the hospital's administration staff (entity type *'AdmStaff'*) and can be specialized to members of the hospital's medical staff (entity type *'MedStaff'*). Moreover, a patient can be specialized to a dialysis patient (entity type *'Dialysis Patient'*). Each staff member either belongs to the hospital's administration staff or belongs to the hospital's medical staff, but cannot belong to both at the same time. Consequently, the set of these two specializations is total and disjoint. Every person can be a patient (even a member of the hospital's staff). Thus, the specialization of person to the entity type *'Patient'* and the specialization of person to the entity type *'Staff'* are non-disjoint. Moreover, not every person need to be a patient or a staff member. As a consequence, this set of specializations is not total but only partial. Every dialysis patient is a patient, but not every patient is a dialysis patient.

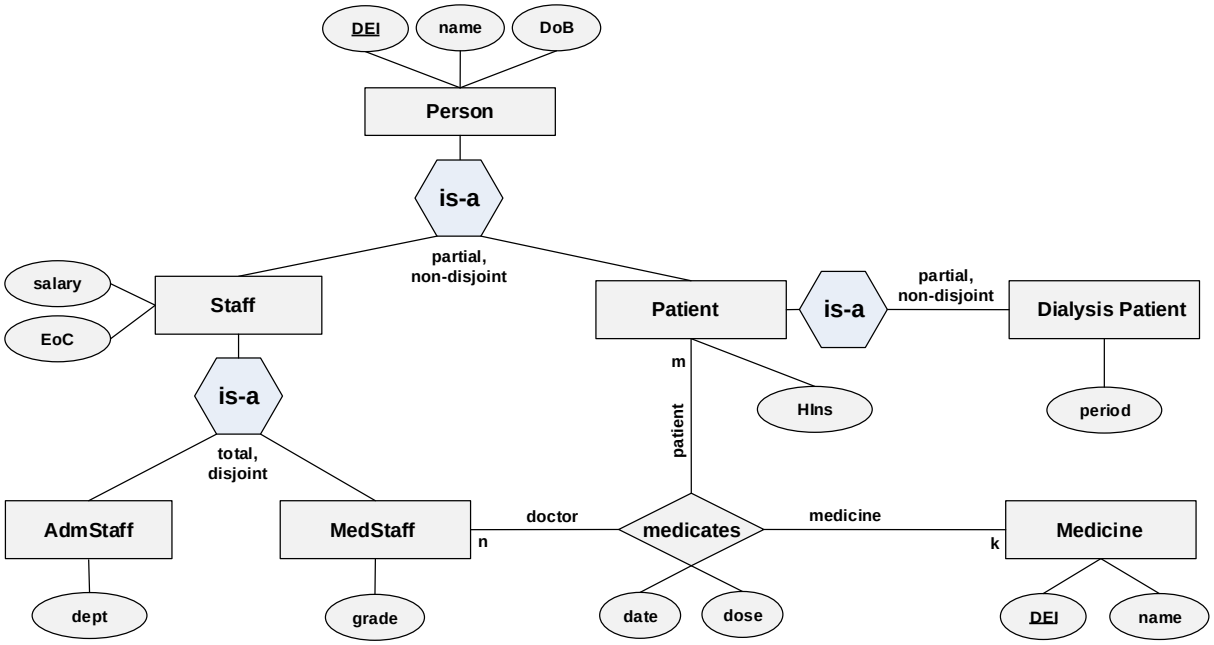


Figure 5.7.: Sample ER-schema with multi-type memberships

Thus, the specialization of patient to dialysis patient is partial. The last entity type *'medicine'* is neither a specialization nor is a generalization of any other entity type of this schema. The entity type *'Staff'* has the attributes *'salaray'* and *'EoC'* (end of contract), the entity type *'AdmStaff'* has the attribute *'dept'* (department), the entity type *'MedStaff'* has the attribute *'grade'*, the entity type *'Patient'* has the attribute *'HIns'* (health insurance), the entity type *'Dialysis Patient'* has the attribute *'period'* (describes the period of days in which a patient gets his dialysis), and the entity type *'Medicine'* has the attributes *DEI* and *'name'*. Recall, every entity can be identified by its *DEI*.

Besides specialization this information schema contains a ternary relationship type. This relationship type has three roles and models the medication that has been applied to a patient by a member of the medical staff. Therefore, the roles are filled by an entity of the type *'MedStaff'* (doctor), an entity of the type *'Patient'*, and an entity of the type *'Medicine'*. Since every patient can be medicated for several times, a member of the medical staff can medicate a patient for several times, and a medicine can be used for medication for several times, the relationship type is many-to-many.

Figure 5.8 shows a relational database its schema results from transforming the entity-relationship schema given in Figure 5.7 to the relational data model and shows a sample instance for this database (recall that we know table *'Medicates'* already from Example 68). The database schema consists of eight tables where the primary key of each table is the combination of the underlined attributes. Foreign keys are presented by additional notes of the form $attr \rightarrow table.attr$ below the corresponding tables. Note, the tables *'Person'*, *'Staff'*, *'AdmStaff'*, *'MedStaff'*, *'Patient'*, and *'Dialysis Patient'* are entity tables, the table *'Medicates'* is a relationship table.

The extension of the database contains seven entities; five persons and two medicines. As an example of multi-table membership, person p_6 belongs to the extension of four different entity tables.

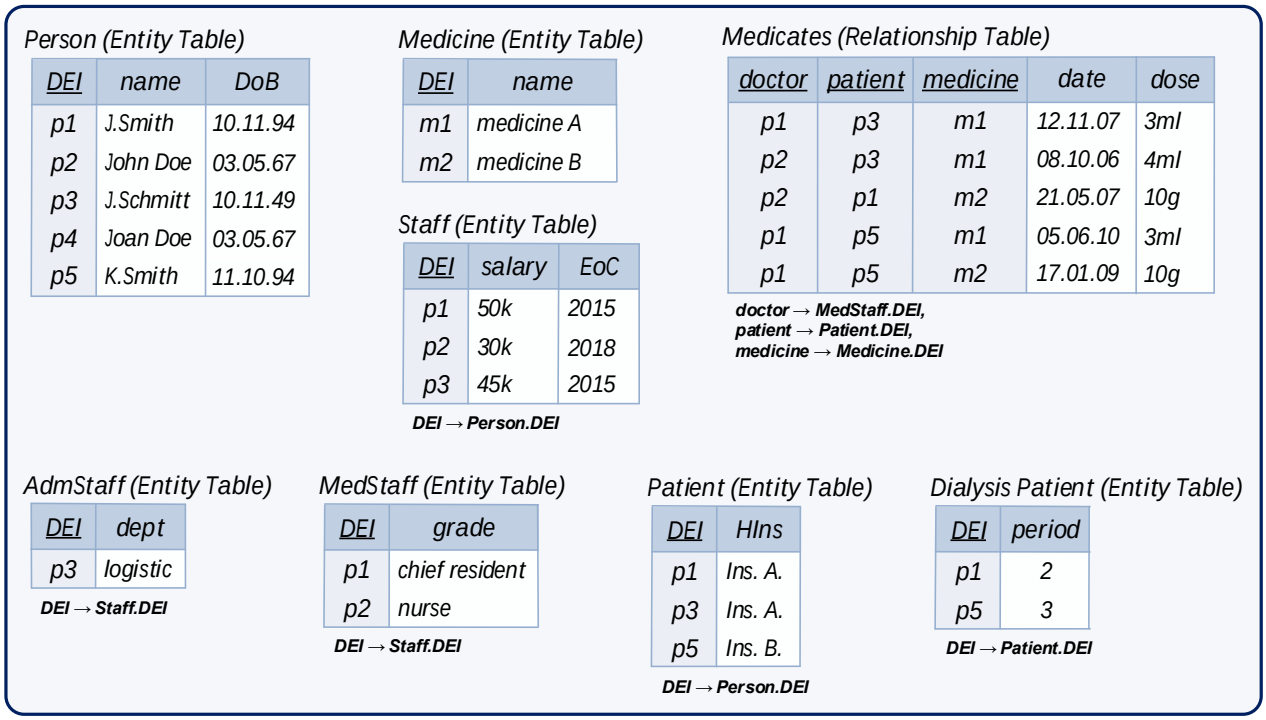


Figure 5.8.: Sample RM-schema with multi-table memberships

Regarding this database it is not clear in which way we can describe the individual persons by a single description. The most intuitive approach is to adopt the Single-Table Approach for modeling inheritance within the relational data model. By doing so, we consider all entity tables that are related by specializations as a single table and hence construct an attribute description type that contains each attribute of one of these tables. This approach enables a simple modeling of membership information into the descriptions, but it cannot avoid that same membership information is redundantly stored in these descriptions. If this effect is not considered in the matching process, the redundantly described information can have a strong influence on the detection result. To regulate this influence, we present an approach that makes use of the previously introduced impact values.

5.3.2. Membership Dependencies between Entity Tables

Before presenting our approach for describing and matching entities with multi-table memberships in detail, we have to introduce two kinds of dependencies that can exist between two entity tables.

Let $\mathcal{T}$ be a set of entity tables.

- **Membership Exclusion:** Two entity tables $T_i, T_j \in \mathcal{T}$ are called to be *membership exclusive* (noted as $\rightleftharpoons^M$), if an entity can never belong to the extensions of both tables at the same time, i.e.:

$$T_i \rightleftharpoons^M T_j \Leftrightarrow \forall \tau \in \mathit{Time}\colon \mathit{Ext}(T_i, \tau) \cap \mathit{Ext}(T_j, \tau) = \emptyset$$

  An example of two membership exclusive entity tables are *'Person'* and *'Medicine'*, because a person can never become a medicine and vice versa. A less restricted example are the tables

'AdmStaff' and *'MedStaff'*, because no person can be a member of the administration staff and the medical staff at the same time.

- **Membership Inclusion:** An entity table $T_i \in \mathcal{T}$ is called to be *membership included* by an entity table $T_j \in \mathcal{T}$ (noted as $\rightharpoonup^M$), if each entity that belongs to the extension of T_j always belongs to the extension of T_i, too, i.e.:

$$T_i \rightharpoonup^M T_j \Leftrightarrow \forall \tau \in \mathit{Time}\colon \mathit{Ext}(T_j, \tau) \subseteq \mathit{Ext}(T_i, \tau)$$

An example of two entity tables where the first is membership included by the second are *'Person'* and *'Patient'*, because every patient is a person.

Membership inclusion between entity tables conceptually corresponds to a specialization (*is-a* relationship) of entity types in the entity-relationship model. Thus, let E_i be the entity type that is represented by entity table T_i, an entity table $T_1 \in \mathcal{T}$ is membership included by an entity table $T_2 \in \mathcal{T}$, if E_2*is-a*E_1. Nonetheless, because of bad schema design or because of considering tables from independent integration sources, membership inclusion is not necessarily restricted to specializations.

However, if all membership inclusions are modeled by specializations, we can also derive the membership exclusions from it. If neither E_i is a specialization nor a generalization of E_j and if for each entity type E_t that is generalization of E_i and E_j the two specializations E_i*is-a*E_t and E_j*is-a*E_t are disjoint (note, this condition is also satisfied if there is no entity type that is a generalization of both), the two entity tables $T_i, T_j \in \mathcal{T}$ are membership exclusive.

Example 75 *In our motivating example, the entity table* 'Person' *is membership included by all other entity tables except* 'medicine'. *The entity table* 'Staff' *is membership included by the entity tables* 'AdmStaff' *and* 'MedStaff'*, and the entity table* 'Patient' *is membership included by the entity table* 'Dialysis Patient'. *The entity table* 'Medicine' *is membership exclusive to all other entity tables and the entity table* 'AdmStaff' *is membership exclusive to the entity table* 'MedStaff'.

Of course, not all cases of membership exclusions need to be modeled in the entity descriptions. For instance, the membership exclusions between the table *'Medicine'* and all other entity tables are automatically implicated by defining different and non-compareable description types. In contrast, the membership exclusion between *'AdmStaff'* and *'MedStaff'* cannot be modeled by non-compareable description types, because although their extensions cannot overlap they both can overlap with the extension of a third entity table and all entities of the same extension should be compareable. For instance, each person should be compareable with another person whether or not one of them is a medical staff member and the other is an administration staff member. For that reason, we introduce the concept of *membership correlation sets*.

Definition 31 ***(Membership Correlation Set:)*** *A set of entity tables $\mathcal{T}$ is called a membership correlation set if it cannot be partitioned in any two disjoint and non-empty subsets so that all two elements of different subsets are pairwise membership exclusive to each other, i.e. $\mathcal{T}$ is called a membership correlation set if:*

$$\nexists \mathcal{T}_i, \mathcal{T}_j \subset \mathcal{T}, \mathcal{T}_i \neq \emptyset, \mathcal{T}_j \neq \emptyset, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset\colon \forall T_i \in \mathcal{T}_i\colon \forall T_j \in \mathcal{T}_j\colon T_i \rightleftharpoons^M T_j$$

For instance, the entity tables of our motivating example can be partitioned into two membership correlation sets. Whereas the first set only contains the table *'Medicine'*, the other set contains all other entity tables, because each of them is a specialization of the table *'Person'* and each of them is membership exclusive to table *'Medicine'*. Obviously, we define an own entity description type per membership correlation set.

5.3.3. Describing Entities with Multi-Table Memberships

In general, there is no need for a specific description approach that handles multi-table memberships, if we can enforce single-table membership by transforming the database schema in another semantically equivalent schema that only contains entity tables with disjoint extensions. One way to enforcing single-table membership is to flatten the hierarchy of tables within a membership correlation set by merging all these tables into a single one and hence to use the Single-Table Approach for modeling inheritance. The resultant table contains every attribute that belongs to one of the merged tables and entities of that table can fill in any relationship role that can be filled in by an entity of the merged tables (recall that we consider attribute names to be clearly distinguishable so that no name conflict between the attributes of different tables can appear). The extension of the resultant table is the union of all the merged tables' extensions. Recall in order to model the membership to the individual entity types (or entity tables respectively), we need to add a membership attribute per entity table.

Of course, we do not actually need to merge tables, but instead generate the entity description types and the concrete entity descriptions of that type like the tables were actually merged. For each membership correlation set we generate a separate entity description type as follows: Let $\mathcal{T}$ be a membership correlation set, we generate an attribute description by adding all attributes of each table of $\mathcal{T}$ and by adding a membership attribute per table of $\mathcal{T}$, i.e.

$$\mathfrak{T}^{Attr}(\mathcal{T}) = \bigcup_{T \in \mathcal{T}} \left(\mathfrak{T}^{Attr}(T) \cup \{A_T\} \right) \tag{5.6}$$

where A_T is the membership attribute that is created for entity table T. If one entity table is membership included by all other tables of this correlation set, this table does not need to be represented by a membership attribute, because it always has the value 'true' for all entities whose descriptions are conform to this type. For instance, in our motivating example, the table *'Person'* does not need to be represented by a membership attribute.

Moreover, the relationship description type of $\mathcal{T}$ contains each label-domain pair of one of its tables. Here we assume that each label is assigned by the same domain in the description type of each table. In that case, the relationship description type of $\mathcal{T}$ results in:

$$\mathfrak{T}^{Rel}(\mathcal{T}) = \bigcup_{T \in \mathcal{T}} \mathfrak{T}^{Rel}(T) \tag{5.7}$$

The concrete entity description of an entity is then constructed as usual, i.e. for each attribute an attribute-value pair is extracted from the database. Each attribute the entity does not have a value for, because it does not belong to the extension of the corresponding table, is paired with the null value. A membership attribute is set to 'true' if the considered entity belongs to the corresponding table and is set to 'false' otherwise. Thus, let e be a database entity, let $\mathcal{T}_M \subseteq \mathcal{T}$ be the set of entity tables in which e is

DEI	name	DoB	isStaff	salary	EoC	isAS	dept	isMS	grade	isPat	HIns	isDPat	period
p1	J.Smith	10.11.94	true	50k	2015	false	⊥	true	chief resident	true	Ins. A.	true	2
p2	John Doe	03.05.67	true	30k	2018	false	⊥	true	nurse	false	⊥	false	⊥
p3	J.Schmitt	10.11.49	true	45k	2015	true	logistic	false	⊥	true	Ins. A.	false	⊥
p4	Joan Doe	03.05.67	false	⊥	⊥	false	⊥	false	⊥	false	⊥	false	⊥
p5	K.Smith	11.10.94	false	⊥	⊥	false	⊥	false	⊥	true	Ins. B.	true	3

(a) Attribute description type and concrete attribute descriptions of the sample entities

DEI	doctor	patient
p1	{p3,p5,m1,m2}	{p2,m2}
p2	{p2,p3,m1,m2}	Ø
p3	Ø	{p1,p2,m1}
p4	Ø	Ø
p5	Ø	{p1,m1,m2}

(b) Relationship description type and concrete relationship descriptions of the sample entities

Figure 5.9.: The attribute descriptions and the relationship descriptions of the sample entities by using the proposed description approach

represented by a tuple, and let $\mathcal{T}_U = \mathcal{T} - \mathcal{T}_M$ be the set of entity tables in which e is not represented by a tuple, the attribute description of e in $\mathcal{T}$ is defined as:

$$f_{AD}(e, \mathcal{T}) = f_{AD}(e, \mathcal{T}_M) \cup f_{AD}(e, \mathcal{T}_U) \tag{5.8}$$

where $f_{AD}(e, \mathcal{T}_M)$ and $f_{AD}(e, \mathcal{T}_U)$ are defined as:

$$f_{AD}(e, \mathcal{T}_M) = \bigcup_{T \in \mathcal{T}_M} \left(f_{AD}(e, \{T\}) \cup (A_T, \text{'true'})\right) \tag{5.9}$$

$$f_{AD}(e, \mathcal{T}_U) = \bigcup_{T \in \mathcal{T}_U} \left(\bigcup_{A \in \mathfrak{T}^{Attr}(T)} (A, \bot) \cup (A_T, \text{'false'})\right) \tag{5.10}$$

The related sets of the relationship descriptions are first extracted for each table separately and then combined. To guarantee the uniqueness of each label, the set per label is constructed by computing the union of all the sets this label has in any of the merged tables, i.e.

$$f_{RD}(e, \mathcal{T}) = \bigcup_{(L,dom) \in \mathfrak{T}^{Rel}(\mathcal{T})} \{(L, \bigcup_{T \in \mathcal{T}} \{S \mid (L, S) \in f_{RD}(e, \{T\})\})\} \tag{5.11}$$

Example 76 *For illustration we reconsider the motivating example of this section. As mentioned before, the tables* 'Person', 'Staff', 'AdmStaff', 'MedStaff', 'Patient', *and* 'Dialysis Patient' *form a membership correlation set. The attribute description type of this correlation set corresponds to the schema of a table that result from merging all these tables in the previously proposed way. The resultant table is presented in Figure 5.9(a). In this example, the membership attributes are named as* 'isStaff', 'isAS' *(shortcut for* 'isAdmStaff'*),* 'isMS' *(shortcut for* 'isMedStaff'*),* 'isPat' *(shortcut for* 'isPatient'*), and* 'isDPat' *(shortcut for* 'isDialysisPatient'*). We omit the membership attribute of table* 'Person'*, because each entity that belongs to any table of this correlation set belongs to the* 'Person' *as well. Since, we only*

consider single-valued attributes, the attribute descriptions of the individual entities correspond to the tuples of the newly constructed table.

The considered database schema contains only a single relationship and only the two roles 'doctor' *and* 'patient' *reference to an entity table of the considered membership correlation set. In this example, we assume that relationship information is modeled by two labels. The first label corresponds to the role* 'doctor' *and refers to all entities that are correlated with the considered person acting as a doctor. The second label corresponds to the role* 'patient' *and refers to all entities that are correlated with the considered person acting as a patient. Consequently, the relationship description type of this membership correlation set consists of two label-domain pairs where the domain is a set of entities for both pairs. The relationship instances of the label* 'doctor' *originates from the entity table* 'MedStaff' *and the relationship instances of the label* 'patient' *originates from the entity table* 'Patient'. *The concrete relationship descriptions of all entities that belong to the considered correlation set are presented in Figure 5.9(b).*

5.3.4. Matching of Entity Descriptions with Membership Attributes

A big advantage of the presented approach for describing entities with multi-table memberships is that the resultant descriptions are structurally equivalent to the conventional case of single-table membership. Therefore, two entity descriptions that contain information on multi-table memberships can be matched as usual, i.e. by matching their attribute descriptions and their relationship descriptions as presented in Section 5.2. Nevertheless, as we will illustrate below, these descriptions can contain same information for several times which in turn can distort the matching result significantly.

Example 77 *For illustration we consider the entity* p1 *from our motivating example. Due to it is a dialysis patient, it is automatically a patient as well and hence the value 'true' in the membership attribute* 'isPat' *is implicated by the value 'true' in the membership attribute* 'isDPat'. *Another dependency can be observed for* p1*'s value in the membership attributes* 'isAS' *and* 'isMS'. *Because a person cannot belong to the administration staff and the medical staff at the same time, the value 'true' in the attribute* 'isMS' *implicates the value 'false' in the attribute* 'isAS'. *Finally, the value 'false' in the membership attributes* 'isAS' *implicates a null value in the attribute* 'dept'.

As we have shown by this example, the values of some attributes can be implicated by the values of other attributes. We can generalize such implications by the following observations:

- An attribute is implicated to contain the null value, if its corresponding membership attribute is 'false'. Similar holds for relationship types, because the related entity set of a specific label is empty if the considered entity does not belong to one of the corresponding entity tables.
- The value of the membership attribute A_{T_j} is implicated to 'false', if the value of a membership attribute A_{T_i} with $T_i \rightharpoonup^M T_j$ (T_i is membership included by T_j) is 'false'. In contrast, the value of a membership attribute A_{T_i} is implicated to 'true', if the value of a membership attribute A_{T_j} with $T_i \rightharpoonup^M T_j$ is 'true'.
- The value of a membership attribute A_{T_i} is implicated to 'false', if the value of a membership attribute A_{T_j} with $T_i \rightleftharpoons^M T_j$ is 'true'.

By matching all attribute value pairs independently, these implications can lead to the case that same information is considered in several similarity scores of the produced feature score and hence will have a stronger influence on the later made duplicate decision than other information. Since these implications are value dependent, i.e. they only appear for specific values in the considered attributes, these dependencies cannot be handled by the attributes' weights.

Example 78 *For illustration we consider some of the entities from the motivating example. First, we compare the two entities p_1 and p_4. The first is a staff member and the second not. Thus, their values do not only disagree in the membership attribute* 'isStaff'*, but also disagree in the attributes* 'salary' *and* 'EoC'*. The disagreement in the latter is caused by the disagreement in the first. Consequently, the information that one is a staff member and the other not is considered for three times. In general, two different values in a membership attribute will imply a low similarity score in a set of further attributes.*

Another situation of redundant matching information appears in comparing entity p1 *with entity* p5*. Both entities are dialysis patients. The agreement in the membership attribute* 'isDPat'*, however, implicates an agreement in the membership attribute* 'isPat'*, because for both entities, the value 'true' in the attribute* 'isDPat' *implicates the value 'true' in the attribute* 'isPat'*. Again, same information is considered in multiple similarity scores.*

A third situation of redundant matching information can be observed by comparing entity p1 *with entity* p2*. Both entities are medical staff members and hence both have the value 'true' in the membership attribute* 'isMS'*. The memberships to the medical staff and the administration staff are excluding and the value 'true' in the attribute* 'isMS' *implicates the value 'false' in the attribute* 'isAS'*. Thus, the agreement in the attribute* 'isMS' *implicates an agreement in the attribute* 'isAS'*.*

Finally, we consider the entities p1 *and* p3*. Whereas the first is a medical staff member, the second is an administration staff member. Thus, the disagreement in the attribute* 'isMS' *implicates a disagreement in the attribute* 'isAS' *and vice versa.*

These negative effects of matching redundantly modeled information, however, can be avoided if we set the impact values of the produced similarity scores by a set of rules. For simplification, we restrict to the case where we do not match values across attributes and therefore the considered similarity scores and the considered impact values are clearly determined by the considered attribute.

- **Rule 1:** If the matched entities have both the value 'false' in a membership attribute A_T, the impact value of each attribute from table T is set to zero, because the agreements in the null values of these attributes are already considered by the agreement in attribute A_T.

  Rule 1: $e_r[A_T] = e_s[A_T] = \text{'false'} \Rightarrow \forall A \in \mathfrak{T}^{Attr}(T)\colon imp(e_r[A], e_s[A]) = 0$

- **Rule 2:** If the matched entities have different values in a membership attribute A_T, i.e. one has false' and the other has 'true', the impact value of each attribute from table T is set to zero, because the disagreements in values of these attributes are already considered by the disagreement in attribute A_T.

  Rule 2: $e_r[A_T] = \text{'false'} \wedge e_s[A_T] = \text{'true'} \Rightarrow \forall A \in \mathfrak{T}^{Attr}(T)\colon imp(e_r[A], e_s[A]) = 0$

- **Rule 3:** If the matched entities have both the value 'false' in a membership attribute A_{T_i}, the impact value of each membership attribute A_{T_j} is set to zero if table T_i is membership included by table T_j, i.e. $T_i \rightharpoonup^M T_j$, because the agreements in the value 'false' of these attributes are already considered by the agreement in attribute A_{T_i}.

  Rule 3: $e_r[A_{T_i}] = e_s[A_{T_i}] =$ 'false' $\Rightarrow \forall T_j \in \mathcal{T}, T_i \rightharpoonup^M T_j \colon imp(e_r[A_{T_j}], e_s[A_{T_j}]) = 0$

- **Rule 4:** If the matched entities have both the value 'true' in a membership attribute A_{T_i}, the impact value of each membership attribute A_{T_j} is set to zero if table T_j is membership included by table T_i, i.e. $T_j \rightharpoonup^M T_i$, because the agreements in the value 'true' of these attributes is already considered by the agreement in attribute A_{T_i}.

  Rule 4: $e_r[A_{T_i}] = e_s[A_{T_i}] =$ 'true' $\Rightarrow \forall T_j \in \mathcal{T}, T_j \rightharpoonup^M T_i \colon imp(e_r[A_{T_j}], e_s[A_{T_j}]) = 0$

- **Rule 5:** Let $\mathcal{A}$ be a set of membership attributes which represent a set of tables that are mutual membership exclusive. If the matched entities have both the value 'true' in one of these attributes $A_{T_i} \in \mathcal{A}$, the impact value of each other attribute in $\mathcal{A}$ is set to zero, because its agreement is already considered by the agreement in attribute A_{T_i}.

  Rule 5: $\exists A_{T_i} \in \mathcal{A} \colon e_r[A_{T_i}] = e_s[A_{T_i}] =$ 'true'
  $\Rightarrow \forall A_{T_j} \in \mathcal{A} - \{A_{T_i}\} \colon imp(e_r[A_{T_j}], e_s[A_{T_j}]) = 0$

- **Rule 6:** Let $\mathcal{A}$ be a set of membership attributes which represent a set of tables that are mutual membership exclusive. If the matched entities have the value 'true' in different attributes of $\mathcal{A}$, i.e. one in the attribute A_{T_i} and the other in the attribute A_{T_j}, they automatically disagree in these two attributes. For that reason, either A_{T_i} or A_{T_j} is chosen and the impact value of the non-chosen attribute is set to zero.

  Rule 6: $\exists A_{T_i}, A_{T_j} \in \mathcal{A} \colon e_r[A_{T_i}] = e_s[A_{T_j}] =$ 'true' $\wedge\, e_r[A_{T_j}] = e_s[A_{T_i}] =$ 'false'
  $\Rightarrow imp(e_r[A_{T_i}], e_s[A_{T_i}]) = 0 \vee imp(e_r[A_{T_j}], e_s[A_{T_j}]) = 0$

  Note that if the entities disagree in the two membership attributes A_{T_i} and A_{T_j}, they automatically agree in all membership attributes $\mathcal{A} - \{A_{T_i}, A_{T_j}\}$. Thus, the information $e_r[A_{T_i}] \neq e_s[A_{T_i}] \wedge e_r[A_{T_j}] \neq e_s[A_{T_j}]$, the information $e_r[A_{T_i}] \neq e_s[A_{T_i}] \wedge \bigwedge_{A_T \in \mathcal{A} - \{A_{T_i}, A_{T_j}\}} e_r[A_T] = e_s[A_T]$ and the information $e_r[A_{T_j}] \neq e_s[A_{T_j}] \wedge \bigwedge_{A_T \in \mathcal{A} - \{A_{T_i}, A_{T_j}\}} e_r[A_T] = e_s[A_T]$ are redundant. Therefore another possible reaction would be to let the impact values of A_{T_i} and A_{T_j} unchanged and to set the impact values of the other membership attributes $\mathcal{A} - \{A_{T_i}, A_{T_j}\}$ to zero. Nevertheless, since the non-membership to some entity sets can be a better indication for a duplicate than the non-memberships to other entity sets, the decision model benefits more from setting the impact value of either A_{T_i} or A_{T_j} to zero than by setting the impact values of all attributes in $\mathcal{A} - \{A_{T_i}, A_{T_j}\}$ to zero.

Example 79 *For illustration, we reconsider the matching situations from the previous example and again assume that we only match values from same attributes and not across attributes. Moreover, for representation purposes, we assume that the default impact computation method always returns the*

	...	isStaff	salary	EoC	...
p1	...	true	30k	2015	...
p4	...	false	⊥	⊥	...
$\vec{c}$	...	0.0	0.0	0.0	...
$\vec{i}$	...	1.0	0.0	0.0	...

(a) Sample scenario of the first impact rule

	...	isPat	HIns	isDPat	...
p1	...	true	Ins. A.	true	...
p5	...	true	Ins. B.	true	...
$\vec{c}$	...	1.0	0.5	1.0	...
$\vec{i}$	...	0.0	1.0	1.0	...

(b) Sample scenario of the third impact rule

	...	isAS	dept	isMS	...
p1	...	false	⊥	true	...
p2	...	false	⊥	true	...
$\vec{c}$	...	1.0	1.0	1.0	...
$\vec{i}$	...	0.0	0.0	1.0	...

(c) Sample scenario of the fourth impact rule

	...	isAS	dept	isMS	...
p1	...	false	⊥	true	...
p3	...	true	logistic	false	...
$\vec{c}$	...	0.0	0.0	0.0	...
$\vec{i}$	...	0.0	0.0	1.0	...

(d) Sample scenario of the fifth impact rule

Figure 5.10.: Sample scenarios for using the impact rules to avoid a multiple consideration of same information in the feature matching phase

value 1.0 *and therefore impact values are only set to a value lower than* 1.0 *by any of the rules described above.*

In the first scenario, due to Rule 2, the impact values of the attributes 'salary' *and* 'EoC' *are set to zero. In contrast, the impact value of the attribute* 'isStaff' *is computed as usual. This scenario is presented in Figure 5.10(a).*

In the second scenario depicted in Figure 5.10(b), we use Rule 4 to set the impact value of the attribute 'isPat' *to zero. The impact value of the attribute* 'isDPat' *is computed as usual. Note, the impact value of the attribute* 'HIns' *is not affected by this rule.*

To resolve the third scenario, we use Rule 5 to set the impact values of the attribute 'isAS' *to zero and compute the impact value of the attribute* 'isMS' *as usual. Note, the impact value of the attribute* 'dept' *is set to zero due to Rule 1. This scenario is presented in Figure 5.10(c).*

Finally, in the comparison of the entities p_1 *and* p_3 *we use Rule 6 to set the impact value of the attribute* 'isAS' *to zero. Note, in this case we have the choice to either set the impact value of the attribute* 'isAS' *to zero or to set the impact value of the attribute* 'isMS' *to zero. In this example we decide to do the first without any specific reason.*

Similar rules need to be defined for the relationship descriptions. For instance, if the compared entities have both the value 'false' in the membership attribute *'isMS'*, the sets of related entities for label *'doctor'* are both implicated to be empty. For that reason, the impact value of this relationship information need to be set to zero because the agreement in this information is already considered by the agreement in the attribute *'isMS'*.

Chapter 6

Uncertain Value Theory

In this chapter, we present a formalism for modeling and processing uncertainty in data and data operations. We will use this formalism in the rest of this thesis as a theoretical foundation for modeling and processing data uncertainty as well as process uncertainty. The formalism is based on the possible worlds semantics and is defined in an abstract way so that it can be reused in many application scenarios. We accomplish this high level of abstraction by defining certainty and uncertainty on domain level where a domain can theoretically be any set of things. First, we informally present the principle idea of the possible worlds semantics in Section 6.1. Then we formally introduce the fundamental concepts of crisp values and uncertain values in Section 6.2. We then discuss several characteristics and possible transformations of uncertain values in Section 6.3, Section 6.4, Section 6.5, and Section 6.6. In Section 6.7, we present concepts to adapt functions that are originally defined for crisp values to the handling of uncertain values. Then, we extend our considerations to the domain of functions and hence introduce the concept of uncertain mappings in Section 6.8. Section 6.9 summarizes the fundamental contributions of the formalism. Finally, we present some measures that can be used to rate the (un)certainty of uncertain values in Section 6.10.

6.1. Possible Worlds Semantics

A variety of formalisms that incorporate uncertainty has been proposed in different research areas. The most of these formalisms are logic-based (for an overview see [Cus07]). Examples of formalisms which incorporate logic and probability are Probabilistic Logic Programming [NS92], Probabilistic Horn abduction [Poo93], the Independent Choice Logic [Poo97], Probabilistic knowledge bases [NH97], Bayesian Logic Programs [KR01], Relational Bayesian Networks [Jae13], or Stochastic Logic Programs [Mug00] to name just a few.

Nonetheless, whereas these formalism are logic-based, many formal foundations of probabilistic databases are not. Therefore, in order to stay consistent with the terminology and formalism used in relational database research in general and probabilistic relational database research in particular, we decided to abstain from using a logic-based approach for modeling uncertainty, but generalized the possible worlds semantics from database instances to arbitrary domains instead. Note that the presented formalism is based on the ideas of the possible worlds semantics as we know them from probabilistic

database research [DS07c, AKG87], and hence it may differ from the interpretations of the possible worlds semantics that are used in other research directions such as logic [Car50, Hin62, Kri63, Hal90] or philosophy [Lew86].

Recall from Chapter 3, the core principle of the possible worlds semantics informally means that if we do not know the exact state of the real world, we have to model it by a set of possible states which are mutually exclusive, i.e. only one of these states can be the 'true' one. Each possible state of the real world is usually called a *possible world*. Moreover, each possible world can be associated with a probability which represent the likelihood that this world corresponds to the real world.

Besides modeling the uncertainty of the fixed state of the real world, the possible world semantics defines how to consider changes in this state as well: If we perform an action in the real world and we want to know the possible results of that action, we have to perform this action in all possible worlds separately. An action result belongs to the set of possible action results, if it results from performing this action in at least one possible world. The probability of an action result is equivalent to the accumulative probability of all possible worlds that lead to that action result.

These principles are illustrated in Figure 6.1.

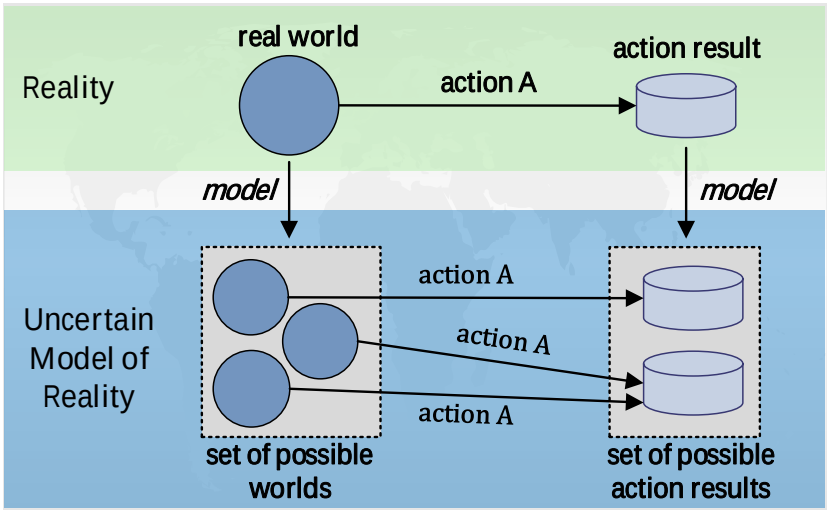


Figure 6.1.: Principle of the possible worlds semantics

6.2. Values for Modeling Uncertainty

As mentioned above, certain information on real-world facts are modeled by the concept of crisp values (also denoted as certain values in this thesis). Since a crisp value represents absolute certainty about the modeled fact, the 'true' value of that fact is exactly known (or at least is assumed to be known[1]).

Definition 32 (Crisp Value): *Let dom be an arbitrary domain. A **crisp value** of dom is a single value $x \in dom$.*

[1] Note, since data can be certain but incorrect, the single value that is assumed to be the 'true' value must not necessarily be the actual 'true' value.

It is obvious that a crisp value of a domain corresponds to an element of that domain. As a consequence, in the rest of the paper, we will use the notation $x \in dom$ to describe a crisp value of domain dom. A crisp value corresponds to a value of absolute certainty. For that reason, we will use the terms *crisp* and *certain* interchangeably in this thesis.

In order to enable a modeling of uncertainty about a fact's 'true' value, we generalize the concepts of incomplete databases and probabilistic databases that we defined in Chapter 3 and therefore introduce the concepts of incomplete values and probabilistic values. An incomplete value is considered as a non-empty and finite set of alternative values so that one of these alternatives is the 'true' value. On attribute value level that concept is also known as OR-Set value [MG88, Imi89] or partial value [Gra79, DeM89, TCY93]. A probabilistic value is an incomplete value where each alternative is associated with a probability. On attribute value level that concept is also known as probabilistic partial value [TCY93]. In contrast to former research we use these definition in a more generalize sense, because we consider a value to be of any domain (even domains of functions). For example, we can use this concept to describe uncertain entity descriptions, uncertain similarity scores, uncertain duplicate decisions, or uncertain duplicate clusterings, but it can be also used to define an indeterministic duplicate detection process that incorporates uncertainty in the detection result.

Definition 33 (Incomplete Value): *Let dom be an arbitrary domain. An **incomplete value** y of dom, denoted as $y \in^{[\iota]} dom$, is a non-empty and finite subset of dom, i.e. $y \subseteq dom \wedge y \neq \emptyset$, where each $x \in y$ is a crisp value of dom and is called an alternative of y. All alternatives are mutually exclusive, i.e. each of them represent a different possible state of the real-world fact that is modeled by y.*

Note, the concept of incomplete values is a generalization of the concept of crisp values, because an incomplete value $y \in^{[\iota]} dom$ with exact one alternative, i.e. $|y| = 1$, corresponds to a crisp value of dom.

The set of all possible incomplete values of domain dom is called the incomplete version of dom and is denoted as $dom^{[\iota]} = \{y \mid y \in^{[\iota]} dom\}$. Theoretically, it holds that $dom^{[\iota]} = 2^{dom} - \emptyset$, but because of the inherit semantics of the mutual exclusion the incomplete version of dom is its power set, but not every domain that is defined as the power set of dom is its incomplete version. Consequently, an incomplete value of the domain dom is a crisp value of the domain $dom^{[\iota]}$, i.e. $y \in^{[\iota]} dom \Leftrightarrow y \in dom^{[\iota]}$.

Definition 34 (Probabilistic Value): *Let dom be an arbitrary domain. A **probabilistic value** z of dom, denoted as $z \in^{[\rho]} dom$, is a pair $z = (y, Pr)$ where Pr is a discrete probability distribution on dom, i.e. $Pr\colon dom \rightarrow [0, 1]$ and $\sum_{x \in dom} Pr(x) = 1$, and $y \in^{[\iota]} dom$ is an incomplete value of dom that contains all alternative values of z, i.e. $y = \{x \mid x \in dom, Pr(x) > 0\}$.*

For ease of presentation, a probabilistic value $z = (y, Pr)$ can be also represented as $z = \bigcup_{x \in y} \{(x\colon Pr(x))\}$.

Note, in contrast to the definition of probabilistic databases, we define the probability distribution Pr on the considered domain dom instead of the set of alternative values y. That is because we sometimes need to use the probability of a domain element that does not belong to the set of alternative values. Of course, such a probability is per definition zero, but using a probability distribution that maps each of the domain elements to a probability simplifies some formalizations notably.

According to Definition 34, the probability of all alternative values have to sum up to one. Nevertheless, it can be useful to relax this requirement in situations where a probabilistic value is temporary split into several smaller probabilistic values for processing reasons, i.e. the set of alternative values is partitioned into smaller sets and each of these sets is processed separately. For that reason, we sometimes relax the condition that the accumulative probability of all alternative values must be one if we consider interim detection results.

Note, as the concept of incomplete values, the concept of probabilistic values is a generalization of the concept of crisp values, because a probabilistic value $z \in^{[\rho]} dom$ with $z = (y, Pr)$ where $|y| = 1$ corresponds to a crisp value of dom.

The set of all the probabilistic values that are possible for the domain dom is called the probabilistic version of dom and is denoted as $dom^{[\rho]} = \{z \mid z \in^{[\rho]} dom\}$. Consequently, a probabilistic value of the domain dom is a crisp value of the domain $dom^{[\rho]}$, i.e. $z \in^{[\rho]} dom \Leftrightarrow z \in dom^{[\rho]}$.

As mentioned in the introduction of this chapter, we summarize the concepts of incomplete value and probabilistic value by the concept of uncertain values.

Definition 35 (Uncertain Value): *Let dom be an arbitrary domain. An **uncertain value** of dom, denoted as $u \in^{[\iota/\rho]} dom$, is either an incomplete value or a probabilistic value of this domain, i.e. $u \in^{[\iota/\rho]} dom \Rightarrow u \in^{[\iota]} dom \vee u \in^{[\rho]} dom$.*

The uncertain version of a domain dom is denoted as $dom^{[\iota/\rho]}$ and is the set of all incomplete values of dom and all probabilistic values of dom, i.e. $dom^{[\iota/\rho]} = dom^{[\iota]} \cup dom^{[\rho]}$.

To make a general discussion on uncertain values possible, we define the two mappings $alt(\cdot)$ and $pd(\cdot)$. The mapping $alt(\cdot)$ maps an uncertain value $u \in^{[\iota/\rho]} dom$ to its sets of alternatives and is defined as follows:

$$alt(u) = \begin{cases} u, & \text{iff } u \in^{[\iota]} dom, \\ y, & \text{iff } u \in^{[\rho]} dom, u = (y, Pr). \end{cases}$$

The mapping $pd(\cdot)$ maps an uncertain value $u \in^{[\iota/\rho]} dom$ to its probability distribution (if applicable) and is defined as follows:

$$pd(u) = \begin{cases} \text{undefined}, & \text{iff } u \in^{[\iota]} dom, \\ Pr, & \text{iff } u \in^{[\rho]} dom, u = (y, Pr). \end{cases}$$

For ease of presentation, we introduce some naming conventions for this chapter. In the remainder of this chapter, we will use the letter 'x' for naming crisp values, the letter 'y' for naming incomplete values, the letter 'z' for naming probabilistic values, and the letter 'u' for naming uncertain values. Note, an uncertain value of one domain can be a crisp value of another domain (simply recall a crisp value of domain $dom^{[\rho]}$ is a probabilistic value of domain dom). Thus this notation is only valid in relation to the considered context which we will always concretize by a specific domain. Moreover, the incomplete value that describes the alternatives of a probabilistic value and the probability distribution of a probabilistic value will always be indicated by the same subscript as the probabilistic value itself, e.g. $z_1 = (y_1, Pr_1)$.

Example 80 *Let the set Colors* $= \{white, yellow, green, red, cyan, blue, black\}$ *be a domain of colors that cars can have. The value* $x = red$ *is a crisp value of the domain Colors and models the color of a car that is exactly known to be red. The value* $y = \{red, blue, black\}$ *is an incomplete value of the domain Colors and models the color of a car that is not exactly known, but is either red, blue, or black. The value* $z = \{(red\colon 0.6), (green\colon 0.4)\}$ *is a probabilistic value of the domain Colors and models the color of a car that is red with a probability of* 0.6 *and is green with a probability of* 0.4.

A domain can be build on other domains. This lead to some interesting circumstances. Naturally, uncertainty can be defined on several levels. For example, the value $z \in^{[\rho]} dom^{[\iota]}$ is a probabilistic value that has incomplete values of dom as alternatives. For this reason we will consider some characteristics of uncertain values in the following subsections.

6.2.1. Analogies

As it can be simply seen, the concept of probabilistic values is closely related to probability mass functions and is closely related to random variables.

A *probability mass function* [Das10] that is defined on a domain dom is a function that associates probabilities to a finite number of elements of dom so that the accumulative probability of all domain elements is exact one, i.e. it is a discrete probability distribution on dom. Thus, from a mathematical point of view, a probabilistic value corresponds to a probability mass function that additionally stores all the domain elements having a probability greater than zero.

A random variable is a function that maps elements of an event space to elements of a measurable space [FG96] where the set of the real numbers is typically used as output space. A random variable corresponds to an incomplete value if the domain of the latter is a measurable space. In that case the variable's range, i.e. its set of possible output values, corresponds to the set of alternatives of the incomplete value. Typically a random variable is combined with a probability mass function that associates probabilities to the elements of the variable's range. In that case, a random variable corresponds to a probabilistic value.

6.3. Uncertain Value Transformation

In some use cases, it is useful to transform an incomplete value into a probabilistic value or vice versa.

A probabilistic value of the domain dom can be transformed to an incomplete value of the same domain by ignoring its probability distribution.

Definition 36 (Probabilistic Value Transformation): *Let* dom *be an arbitrary domain and let* $z_i = (y_i, Pr_i) \in^{[\rho]} dom$ *be a probabilistic value of* dom. *The function* $f_{\rho\to\iota}\colon dom^{[\rho]} \to dom^{[\iota]}$ *transforms* z_i *to an incomplete value* $y_j \in^{[\iota]} dom$ *and is defined as:*

$$y_j = f_{\rho\to\iota}(z_i) = alt(z_i) = y_i$$

Since this transformation discards probabilities, it takes an information loss.

Under the assumption that all alternatives of an incomplete value are equal probable, an incomplete value of the domain dom can be transformed to a probabilistic value of the same domain by adding a probability distribution. Due to it makes some unfunded assumption by adding probabilities, this transformation contrives some information. We consider only point probability distributions in this thesis. Therefore, the amount of unfoundedly added information is minimized by using a uniform probability distribution.

Definition 37 (Incomplete Value Transformation): *Let dom be an arbitrary domain and let $y_i \in^{[\iota]} dom$ be an incomplete value of dom. The function $f_{\iota\to\rho}\colon dom^{[\iota]} \to dom^{[\rho]}$ transforms y_i to a probabilistic value $z_j \in^{[\rho]} dom$ and is defined as:*

$$z_j = f_{\iota\to\rho}(y_i) = (y_j, Pr_j)$$

$$\textit{where} \qquad y_j = y_i \qquad \textit{and} \qquad \forall x \in dom_j : Pr_j(x) = \begin{cases} \frac{1}{|y_j|}, & \textit{if } x \in y_j, \\ 0, & \textit{else.} \end{cases}$$

Example 81 *The probabilistic value $z_1 = \{(red\colon 0.2), (green\colon 0.8)\}$ can be transformed to the incomplete value $y_2 = f_{\rho\to\iota}(z_1) = \{red, green\}$ by ignoring probabilities. On the other hand, the incomplete value $y_3 = \{red, blue, black\}$ can be transformed to the probabilistic value $z_4 = f_{\iota\to\rho}(y_3) = \{(red\colon 1/3), (blue\colon 1/3), (black\colon 1/3)\}$.*

6.4. Uncertain Value Flatting

As mentioned above, uncertainty can theoretically exists on different levels, i.e. an incomplete value can be a set of incomplete values where each of these values is in turn a set of incomplete values and so on. If the uncertain values of all levels are based on the same domain and are of the same type of uncertainty, i.e. incomplete values or probabilistic values, the different levels can be flattened to a single level without losing any semantical information.

In other words: an incomplete value $y_i \in^{[\iota]} dom^{[\iota]}$ is semantically equivalent to an incomplete value $y_j \in^{[\iota]} dom$ if $y_j = \bigcup y_i$. This equivalence can be illustrated as follows: The incomplete value y_i means that one of its alternatives, i.e. the incomplete values $y_k \in y_i$, is a 'true' representation of the real-word fact that is modeled by y_i. An incomplete value y_k is in turn a 'true' representation, if one of the values $x \in y_k$ is the 'true' value. Consequently one of the values $\bigcup y_i$ is the 'true' value. This in turn is exactly the semantical information that is modeled by y_j.

Definition 38 (Incomplete Value Flatting): *Let $y_i \in^{[\iota]} dom^{[\iota]}$ be an incomplete value of the domain $dom^{[\iota]}$, i.e. a set of incomplete values of the domain dom. The value y_i can be flattened to an incomplete value $y_j \in^{[\iota]} dom$ by using the function flat-i$\colon (dom^{[\iota]})^{[\iota]} \to dom^{[\iota]}$ that performs the set union of all its alternatives: $y_j =$ flat-i$(y_i) = \bigcup y_i$.*

Example 82 *Let $y = \{\{red, green\}, \{blue, green\}\}$ be an incomplete value of Colors$^{[\iota]}$. This means that the color represented by y is either red or green, or blue or green and hence is either red, green, or blue. Thus, y can be flattened to flat-i$(y) = \{red, green, blue\}$.*

The same principle can be applied to a probabilistic value. A probability distribution on a probability distribution on a domain dom means that a value $x \in dom$ is the 'true' value, if it is the 'true' value of the 'true' distribution. Since a value can be possible in multiple possible distributions, its probability to be the 'true' value result by suming up its probability in every possible distribution.

Definition 39 (Probabilistic Value Flatting): *Let $z_i \in^{[\rho]} dom^{[\rho]}$ with $z_i = (y_i, Pr_i)$ be a probabilistic value of the domain $dom^{[\rho]}$, i.e. y_i is a set of probabilistic values of the domain dom. The value z_i can be flattened to a probabilistic value $z_j \in^{[\rho]} dom$ with $z_j = (y_j, Pr_j)$ by using the function flat-p$\colon (dom^{[\rho]})^{[\rho]} \rightarrow dom^{[\rho]}$:*

$$
\begin{aligned}
& & z_j &= \textit{flat-p}(z_i) = (y_j, Pr_j) \\
& \textit{where} & y_j &= \bigcup \{y_k \mid z_k \in y_i, z_k = (y_k, Pr_k)\} \\
& \textit{and} & \forall x \in dom_j : Pr_j(x) &= \sum\nolimits_{z_k=(y_k,Pr_k)\in y_i, x\in y_k} Pr_i(z_k) \times Pr_k(x)
\end{aligned}
$$

Example 83 *Let $z_1 = \{(z_{11}\colon 0.4), (z_{12}\colon 0.6)\}$ be a probabilistic value of Colors$^{[\rho]}$ where $z_{11} = \{(\textit{red}\colon 0.8), (\textit{green}\colon 0.2)\}$, and $z_{12} = \{(\textit{blue}\colon 0.6), (\textit{green}\colon 0.4)\}$. The value z_1 can be flattened to the value $z_2 =$ flat-p$(z_1) = (\{\textit{red}, \textit{green}, \textit{blue}\}, Pr_2)$ where:*

$$
\begin{aligned}
Pr_2(\textit{red}) &= Pr_1(z_{11}) \times Pr_{11}(\textit{red}) = 0.32 \\
Pr_2(\textit{green}) &= Pr_1(z_{11}) \times Pr_{11}(\textit{green}) + Pr_1(z_{12}) \times Pr_{12}(\textit{green}) = 0.32 \\
Pr_2(\textit{blue}) &= Pr_1(z_{12}) \times Pr_{12}(\textit{blue}) = 0.36
\end{aligned}
$$

By using uncertain value transformations first, a probabilistic value of the domain $dom^{[\iota]}$ or an incomplete value of the domain $dom^{[\rho]}$ can be flattened to a probabilistic value of domain dom if we transform all incomplete values to probabilistic values, or can be flattened to an incomplete value of domain dom if we transform all probabilistic values to incomplete values.

It is important to note that such a flatting is only semantically lossless if all 'levels' are based on the same domain. As a negative example, we consider an example from the field of clustering. Recall from Section 2.2 that *C-All*$_{\otimes}(S)$ is the set of all possible cluster-disjoint clusterings of the element set S. An incomplete value of domain *C-All*$_{\otimes}(S)$, i.e. an incomplete clustering, is a set of alternative cluster-disjoint clusterings of S. Each clustering is in turn a set of clusters and each cluster is a subset of S. Theoretically an incomplete clustering is a set of sets of sets and hence looks like it could be flattened to a single set. However, a clustering does not model uncertainty, i.e. not only one of its cluster is the 'true' cluster, but all of them together form a 'true' set of clusters. As a consequence, to flatten the incomplete clustering means to modify it in a mutually exclusive set of clusters and hence changes its semantics as well. As we take a closer look we see that this change of semantics is caused by the fact that not all levels are based on the same domain. The domain of the clusters is $dom_A = 2^S - \emptyset$, whereas the domain of the clusterings is $dom_B =$ *C-All*$_{\otimes}(S)$ and hence is non-equal to dom_A.

6.5. Uncertain Value Dependencies

Some uncertain values are correlated to each other, that means that the 'true' alternative of one value depends on the 'true' alternative of the other values.

In our formalism, correlations between uncertain values cannot be derived from the values themselves, but need to be explicitly modeled by an own uncertain value that models the possible combinations of alternatives of the correlated values. In the probabilistic case, such an uncertain value can be considered as a joint probability distribution on the correlated values.

With respect to independence, we have to distinguish between the pairwise independence between two uncertain values and the mutual independence of a set of uncertain values. Due to the first is a special case of the latter, we restrict our consideration to mutual (in)dependencies.

Since we sometimes compare different alternative scenarios, it can be the case that for the same set of uncertain values different sets of correlations exist. For that reason, we interpret the meaning of (in)dependence always within a given context.

Definition 40 (Incomplete Value Dependencies): *Let $S = \{y_1, \ldots, y_k\}$ be a set of incomplete values where $\forall i \in \{1, \ldots, k\} : y_i \in^{[\iota]} dom_i$. Moreover, let y_S be an incomplete value of the domain $dom_S = dom_1 \times \ldots \times dom_k$ that models all possible combinations of the alternatives of the values in S. The values $y_1, \ldots, y_k$ are mutually*

- *independent w.r.t. y_S, iff: $y_S = \{(x_1, \ldots, x_k) \mid x_1 \in y_1, \ldots, x_k \in y_k\}$*
- *correlated w.r.t. y_S, iff: $y_S \subsetneq \{(x_1, \ldots, x_k) \mid x_1 \in y_1, \ldots, x_k \in y_k\}$*

The same principle can be applied to probabilistic values. The only difference is that we have to take the probabilities of the individual alternatives additionally into account. For that reason, we distinguish between strong correlations and weak correlations.

Definition 41 (Probabilistic Value Dependencies): *Let $S = \{z_1, \ldots, z_k\}$ be a set of probabilistic values where $\forall i \in \{1, \ldots, k\} : z_i \in^{[\rho]} dom_i$ and $z_i = (y_i, Pr_i)$. Moreover, let $z_S = (y_S, Pr_S)$ be a probabilistic value of the domain $dom_S = dom_1 \times \ldots \times dom_k$ that models all possible combinations of the alternatives of the values in S. The values $z_1, \ldots, z_k$ are mutually*

- *independent with respect to z_S, iff: $y_1, \ldots, y_k$ are mutually independent and*

$$\forall x_1 \in dom_1 \colon \ldots \forall x_k \in dom_k \colon Pr_S(x_1, \ldots, x_k) = \prod\nolimits_{i \in \{1,\ldots,k\}} Pr_i(x_i)$$

- *weak correlated with respect to z_S, iff: $y_1, \ldots, y_k$ are mutually independent and*

$$\exists x_1 \in dom_1 \colon \ldots \exists x_k \in dom_k \colon Pr_S(x_1, \ldots, x_k) \neq \prod\nolimits_{i \in \{1,\ldots,k\}} Pr_i(x_i)$$

- *strong correlated with respect to z_S, iff: $y_1, \ldots, y_k$ are mutually correlated*

Since correlations need to be explicitly modeled, we assume a mutual independence between the k uncertain values $U = \{u_1, \ldots, u_k\}$, if in the considered context no value exists that models the correlations of any subset of U.

For simplification, we usually denote a probabilistic value that models the joint probability distribution of the probabilistic values $z_i = (y_i, Pr_i)$ and $z_j = (y_j, Pr_j)$ as $z_{i,j} = (y_{i,j}, Pr_{i,j})$. Moreover, we sometimes use the notion of a probabilistic value $z_{i|j} = (y_{i|j}, Pr_{i|j})$ that models the conditional probability of z_i with $z_j = x_j$ given. Nevertheless, this probabilistic value does not need to be separately modeled, but can be derived from $z_{i,j}$ as $y_{i|j} = y_{i,j}$ and $Pr_{i|j}(x) = \frac{Pr_{i,j}(x)}{Pr_j(x)}$.

Example 84 *Let $y_1 = \{red, blue\}$, $y_2 = \{red, green\}$, and $y_3 = \{red, black\}$ be three incomplete values that each models the color of a car. Moreover, let us assume that the incomplete value $y_{1,2,3}$ models all possible combinations of the cars' colors. The values y_1, y_2, and y_3 are mutually independent w.r.t. $y_{1,2,3}$, iff*

$$\begin{aligned} y_{1,2,3} \quad = \quad & \{(red, red, red), (red, red, black), (red, green, red), (red, green, black), \\ & (blue, red, red), (blue, red, black), (blue, green, red), (blue, green, black)\} \end{aligned}$$

Now, let us assume that it is known that at most one of the cars is red. Consequently, all values are correlated and we redefine $y_{1,2,3}$ to:

$$y_{1,2,3} \quad = \quad \{(red, green, black), (blue, red, black), (blue, green, red), (blue, green, black)\}$$

6.6. Uncertain Value (De)Composition

Sometimes it is useful to compose several uncertain values to a single uncertain value or to decompose a single uncertain value to multiple uncertain values.

We start with the decomposition of an incomplete value.

Definition 42 (Incomplete Value Decomposition): *Let dom_S be a domain that can be defined by the cross product of the domains dom_1 to dom_k. An incomplete value $y \in^{[i]} dom_S$ can be decomposed into k incomplete values $\{y_1, \ldots, y_k\}$ where $\forall i \in \{1, \ldots, k\}$: $y_i \in^{[i]} dom_i$ by using the function dec-i:*

$$\begin{aligned} & \textit{dec-i}(y, [dom_1, \ldots, dom_k]) = \{y_1, \ldots, y_k\} \\ & \textit{so that:} \qquad \forall i \in \{1, \ldots, k\}: y_i = \{x[i] \mid x \in y\} \end{aligned}$$

where $x[i]$ refers the crisp value at the ith position of a tuple x.

Example 85 *For illustration, we consider the incomplete values y_1, y_2, and y_3 from Example 84 and assume that their correlations are modeled by the incomplete value $y_{1,2,3}$ that is defined as:*

$$y_{1,2,3} \quad = \quad \{(red, green, black), (blue, red, black), (blue, green, red), (blue, green, black)\}$$

This value can be decomposed into the values y_1, y_2 and y_3, but can be also decomposed into the values $y_{1,2}$ and y_3. The domain of $y_{1,2,3}$ is actually $dom_{3C} = Colors \times Colors \times Colors$. In the first case, we decompose $y_{1,2,3}$ into three incomplete values of the domain Colors. In the second case, we consider the domain to be $dom_{3C} = dom_{2C} \times Colors$ where $dom_{2C} = Colors \times Colors$ and then decompose $y_{1,2,3}$ into two incomplete values: one of the domain Colors (value y_3) and one of the domain dom_{2C} (value $y_{1,2}$). From decomposing our sample value $y_{1,2,3}$ in the described ways, the following sets of incomplete values result:

$$\begin{aligned} & \textit{dec-i}(y, [Colors, Colors, Colors]) = \{y_1 = \{red, blue\}, y_2 = \{red, green\}, y_3 = \{red, black\}\} \\ & \textit{dec-i}(y, [dom_{2C}, Colors]) = \{y_{1,2} = \{(red, green), (blue, red), (blue, green)\}, y_3 = \{red, black\}\} \end{aligned}$$

As demonstrated by this example, decomposition can be used to derive an incomplete value that models the correlations between the incomplete values of a set S from an incomplete value that models the correlations between the incomplete values of a set $S' \supset S$.

Decomposition can be used to illustrate that pairwise independence does not imply mutual independence.

Example 86 *For demonstration, we consider the incomplete values y_1, y_2, and y_3 from Example 84 and assume that their correlations are modeled by the incomplete value $y_{1,2,3}$ that is defined as:*

$$y_{1,2,3} = \{(red, red, red), (red, green, black), (blue, green, red), (blue, red, black)\}$$

Using decomposition from this value the uncertain values $y_{1,2}$, $y_{1,3}$, and $y_{2,3}$ can be derived as follows:

$$\begin{aligned} y_{1,2} &= \{(red, red), (red, green), (blue, green), (blue, red)\} \\ y_{1,3} &= \{(red, red), (red, black), (blue, red), (blue, black)\} \\ y_{2,3} &= \{(red, red), (green, black), (green, red), (red, black)\} \end{aligned}$$

Due to $y_{1,2} = y_1 \times y_2$, $y_{1,3} = y_1 \times y_3$, and $y_{2,3} = y_2 \times y_3$ the values y_1, y_2, and y_3 are all pairwise independent to each other. However, because $y_{1,2,3} \neq y_1 \times y_2 \times y_3$ they are not mutually independent. This is because any pairwise combination of their alternatives is included in an alternative of $y_{1,2,3}$, but not all ternary combinations of alternatives are included in an alternative of $y_{1,2,3}$.

A decomposition of probabilistic values can be defined accordingly.

Definition 43 (Probabilistic Value Decomposition): *Let dom_S be a domain that can be defined by the cross product of the domains dom_1 to dom_k. A probabilistic value $z \in^{[\rho]} dom_S$ with $z = (y, Pr)$ can be decomposed into k probabilistic values $\{z_1, \ldots, z_k\}$ where $\forall i \in \{1, \ldots, k\}: z_i \in^{[\rho]} dom_i, z_i = (y_i, Pr_i)$ by using the function dec-p:*

$$\begin{aligned} & \textit{dec-p}(z, [dom_1, \ldots, dom_k]) = \{z_1, \ldots, z_k\} \\ \textit{so that:} \quad & \{y_1, \ldots, y_k\} = \textit{dec-i}(y, [dom_1, \ldots, dom_k]) \\ \textit{and} \quad & \forall i \in \{1, \ldots, k\}: \forall x \in dom_i: Pr_i(x) = \sum\nolimits_{x' \in y, x'[i]=x} Pr(x') \end{aligned}$$

This approach corresponds to probability theory, where the joint probability distribution on the probabilistic values in a set S can be derived from the joint probability distribution on the probabilistic values in a set $S' \supset S$ by summing up the probabilities of all combinations in S' that have same values in S.

We call a decomposition to be lossless, if the incomplete version of dom_S can be expressed by the cross product of the incomplete versions of dom_1 to dom_k:

$$dom_S^{[\iota]} = (dom_1 \times \ldots \times dom_k)^{[\iota]} = dom_1^{[\iota]} \times \ldots \times dom_k^{[\iota]}$$

Note that a decomposition is lossless, if the resultant values are mutually independent. The same holds for the probabilistic case.

From the principle of decomposition we can derive the counterprinciple of composition, i.e. a set of uncertain values can be composed to a single uncertain value.

In our formalism we use composed values to model correlations. As a consequence, we have a chicken-and-egg problem because correlations can only be incorporated into composition if the composition already exists. Thus, for any set of correlated values, a composed value need to be explicitly specified and a general function for computing the composition of a set of uncertain values can be only defined for the case of mutual independence. For that reason, the input to a composition function is always assumed to be a set of mutually independent values.

Thus, according to Definition 40, the composition of a set of incomplete values is defined as:

Definition 44 (Incomplete Value Composition): *Let $S = \{y_1, \ldots, y_k\}$ be a set of incomplete values where $\forall i \in \{1, \ldots, k\}\colon y_i \in^{[\iota]} dom_i$. These values can be composed to a single incomplete value $y_S \in^{[\iota]} dom_S$ where $dom_S = dom_1 \times \ldots \times dom_k$ by using the function comp-i:*

$$y_S = \textit{comp-i}(y_1, \ldots, y_k) = \{(x_1, \ldots, x_k) \mid x_1 \in y_1, \ldots, x_k \in y_k\}\}$$

According to Definition 41, the composition of a set of probabilistic values is defined as:

Definition 45 (Probabilistic Value Composition): *Let $S = \{z_1, \ldots, z_k\}$ be a set of probabilistic values where $\forall i \in \{1, \ldots, k\}\colon z_i \in^{[\rho]} dom_i, z_i = (y_i, Pr_i)$. These values can be composed to a single probabilistic value $z_S \in^{[\rho]} dom_S$ where $dom_S = dom_1 \times \ldots \times dom_k$ by using the function comp-p:*

$$\begin{aligned} & z_S = \textit{comp-p}(z_1, \ldots, z_k) = (y_S, Pr_S) \\ \textit{so that:} \quad & y_S = \{(x_1, \ldots, x_k) \mid x_1 \in y_1, \ldots, x_k \in y_k\}\} \\ \textit{and} \quad & \forall (x_1, \ldots, x_k) \in dom_S\colon Pr_S(x_1, \ldots, x_k) = \prod\nolimits_{i \in \{1,\ldots,k\}} Pr_i(x_i) \end{aligned}$$

Following the principle of composition, a cross product of the incomplete versions of dom_1 to dom_k can be expressed by the incomplete version of dom_S:

$$dom_1^{[\iota]} \times \ldots \times dom_k^{[\iota]} = (dom_1 \times \ldots \times dom_k)^{[\iota]} = dom_S^{[\iota]}$$

The same holds for the probabilistic case.

6.7. Mapping Uncertain Values

Until now, we focus on properties and transformations of incomplete values and probabilistic values. Nevertheless, we sometimes need to apply conventional function to these values and therefore need to adapt functions that have been defined for crisp values to incomplete input values or probabilistic input values respectively.

An action applied to the real world can be formalized by a function that gets the world as input and produces the action result as output. Thus, according to the possible worlds semantics (see Section 6.1), a function is adapted to incomplete values by applying the function to each of the value's alternatives and hence by creating an incomplete value of the function's output domain.

Definition 46 (Mapping Incomplete Values): *Let $f : dom_A \rightarrow dom_B$ be a function that maps a crisp value of the domain dom_A to a crisp value of the domain dom_B and let y_i be an incomplete value of domain dom_A. The function $f : dom_A^{[\iota]} \rightarrow dom_B^{[\iota]}$ is called the incomplete version of f and is defined as:*

$$f(y_i) = y_j = \{f(x) \mid x \in y_i\}$$

Example 87 *Let f_{colful}: Colors $\rightarrow \mathbb{R}$ be a function that computes the colorfulness of a specific color, i.e. the degree of difference between this color and gray. Let $y = \{red, black, white\}$ be an incomplete value of the domain Colors that models the uncertain information on the color of a specific object. The colorfulness of this object's color is then represented by $y_{cf} = f_{colful}(y)$ that is an incomplete value of the domain $\mathbb{R}$. Let us assume that the colorfulness of red is 0.8, and the colorfulness of black and white is both 0.0. Consequently, the colorfulness of the color modeled by y is modeled by $y_{cf} = \{0.0, 0.8\}$.*

According to the possible worlds semantics, a function is adapted to probabilistic values by applying the function to each of the value's alternatives and by defining the probability of a resultant alternative as the sum of the probabilities of the input alternatives that are mapped to that output alternative. Consequently, the function produces a probabilistic value of its output domain as a result.

Definition 47 (Mapping Probabilistic Values): *Let $f : dom_A \rightarrow dom_B$ be a function that maps a crisp value of the domain dom_A to a crisp value of the domain dom_B and let $z_i = (y_i, Pr_i)$ be a probabilistic value of domain dom_A. The function $f : dom_A^{[\rho]} \rightarrow dom_B^{[\rho]}$ is called the probabilistic version of f and is defined as:*

$$f(z_i) = z_j = (y_j, Pr_j)$$
$$\textit{where} \quad y_j = f(y_i) \quad \textit{and} \quad \forall x \in dom_B : Pr_j(x) = \sum\nolimits_{x' \in y_i, f(x')=x} Pr_i(x')$$

Example 88 *Let $z = \{(red\colon 0.6), (black\colon 0.3), (white\colon 0.1)\}$ be a probabilistic value of the domain Colors that models the uncertain information on the color of a specific object. The colorfulness of this object's color is then represented by $z_{cf} = f_{colful}(z) = (y_{cf}, Pr_{cf})$ that is a probabilistic value of the domain $\mathbb{R}$. By assuming the same values of colorfulness as in Example 87, the alternatives of z_{cf} are $y_{cf} = \{0.0, 0.8\}$ and the probabilities result in $Pr_{cf}(0.8) = Pr(red) = 0.6$ and $Pr_{cf}(0.0) = Pr(black) + Pr(white) = 0.4$.*

Since a domain can be built on other domains, this concept is not restricted to functions with a single value as input. We will illustrate this by the following simple example.

Example 89 *Let $dom_B = \mathbb{N}$ be the domain of natural numbers and let $dom_A = \mathbb{N} \times \mathbb{N}$ be the domain of all pairs of natural numbers. Moreover, let $f^+ : dom_A \rightarrow dom_B$ be a function that maps a pair of natural numbers to a single natural number by computing their sum, i.e. for any input value $x = (x_1, x_2)$ the result from applying function f^+ on x is defined as $f^+(x) = x_1 + x_2$. Since the input is a pair of*

numbers, f^+ can be also written as $f^+ : dom_B \times dom_B \rightarrow dom_B$ and hence can be defined as $f^+(x_1, x_2) = x_1 + x_2$.

Let $y \in^{[\iota]} dom_A$ be an incomplete value of dom_A. The function $f^+ : dom_A^{[\iota]} \rightarrow dom_B^{[\iota]}$ is the incomplete version of f^+ and is defined as:

$$\begin{aligned} f^+(y) &= \{f^+(x) \mid x \in y\} \\ &= \{f^+(x_1, x_2) \mid x \in y, x = (x_1, x_2)\} \\ &= \{x_1 + x_2 \mid x \in y, x = (x_1, x_2)\} \end{aligned}$$

It is simply to see that y models the possible combinations of two incomplete values y_1 and y_2 of domain dom_B and hence is a composition of both. In the case y_1 and y_2 are mutually independent, the incomplete version of f^+ can be also written as:

$$\begin{aligned} f^+(y_1, y_2) &= \{f^+(x_1, x_2) \mid x_1 \in y_1, x_2 \in y_2\} \\ &= \{x_1 + x_2 \mid x_1 \in y_1, x_2 \in y_2\} \end{aligned}$$

As a consequence, let $\{y_1, \ldots, y_k\}$ be a set of mutually independent incomplete values, the incomplete version of the sum operator produces the incomplete value y_l and is defined as:

$$y_l = \sum_{i=1}^{k} y_i = \{\sum_{i=1}^{k} x_i \mid x_1 \in y_1, \ldots, x_k \in y_k\} \tag{6.1}$$

The probabilistic version of the sum operator is defined accordingly.

Obviously, we can adopt this principle from the sum operator to any other k-nary operator. As a consequence, let $\{y_1, \ldots, y_k\}$ be a set of mutually independent incomplete values, the incomplete version of the operator Θ produces the incomplete value y_l and is defined as:

$$y_l = \Theta_{i=1}^{k} y_i = \{\Theta_{i=1}^{k} x_i \mid x_1 \in y_1, \ldots, x_k \in y_k\} \tag{6.2}$$

The probabilistic version of Θ is defined accordingly.

Often we need to process an uncertain value with a crisp value.

Theorem 4 *Let f: $dom_A \times dom_B \rightarrow dom_C$ be a function that maps two crisp values of domain dom_A or dom_B respectively to a crisp value of domain dom_C, let $y_1 \in^{[\iota]} dom_A$ be an incomplete value of domain dom_A and let $x_2 \in dom_B$ be a crisp value of domain dom_B, the incomplete version of f computes the incomplete value $y_3 \in^{[\iota]} dom_C$:*

$$y_3 = f(y_1, x_2) = \{f(x_1, x_2) \mid x_1 \in y_1\}$$

Proof 1 *Theorem 4 can be proved by composing y_1 and x_2. Let $y_2 \in^{[\iota]} dom_A \times dom_B$ be an incomplete value that models all possible combinations of y_1 and x_2, y_2 is defined as: $y_2 = \{(x_1, x_2) \mid x_1 \in y_1\}$. Consequently, the incomplete version of f with only y_2 as input is defined as:*

$$f(y_2) = \{f(x) \mid x \in y_2\} = \{f(x_1, x_2) \mid x_1 \in y_1\}$$

Special cases of such functions are functions that multiply a constant value to an incomplete value and functions that add a constant value to an incomplete value. Let $f^{\times}$ be a function that multiplies two crisp numerical values, let f^{+} be the aforementioned function that adds two crisp numerical values. Moreover, let y_1 be an incomplete numerical value and let x_2 be a crisp numerical value, the incomplete versions of these functions each computes an incomplete numerical value as follows:

$$\begin{aligned} f^{\times}(y_1, x_2) &= y_1 \times x_2 = \{x_1 \times x_2 \mid x_1 \in y_1\} \\ f^{+}(y_1, x_2) &= y_1 + x_2 = \{x_1 + x_2 \mid x_1 \in y_1\} \end{aligned}$$

The probabilistic versions of functions with several input values can be derived accordingly.

6.7.1. Uncertainty Including Functions

In many use cases, besides the input data, the used mapping can include uncertainty as well and hence is of type $f : dom_A \rightarrow dom_B^{[\iota/\rho]}$. It is interesting to note that such cases are already covered by Definition 46 and Definition 47, because we can assume that the target domain dom_C of a conventional function is the uncertain version of dom_B, i.e. $dom_C = dom_B^{[\iota/\rho]}$.

For illustration, we consider the incomplete version of a function that maps a crisp value of domain dom_A on an incomplete value of domain dom_B. Let $y_1 \in^{[\iota]} dom_A$. The incomplete version of function $f : dom_A \rightarrow dom_B^{[\iota]}$, i.e. $f : dom_A^{[\iota]} \rightarrow (dom_B^{[\iota]})^{[\iota]}$, is then defined as:

$$\begin{aligned} f(y_1) &= y_2 = \{f(x) \mid x \in y_1\} \\ &= \{\{x' \mid x' \in f(x)\} \mid x \in y_1\} \end{aligned}$$

Since, the resultant value y_2 has two uncertainty levels that are both based on the domain dom_B, it is semantically equivalent to its flattened value *flat-i*(y_2):

$$f(y_1) = y_3 = \textit{flat-i}(y_2) = \{x' \mid x \in y_1, x' \in f(x)\}$$

Example 90 *Two colors are called complementary, if their mix results in a neutral hue/shade of gray. What color complements another color depends on the considered color model, e.g. the RGB color model or the Natural Color System (NCS). Let us assume a situation in which we do not know the considered color model and hence define an uncertainty including function* $f_{\textit{compl-i}}$*: Colors* $\rightarrow$ *Colors*$^{[\iota]}$ *that computes a set of possible complementations of a specific color. Since in the given context only one of the color models can be the actually considered one, only one of the possible colors is the 'true' complementation and we can use the concept of incomplete values to model this uncertainty.*

Let $x =$ *red be the color we want to compute the complementary color for. In the RGB color model, the complementary color of red is cyan. In contrast, in the NCS, the complementary color of red is green. Consequently, the function* $f_{\textit{compl-i}}(x)$ *produces an incomplete value* y_1 *of the domain Colors that contains cyan and green as alternatives, i.e.* $y_1 = \{\textit{cyan}, \textit{green}\}$ *(here we assume that no other color model is additionally used).*

Let $y_2 = \{\textit{red}, \textit{green}\}$ *be an incomplete value that models the color we want to compute the complementary color for. The complementary color of green is magenta (RGB color model) or red (NCS). Consequently, the function* $f_{\textit{compl-i}}(y_2)$ *produces the incomplete value* $y_3 = \{\{\textit{cyan}, \textit{green}\}, \{\textit{magenta}, \textit{red}\}\}$.

Since any of the four colors is the true complementary color of the color modeled by y_2, this value can be flattened to $y_4 = \{cyan, green, magenta, red\}$.

6.7.2. Uncertain Value Functions

As an uncertain value function we consider such functions that are designed with uncertain input values in mind. Since the input of such functions is already an uncertain value, they are extra defined for processing uncertainty and hence do not need to be adapted to uncertain values anymore. Nevertheless, this means not that uncertain value function does not have an incomplete version or a probabilistic version, because these are required if uncertainty exists on multiple levels. Two special kinds of uncertain value functions are *uncertainty resolution functions* and *uncertainty reduction functions* which we will discuss in the following two subsections.

6.7.2.1. Uncertainty Resolution Functions

A function f is uncertainty resolving, if it maps an uncertain value of domain dom_A to a crisp value of domain dom_B and hence is of type $f\colon dom_A^{[\iota/\rho]} \to dom_B$.

If we restrict to the case where the output domain corresponds to the input domain, i.e. $dom_B = dom_A$, the uncertainty of an uncertain value (incomplete or probabilistic) can be resolved in two ways[2]: (i) by selecting one of its alternatives as resolution output (deciding strategy), or (ii) by combining its alternatives to any single element of the output domain (mediating strategy). Whereas from using a deciding strategy always one of the given input alternatives is returned as output value, a function that uses a mediating strategy may produce a resolution output that does not necessarily correspond to an alternative of the resolved uncertain value.

Definition 48 (Deciding Uncertainty Resolution Function): *Let $f\colon dom^{[\iota/\rho]} \to dom$ be an uncertainty resolution function. The function f is called to be* deciding, *if $\forall u \in dom^{[\iota/\rho]}\colon f(u) \in alt(u)$.*

Example 91 *In the color example, we can resolve uncertainty by selecting the brightest color of all possible colors. Let $y = \{green, blue, yellow\}$ be an incomplete value of the domain Colors and let $f_{brightness}\colon Colors \to \mathbb{R}$ be a function that computes the brightness of a color. A corresponding uncertainty resolution function $f_{brightest}\colon Colors^{[\iota]} \to Colors$ is then defined as:*

$$f_{brightest}(y) = \mathbf{argmin}_{x \in y} f_{brightness}(x)$$

Definition 49 (Mediating Uncertainty Resolution Function): *Let $f\colon dom^{[\iota/\rho]} \to dom$ be an uncertainty resolution function. The function f is called to be* mediating, *if $\exists u \in dom^{[\iota/\rho]}\colon f(u) \notin alt(u)$.*

[2]Here we adopt the distinction into *deciding* strategies and *mediating* strategies from current research on certain data fusion that has been published by Bleiholder et al. [BN06, BN08].

Example 92 *Colors can be combined by mixing them. Therefore, the uncertainty of an incomplete value y can be resolved by mapping it to the color that result from mixing all of its alternatives. Let $y = \{green, red\}$ be an incomplete value of the domain Colors and let us consider a function that uses the RGB color model to compute the mixture of a set of colors. Due to mixing the colors green and red results in the color yellow, the incomplete value y can be resolved to the crisp value $x = yellow$.*

Of course, in the case of probabilistic values the probability masses of the individual alternatives can be additionally taken into account. For instance, we can select the most probable color as resolution output (deciding strategy) or we can mix the colors by considering the probability masses as color masses (mediating strategy).

6.7.2.2. Uncertainty Reduction Functions

A function is uncertainty reducing, if its output value is always less or equal uncertain than its input value. In this section, we restrict to simple reduction functions where the input domain corresponds to the output domain and that do not have any functionality except uncertainty reduction. Nevertheless, more complex uncertainty reduction functions can be built by composing such a simple uncertainty reduction function with a function that maps the uncertain value of the considered domain to an uncertain value of another domain.

The meaning of uncertainty reduction essentially depends on the meaning of uncertainty (or certainty respectively) and therefore essentially depends on the measure that is used to rate uncertainty (see Section 6.10). An uncertainty measure is an uncertain value function that maps an uncertain value to a crisp numerical value. Since a crisp value is per definition certain, a measure M is only a valid measure of uncertainty, if it maps each uncertain value that has exact one alternative to the minimal possible value of its output domain v_{min}, i.e. $\forall u \in^{[\iota/\rho]} dom\colon M(u) = v_{min} \vee |alt(u)| > 1$. If the output domain of the quality measure is normalized, the minimal value is 0.

Definition 50 (Uncertainty Reduction): *Let M be a valid measure of uncertainty. A function $f\colon dom^{[\iota/\rho]} \rightarrow dom^{[\iota/\rho]}$ is called an uncertainty reduction function with respect to M, iff $\forall u \in dom^{[\iota/\rho]}\colon M(f(u)) \leq M(u)$ and $\exists u \in dom^{[\iota/\rho]}\colon M(f(u)) < M(u)$.*

Example 93 *For illustration we consider a simple measure M that rates the uncertainty of an uncertain value by its number of alternatives, i.e. the more alternatives an uncertain value has the greater is its uncertainty. Thus, let u be an uncertain value (incomplete or probabilistic), its uncertainty is defined as $M(u) = |alt(u)|$. Consequently, a simple example for an uncertainty reduction function is a function that maps an uncertain value to a subset of its alternatives (e.g. the k most probable alternatives in the probabilistic case).*

We can generalize the concepts of deciding strategies and mediating strategies from uncertainty resolution functions to uncertain value functions. We call an uncertainty value function to be *deciding*, if it simply selects some of the alternatives of the input value as the alternatives of the output value.

Definition 51 (Deciding Function): *Let $f\colon dom^{[\iota/\rho]} \rightarrow dom^{[\iota/\rho]}$ be an uncertain value function. The function f is called to be* deciding, *if $\forall u \in dom^{[\iota/\rho]}\colon alt(f(u)) \subseteq alt(u)$.*

In contrast, an uncertainty value function is called to be *mediating*, if its output value always has equal or less alternatives than its input value and if it is not deciding, i.e. some alternatives of the output value are no alternatives of the input value.

Definition 52 (Mediating Function): *Let* $f\colon dom^{[\iota/\rho]} \rightarrow dom^{[\iota/\rho]}$ *be an uncertain value function. The function* f *is called to be* mediating*, if* $\forall u \in dom^{[\iota/\rho]}\colon |alt(f(u))| \leq |alt(u)| \wedge \exists u' \in dom^{[\iota/\rho]}\colon alt(f(u')) \not\subseteq alt(u')$.

Since the accumulative probability of all alternatives of the output must be one, a deciding uncertainty reduction function for probabilistic values always includes a final normalization. In the case of mediating functions, such a normalization is not always required.

Note that we define both properties on the level of uncertain value functions instead of the level of uncertainty resolution functions. For that reason, they are independent from the used uncertainty measure M.

Example 94 *An example for a deciding function is a function that clusters all alternatives of its input value, then selects one alternative per cluster, and finally takes the selected alternatives as the alternatives of the output value. An example for a mediating function is a function that clusters all alternatives of its input value, then computes a single representative per cluster, and finally takes the newly created representatives as the alternatives of the output value.*

With respect to the most measures of uncertainty, a meaningful uncertainty reduction function is either deciding or mediating, i.e. it does not increase the number of alternatives.

Each uncertainty resolution function is an uncertain value function that produces a crisp output value. Therefore Definition 48 is conform to Definition 51 (deciding functions) and Definition 49 is conform to Definition 52 (mediating functions). Since the uncertainty of a crisp value is zero for each valid measure of uncertainty, an uncertainty resolution function is a special kind of uncertainty reduction function.

6.8. Uncertain Mappings

Since the definition of uncertain values is made in an abstract way, we can also apply it to a domain of functions. For ease of representations, we use specific symbols to name the different kind of functions, i.e. '$f^{[\iota]}$' to name incomplete functions, '$f^{[\rho]}$' to name probabilistic functions, and '$f^{[\iota/\rho]}$' to name uncertain functions. Crisp functions are always conventionally named by 'f'. As for the simplified naming of uncertain values, this naming convention always depends on the considered domain.

Definition 53 (Incomplete Function): *Let F-All*(dom_A, dom_B) *be the set of all crisp functions of type* $f\colon dom_A \rightarrow dom_B$ *and let* x *be a crisp value of domain* dom_A*. An incomplete function of type* f *is defined as an incomplete value* $f^{[\iota]} = \{f_1, \ldots, f_k\}$ *of the domain F-All. The result of applying* $f^{[\iota]}$ *to* x *is an incomplete value* $y \in^{[\iota]} dom_B$ *that is defined as:*

$$y = f^{[\iota]}(x) = \{f(x) \mid f \in f^{[\iota]}\}$$

Theorem 5 *Let $f^{[\iota]} \in^{[\iota]}$ F-All(dom_A, dom_B) be an incomplete function and let $y_1 \in^{[\iota]} dom_A$ be an incomplete value of domain dom_A. The incomplete version of $f^{[\iota]}$ applied to y_1 produces the incomplete value $y_2 \in^{[\iota]} dom_B$ so that:*

$$y_2 = f^{[\iota]}(y_1) = \{f(x) \mid f \in f^{[\iota]}, x \in y_1\}$$

Proof 2 *According to Definition 53, Definition 46, and Definition 38 the incomplete version of $f^{[\iota]}$ applied to y_1 results in:*

$$\begin{aligned} y_2 = f^{[\iota]}(y_1) &\overset{\text{Def. 53}}{=} \{f(y_1) \mid f \in f^{[\iota]}\} \\ &\overset{\text{Def. 46}}{=} \{\{f(x) \mid x \in y_1\} \mid f \in f^{[\iota]}\} \\ &\overset{\text{Def. 38}}{=} \{f(x) \mid f \in f^{[\iota]}, x \in y_1\} \end{aligned}$$

Theorem 6 *Let $f^{[\iota]} \in^{[\iota]}$ F-All(dom_A, dom_B) be an incomplete function and let $z_1 = (y_1, Pr_1)$ be a probabilistic value of domain dom_A. The probabilistic version of $f^{[\iota]}$ applied to z_1 produces the incomplete value $y_2 \in^{[\iota]} dom_B^{[\rho]}$ so that:*

$$\begin{aligned} y_2 &= f^{[\iota]}(z_1) = \{(f(y_1), Pr_f) \mid f \in f^{[\iota]}\} \\ \textit{where} \quad & \forall f \in f^{[\iota]} \colon \forall x \in dom_B \colon Pr_f(x) = \sum\nolimits_{x' \in y_1, f(x')=x} Pr_i(x') \end{aligned}$$

Proof 3 *According to Definition 53 and Definition 47 the probabilistic version of $f^{[\iota]}$ applied to z_1 results in:*

$$\begin{aligned} y_2 = f^{[\iota]}(z_1) &\overset{\text{Def. 53}}{=} \{f(z_1) \mid f \in f^{[\iota]}\} \\ &\overset{\text{Def. 47}}{=} \{(f(y_1), Pr_f) \mid f \in f^{[\iota]}\} \\ \textit{where} \quad & \forall f \in f^{[\iota]} \colon \forall x \in dom_B \colon Pr_f(x) = \sum\nolimits_{x' \in y_1, f(x')=x} Pr_i(x') \end{aligned}$$

By transforming the incomplete function to a probabilistic function (see Definition 37) or by transforming the probabilistic value to an incomplete value (see Definition 36), the result can be flattened to a probabilistic value of domain dom_B or can be flattened to an incomplete value of domain dom_B respectively.

Definition 54 (Probabilistic Function): *Let F-All(dom_A, dom_B) be the set of all crisp functions of type $f \colon dom_A \to dom_B$ and let x be a crisp value of domain dom_A. A probabilistic function of type f is defined as a probabilistic value $f^{[\rho]} = (f^{[\iota]}, Pr_f)$ of the domain F-All. The result of applying $f^{[\rho]}$ to x is a probabilistic value $z \in^{[\rho]} dom_B$ that is defined as:*

$$\begin{aligned} & f^{[\rho]}(x) = z = (y, Pr) \\ \textit{where} \quad & y = f^{[\iota]}(x) = \{f(x) \mid f \in f^{[\iota]}\} \\ \textit{and} \quad & \forall x' \in dom_B : Pr(x') = \sum\nolimits_{f \in f^{[\iota]}, f(x)=x'} Pr_f(f) \end{aligned}$$

Theorem 7 *Let $f^{[\rho]} = (f^{[\iota]}, Pr_f)$ be a probabilistic function of the domain F-All(dom_A, dom_B) and let $z_1 = (y_1, Pr_1)$ be a probabilistic value of domain dom_A. The probabilistic version of $f^{[\rho]}$ applied to z_1 produces the probabilistic value $z_2 \in^{[\rho]} dom_B$ so that:*

$$\begin{array}{rcl} z_2 & = & f^{[\rho]}(z_1) = (y_2, Pr_2) \\ \textit{where} & & y_2 = \{f(x) \mid f \in f^{[\iota]}, x \in y_1\} \\ \textit{and} & & \forall x \in dom_B \colon Pr_2(x) = \sum\nolimits_{f \in f^{[\iota]}, x' \in y_1, f(x') = x} Pr_1(x') \times Pr_f(f) \end{array}$$

Proof 4 *According to Definition 54, Definition 47, and Definition 39 the probabilistic version of $f^{[\rho]}$ applied to z_1 results in:*

$$\begin{array}{rcl} z_2 = f^{[\rho]}(z_1) & \overset{\text{Def. 54}}{=} & = (y_2, Pr_2) \\ & \textit{where} & y_2 = \{f(z_1) \mid f \in f^{[\iota]}\} \in^{[\iota]} dom_B^{[\rho]} \\ & \textit{and} & \forall x \in dom_B^{[\rho]} \colon Pr_2(x) = \sum\nolimits_{f \in f^{[\iota]}, f(z_1) = x} Pr_f(f) \\ & \overset{\text{Def. 47}}{=} & (y_2, Pr_2) \\ & \textit{where} & y_2 = \{(y_{1f}, Pr_{1f}) \mid f \in f^{[\iota]}\} \in^{[\iota]} dom_B^{[\rho]} \\ & \textit{and} & \forall x \in dom_B^{[\rho]} \colon Pr_2(x) = \sum\nolimits_{f \in f^{[\iota]}, f(z_1) = x} Pr_f(f) \\ & \textit{and} & \forall f \in f^{[\iota]} \colon \big(y_{1f} = \{f(x) \mid x \in y_1\} \in^{[\iota]} dom_B \\ & & \quad \textit{and} \quad \forall x \in dom_B \colon Pr_{1f}(x) = \sum\nolimits_{x' \in y_1, f(x') = x} Pr_1(x')\big) \\ & \overset{\text{Def. 39}}{=} & (y_2, Pr_2) \\ & \textit{where} & y_2 = \{f(x) \mid f \in f^{[\iota]}, x \in y_1\} \in^{[\iota]} dom_B \\ & \textit{and} & \forall x \in dom_B \colon Pr_2(v) = \sum\nolimits_{f \in f^{[\iota]}, x' \in y_1, f(x') = x} Pr_1(x') \times Pr_f(f) \end{array}$$

Theorem 8 *Let $f^{[\rho]} = (f^{[\iota]}, Pr_f)$ be a probabilistic function of the domain F-All(dom_A, dom_B) and let $y_1 \in^{[\iota]} dom_A$ be an incomplete value of domain dom_A. The incomplete version of $f^{[\rho]}$ applied to y_1 produces the probabilistic value $z_2 \in^{[\rho]} dom_B^{[\iota]}$ so that:*

$$\begin{array}{rcl} z_2 & = & f^{[\rho]}(y_1) = (y_2, Pr_2) \\ \textit{where} & & y_2 = \{y_f = \{f(x) \mid x \in y_1\} \mid f \in f^{[\iota]}\} \\ \textit{and} & & \forall x \in dom_B^{[\iota]} \colon Pr_2(x) = \sum\nolimits_{f \in f^{[\iota]}, f(y_1) = x} Pr_f(f) \end{array}$$

Proof 5 *Theorem 7 can be proved by using Defintion 54 and Defintion 46:*

$$
\begin{aligned}
z_2 = f^{[\rho]}(y_1) &\overset{\text{Def. 54}}{=} (y_2, Pr_2) \\
\textit{where} &\quad y_2 = \{f(y_1) \mid f \in f^{[\iota]}\} \in^{[\iota]} dom_B^{[\iota]} \\
\textit{and} &\quad \forall x \in dom_B^{[\iota]} \colon Pr_2(x) = \sum\nolimits_{f \in f^{[\iota]}, f(y_1)=x} Pr_f(f) \\
&\overset{\text{Def. 46}}{=} (y_2, Pr_2) \\
\textit{where} &\quad y_2 = \{y_f = \{f(x) \mid x \in y_1\} \mid f \in f^{[\iota]}\} \in^{[\iota]} dom_B^{[\iota]} \\
\textit{and} &\quad \forall x \in dom_B^{[\iota]} \colon Pr_2(x) = \sum\nolimits_{f \in f^{[\iota]}, f(y_1)=x} Pr_f(f)
\end{aligned}
$$

By transforming the probabilistic function to an incomplete function (see Definition 36) or by transforming the incomplete value to a probabilistic value (see Definition 37) , the result can be flattened to an incomplete value of domain dom_B or can be flattened to a probabilistic value of domain dom_B respectively.

Example 95 *For illustration, we reconsider the example scenario of color complementation. Let* $f_{\textit{compl-CM}} \colon \textit{Colors} \times \textit{ColorModels} \rightarrow \textit{Colors}$ *be a function that computes the complementary color* c_2 *of a specific color* c_1 *with respect to a specific color model CM, i.e.* $c_2 = f_{\textit{compl-CM}}(c_1, \textit{CM})$*. Let CM-All be the domain of all color models. The uncertainty including function* $f_{\textit{compl-i}}$ *from Example 90 can then be defined as an incomplete function* $f_{\textit{compl}}^{[\iota]}$ *that contains one function per considered color model, i.e.* $f_{\textit{compl}}^{[\iota]}(x) = \{f_{\textit{compl-CM}}(x, \textit{CM}) \mid \textit{CM} \in \textit{CM-All}\}$.

A function that produces multiple possible output values from a single input value must have at least one point in its process flow where several choices can be made. Let us call such point as choice points. Consequently, each uncertainty including function can be represented by an uncertain function, if we consider each possible combination of choices (one choice per choice point) as an own function.

6.8.1. Composition of Uncertain Mappings

Composing functions is to use the output of one function as the input to another [Haz01]. For example, let $f_1 \colon dom_A \rightarrow dom_B$ and $f_2 \colon dom_C \rightarrow dom_D$ two crisp functions where the input domain of f_2 is a superset of the output domain of f_1, i.e. $dom_B \subseteq dom_C$. These two functions can be composed to a third function $f_3 = f_2 \circ f_1 : dom_A \rightarrow dom_D$ that is defined by $f_3(x) = (f_2 \circ f_1)(x) = f_2(f_1(x))$ for all $x \in dom_A$.

The composition of uncertain functions is based on the same principle, i.e. a composition of k uncertain functions is equivalent to a subsequent application of these functions. Since the result of an uncertain function is always an uncertain value, the meaning of composition can be derived from our former definitions on uncertain functions.

Theorem 9 *Let* $f_1^{[\iota]}$ *be an incomplete function of the domain F-All*(dom_A, dom_B) *and let* $f_2^{[\iota]}$ *be an incomplete function of the domain F-All*(dom_C, dom_D) *where* $dom_B \subseteq dom_C$*. Moreover, let* $x_1 \in$

dom_A be a crisp value of domain dom_A. The composition $f_2^{[\iota]} \circ f_1^{[\iota]}$ of both functions is an incomplete function $f_3^{[\iota]}$ of the domain F-All(dom_A, dom_D) that is defined as:

$$f_3^{[\iota]}(x_1) \quad = \quad (f_2^{[\iota]} \circ f_1^{[\iota]})(x_1) = \{(f \circ f')(x_1) \mid f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}\}$$

Proof 6 *Theorem 9 can be proved by the use of Defintion 53 and Theorem 5:*

$$\begin{aligned}
f_3^{[\iota]}(x_1) &= (f_2^{[\iota]} \circ f_1^{[\iota]})(x_1) \\
&= f_2^{[\iota]}(f_1^{[\iota]}(x_1)) \\
&\overset{\text{Def. 53}}{=} \{f(f_1^{[\iota]}(x_1)) \mid f \in f_2^{[\iota]}\} \\
&\overset{\text{Def. 53}}{=} \{f(\{f'(x_1) \mid f' \in f_1^{[\iota]}\}) \mid f \in f_2^{[\iota]}\} \\
&\overset{\text{Theo. 5}}{=} \{f(f'(x_1)) \mid f' \in f_1^{[\iota]}, f \in f_2^{[\iota]}\} \\
&= \{(f \circ f')(x_1) \mid f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}\}
\end{aligned}$$

Theorem 10 *Let $f_1^{[\rho]} = (f_1^{[\iota]}, Pr_{f_1})$ be a probabilistic function of the domain F-All(dom_A, dom_B) and let $f_2^{[\rho]} = (f_2^{[\iota]}, Pr_{f_2})$ be a probabilistic function of the domain F-All(dom_C, dom_D) where $dom_B \subseteq dom_C$. Moreover, let $x \in dom_A$ be a crisp value of domain dom_A. The composition $f_2^{[\rho]} \circ f_1^{[\rho]}$ of both functions is a probabilistic function $f_3^{[\rho]} =$ of the domain F-All(dom_A, dom_D) that is defined as:*

$$\begin{aligned}
f_3^{[\rho]}(x) &= f_2^{[\rho]} \circ f_1^{[\rho]}(x) = z = (y, Pr) \\
\textit{where} \quad & y = \{(f \circ f')(x) \mid f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}\} \\
\textit{and} \quad & \forall x' \in dom_D \colon Pr(x') = \sum\nolimits_{f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}, f(f'(x)) = x'} Pr_{f_1}(f') \times Pr_{f_2}(f)
\end{aligned}$$

Proof 7 *Theorem 10 can be proved by the use of Definition 54, Definition 38, and Theorem 7:*

$$
\begin{aligned}
f_3^{[\rho]}(x_1) &= (f_2^{[\rho]} \circ f_1^{[\rho]})(x_1) \\
&= f_2^{[\rho]}(f_1^{[\rho]}(x_1)) \\
&\overset{\text{Def. 54}}{=} f_2^{[\rho]}(z_2 = (y_2, Pr_2)) \\
&\quad \textit{where} \quad y_2 = \{f'(x_1) \mid f' \in f_1^{[\iota]}\} = f_1^{[\iota]}(x_1) \\
&\quad \textit{and} \quad \forall x \in dom_B : Pr_2(x) = \sum\nolimits_{f' \in f_1^{[\iota]}, f'(x_1)=x} Pr_{f_1}(f') \\
&\overset{\text{Theo. 7}}{=} z_3 = (y_3, Pr_3) \\
&\textit{where} \quad y_3 = \{f(x) \mid f \in f_2^{[\iota]}, x \in y_2\} \\
&\textit{and} \quad \forall x \in dom_D\colon Pr_3(x) = \sum\nolimits_{f \in f_2^{[\iota]}, x' \in y_2, f(x')=x} Pr_2(x') \times Pr_{f_2}(f) \\
&= z_3 = (y_3, Pr_3) \\
&\textit{where} \quad y_3 = \{f(x) \mid f \in f_2^{[\iota]}, x \in IV_2\} \\
&\textit{and} \quad \forall x \in dom_D\colon Pr_3(x) = \sum\nolimits_{f \in f_2^{[\iota]}} \left(Pr_{f_2}(f) \times \sum\nolimits_{x' \in y_2, f(x')=x} Pr_2(x')\right) \\
&\overset{\text{subst. } z_2}{=} z_3 = (y_3, Pr_3) \\
&\textit{where} \quad y_3 = \{f(x) \mid f \in f_2^{[\iota]}, x \in \{f'(x_1) \mid f' \in f_1^{[\iota]}\}\} \\
&\textit{and} \quad \forall x \in dom_D\colon Pr_3(x) = \sum\nolimits_{f \in f_2^{[\iota]}} \left(Pr_{f_2}(f) \times \sum\nolimits_{x' \in \{f'(x_1) \mid f' \in f_1^{[\iota]}\}, f(x')=x} \mathcal{X}\right) \\
&\quad \textit{where } \mathcal{X} = \sum\nolimits_{f' \in f_1^{[\iota]}, f'(x_1)=x} Pr_{f_1}(f') \\
&\overset{\text{Def. 38}}{=} z_3 = (y_3, Pr_3) \\
&\textit{where} \quad y_3 = \{f(f'(x_1)) \mid f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}\} \\
&\textit{and} \quad \forall x \in dom_D\colon Pr_3(x) = \sum\nolimits_{f \in f_2^{[\iota]}} \left(Pr_{f_2}(f) \times \sum\nolimits_{f' \in f_1^{[\iota]}, f(f'(x_1))=x} Pr_{f_1}(f')\right) \\
&= z_3 = (y_3, Pr_3) \\
&\textit{where} \quad y_3 = \{f(f'(x_1)) \mid f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}\} \\
&\textit{and} \quad \forall x \in dom_D\colon Pr_3(x) = \sum\nolimits_{f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}, f(f'(x_1))=x} Pr_{f_2}(f) \times Pr_{f_1}(f') \\
&= z_3 = (y_3, Pr_3) \\
&\textit{where} \quad y_3 = \{(f \circ f')(x_1) \mid f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}\} \\
&\textit{and} \quad \forall x \in dom_D\colon Pr_3(x) = \sum\nolimits_{f \in f_2^{[\iota]}, f' \in f_1^{[\iota]}, f(f'(x_1))=x} Pr_{f_1}(f') \times Pr_{f_2}(f)
\end{aligned}
$$

6.9. Summary

In this chapter we have introduced a formalism that can be used to model and handle uncertainty of data as well as data operations. For that purpose, we first presented the principles of the possible worlds semantics in Section 6.1. Then we introduced the concepts of incomplete values and probabilistic values (both summarized under the concept of uncertain values) as some specific constructs to model uncertainty in Section 6.2. Next, we considered in which way these concepts can be transformed from one into the other (Section 6.3), discussed in which way uncertainty that is modeled at different levels can be combined (Section 6.4), and presented in which way correlations between uncertain values can be

modeled within our formalism (Section 6.5 and Section 6.6). In Section 6.7, we adapted conventional functions to uncertain input values and considered functions that are especially designed to get uncertain input values or functions that are intended to produce uncertain output values. Finally, we extended our considerations to mappings that are inherent uncertain in Section 6.8.

At a glance, the most important concepts that we have introduced in this chapter can be summarized by considering the following simple example (for ease of presentation we use the concept of incomplete values for illustration). Let $f : dom_A \rightarrow dom_B$ be a function that is originally defined to map a crisp value of dom_A to a crisp value of dom_B, let $f^{[\iota]}$ be an incomplete function of the same type, and let y_1 (or x) be an incomplete value (or a crisp value respectively) of domain dom_A. There are three different scenarios in which uncertainty can arise and hence an incomplete output value y_2 is produced. The input data can be uncertain (see Equation 6.3), the function can be uncertain (see Equation 6.4), or both of them can be uncertain (see Equation 6.5).

$$f(y_1) = y_2 = \{f(x) \mid x \in y_1\} \tag{6.3}$$

$$f^{[\iota]}(x) = y_2 = \{f(x) \mid f \in f^{[\iota]}\} \tag{6.4}$$

$$f^{[\iota]}(y_1) = y_2 = \{f(x) \mid f \in f^{[\iota]}, x \in y_1\} \tag{6.5}$$

Whereas, the first scenario can be used to define how conventional data operations have to work on uncertain input data, the second scenario can be used to define how uncertain data operations have to process certain input data. The last scenario presents the combination of both and hence can be used to define how uncertain data operations have to process uncertain input data.

6.10. Measures for Certainty

Sometimes it is useful to measure the certainty (or uncertainty respectively) of an uncertain value. For that reason, we examine existing literature for such measures and discuss some of them in the rest of this section. For ease of presentation, we present them based on the notion of the uncertain value theory.

6.10.1. Number of Alternatives

An intuitve and often used measure to rate the degree of uncertainty of an uncertain value is to use its number of alternatives, because the more alternatives an uncertain value has, the more uncertain that value can be considered to be. Obviously, we strive after a normalized measure of certainty that is 1 if the uncertain value has only a single alternative (and hence is a crisp value) and that is 0 if every element of the corresponding domain is an alternative. We detect two ways to consider the number of alternatives in a normalized measure. Let $y \in^{[\iota]} dom$ be an incomplete value, we define the Ratio of Excluded Alternatives (short *RExA*) and the Inverse Number of Alternatives (short *invNA*) as:

$$\mathit{RExA}(y) = \frac{|dom| - |y|}{|dom| - 1} \tag{6.6}$$

$$\mathit{invNA}(y) = \frac{1}{|y|} \tag{6.7}$$

The Ratio of Excluded Alternatives is based on the idea that the more domain elements are excluded for sure, the less alternatives remain and the greater is the certainty of the considered uncertain value [Pan09a]. The benefit of the Ratio of Excluded Alternatives is that it scales cardinally which increases

the interpretability of the resultant scores [HKK07]. In contrast, the benefit of the Inverse Number of Alternatives is that it is independent of the domain size, because that size can be extremely large in some application areas and in such cases almost all uncertain value are rated by scores close to 1 even if their degree of certainty are different to a large extent.

Both measures have the drawback that they are not able to capture certainty information that is described by the alternatives' probabilities. Moreover, these measures can become unrepresentative if the considered incomplete value can be decomposed into several mutually independent incomplete values, because increasing the number of alternatives of one of the independent values from n to $n+1$ increases the number of alternatives of the composed value by $\frac{n+1}{n}$

6.10.2. Shannon Entropy

A commonly used measure for uncertainty in information theory is the *Shannon entropy* [Tan01, Sin13] that rates the amount of uncertainty of a discrete random variable. For a random variable X with a finite number of possible outcomes $\{x_1, \ldots, x_k\}$ and a probability mass function Pr, the Shannon entropy is defined as:

$$H(X) = \sum_{i=1}^{k} Pr(x_i) \times h(x_i)$$

where $h(x_i)$ can be interpreted as the uncertainty that is associated with the event that leads to $X = x_i$. The value $h(x_i)$ is often also referred to as the information content of x_i (or its corresponding event respectively) and hence is often denoted as $I(x_i)$. The information content of a possible outcome x_i is in turn defined as $h(x_i) = C \times \log_b\left(\frac{1}{Pr(x_i)}\right) = -C \times \log_b\left(Pr(x_i)\right)$ where $C > 0,\ b > 1$. The variables C and b are usually set to $C = 1$ and $b = 2$ (other common settings of b are: $b = e$ (euler's number) or $b = 10$). Using this definition of $h(x_i)$ and using the setting $C = 1$ and $b = 2$, the entropy of X can be also described as:

$$H(X) = -\sum_{i=1}^{k} Pr(x_i) \times \log_2(Pr(x_i))$$

The entropy is maximal if the probability mass function is a uniform distribution on the variable's corresponding domain and hence if all outcomes are equally likely [Sin13]. Let X be a random variable that is defined on a domain with N elements, the entropy of X is maximal if it models a uniform distribution on all N elements. Thus, the maximal entropy of X is defined as:

$$H_{max(X)} = -\sum_{i=1}^{N} \frac{1}{N} \times \log_2\left(\frac{1}{N}\right) = \log_2(N)$$

The normalized entropy of the random variable X consequently results in:

$$H_{norm}(X) = \frac{H(X)}{H_{max(X)}} = -\sum_{i=1}^{k} Pr(x_i) \times \frac{\log_2(Pr(x_i))}{\log_2(N)} = -\sum_{i=1}^{k} Pr(x_i) \times \log_N(Pr(x_i))$$

As described in Section 6.2.1, a probabilistic value can be considered as a discrete random variable. Consequently, the entropy can be used to rate the uncertainty of a probabilistic value. Uncertainty has the inverse meaning of certainty. Therefore, let $z \in^{[\rho]} dom$ be a probabilistic value with $z = (y, Pr)$ and $N = |dom|$, the certainty of z can be rated by its Inverse Normalized Entropy:

$$invH_{norm}(z) = 1 - H_{norm}(z) = 1 + \sum_{x \in y} Pr(x) \times \log_N(Pr(x))$$

6.10.3. Uncertain Density / Answer Decisiveness

In [dKvK07a] de Keijzer and van Keulen propose two normalized measures to rate the uncertainty of an uncertain data set. Whereas the first measure *Uncertainty Density* is defined for incomplete data, the second measure *Answer Decisiveness* is defined for probabilistic data. Both measures are based on the concepts of *choice points*. In this context, a choice point is considered as a data construct that models several mutual exclusive events. For example, in a set of mutual independent uncertain values, each value serves as an individual choice point, because each of them has several alternatives.

Let S_{cp} be a set of choice points, let $N_{cp} = |S_{cp}|$ be the number of choice points, let $N_{poss,cp}$ be the number of alternatives of choice point $cp \in S_{cp}$, and let Pr_{cp}^{max} be the probability of the most probable alternative of choice point $cp \in S_{cp}$, the Uncertainty Density (short *UDens*) and the Answer Decisiveness (short *ADec*) of S_{cp} are defined as:

$$UDens(S_{cp}) = 1 - \frac{1}{N_{cp}} \sum_{i=1}^{N_{cp}} \frac{1}{N_{poss,i}} \tag{6.8}$$

$$ADec(S_{cp}) = \frac{1}{N_{cp}} \sum_{i=1}^{N_{cp}} \frac{Pr_{cp}^{max}}{(2 - Pr_{cp}^{max}) \times \log_2 \left(\mathbf{max}(2, N_{poss,i})\right)} \tag{6.9}$$

The Uncertain Density is 0 in the case of absolute certainty and is 1 in the case of absolute uncertainty. In contrast, the Answer Decisiveness is 1 in the case of absolute certainty and is 0 in the case of absolute uncertainty. Therefore, Answer Decisiveness already rates certainty, but Uncertain Density rates uncertainty. Nevertheless, since Uncertain Density is normalized, we can use the inverse of Uncertain Density to rate certainty. We call this measure Certain Density (short *CDens*) in the remainder of this thesis. Since we define Certain Density as the inverse of Uncertain Density, it holds: $CDens(S_{cp}) = 1 - UDens(S_{cp})$. Note, for an input set with only a single choice point, e.g. a single uncertain value, Certain Density is equivalent to the Inverse Number of Alternatives.

It is important to note that in contrast to the entropy, the Answer Decisiveness only takes the probability of the most probable alternative and the overall number of alternatives into account. The idea behind that approach is that the greater this probability is and the less the number of alternatives is, the more certain is the data represented by the corresponding choice point. Nevertheless, as we think it is not inessential how probable the other alternatives are, because a choice point that has one alternative with probability 0.5 and has ten other alternatives each with a probability of 0.05 seems not be equivalently certain as a choice point that has one alternative with probability 0.5, has one alternative with probability 0.4 and has ten alternatives with probability 0.01. Whereas the entropy captures such a difference, the Answer Decisiveness does not.

De Keijzer and van Keulen define both measures with regard to entity independent uncertain data where each database entity (XML element or ULDB x-tuple) can be considered as an individual choice point. In the presence of dependencies, however, considering each entity as an individual choice point implicates to ignore the entity dependencies in measuring uncertainty. Nevertheless, since dependencies can reduce uncertainty a lot, such an ignorance can distort the rated quality score to a large extent. Moreover, if we incorporate these dependencies into the computation process by considering the set of correlated values as a single choice point, the measured uncertainty increases compared to the case of

independence although the number of possible combination of value alternatives decreased. It is obvious that such a behavior is counterintuitive and does not reflect the expected meaning of certainty.

Example 96 *For illustration, consider two incomplete values each with two alternatives. In the case of independence, we have four possible combinations of the values' alternatives, but also have two choice points each with two choices. Thus, the Certain Density of both values results in* 0.5*. Now let us assume that the two values are correlated and the first alternative of the first value implicates the first alternative of the second value. Thus, the dependency decreases the number of possible combinations from four to three. As a consequence, assigning the same Certain Density to these values However, in this case the values are not independent anymore and therefore cannot be considered as separate choice points. In contrast, we need to consider them as a single choice point that has three alternatives. As a consequence, the Certain Density of both values results in* $1/3$ *which is less than their Certain Density has been in the case of independence although the number of possible combinations decreased.*

6.10.4. Precision

A different but interesting certainty measure is the *Precision* of measurements[3]. In the ISO/WD 15725-1 standard the precision of a measurement is defined as the:

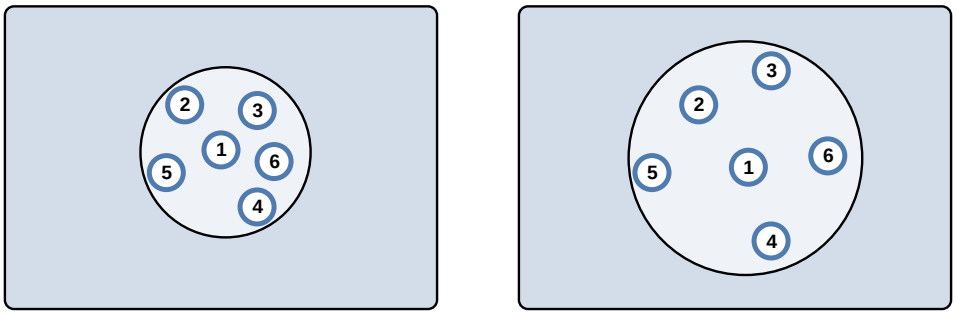
> "*closeness of agreement between independent test/measurement results obtained under stipulated conditions*" [ISO11]

It is obvious that we can adopt this principle from measurements to uncertain values because the closer the alternatives of an uncertain value are, the more information they have in common and consequently the greater is the amount of the value's information that can be considered to be certain.

In contrast to the certainty measures presented so far, Precision focus on a different aspect of certainty. Whereas the Number of Alternatives, the Shannon Entropy, the Certain Density and the Answer Decisiveness can be considered as measures that rates the certainty of an uncertain value (or a set of uncertain values) based on the certainty of *point query* answers, Precision can be considered as a measure that focus on *range query* answers. As a *point query* we consider a predicate that checks whether or not a single domain element is the true alternative of the queried uncertain value, e.g. the predicate $u = x$ that compares an uncertain value u with a crisp value x is a point query. In contrast, we consider a *range query* to be a predicate that checks whether or not the true alternative of the queried uncertain value is within a specific range, e.g. the predicate $u \leq x$ is a range query. In general, the closer the alternatives of an uncertain value are, for more range queries a certain result can be returned.

Example 97 *We will illustrate the difference of both concept on a simple example. Let* $y_1 = \{1, 2, 4\}$*,* $y_2 = \{1, 5, 7\}$*, and* $y_3 = \{1, 2, 3, 4\}$ *be three incomplete values of the domain* $dom = \{1, \ldots, 10\}$*. Considering point queries,* y_1 *as well as* y_2 *return a certain query result for seven elements, but* y_3 *can only return a certain query result for six elements. Consequently,* y_1 *and* y_2 *can return certain results for more point queries than* y_3 *and indeed by using the measures RExA, invNA, or CDens the values* y_1

[3] Not to be confused with the precision of a binary classification that is typically used to rate the quality of duplicate detection results (see Section 4.3.8).



(a) High precise set (b) Less precise set

Figure 6.2.: The generalized interpretation of range query certainty

and y_2 are considered to be more certain than y_3. On the other hand, the values y_1 as well as y_3 can return a certain result for all range queries that check whether or not the true alternative is greater than 4 which are twenty one in numbers, but the value y_2 can return a certain result only for range queries that check whether or not the true alternative is greater than 7 which are six in numbers. Thus, under this aspect, the value y_3 can be considered to be more certain than the value y_2. A similar example can be constructed for probabilistic values.

It is essential to note that range queries are not exclusive for ordered domains, but they can be defined for any domain on which a similarity measure or a distance measure can be defined. That is because each range query can be generalized to a predicate of the form $\theta_1 \geq dst(u, x) \leq \theta_2$. For example, the predicate $u > 4$ that is defined on the domain $dom = \{1, \ldots, 10\}$ can be formulated as $dst(u, 10) < 4$, if the distance between two domain elements x_1 and x_2 is defined as $dst(x_1, x_2) = |x_1 - x_2|$. This generalized interpretation of range queries is simplified illustrated in Figure 6.2 where the dark blue colored area represents the range for which a certain query result can be returned.

A set is considered to be more precise the closer its elements are. Thus, the meaning of Precision is not clearly defined, because the term 'closeness' is vague by nature. One option is to define the Precision of a set as the aggregated closeness between each pair of set elements. Another option is to compute a centroid and then to define the Precision of a set as the aggregated closeness of all set elements to that centroid. For aggregation purposes several functions such as the average value, the minimal value, or the maximal value are conceivable. For rating the closeness between two domain elements we can use any similarity measure (or distance measure respectively) that is defined on the corresponding domain.

We can simply adopt the concept of Precision to rate the certainty of an incomplete value by considering the value's alternatives as the set's elements and can simply adopt it to rate the certainty of a probabilistic value by replacing the average function by the expected value function (of course, the aggregation functions **min** and **max** can be used as well) and by considering probabilities for computing the centroid.

For instance, let $z \in^{[\rho]} dom$ be a probabilistic value with $z = (y, Pr)$, and let sim be a similarity measure that is defined on elements of dom, we define the *Expected Pair Precision* of z as:

$$EPPrec(z) = \sum_{x_i \in y} \sum_{x_j \in y} Pr(x_i) \times Pr(x_j) \times sim(x_i, x_j) \tag{6.10}$$

This in turn corresponds to the expected value of the probabilistic similarity score that result from using the probabilistic version of sim to compare z with itself, i.e. $\mathit{EPPrec}(z) = \mathit{EXP}(sim(z, z))$.

Chapter 7

Deterministic Duplicate Detection in Uncertain Databases

In this Chapter, we present our research findings on deterministic duplicate detection in uncertain databases. We start with presenting some challenges that we need to face for guaranteeing a meaningful and effective duplicate detection result in Section 7.1. Since deterministic detection results need to be accomplished by resolving the uncertainty of the input database at some moment of the detection process, we discuss aggregation methods for several domains in Section 7.2. Then we propose two detection approaches. Whereas the world-based detection approach (Section 7.3) follows the underlying idea of the possible worlds semantics and therefore performs a conventional duplicate detection process in some of the possible worlds separately, the description-based detection approach (Section 7.4) compares database entities by matching probabilistic entity descriptions. Section 7.5 is devoted to the challenge of detecting scattered duplicates and introduce the Probabilistic Monge-Elkan Similarity as an efficient and effective solution. A cost optimization of matching probabilistic entity descriptions is considered in Section 7.6. Section 7.7 introduces two preparation activities that help to overcome the negative effects of entity correlations and/or poor schema designs. In Section 7.8 we propose an approach for incorporating process uncertainty into the detection process. Finally, Section 7.9 provides a theoretical comparison of the world-based detection approach and the description-based detection approach.

7.1. Problem Description

As in certain databases, duplicate detection in uncertain databases is a process that identifies database entities referring to the same real-world entity. We consider the concept of indeterministic duplicate detection in this thesis (Chapter 8) and therefore propose approaches that use the modeling power of uncertain databases in order to incorporate the uncertainty of ambiguous duplicate decisions into the detection result. In many real-life scenarios, however, using an indeterministic duplicate detection approach is not an option, because a lot of commonly used representation systems as TI-databases or BID-databases are not powerful enough to model indeterministic deduplication results without the additional use of views. Since it is often required that the deduplicated database is represented within the same system as the input database, the considered representation system need to be closed under the applied deduplication

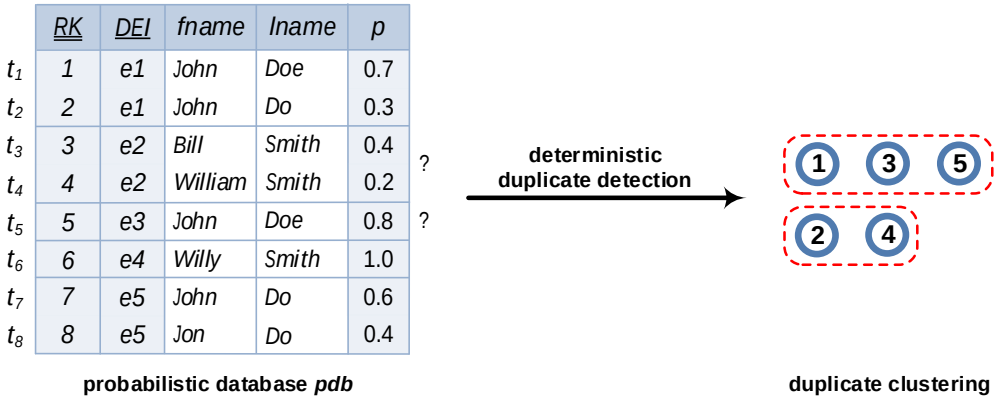


Figure 7.1.: Illustrating example for deterministic duplicate detection in probabilistic data

process. For that reason, we consider deterministic duplicate detection for uncertain databases in this thesis as well.

Recall from Section 3.4 that the set of all possible database entities of an uncertain database (incomplete or probabilistic) *udb* is denoted as $Ext_{Poss}(udb)$. Formally, a process for deterministic duplicate detection in an uncertain database is defined as it is defined for a certain database (see Definition 22), i.e. it is a functions that partitions the considered set of database entities into duplicate clusters. Thus, let *udb* be an uncertain database, a deterministic duplicate detection process performed on this database maps the set $Ext_{Poss}(udb)$ to a cluster-disjoint clustering of $Ext_{Poss}(udb)$[1].

Example 98 *For illustration, we consider the probabilistic BID-database presented on the left side of Figure 7.1. This database consists consists of a single table and contains five database entities. Since each of these entities is represented by a single x-tuple, we can consider entities and x-tuples synonymously in this example. Some database entities (e_1, e_2, and e_5) have multiple alternatives, e.g. the entity e_1 is represented by the alternatives t_1 and t_2, and some entities (e_2 and e_5) are only maybe, i.e. the membership of these entities to the considered table is uncertain. It seems intuitive that the entities e_1, e_3, and e_5 as well as the entities e_2 and e_4 represent the same person and hence are duplicates. Thus, we construct the duplicate clustering that is depicted on the right side of Figure 7.1. This clustering contains two clusters: One with e_1, e_3, and e_5, and one with e_2 and e_4.*

As we will show in the remainder of this section, a deterministic detection of duplicates in uncertain databases is not straightforward and poses some interesting challenges. The first and most obvious challenge is the representation heterogeneity discussed in the following subsection. Representation heterogeneity is a common problem in processing uncertain databases and is usually solved by utilizing the

[1] Due to we cannot conclude certain information on non-membership from the stored information on membership if we use the *Open World Assumption* [Rei77], some applications (e.g. data integration [ASUW10]) assume that uncertain database systems manage a separate entity set that contains all the database entities the database has information on. This set can also contain database entities on which we are sure that they do not belong to the considered universe of discourse and hence can contain database entities that do not belong to any of the database's possible worlds, e.g. a person that is not a student anymore. In that case, the stored set of entities $Ext_*(udb)$ must be a superset of all possible entities, i.e. $Ext_*(udb) \supseteq Ext_{Poss}(udb)$. Of course, if we do not have any information on these entities except their *DEI*s, a consideration of them in the duplicate detection process makes only sense if we also detect duplicates across multiple databases and therefore can use the given information on non-membership to meaningfully combine data from several sources.

possible worlds semantics for process definition. Defining deterministic duplicate detection according to the possible worlds semantics, however, causes some further challenges. Since all these challenges are interrelated and some possible solutions to one challenge pose other challenges, we start with representation heterogeneity and then proceed in an order that we think reflects the interdependencies between the individual challenges best.

7.1.1. Representation Heterogeneity

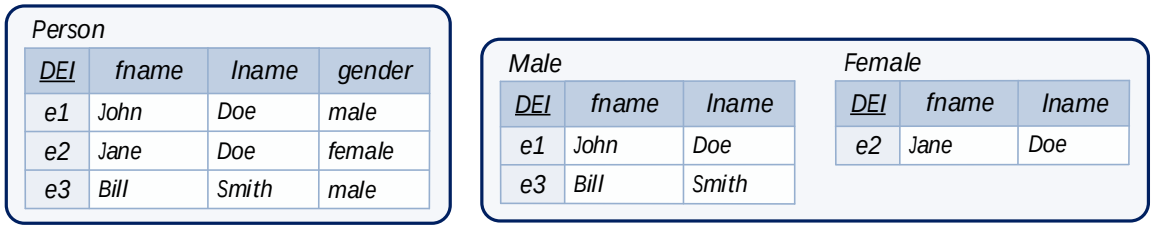
As presented in Section 3.3, a variety of representation systems has been proposed for uncertain databases in recent years. Nevertheless, the goal of this thesis is not to define a specific duplicate detection approach for each of them, but we strive for an approach that can work on a database that is represented in any of these systems and always produces the same result if same input information is given. As a consequence, we have to deal with the problem of *Representation Heterogeneity*, i.e. same information can be represented in different ways. Sometimes, representation heterogeneity is even given within a single representation system because some systems enable a modeling of same information in several way (even if a fixed world schema is predefined).

In general, cases of representation heterogeneity can be divided into two categories:

- The first category considers cases in which same information is represented by different world schemas.

  Example 99 *For demonstration, we consider the same sample that we already have used in Section 3.4 to illustrate a transformation from value uncertainty into membership uncertainty. Therefore, we consider a database db_1 whose world schema contains the single table* 'Person' *having the three attributes* 'fname', 'lname', *and* 'gender'. *By assuming that each person is either a woman or a man, this database can be transformed into a database db_2 whose world schema contains the two tables* 'Male' *and* 'Female' *each having the two attributes* 'fname' *and* 'lname'. *Even the world schemas of both databases are different, the transformation is lossless, i.e. for each possible instance of the first database it exists an instance of the second database that is different in the data, but represents the same information. This case of representation heterogeneity is illustrated in Figure 7.2.*

  Representation heterogeneity that is caused by different world schemas is also a problem for duplicate detection in certain databases. In certain databases, the concept of entity descriptions helps to overcome schema heterogeneity and thus enables a design of duplicate detection processes that are representation independent, i.e. these processes are defined on the extracted descriptions and therefore abstract from the way by which these descriptions are extracted from the databases. Note, for extracting uniform descriptions from databases with heterogeneous schemas, concepts of data integration can be adopted. One way to resolve schema heterogeneity in data integration is to define mappings from the local schemas to a user defined global schema [RB01, HMH01, BBR11] whereas in our case the global schema corresponds to the considered description type. A another way is the utilize a multidatabase query language like SchemaSQL [LSS96a, LSS99, LSS01] that enables an incorporation of schema properties into query formulation.



Person

DEI	fname	lname	gender
e1	John	Doe	male
e2	Jane	Doe	female
e3	Bill	Smith	male

(a) Database db_1

Male

DEI	fname	lname
e1	John	Doe
e3	Bill	Smith

Female

DEI	fname	lname
e2	Jane	Doe

(b) Database db_2

Figure 7.2.: Sample for representation heterogeneity that is caused by heterogeneous schemas

- The second category of representation heterogeneity considers cases in which same world schemas are used, but the way of uncertainty representation differs, i.e. the possible worlds representations are equal, but the compact representations are not. For example, an A-tuple of an AOR-database can be also represented by an x-tuple of a BID-database as we already have pointed out in Section 3.3.4.

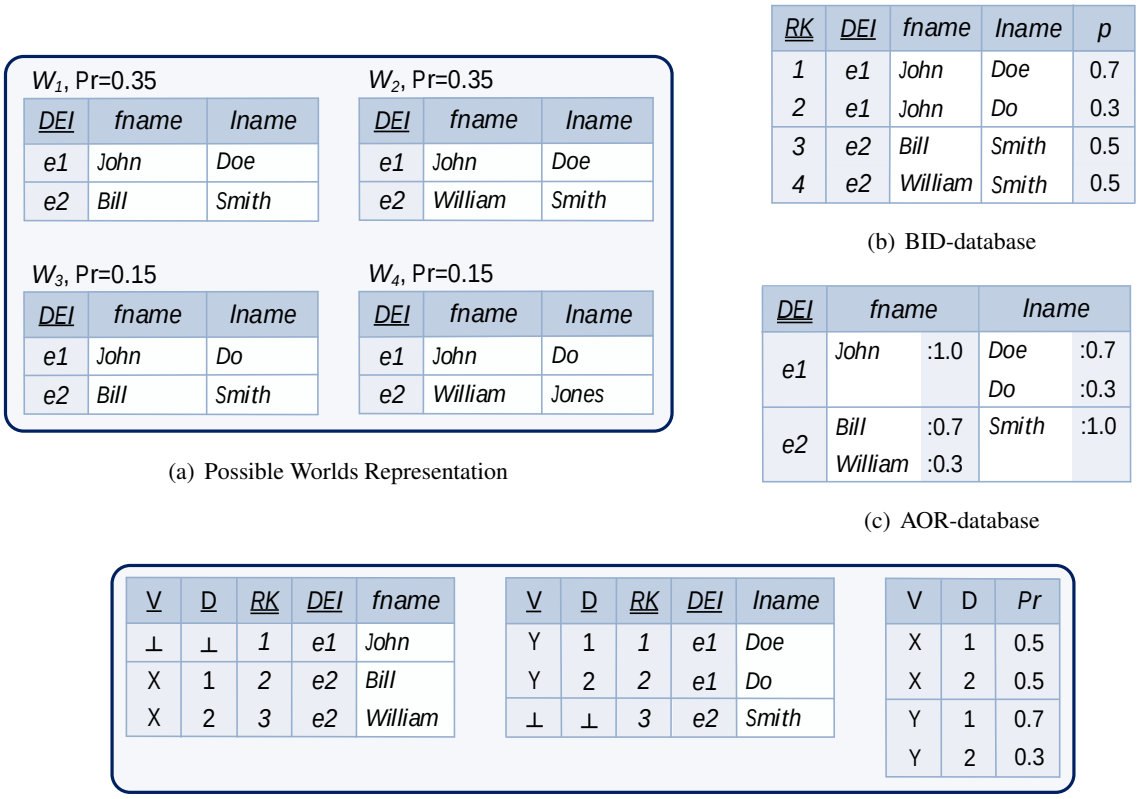
  Example 100 *For illustration we consider the possible worlds representation that is presented in Figure 7.3(a). By using a BID-database this probability distribution on four possible worlds can be represented as shown in Figure 7.3(b). In contrast, an AOR-database can compactly represent this world space as shown in Figure 7.3(c). Finally, the same possible worlds can be represented by an U-database (MayBMS) as presented in Figure 7.3(d).*

In the presence of schema heterogeneity, the sets of possible worlds differ from each other and representation heterogeneity is only hard to resolve by the duplicate detection process itself (it is important to note that same worlds implies same information, but same information does not necessarily imply same worlds). Moreover, this problem is also present in the handling of certain databases and is not a particular problem for uncertain databases. For that reason, we consider a resolution of representation heterogeneity only for fixed world schemas and assume that the heterogeneity of world schemas is resolved by using a schema-specific function for extracting entity descriptions.

All representation systems for uncertain databases have in common that their data can be transformed into the possible worlds representation. Thus, we denote a data operation to be *representation independent*, if its result only depends on the possible worlds representation of the input database and does not depend on the system this database is actually represented with.

Definition 55 (Representation Independence): *Let $\mathfrak{S}_{Rep}$ be the set of all representation systems that are defined for compactly modeling uncertain databases, let $\mathfrak{R}^{db}$ be the set of all representations that are possible in system $\mathfrak{R} \in \mathfrak{S}_{Rep}$, let ϱ be an operation that is defined for uncertain databases. The operation ϱ is called to be representation independent if the result of applying ϱ to an uncertain database only depends on the modeled possible worlds, but does not depend on the system these worlds are actually represented with. Thus, ϱ is representation independent, iff there exists a semantically equivalent implementation $\varrho^{\mathfrak{R}}$ for every representation system $\mathfrak{R} \in \mathfrak{S}_{Rep}$ so that:*

$$\forall \mathfrak{R}_1, \mathfrak{R}_2 \in \mathfrak{S}_{Rep}: \quad \forall udb_1 \in \mathfrak{R}_1^{db}: \quad \forall udb_2 \in \mathfrak{R}_2^{db}:$$
$$pwr(udb_1) = pwr(udb_2) \quad \Rightarrow pwr(\varrho^{\mathfrak{R}_1}(udb_1)) = pwr(\varrho^{\mathfrak{R}_2}(udb_2))$$



W_1, Pr=0.35

DEI	fname	lname
e1	John	Doe
e2	Bill	Smith

W_2, Pr=0.35

DEI	fname	lname
e1	John	Doe
e2	William	Smith

W_3, Pr=0.15

DEI	fname	lname
e1	John	Do
e2	Bill	Smith

W_4, Pr=0.15

DEI	fname	lname
e1	John	Do
e2	William	Jones

(a) Possible Worlds Representation

RK	DEI	fname	lname	p
1	e1	John	Doe	0.7
2	e1	John	Do	0.3
3	e2	Bill	Smith	0.5
4	e2	William	Smith	0.5

(b) BID-database

DEI	fname		lname	
e1	John	:1.0	Doe	:0.7
			Do	:0.3
e2	Bill	:0.7	Smith	:1.0
	William	:0.3		

(c) AOR-database

V	D	RK	DEI	fname
⊥	⊥	1	e1	John
X	1	2	e2	Bill
X	2	3	e2	William

V	D	RK	DEI	lname
Y	1	1	e1	Doe
Y	2	2	e1	Do
⊥	⊥	3	e2	Smith

V	D	Pr
X	1	0.5
X	2	0.5
Y	1	0.7
Y	2	0.3

(d) U-database (MayBMS)

Figure 7.3.: Sample for representation heterogeneity that is caused by a heterogeneous modeling of uncertainty within in the database

Thus, a data operation ϱ is representation independent, if

$$\forall \mathfrak{R} \in \mathfrak{S}_{Rep} \colon \forall udb \in \mathfrak{R}^{db} \colon \varrho(pwr(udb)) = pwr(\varrho^{\mathfrak{R}}(udb))$$

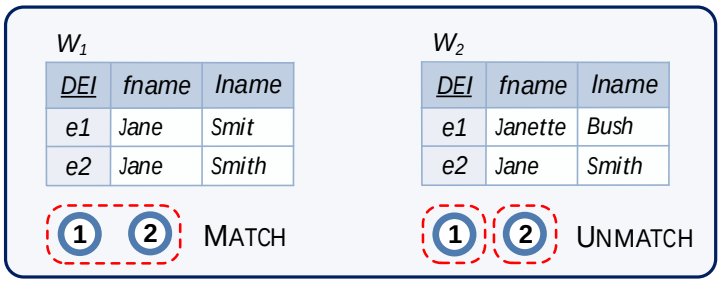
In uncertain data processing, the representation independence of a data operation is commonly guaranteed by first defining its semantics based on a naive implementation that gets the possible worlds representation as input and then by implementing a system-specific version per representation system that is semantically equivalent to the naive implementation. Similar to the made distinction of database queries in Section 3.2, we have to distinguish between two classes of operations.

- Operations of the first class explicitly work on the uncertainty of the database (e.g. an operation that reduces uncertainty by discarding possible worlds or an operation that computes the uncertainty of the database) and hence has no equivalent operation in certain data processing. Operations of this class are considered to get a possible worlds representation as input, but are intended to combine the data of the individual worlds at some processing step. The class of *Uncertainty Processing Database Queries* that we have presented in Section 3.2 is a subclass of this class of data operations.

- In contrast, operations of the second class each have an equivalent operation in certain data processing. Thus, these operations can be considered as a result of adapting its certain data equivalent

RK	DEI	fname	lname
1	e1	Jane	Smit
2	e1	Janette	Bush
3	e2	Jane	Smith

(a) BID-database *idb*



(b) Possible world space $\{W_1, W_2\}$ of *idb*

Figure 7.4.: Problem of contrary duplicate decisions

to uncertain data processing and hence its underlying semantics is originally defined with no uncertainty in mind. In this case, the semantics of an uncertain data operation is traditionally defined according to the possible worlds semantics (Section 3.2). Thus, performing an uncertain data operation on a set of possible worlds is defined to be semantically equivalent to perform its certain data equivalent in each of these possible worlds separately. The class of *Conventional Database Queries* and the class of *Uncertainty Including Database Queries* that we have presented in Section 3.2 are subclasses of this operation class.

In our situation, we indeed have a certain data equivalent namely a deterministic duplicate detection process as described in Chapter 4. Therefore, it seems intuitive to define the semantics of a deterministic duplicate detection approach δ^U_{det} on an uncertain database by performing its certain data equivalent δ_{det} on each of the database's possible worlds separately. Thus, let *pdb* a probabilistic database with the possible worlds representation $(\mathbf{W}, Pr)$, the output of applying δ^U_{det} on *pdb* is defined as the probability space $\delta^U_{\text{det}}(pdb) = (\mathbf{W}_\delta, Pr_\delta)$ with:

$$\mathbf{W}_\delta = \{\delta_{\text{det}}(W) | W \in \mathbf{W}\} \tag{7.1}$$

$$\text{and} \quad \forall W \in \mathbf{W}_\delta \colon Pr_\delta(W) = \sum\nolimits_{W' \in \mathbf{W}, \delta(W') = W} Pr(W')$$

7.1.2. Contrary Duplicate Decisions

In Equation 7.1 we define the semantics of deterministic duplicate detection in uncertain databases based on the possible worlds semantics. Processing duplicate detection in each possible world separately, however, can lead to a set of contrary decisions on the same pair of database entities, because an entity pair can be classified as a MATCH in one world and can be classified as an UNMATCH in another world.

Example 101 *This problem should be illustrated by the BID-database presented in Figure 7.4. The incomplete database contains two entities e_1 and e_2 that are each modeled by an x-tuple with two alternatives or one alternative respectively (see Figure 7.4(a)). The possible world space of this database contains the two possible worlds that are depicted in Figure 7.4(b). Whereas the two entities are classified as a* MATCH *in world W_1 they are classified as an* UNMATCH *in world W_2.*

As illustrated by this simple example, we cannot interpret the probability space $\delta^U_{\text{det}}(pdb) = (\mathbf{W}_\delta, Pr_\delta)$ as a deterministic duplicate detection result in many application scenarios. For that reason, using the

possible worlds semantics to define a semantical reference for duplicate detection in uncertain data is not an option if we strive for a deterministic duplicate detection result. This conflict is obvious because using the possible worlds semantics means to push uncertainty of the operation input instantly to the operation output without resolving it. To our purpose, however, we need a certain detection result and therefore need to resolve the uncertainty of the input data within the operation.

As a consequence, although there is a certain data equivalent available we have to consider deterministic duplicate detection as an operation of the first class that is allowed to combine the data from the individual worlds. By doing so, the problem of contrary decisions can be solved by finally aggregating them to a single decision (pairwise decisions or clustering). In general, aggregation can be performed at six different moments of an iterative duplicate detection process:

- aggregation of the database's possible worlds
- aggregation of the possible descriptions per entity
- aggregation of the possible feature scores per candidate pair
- aggregation of the possible similarity scores per candidate pair
- aggregation of the possible duplicate decisions per candidate pair
- aggregation of the possible duplicate clusterings

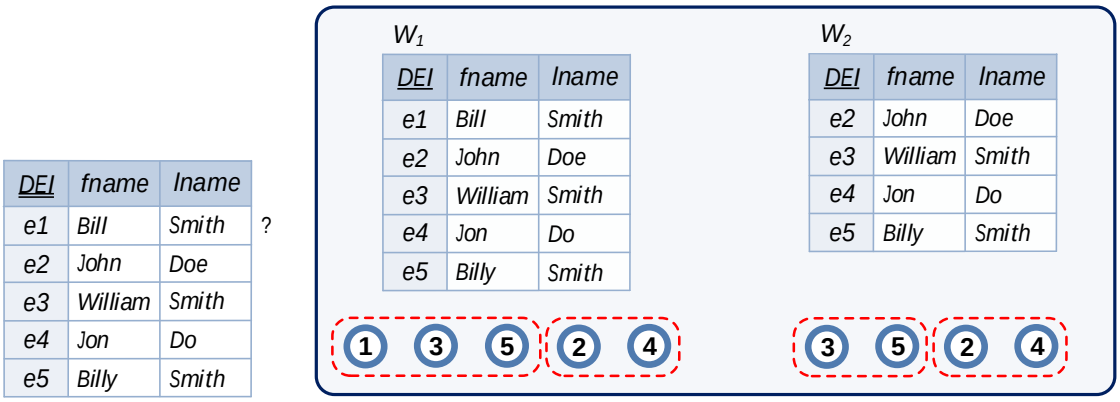
We will discuss aggregation methods for the different moments (and hence the different input domains) in Section 7.2.

Notice, a similar approach of pairwise aggregation has been proposed by Lian and Chen for set similarity join processing in uncertain data [LC10, LC11b], because they consider two uncertain sets to satisfy the join condition, if they satisfy the join condition in a set of worlds having an accumulative probability greater than a specific threshold. Thus, designing uncertain data operations that release from using the possible worlds semantics as a semantical reference is not unusual in uncertain data processing and has been already used in previous proposals. However, the procedure proposed by Lian and Chen is applicable for similarity join computation, but sets up some essential challenges for duplicate detection as we will discuss in the following sections.

7.1.3. Handling of Membership Uncertainty

We observe four challenges for deterministic duplicate detection that emerge from the uncertainty on the membership of a database entity to a specific database table:

- Some worlds can produce incomplete detection result, i.e. some entities are missing in the duplicate clustering that is finally produced for each of these worlds.
- The influence of the membership information on the duplicate detection result is often much stronger than useful.
- Incorrect dependencies between attributes (or relationship roles) and table memberships can increase the negative influence of membership information on the detection result further on.
- Due to incorrect dependencies, the exact interpretation of membership information is sometimes not clear.



(a) TI-database *idb* (b) Possible world space $\{W_1, W_2\}$ of *idb*

Figure 7.5.: Problem of incomplete detection results

In the following we present all four problems in detail and point out some intuitive solutions for each of them.

7.1.3.1. Incomplete Detection Results

Due to the membership of a database entity to a specific database table can be uncertain, the different worlds of a possible world space usually contain different sets of entities, i.e. some entities are missing in some worlds. Obviously, a duplicate clustering only contains entities that are present in its corresponding world. Therefore, in worlds were some entities are missing, the final clustering is incomplete, i.e. some of the considered database entities do not belong to any of its clusters. Similar holds for pairwise computed matching results, because it can happen that no feature score, no similarity score, and no duplicate decision is available for an entity pair in some worlds.

Example 102 *For illustration, we consider the TI-database presented in Figure 7.5(a). This incomplete database contains five database entities each represented by another tuple. Whereas the first tuple is only maybe, the other tuples do not contain any uncertainty. Thus, this database represents the two possible worlds that are depicted in Figure 7.5(b). The first world contains all entities and the duplicate detection process applied to this world produces a final clustering with two clusters: One with the entities* e_1*,* e_3 *and* e_5*, and one with the entities* e_2 *and* e_4*. In contrast, in world* W_2 *the first entity is missing and the final duplicate detection result is only a clustering on the remaining four entities where again* e_3 *and* e_5 *as well as* e_2 *and* e_4 *belong to the same cluster.*

The challenge that emerge from this circumstance is how to aggregate detection results that are not complete, e.g. how to aggregate two clusterings that are defined on different sets of entities or how to define the duplicate decision on two entities if one of them (or even both) is missing in some worlds.

We identify two intuitive solution for this problem: The first solution is to guarantee a complete result for every world from the start and the second solution is to utilize aggregation functions (for worlds, feature scores, similarity scores, etc.) that can work on incomplete detection results.

- Complete detection results can be simply ensured by defining a fixed result for all missing entity pairs or entities respectively. A straightforward solution that is used in the aforementioned similarity join approach proposed by Lian and Chen [LC11b, LC10] is to define the similarity between two entities as zero, if one of them is missing in the considered world. By doing so, each missing entity pair is associated with an empty feature score, is automatically classified as an UNMATCH and each missing entity finally forms a duplicate cluster by its own.

  On one hand defining the similarity of each missing entity pair to zero seems to be intuitive. On the other hand, until the moment of aggregation we consider each possible world separately and therefore assume that the duplicate detection process that works on a specific world has only information available that is stored in this world. In this case, the only information on missing entities is that the entities are missing and therefore likely do not belong to the considered universe of discourse. By assuming zero similarity between two missing entities, however, we conclude that the entities have no resemblance although we do not have any information on them. Such a conclusion seems extreme counter-intuitive. Moreover, this assumption leads to a set of further problems as we will discuss in the following sections.

- The second way of solution is to allow incomplete detection results in the aggregation input and hence to use a method for meaningfully aggregating results that are defined on different sets of entities (or entity pairs respectively). Such an aggregation is hard to define for the domains of possible worlds and clusterings. In contrast, if we consider an aggregation of pairwise results, e.g. feature scores, similarity scores, or duplicate decisions, we can simply distinguish between worlds in which the considered entities have been compared and worlds in which these entities have not been compared because one of them (or both) is missing. Thus, we can restrict the actual aggregation process to the worlds in which both entities are present. Nonetheless, by doing so any information on membership uncertainty is ignored by the detection process and we need to define a fixed aggregation output for all entities that do not coexist in any of the possible worlds.

It can be simply seen that both solutions are contrary and represent two extreme approaches. In the first solution we consider the non-coexistence of two entities in a possible world as a strong indication that they are non-duplicates and in the second solution we completely ignore membership uncertainty in decision making. For that reasons, we propose an approach for handling incomplete detection result in Section 7.2.1 that is based on the concept of impact values and that is a mix of the both aforementioned solutions. Moreover, it enables a trade-off between the two presented extremes and hence can be adjusted to individual user requirements.

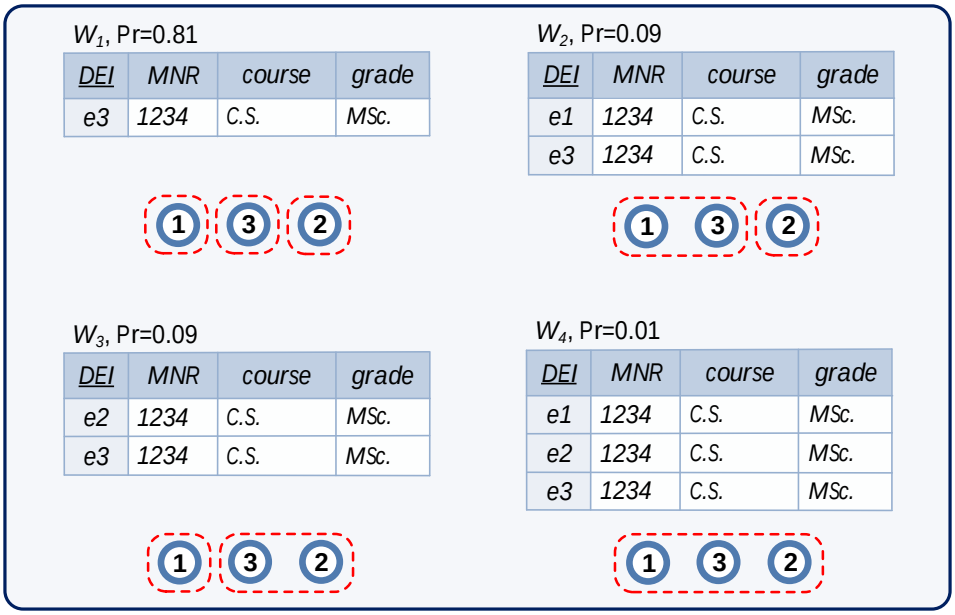
7.1.3.2. Influence of Membership Uncertainty on Duplicate Detection Results

Defining the similarity between two entities as zero if at least one of them is missing leads to the problem that membership information can dominate attribute value information as well as relationship information and hence has too much influence on the final detection result.

Example 103 *We will illustrate such a domination by the example presented in Figure 7.6. Figure 7.6(a) shows a TI-database pdb containing the table* 'Student' *that has a schema with the three attributes*

DEI	MNR	course	grade	p
e1	1234	C.S.	MSc.	0.1
e2	1234	C.S.	MSc.	0.1
e3	1234	C.S.	MSc.	1.0

(a) TI-database *pdb*



(b) Possible worlds representation of *pdb* with the most likely detection results

Figure 7.6.: Sample that illustrates the large influence of uncertain table membership on duplicate detection results

'MNR' *(matriculation number),* 'course' *and* 'grade' *as well as contains the three database entities* e_1, e_2, *and* e_3. *Whereas the last entity is definitely a student, the first two are probably not. Nevertheless, the attribute values of all entities are equal. At this point, it is important to note that we restrict to attributes that are exclusive for students and therefore do not use attributes like* 'firstname' *and* 'lastname' *by purpose, because we intend a clear differentiation between this problem and the problem discussed in Section 7.1.3.3.*

Let us assume that all three entities are duplicates, i.e. $\omega(e_1) = \omega(e_2) = \omega(e_3)$ *and let us assume that the differences in membership probability are caused by the different times these entities have been inserted into the database. For example, at the moment when database entity* e_3 *was inserted into the database, it was certain that the corresponding person* $\omega(e_3)$ *studies (maybe she was close to her final degree), but at the time the entities* e_2 *and* e_3 *find their way in the database this certainty was not given anymore, but instead it was high likely that the considered person already got her final degree and hence was assumed to be not a student anymore with a high confidence.*

Figure 7.6(b) presents the possible worlds representation of pdb. Since e_1 *and* e_2 *are only maybe students, we have four possible worlds, where the first one is by far the most probable. The two entities* e_1 *and* e_2 *only both study in world* W_4. *Thus, by assuming a similarity of zero for pairs where one entity is missing, they are classified as a* MATCH *only in worlds with an accumulative probability of*

0.01. *Moreover, the accumulative probability of all worlds in which* e_1 *and* e_3 *(as well as* e_2 *and* e_3*) are classified as a* MATCH *is only* 0.1. *All these probabilities are extremely low considering the fact that the entities' attribute values are completely identical. Curiously, the accumulative probability of worlds in which* e_1 *and* e_3 *are classified as a* MATCH *is greater than the accumulative probability of these worlds in which* e_1 *and* e_2 *are classified as a* MATCH*, even the latter two do not only agree in their attribute values, but also share the same information on being a student.*

The probabilities of whether or not $\{e_1, e_3\}$*, and* $\{e_2, e_3\}$ *are classified as a* MATCH *are extreme low because of the differences in membership uncertainty. Nevertheless, this membership only models the fact whether or not the considered entities are students. This information, however, is not as useful for detecting real-world equivalence as the given attribute values because (as discussed above) the concrete membership probabilities often depend much more on the time of data insertion than the attribute values although their existence depends on this membership.*

In summary, this examples illustrates the domination of membership information that result by defining the similarity between two entities as zero if one or both of them is missing. Moreover, even in cases where the membership information is equivalent (e.g. e_1 *and* e_2*), the high influence of non-membership that is caused by using a zero similarity for missing entity pairs avoids a detection of high likely duplicates.*

As we will see in the following section, the high influence of membership uncertainty on the detection result can become even more worse, if the database contains incorrect dependencies of attributes or relationship roles to table memberships.

7.1.3.3. Incorrect Dependencies of Attributes and Relationship Roles to Table Memberships

In uncertain data, attributes and/or relationship roles often incorrectly depend on table memberships because of imperfect or minimalistic schema designs. We will denote such dependencies as *Attribute/Relationship-Membership Dependencies* (short *ARM-dependencies*). Note, ARM-dependencies typically occur if the schema of an uncertain database is designed with an approach that has been conceptualized for certain databases.

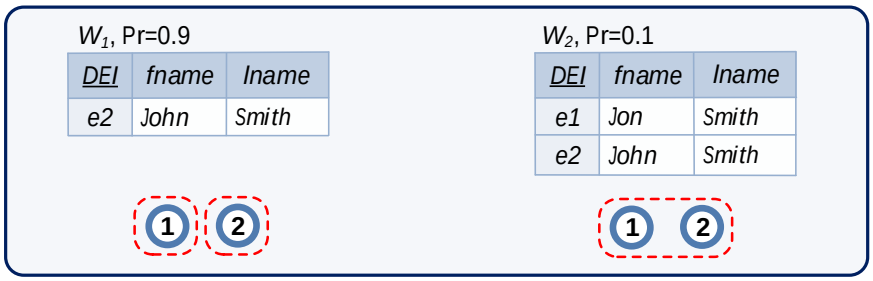
Example 104 *For illustration we consider the TI-database* pdb_1 *with the single table* 'Student' *as presented in Figure 7.7(a) and consider its possible worlds representation that is shown in Figure 7.7(c). The table contains two entities and has a schema with the two attributes* 'fname' *(firstname) and* 'lname' *(lastname). Since entity* e_1 *only maybe belongs to this table, the possible world space contains two worlds; one in which both entities are present and one in which only* e_2 *is present. Each membership probability in table* 'Student' *models the likelihood that the considered entity is a student, and therefore has only less to do with the likelihood that the entity owns the described values in the attributes* 'fname' *and* 'lname'*, because both attributes are not only applicable for students in particular, but are applicable for persons in general. Consequently, the values of these attributes are actually not restricted to worlds in which the corresponding entities are considered to be students, but have to be present in all worlds where these entities are considered to be persons. Since uncertainty on being a student does not imply uncertainty on being a person, both attributes incorrectly depends on the membership of the table* 'Student'.

DEI	fname	lname	p
e1	Jon	Smith	0.1
e2	John	Smith	1.0

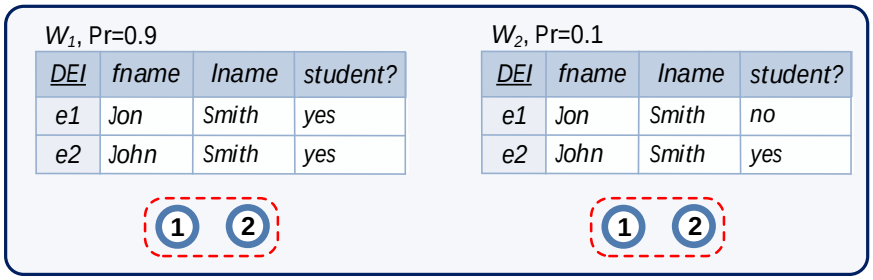
(a) TI-database pdb_1

RK	DEI	fname	lname	student?	p
1	e1	Jon	Smith	yes	0.1
2	e1	Jon	Smith	no	0.9
3	e2	John	Smith	yes	1.0

(b) BID-database pdb_2



(c) Possible worlds representation of pdb_1 with the most likely detection results



(d) Possible worlds representation of pdb_2 with the most likely detection results

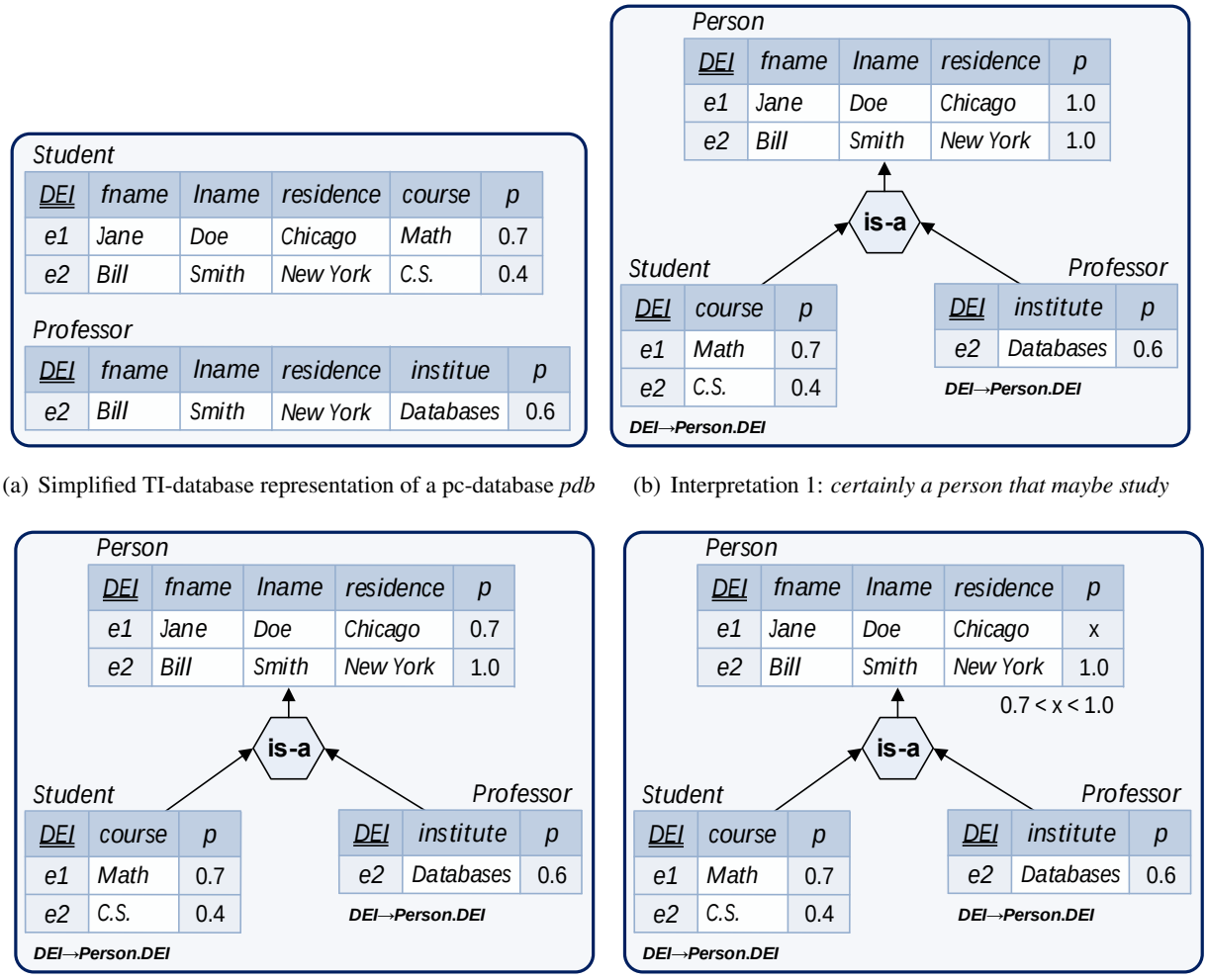
Figure 7.7.: Sample for the large influence of uncertain table membership on the duplicate detection result if incorrect ARM-dependencies are given

Such incorrect dependencies usually exist in certain data, too. Nevertheless, since in certain data all memberships are certain, these dependencies do not have any negative influence on the detection process. In uncertain data, however, such dependencies can change the detection result to the worse.

Example 105 *By assuming that every potential student is certainly a person and thus by assuming that neither* 'fname' *nor* 'lname' *depends on the membership to the table* 'Student'*, the information stored in the TI-database pdb_1 from Figure 7.7(a) is equivalent to the information stored in the BID-database pdb_2 that contains the single BID-table* 'Person' *and is presented in Figure 7.7(b). The possible worlds representation of pdb_1 and pdb_2 are presented in Figure 7.7(c) or Figure 7.7(d) respectively.*

If we define the similarity between two entities as zero if at least one of them is missing, in pdb_1 the entities e_1 and e_2 are classified as a MATCH *only in a set of worlds that has an accumulative probability of* 0.1*. In contrast, in database pdb_2 the accumulative probability of all worlds in which these entities are classified as a* MATCH *is* 1.0*, because they are present and high similar in both possible worlds.*

This example illustrates the dominance of membership information on the duplicate detection result, if attributes incorrectly depend on table memberships. The presented domination can become even more worse, if the two compared entities belong to different membership-independent tables, e.g. *'Student'*



(a) Simplified TI-database representation of a pc-database *pdb*

(b) Interpretation 1: *certainly a person that maybe study*

(c) Interpretation 2: *maybe a person that certainly study*

(d) Interpretation 3: *maybe a person that maybe study*

Figure 7.8.: A sample pc-database *pdb* (simplified presented in the form of a TI-database) with incorrect ARM-dependencies and the three possible interpretations of membership uncertainty that result from resolving the incorrect ARM-dependencies

and *'Customer'*, and hence the membership probabilities are not (or only less) comparable. Similar consequences can result from incorrect dependencies of relationship roles to table memberships.

7.1.3.4. Ambiguous Interpretations of Membership Uncertainty

A resolution of incorrect ARM-dependencies is not as simple as presented in Example 105, if the considered table also contains attributes or relationship roles that correctly depend on its membership. In that case, the table needs to be split into two or more tables instead (note that such a splitting would only be performed for the duplicate detection process and the deduplicated data need to be represented in its original schema).

Example 106 *As an example, we consider the TI-database presented in Figure 7.8(a). The database's schema has the two tables* 'Student' *and* 'Professor' *that have four attributes and that have disjoint extensions. Note, actually this database has to be modeled by a pc-database because we have correlations*

between the tuples from different tables, e.g. all tuples in 'Student' *and all tuples in* 'Professor' *that share the same DEI are mutually exclusive. For ease of presentation, however, we omit these correlations from our graphical presentations and therefore represent the database as a TI-database, but always keep the mentioned correlations in mind.*

It is obvious that the schema contains six incorrect ARM-dependencies, because the attributes 'fname', 'lname', *and* 'residence' *are not specific for students and professors, but describe persons instead. To resolve these dependencies, we can transform the schema into the database schema that is presented in Figure 7.8(b) and then export the data from the first schema to the second.*

The problem with such a exportation is that the interpretation of membership information of the source database is often ambiguous and we cannot derive a certain membership probability for each entity to the individual tables of the target database.

Example 107 *For illustration we reconsider the TI-database (pc-database) from Figure 7.8(a). The database instance contains two entities that both have certain attribute value information, but have an uncertain membership to both tables. Entity* e_1 *belongs to table* 'Student' *with a probability of* 0.7*, and entity* e_2 *belongs to the table* 'Student' *with a probability of* 0.4 *and belongs to the table* 'Professor' *with a probability of* 0.6*. The extensions of both tables are disjoint. Thus, whereas* e_1 *is maybe a student but definitely not a professor,* e_2 *is either a student or a professor. All students and all professors are persons. Therefore,* e_2 *is definitely a person, because the memberships to* 'Student' *and* 'Professor' *complement each other and sum up to one. In contrast, the probability that* e_1 *is a person cannot be derived from the data, but is ambiguous instead. We only know that* e_1 *is a student with* 0.7 *probability, but we do not know to what probability it is a person if it is not a student, because such a case is not captured in the database. There are three possible interpretation for that situation:*

1. *Entity* e_1 *is definitely a person, but this person is only a student with a probability of* 0.7.
2. *Entity*e_1 *is only maybe a person (with probability* 0.7*), but if so she is definitely a student.*
3. *Both memberships are uncertain, i.e.* e_1 *is maybe a person and if so she is maybe a student.*

Due to the given ambiguity, the exportation of our sample data from the source schema to the target schema is not clear, but each of the three possible interpretations result in another database instance. The TI-databases that represent the three possible exported instances are presented in Figure 7.8 (a)-(c). The difficult of the last interpretation is that we do not know the exact probabilities of the case that the entity is a person not being a student and of the case that this entity is not a person at all. Nevertheless, since we know that e_1 *is a student with* 0.7 *probability, the probability that* e_1 *is a person is restricted to the range* $]0.7, 1.0[$.

Note, the first two interpretations can be considered as the extreme cases of the third interpretation if we define x *within the range* $[0.7, 1.0]$ *instead of the range* $]0.7, 1.0[$*, because the first interpretation corresponds to the setting* $x = 1.0$ *and the second interpretation corresponds to the setting* $x = 0.7$

In conclusion, we sometimes cannot say in which membership the uncertainty resides: being a student, being a person or in both. This information, however, can be very important for the duplicate detection

process, because if we compare a student with a professor, the memberships to these two tables are only less relevant than the memberships to a table that contains both entities. Moreover, student membership and professor membership can change over time (sometimes even for multiple times). In contrast, person membership does not change (a person exist or not) or change at most once. As a consequence, differences in person membership are an higher indication for a non-duplicate than differences in the student membership or professor membership respectively.

Of course, the ambiguity discussed above can be extended to a consideration that captures more than one level, because we could also have attributes that are not specific for persons, but can be also available for other objects (e.g. the attribute *'age'* is not restricted to persons). However, the principle remains the same. For that reason, we restrict our discussion to one level.

Possible solutions to the problem of ambiguous membership probabilities are:

(a) An introduction of interval probabilities (or interval conditions respectively) enables a modeling of the general interpretation that covers all three possible interpretations described above. Nevertheless, an handling of interval probabilities makes duplicate detection more complex. For that reason, we abstain from such a solution at the moment and plan an extension of our duplicate detection approaches to interval probabilities in future research. Note, the schema transformation as well as the incorporation of interval probabilities would be restricted to the detection process and thus the deduplicated database would contain point probabilities if the input database contains point probabilities.

(b) A deterministic decision which of the individual memberships is considered to be uncertain, and hence to chose a fixed value for variable x in the aforementioned example.

We discuss a resolution of incorrect ARM-dependencies in detail in Section 7.7.2.

7.1.4. Detecting Scattered Duplicates

An essential challenge is the detection of equivalent duplicates, because inserting the same information mistakenly twice (or more often) is a common source of duplicates, but detecting equivalent duplicates is not straightforward by considering each possible world separately.

As already shown in Example 103 in Section 7.1.3.2, the detection of equivalent duplicates can become difficult if memberships are uncertain and the large influence of membership information on the detection result is not restricted. Nevertheless, as we will demonstrate by Example 133 this detection becomes also a challenge in cases where all entities are present in each world and no membership uncertainty is given, but an entity has dissimilar alternatives. We will refer to such entities as *scattered entities* in the rest of this thesis and denote two entities e_r and e_s to be *scattered duplicates* if they are real-world equivalent, i.e. $\omega(e_r) = \omega(e_s)$, and at least one of these entities is scattered.

Two data equivalent but scattered entities are very dissimilar in the majority of worlds and hence will only be detected as a MATCH in a small amount of the possible world space. The more alternatives the entities have, the greater becomes the amount of worlds in which they do not have similar data and hence the more difficult becomes a world-based detection of them.

RK	DEI	fname	lname	residence
1	e1	Jane	Taylor	Chicago
2	e1	Jane	Smith	Dayton
3	e1	Jane	Bush	New York
4	e2	Jane	Taylor	Chicago
5	e2	Jane	Smith	Dayton
6	e2	Jane	Bush	New York

(a) BID-database *idb* with two equivalent x-tuples (blocks)

W_1 MATCH

DEI	fname	lname	residence
e1	Jane	Taylor	Chicago
e2	Jane	Taylor	Chicago

W_2 UNMATCH

DEI	fname	lname	residence
e1	Jane	Smith	Dayton
e2	Jane	Taylor	Chicago

W_3 UNMATCH

DEI	fname	lname	residence
e1	Jane	Bush	New York
e2	Jane	Taylor	Chicago

W_4 UNMATCH

DEI	fname	lname	residence
e1	Jane	Taylor	Chicago
e2	Jane	Smith	Dayton

W_5 MATCH

DEI	fname	lname	residence
e1	Jane	Smith	Dayton
e2	Jane	Smith	Dayton

W_6 UNMATCH

DEI	fname	lname	residence
e1	Jane	Bush	New York
e2	Jane	Smith	Dayton

W_7 UNMATCH

DEI	fname	lname	residence
e1	Jane	Taylor	Chicago
e2	Jane	Bush	New York

W_8 UNMATCH

DEI	fname	lname	residence
e1	Jane	Smith	Dayton
e2	Jane	Bush	New York

W_9 MATCH

DEI	fname	lname	residence
e1	Jane	Bush	New York
e2	Jane	Bush	New York

(b) The possible worlds $\{W_1, \ldots, W_9\}$ of *idb* with likely resulting matching classes

Figure 7.9.: Sample for illustrating the difficulty of detecting scattered duplicates by considering possible worlds separately

Example 108 *For illustration, we consider a commonly occurring situation where the data of a person should be stored in a database. The considered person has married (and hence has changed her lastname) and moved twice, but we do not know what combination of lastname and residence is the current one. We use three alternatives to store this uncertain information into one x-tuple (database entity e_1). Let us assume that the same information is mistakenly inserted into the database for a second time and therefore the database contains another x-tuple (database entity e_2) that exept of the DEI has equivalent data to x-tuple representing e_1. The BID-database idb containing both entities is presented in Figure 7.9(a). Due to we do not have any dependencies, idb represents the nine possible worlds that are depicted in Figure 7.9(b). Caused by the dissimilarity of the alternatives, both entities are similar only in three of these nine worlds and hence are likely classified as a* MATCH *only in 33% of all worlds. Thus, both entities are only finally classified as a* MATCH *if the used aggregation method is configured extremely lax. Such a lax configuration, however, implicates a high risk of producing many false positive in cases of non-equivalent x-tuples.*

A detection of real-world equivalence between scattered database entities would be much easier on the original database where x-tuples can be compared. However, such a comparison becomes non trivial in multi-table databases where each entity is represented by x-tuples from several tables. Moreover, during data insertion typos can corrupt the data (attribute values as well as probabilities) and the scattered x-tuples are not completely equivalent anymore. Thus, simply comparing the x-tuples' alternatives on equivalence is not sufficient and similarity measures need to be used instead. Sometimes it seems plausible to design a duplicate detection approach that is tailor-made for the considered representation system, which is a disadvantage and advantage at the same time. On one hand, we strive for a duplicate detection approach that is representation independent, but on the other hand, the typical error patterns often strongly depends on the used representation system, because the data is inserted in exactly that way of representation. Moreover, error pattern can also depend on the tools that have been used for data insertion (command line, web application, etc.).

In the two deterministic detection approaches that we present in this thesis, we aim to handle that challenge by specific aggregation methods (see Section 7.5) and hence stay independent from the representation system the underlying data is modeled in. Nevertheless, additional methods that are designed with a specific representation system (or application) in mind could improve the quality of detecting scattered duplicates further on.

7.1.5. Trustability and Affections of Entity Dependencies

The possible world space is constructed based on dependencies between the memberships, attributes, or relationships of the individual database entities. Thus, if we perform duplicate detection in each world separately, we automatically agree with these dependencies and the affect of these dependencies on the detection result is automatically implied by the processed world space. However, as we will show in this section, entity dependencies can be the result of incorrect assumptions and hence can change the duplicate detection result to the worse. Moreover, even in the case of correct dependencies, the affections on the duplicate detection process that are implied by the world space sometimes do not correspond to the affections the user would expect. For that reason, we will discuss both issues in more detail.

First we consider the trustability of entity dependencies and then discuss the different ways dependencies can affect the duplicate detection process in a meaningful way.

7.1.5.1. Trustability of Entity Dependencies

During data management incorrect dependencies between different database entities, i.e. dependencies that contradict real-world, can find their way into the data. Such a situation is illustrated in Example 109 where a certain database is transformed into an uncertain database by building a set of possible repairs.

Example 109 *Consider the certain database db presented in Figure 7.10(a). This database contains a single table with the three database entities e_1, e_2, and e_3. Assume that all these entities are true duplicates ($\omega(e_1) = \omega(e_2) = \omega(e_3)$), but are not known to be duplicates so far. Whereas e_1 and e_2 have the same value in the* social security number *(attribute* 'SSN'*), e_3 has a unique one. In real world, the SSN is unique and therefore we can simply realize that something is wrong with the data. However, the type of error is not simple to identify, because e_1, e_2 could be a duplicate, one (or both) of their SSNs can*

contain a typo, or in one of the entities the SSN is transposed with the value of another attribute as for example the bank account number ('BAN'*), etc.. Nevertheless, some applications require that the unique constraint is satisfied and one option to guarantee uniqueness is to materialize a set of possible repairs [Ber11, DPW12] within an uncertain database. Let us assume that for guaranteeing the uniqueness of the SSN the database administrator performs an operation* possRepair *on this database that builds the set of all possible repairs to this unique constraint. The result of this operation is an incomplete database idb that has two possible worlds, i.e. the two possible repairs produced by* possRepair*, that are presented in Figure 7.10(b). Since e_1 and e_2 have the same value in SSN, both entities exclude each other, i.e. no repair consists both of them. However, all three entities are still duplicates.*

As we illustrate by this example, duplicate entities can remain in the same possible world (called *intra-world duplicates*), but can also exist across different worlds (called *inter-world duplicates*).

Definition 56 ***(Intra-World Duplicates:)*** *Let udb be an uncertain database that has the possible world space $\boldsymbol{W}$. Two database entities $e_r, e_s \in Ext_{Poss}(udb)$ are denoted as* intra-world duplicates*, if they are duplicates, i.e. $\omega(e_r) = \omega(e_s)$, and if they coexist in a single possible world: $\exists W \in \boldsymbol{W}\colon e_r, e_s \in W$.*

Definition 57 ***(Inter-World Duplicates:)*** *Let udb be an uncertain database that has the possible world space $\boldsymbol{W}$. Two database entities $e_r, e_s \in Ext_{Poss}(udb)$ are denoted as* inter-world duplicates*, if they are duplicates, i.e. $\omega(e_r) = \omega(e_s)$, and if they do not coexist in any possible world: $\nexists W \in \boldsymbol{W}\colon e_r, e_s \in W$.*

In Example 109, e_1 and e_3 as well as e_2 and e_3 are intra-world duplicates, because they coexist in a single world, that is e_1, e_3 in W_1 and e_2, e_3 in W_2. In contrast, e_1 and e_2 are inter-world duplicates, because they only exist across different worlds, i.e. e_1 in W_1, but e_2 only in W_2.

From this example, we can conclude that excluding entities can be indeed true duplicates. Moreover, we also can conclude that duplicate-free worlds do not necessarily imply a duplicate-free database.

Theorem 11 *An uncertain database can contain true duplicates, even if each of its worlds is free from true duplicates.*

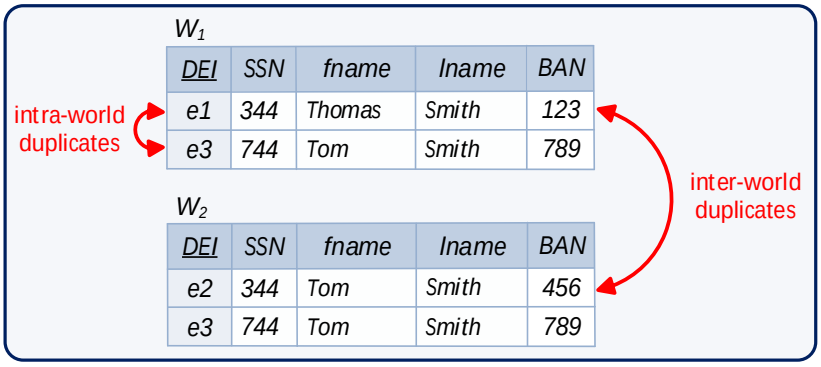
Theorem 11 is proved by Example 109.

The problem that incorrectly introduced dependencies can change the duplicate detection result to the worse also holds for other types of entity dependencies besides exclusions on table membership.

Example 110 *For demonstration, we consider the possible world space $\boldsymbol{W}_1$ presented in Figure 7.11(a). This space consists of four possible worlds, where each of these worlds consists of the two duplicate database entities e_1 and e_2 ($\omega(e_1) = \omega(e_2)$). For both entities, the SSN and the last name are uncertain. Whereas the uncertain SSNs could result by an erroneous data insertion (maybe a manually completed document could not be exactly read by the corresponding secretary) the uncertain last name could result from incomplete information on a marriage (it is known that the person married and changed her name, but it is unknown which of the two names is the current one). It is important to note that this database could result from inserting the uncertain information on the same real-world entity twice.*

DEI	SSN	fname	lname	BAN
e1	344	Thomas	Smith	123
e2	344	Tom	Smith	456
e3	744	Tom	Smith	789

(a) Certain database *db*



(b) Possible world space $\{W_1, W_2\}$ of an incomplete database $idb = possRepair(db)$

Figure 7.10.: The incomplete database *idb* that models all possible repairs of the certain database *db* with respect to the constraint that the attribute *'SSN'* is unique

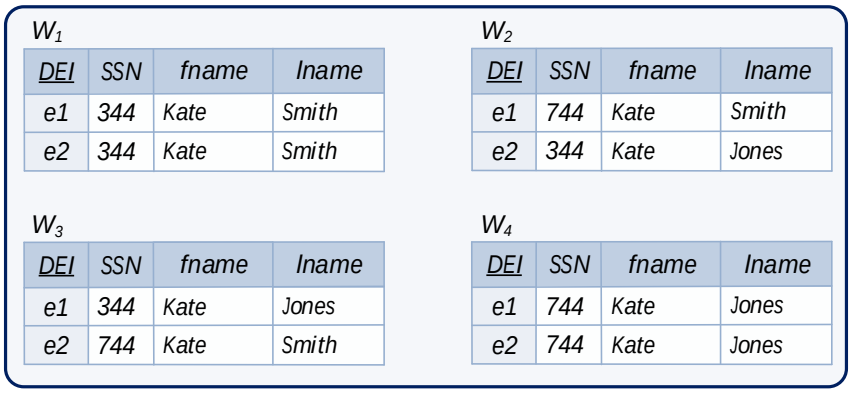
Because the SSN need to be unique, the database administrator discards all worlds that violates this constraint and hence restrict the considered possible world space $\boldsymbol{W}_1$ *to the world space* $\boldsymbol{W}_2 = \{W_2, W_3\}$ *that is presented in Figure 7.11(b). The consequence of this restriction is that in both remaining worlds* e_1 *and* e_2 *are high dissimilar and hence it is unlikely that they are classified as a* MATCH. *In contrast, by considering the initial set of four possible worlds, both entities are equivalent in two of the four worlds and hence the likelihood that they are classified as a* MATCH *is much greater.*

As a consequence, the incorrectly introduced dependencies probably cause a false negative, because the considered entities are dissimilar in the remaining worlds.

The logical conclusion of Theorem 11 and Example 110 is that performing duplicate detection in each world separately does not necessarily lead to a complete result (recall< 1) if the database contains incorrect entity dependencies. For that reason, before detecting duplicates the uncertain database needs to be cleaned from incorrect entity dependencies. The problem in dependency cleaning is that the incorrectness of a dependency cannot be concluded from the dependency itself, but instead require context information on the dependency's origin. We will discuss dependency cleaning in Section 7.7.1.

7.1.5.2. Meaningful Affections of Entity Dependencies

The above discussion on the correctness of dependencies leads to the fundamental question whether we should consider dependencies at all or we should ignore any kind of dependency instead. As we will show in this section, trustable entity dependencies can affect the duplicate detection process in a meaningful way.



W_1

DEI	SSN	fname	lname
e1	344	Kate	Smith
e2	344	Kate	Smith

W_2

DEI	SSN	fname	lname
e1	744	Kate	Smith
e2	344	Kate	Jones

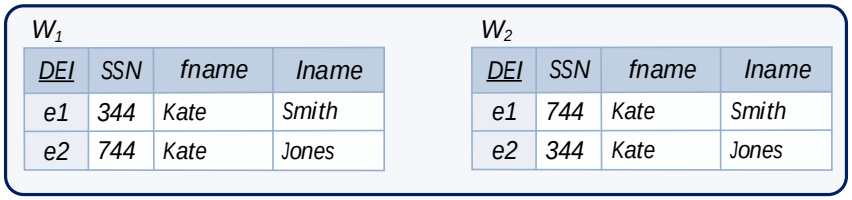
W_3

DEI	SSN	fname	lname
e1	344	Kate	Jones
e2	744	Kate	Smith

W_4

DEI	SSN	fname	lname
e1	744	Kate	Jones
e2	744	Kate	Jones

(a) Possible world space $\mathbf{W}_1 = \{W_1, W_2, W_3, W_4\}$



W_1

DEI	SSN	fname	lname
e1	344	Kate	Smith
e2	744	Kate	Jones

W_2

DEI	SSN	fname	lname
e1	744	Kate	Smith
e2	344	Kate	Jones

(b) Possible world space $\mathbf{W}_2 = \{W_2, W_3\}$

Figure 7.11.: The possible world space $\mathbf{W}_2$ that models the possible repairs of the possible world space $\mathbf{W}_1$ by guaranteeing the uniqueness of the attribute *'SSN'*

Example 111 *For example, consider an uncertain database where each possible world result from the same scientific experiment performed under different conditions/configurations. In that case, it makes no sense to compare the entities across worlds, if the worlds' underlying conditions/configurations are not comparable. Thus, in this example the given entity dependencies contain useful information it make sense to incorporate them into the detection process.*

Nevertheless, the answer to the question in which way an entity dependency affect the duplicate detection process in a meaningful way is not always clear and can differ from the affection that is implicated by performing duplicate detection in each world separately.

Example 112 *Consider a situation of storing information on four persons' with the same last name where each of the person's residence is either 'City A' or 'City B'. Assume that the corresponding secretary knew that all four persons belong to the same family and hence either all of them live in 'City A' or all of them live in 'City B'. Thus, the given information is stored in an incomplete database with the two possible worlds presented in Figure 7.12.*

If we perform the duplicate detection process in both worlds separately, the likelihood that two of these persons are classified as a MATCH *is greater than in the case we do not consider these dependencies, because in the first case the residences are equivalent in each of the two worlds, but would be dissimilar in some of the sixteen worlds of the second case. Thus, the dependencies affect the detection process in the way that they increase the likelihood that some pairs of these entities are classified as a* MATCH.

The question now is whether this affection is desired or not? On one hand, it could be the case that one (or more) family members are mistakenly inserted into the database for several times, e.g. e_3 *and* e_4 *could*

W_1

DEI	fname	lname	residence
e1	Thomas	Smith	City A
e2	Kim	Smith	City A
e3	Tim	Smith	City A
e4	Tom	Smith	City A

W_2

DEI	fname	lname	residence
e1	Thomas	Smith	City B
e2	Kim	Smith	City B
e3	Tim	Smith	City B
e4	Tom	Smith	City B

Figure 7.12.: A possible world space $\mathbf{W} = \{W_1, W_2\}$ with two possible worlds

describe the same child. In this case, the affection implicated by the world space is appropriate because it helps to detect true duplicates. On the other hand, we are inclined to assume that the information that these persons belong to the same family implies the information that each of these database entities refer to another real-world person, even if their data is similar to a large extent. In that case, the affection implicated by the world space would be counter-productive and implies a high risk of causing many false positives.

This example should illustrate that the most meaningful affection of a set of entity dependencies on a duplicate detection process, (a) can differ from the affection that is implicated by a world-based consideration, (b) is often only hard to derive from the given set of dependencies, and (c) can essentially depend on the considered context.

For that reason, we have to differentiate between the dependencies themselves and the affections that are caused by these dependencies. We realize this differentiation by a data preparation activity that is presented in Section 7.7.1. The preparation activity (i) extracts all dependencies from the data, (ii) verifies the dependencies on correctness, (iii) derives specific affections from the correct dependencies, (iv) removes the original dependencies from the data, and (v) finally passes the prepared data as well as the derived activities to the duplicate detection process.

7.1.6. Outline

In the remainder of this chapter, we propose two approaches for deterministic duplicate detection in uncertain databases. The first approach consider uncertainty on database level and is denoted as *World-Based Approach* (Section 7.3). In contrast, the second approach consider uncertainty on the level of entity descriptions and is denoted as *Description-Based Approach* (Section 7.4). Since an adaptation to incomplete databases is usually straight forward, we restrict our consideration to probabilistic databases. We guarantee representation independence by defining both detection approaches based on the possible worlds representation and restrict the need of adaptation to a specific representation system on the step of possible world generation (world-based approach) or to the step of extracting probabilistic entity descriptions (description-based approach) respectively.

We ensure deterministic detection results by resolving uncertainty at one moment of the detection process (at least by aggregating the final clusterings) and compensate incomplete detection results and hence the strong influence of information on non-membership by using the concept of impact values (see Section 7.2). In Section 7.5 we propose a specific uncertainty resolution method for increasing

the effectiveness of detecting scattered duplicates. Duplicate detection processes should scale well even for large databases. For that reason, we consider several optimization techniques in Section 7.6. To resolve incorrect ARM-dependencies (and hence to clearly interpret ambiguous situations of membership uncertainty) and to handle correlation between database entities in an appropriate way, we propose two new activities of data preparation in Section 7.7. Uncertainty typically does not only result from the input database, but is also a consequence of several potential process configurations. An incorporation of process uncertainty into the detection process is therefore considered in Section 7.8. Finally, we compare the world-based detection approach and the description-based detection approach from a theoretical point of view in Section 7.9.

7.2. Aggregation Methods for Uncertainty Resolution

We propose two duplicate detection approaches in this chapter. Both of them accomplish deterministic duplicate detection results by resolving emerging uncertainty between any of the corresponding detection phases. The purpose of uncertainty resolution is to transform an uncertain value (probabilistic or incomplete) to a crisp value of the same domain. If we consider an uncertain value as its set of alternatives, the resolution of an uncertain value is semantically equivalent to the aggregation of a set of crisp values.

Probabilistic value alternatives are associated with probabilities. Moreover, each alternative can be associated with an individual impact value (see Section 7.2.1). For that reason, we consider all elements of the aggregation input to be associated with an element-specific weight. We distinguish between absolute weights and relative weights. Let dom be an arbitrary domain, let $u \in^{[\iota/\rho]} dom$ be the uncertain value of this domain we want to resolve, and let $imp(v)$ be the impact value that has been computed for each element $v \in alt(u)$. The most intuitive way to compute the absolute weight of domain element is to use the impact value as weight if the considered uncertain value is incomplete and to use the product of impact value and probability as weight in the considered uncertain value is probabilistic (of course, other computation methods are possible). Therefore, we define the absolute weight of an element $v \in dom$ as:

$$absW(v) = \begin{cases} imp(v), & \text{if } u \in^{[\iota]} dom, \\ imp(v) \times Pr(v), & \text{if } u \in^{[\rho]} dom \wedge u = (y, Pr). \end{cases} \tag{7.2}$$

Note, the domain of feature scores is an exception because these scores already contain some impact values and therefore does not need to be associated with one. For that reason, we consider the absolute weight of a feature score to be computed on probabilities only. As a consequence, for an incomplete feature score the weights are uniformly distributed on its set of alternatives.

The relative weight of a domain element v is considered to be the ratio between its absolute weight and the accumulative absolute weight of all domain elements and is therefore defined as:

$$relW(v) = \frac{absW(v)}{\sum_{v' \in dom} absW(v')} \tag{7.3}$$

The relative weight of a set $S \subseteq dom$ is defined as $relW(S) = \sum_{v \in S} relW(v)$. In the rest of this section, we always refer to relative weights if we talk about weights.

As mentioned in Section 7.1.2, uncertainty can be resolved at different moments in the detection process. For that reason, we consider aggregation methods for different input domains. An aggregation

methods gets a set of elements from the same domain as input and produces an element of this domain as output. We start with the domain of possible worlds (each a certain databases instance) as well as the domain of entity descriptions and then continue with the domains of similarity scores, feature scores, matching classes, duplicate decisions, and cluster-disjoint clusterings[2]. Since the input set contains all domain elements that have a weight larger than zero, we sometimes define the input set only implicitly by defining a weight function.

7.2.1. World-Impact Values

In Section 7.1.3.1 we present the challenge of handling incomplete duplicate detection results and sketch two intuitive solutions. Whereas the first solution considers two entities to be an UNMATCH in all possible worlds in which they do not coexist, the second solution simply ignores information on uncertain memberships. Both solutions are only less satisfying in many application scenarios because they correspond to two extreme ways of considering membership uncertainty in the detection process: The first solution considers membership uncertainty unmitigated with all the strong influence it has and the second solution does not consider membership uncertainty at all.

For that reason, we utilize the concept of impact values in order to enable a trade-off between these two extremes. Recall, we introduce the concept of impact values in Section 5.2.2 in order to rate the significance (or the reliability) of matching results. Obviously, we can adopt this concept to rate the significance of worlds and hence can use it to configurate the influence of membership uncertainty on the detection result. Combining impact values with an assumption of zero similarity for missing entity pairs results in a mix of the two aforementioned intuitive solutions, because we do not only assume a zero similarity for missing entity pairs, but assign them with a low impact value and therefore can differentiate between a zero similarity that results from an actual matching process and a zero similarity that is assumed by a missing entity. By adjusting the size of the impact that we use for a missing entity pair, we can adjust the desired effect of membership information on the detection result. For instance, if we set the impact value to zero this approach is equivalent to the second intuitive solution that completely ignore membership uncertainty and if we set the impact value to one this approach is equivalent to the first intuitive solution that does not compensate the strong influence of membership uncertainty on the detection result at all.

The advantage of using a pairwise aggregation method that can work on incomplete detection results is the enabled differentiation between worlds that contain the two compared entities and worlds that do not. Since entities can belong to multiple entity tables, it makes sense to refine this differentiation further on and therefore to use specific world-impact values not only for missing entity pairs but rather for all entity pairs. For instance, assume that in one world the compared entities are each represented in multiple tables in each with several attributes, but in another world they are each represented only in a single table with one or two attributes. Consequently, the matching result (feature score, similarity score, or duplicate decision) in the first world is more significant for the aggregation output than the matching result of the second world. To capture these differences in significance, we compute an individual world-impact

[2] Although the feature matching phase is executed in front of the similarity computation phase, we discuss an aggregation of similarity scores before discussing an aggregation of feature scores because an aggregation of the former is much more intuitive than an aggregation of the second and aggregation concepts of the first can be adopted to the latter.

value per combination of entity pair and world, and store all the impact values of one pair into a so-called *world-impact vector*. This world-impact vector of an entity pair is than used to determine a pair-specific weighting scheme for the aggregation process. Let $\{e_r, e_s\}$ be an entity pair, the world-impact vector of $\{e_r, e_s\}$ is in the following denoted as $\vec{\iota}_{pw}(e_r, e_s) = \{v_1, \ldots, v_k\}$ where v_i is the impact value of the detection result that has been computed for $\{e_r, e_s\}$ in world W_i.

Recall that we have proposed a method for computing the impact of the similarity between two entities in Section 5.2.2.4. We adopt that computation method to feature scores and duplicate decisions. This can be simply justified. The proposed method computes the impact of a similarity score from the impact values that are stored in its corresponding feature score. Consequently, that impact value is also suitable to rate the significance of this feature score. The matching class that is computed in the classification phase is based on the similarity score. Thus, the reliability of the produced duplicate decision is the same as the reliability of the processed similarity score.

Of course, we can use the concept of impact values also for aggregating possible worlds or for aggregating whole duplicate clusterings. In that case, however, we need to consider all entity pairs collectively and therefore can only compute one impact value per world. Let W be a possible world, the world impact value of world W is denoted as $imp(W)$. Recall that we have proposed a method for computing the impact of a certain database in Section 5.2.2.4. The differentiation between the individual worlds, however, cannot be as fine as for individual entity pairs, because an impact value must reflect the significance (reliability) of all database entities that belong to the corresponding world and cannot be adjusted to individual pairs.

In summary, the concept of world-impact values provides a powerful mechanism to handle the challenge of missing entities. Nevertheless, the computation of appropriate world-impact values is often not straightforward. Moreover it is not simply to verify whether or not the computed impact values are appropriate for the considered use case. As a consequence, world-impact values often cannot completely compensate the negative effects that result from the assumption that missing entity pairs have zero similarity. For that reason, we combine them with a preparation phase that resolves incorrect ARM-dependencies (see Section 7.7.2) and propose a specific aggregation method for scattered duplicates that does not only consider the similarity between the alternatives of two compared probabilistic values, but also takes the similarity between their probability distributions into account (see Section 7.5).

7.2.2. Aggregation Requirements

Whereas possible worlds and entity descriptions represent duplicate detection input, similarity scores, feature scores, matching classes, duplicate decisions, and duplicate clusterings represent duplicate detection results (interim or final). Consequently, the requirements for an appropriate aggregation with respect to the first two domains are different to the requirements for an appropriate aggregation with respect to the last five domains. Since worlds and descriptions only serve as detection input, an aggregation method is the more useful the better the aggregation output represents the set of input values. In contrast, methods for aggregating detection results additionally need to take application-specific requirements on duplicate detection quality (e.g. the application considers false positives to be double worse than false negatives) into account. The most meaningful audacious method (least number of false negatives) is to take the most 'positive' input value as aggregation output because by doing so two entities are suspected

to be duplicates if they are suspected to be duplicates in any of the plausible scenarios. In contrast, the most meaningful cautious method (least number of false positives) is to take the most 'negative' input value as aggregation output, because by doing so two entities are suspected to be non-duplicates if they are suspected to be non-duplicates in any of the plausible scenarios. Of course, the definition of most 'positive' input value and most 'negative' input value depends on the considered domain. For instance, the most 'positive' input value and the most 'negative' input value in the domain of similarity scores are intuitively defined as 1.0 or 0.0 respectively. In contrast, in the domain of clusterings, the most 'negative' input value is defined as the clustering with the least number of MATCHES and the most 'positive' input value is defined as the clustering with the most number of MATCHES.

7.2.3. Deciding Methods vs. Mediating Methods

Note, as for string matching (see Section 4.3.5), there is a large space of methods which can be used for aggregating a set of uncertain value alternatives of a specific domain to a single one. Although all these methods are based on specific ideas for aggregation, it is hard to predict their effectiveness without experimental verifications. Moreover, most of these methods are more or less application dependent, because they are designed with some specific error pattern in mind and the best fitting aggregation method is generally a trade off between effectiveness and runtime requirements. Therefore, it is not the goal of this section to completely explore the space of convenient aggregation methods for each of the considered domains, but rather to point out some intuitive solutions instead. As a consequence, we set some first benchmarks for further explorations and therefore present a toolbox for each input domain from which the user can select a method that is best fitting for his purpose or that inspires the user to design her own. Of course, the larger the toolbox, the more difficult becomes the selection for the user without making deeper considerations on the provided methods first. Nonetheless, a powerful toolbox also gives him the opportunity to find an appropriate method if he is willing to put some effort into the selection process.

Recall from Section 6.7.2.1 that each uncertainty resolution function is either of a deciding nature, i.e. one of the input alternatives is selected as resolution output, or is of a mediating nature, i.e. the resolution output is a domain element that does not necessarily belong to the set of input alternatives. Obviously, we can adopt the same distinction for aggregation methods. The deciding method that is most intuitive in the presence of weights is to return the most weighted input value as aggregation output. Nonetheless, the most weighted input value often does not represent the input set best. For that reason, it can be a better solution to return the input value that has the shortest distance to all other input values, e.g. the median in ordered domains, instead. Of course, for this purpose a distance measure is required and as we will see in the rest of this section defining an appropriate distance measure is not straightforward for some domains. Further deciding strategies are to return the most informative input value (e.g. the longest string or the most precise decimal) or to return one of the input values randomly.

A typical mediating method is not to return the input value that has the shortest distance to all input values as aggregation output, but to return the domain element that has the shortest distance to them. In that case, the returned value does not necessarily belong to the input set. In numerical domains, this approach is usually realized by computing the (weighted) average of all input values. Another commonly used mediating method is to chain the input values to a single one as for example by concatenating string values or by computing the union of sets.

In the remainder of this section, we discuss aggregation for different input domains. Since the aggregation aspects discussed above (aggregation requirements, deciding strategies vs. mediating strategies) are applicable for all these domains, we will face them in the individual discussions, but usually need to utilize them in different circumstances.

7.2.4. Aggregation of Database Instances (Possible Worlds)

The earliest moment of resolution is to aggregate the possible worlds of the input database to a single world (and hence a certain database). As mentioned above, the most intuitive selection strategy is to take the most weighted world as aggregation output. Nevertheless, if some of the database entities are missing in this world, they will not belong to any of the produced duplicate clusters. For that reason, it is more meaningful to select the most weighted world that has all database entities present (note, if the input database is entity independent, for each possible world with missing entities there is another possible world that includes this world and has all entities present).

A database instance is a set of database tuples. Thus, the distance between two worlds can be computed by using a hybrid similarity measure as the Extended Jaccard Coefficient or the Monge-Elkan Similarity (see Section 4.3.5.6) that computes the similarity between two sets by utilizing an internal measure for element similarity. In the considered scenario, the internal measure need to compute the similarity between two tupels. Thus, the most of the decision models that we have discussed in Section 4.3.6 can be used for this purpose. Nonetheless, the number of input worlds is usually extremely large and a pairwise comparison of all these worlds becomes mostly impractical. For that reason, we cannot compute the world with the shortest distance to all other worlds by comparing these worlds in a pairwise fashion. This problem can be solved in two ways.

- We select a small subset of possible worlds (maybe the most weighted worlds) beforehand and then perform the pairwise world comparison only for the selected ones.

- We compare all possible worlds indirectly by comparing the alternative instances per database entity. Due to we are interested in a world that contains all entities, we can compute a representative world by selecting the most representative alternative instance per database entity. Of course, this approach automatically assume that all database entities are mutually independent. Thus, if the input database contains entity dependencies, the computed world must not necessarily belong to the database's possible world space.

Aggregating whole database instances in a mediating fashion is not straightforward. The most intuitive solution is to perform aggregation on entity level and thus to aggregate the alternative instances of each database entity to a single one by using a mediating aggregation method for entity instances. Nonetheless, an aggregation of possible entity instances is in turn non-trivial if the considered database is complex so that each entity can be represented by a set of tuples and relationships between these entities are modeled by foreign key constraints.

Another approach is to mediate the input databases by computing the set union of their tuples per table. Nevertheless, the consequential problem is that database entities are then represented by multiple tuples in one table which violates the principles of the entity-based interpretation of a relational database. One

conflict resolution function	type	domain	description
cry with the wolves	deciding	ALL	selects the most often occurring value
most weighted value	deciding	ALL	selects the most weighted value
roll the dice	deciding	ALL	selects a value randomly
longest value	deciding	string	selects the longest value
concat	mediating	string	concatenates a set of string values
median/average	mediating	numerical	computes the median/average of all values
weighted average	mediating	numerical	computes the weighted average

Table 7.1.: Conflict resolution functions that can be used for mediating strategies

solution for this problem is to treat all possible instances of one database entity as own database entities and to initially mark them as MATCHES in the belief map.

7.2.5. Aggregation of Entity Descriptions

A second moment of resolution is to aggregate several descriptions on the same entity to a single one. In general, we can make use of the same ideas as we have presented for entity-based aggregation of worlds, but in this case the aggregation is simpler to define and simpler to compute, because each attribute description corresponds to a single tuple and we consider each relationship description as a collection of entity sets (one set per label).

Aggregating a set of certain attribute descriptions is similar to computing a representative for multiple conflicting duplicate tuples in the fusion of certain data [BN06, BN08] and is similar to computing a canonical entity in clustering-based decision models (see Section 4.3.6.5). The only difference is that we consider the input descriptions to be associated with weights whereas certain duplicate tuples (or certain database entities respectively) are usually not.

- **Deciding Strategies:** Obviously, the most intuitive deciding strategy is to take the most weighted input description as aggregation output. A more sophisticated deciding strategy is based on the Distributional Cluster Feature (DCF). Andritsos et al. [AFM06] utilize the DCF to compute a representative per duplicate cluster that is then used to derive a probability for each of the cluster's tuples based on the tuple's distance to the computed representative. Due to the representative computed by the DCF is not an element of the considered domain but a numerical vector, the representative itself cannot be used as aggregation output. However, we can use the input description that has the shortest distance to the representative as aggregation output instead.

- **Mediating Strategies:** In many cases, a description that is computed by a mediating strategy represents the given input set better than any of the input descriptions. In the fusion of certain duplicate tuples, mediating strategies are usually applied on an attribute-by-attribute basis, i.e. the representative tuple results from computing a representative value for each attribute. Functions for merging several values of the same attribute are denoted as conflict resolution function [BN06,

	fname	lname	residence	semester	ALL	relW
d_1	John	Doe	Rome	4	{e4}	0.35
d_2	John	Smith	Paris	6	{e4,e5}	0.4
d_3	Jon	Doe	Prague	6	{e4,e5}	0.25

(a) Three input entity descriptions

	fname	lname	residence	semester	ALL
d_4	John	Doe	Paris	6	{e4,e5}
d_5	John|Jon	Doe|Smith	Paris|Rome|Prague	6|4	{e4}

(b) Two sample aggregation outputs

Figure 7.13.: Sample for aggregating entity descriptions with mediating strategies

BN08], because a single value is computed from a set of conflicting values. Obviously, to each attribute another conflict resolution function can be applied.

Since functions for conflict resolution are specific aggregation methods, resolution functions can be of a deciding or a mediating nature. Deciding functions that are typically used to resolve conflicts of attribute values are *cry with the wolves* that select the most often occurring value as the representative or *roll the dice* that select one of the values randomly [BN08]. A typical mediating function is *meet in the middle* that computes the average or the median of the given input values. Another typically used function for string attributes is *concat* that builds a single large string by concatenating some of the input values by using a specific delimiter symbol. Due to portion at the beginning of a value are often more important than subsequent portions, it make sense to concatenate the values in the descending order of their weights. Notice, in the context of duplicate detection, a concatenation of string values is especially useful if we use a token-based similarity measure in the feature matching phase.

Because the input descriptions are associated with weights, additional conflict resolution function are possible and often even more convenient. Examples are a deciding function that selects the most weighted value or a mediating function that computes the weighted average. Of course, different resolution functions can be used for different attributes. An overview on some conflict resolution functions that can be used for computing a representative of a set of conflicting attribute values is presented in Table 7.1.

Sets of related entities can be aggregated by a voting approach, i.e. an entity belongs to the aggregation output if all the input sets it belongs to have an accumulative weight of at least $k \in [0, 1]$. If we set $k = 0$, the voting approach corresponds to the set union and if we set $k = 1$, the voting approach corresponds to the set intersection. Of course, the set of related entities are aggregated for each label separately.

Example 113 *We will illustrate the different aggregation methods with the three entity descriptions d_1, d_2, and d_3 presented in Figure 7.13(a). The attribute descriptions consist of four attributes and are presented in tabular form. For ease of presentation we consider the corresponding relationship descriptions to consists of a single set of related entities and therefore incorporate them with the extra attribute* 'ALL'

into the table-based presentation. Selecting the most weighted description would result in description d_2. Depending on the used distance measure, it is likely that description d_3 would be selected as aggregation output because it has a similar first name to the other two descriptions, has the same last name as d_1, and has the same set of related entities as d_2.

If we select the most weighted value (set) per attribute (relationship label), we select 'John' for 'fname'*, 'Doe' for* 'lname'*, 'Paris' for* 'city'*, '6' for* 'semester'*, and the set $\{e_4, e_5\}$ for the relationship description. The corresponding aggregation output is presented by description d_4 in Figure 7.13(b). This sample demonstrates the difference between using a deciding strategy on description level and using deciding strategies on attribute (relationship) level, because the descriptions d_2 (most weighted description) and d_4 (description with most weighted values) are different.*

In contrast, by concatenating the values of each attribute in the descendant order of their weights (we use the symbol '|' as delimiter) and by selecting each entity from the relationship description that has an accumulative weight larger than 0.7, the aggregation method produces description d_5 as output (see Figure 7.13(b)).

7.2.6. Aggregation of Similarity Scores

Since we consider similarity scores to be normalized, methods for aggregating similarity scores are defined on the domain of all real numbers between zero and one, i.e. $dom = [0, 1]$. Nevertheless, the number of alternatives of an uncertain similarity score is always finite so that the continuous domain can be reduced to a discrete set $dom^* = \{v \mid v \in dom, \mathit{relW}(v) > 0\}$ without changing the aggregation output.

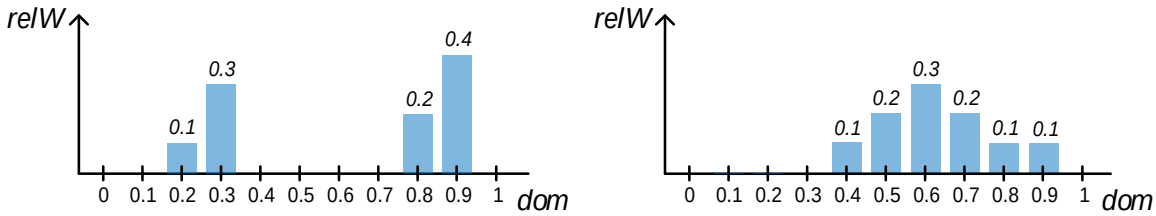
The most intuitive mediating method for aggregating a set of weighted similarity scores is to compute the weighted average similarity, i.e.

$$\mathit{WAvgSim} = \sum\nolimits_{v \in dom^*} \mathit{relW}(v) \times v \tag{7.4}$$

To diminish the strong impact of outliers, it can be useful to discard the k lowest scores and to discard the k highest scores from the input set and then to compute the weighted average of the remaining scores.

Typical deciding strategies for aggregating a set of weighted similarity scores is to select the most weighted score, to select the median score, or to select the score that has the shortest distance to the weighted average.

Besides aggregating several similarity scores to a single score, it can be useful to directly derive a matching class from the given similarity scores instead. Such aggregation methods can be learned with machine learning techniques as Decision Trees or Support Vector Machines. Most learning techniques require a finite vector of numerical values as input instead of a continuous set. However, by restricting the continuous domain $dom \in [0, 1]$ to a discrete ordered domain dom^D and than by mapping each real value of dom to a discrete value of dom^D, the derivation input can be considered as a vector where the ith element in dom^D represents the ith vector position and the weight of this element represents the value of that vector position. For example, let us assume the discrete ordered domain $dom^D = \{0, 0.1, 0.2, \ldots, 0.9, 1\}$, each value in $dom \in [0, 1]$ can be mapped to dom^D by rounding on one decimal place. To stay conform with the initially made definition that the aggregation output must be an element of the same domain as the aggregation input, we simply map the three matching classes to the fixed



(a) Scenario 1 (b) Scenario 2

Figure 7.14.: Two input sets of weighted similarity scores

similarity scores 0 (UNMATCH), 1 (MATCH), and β (POSSIBLE MATCH) where β is a real number within the range $[0, 1]$.

7.2.6.1. Incorporation of Quality Requirements

In general, aggregating similarity scores is always a trade-off between the given quality requirements.

Example 114 *To illustrate the consequences of these requirements on the aggregation method we consider the sample scenario that is depicted in Figure 7.14(a). The given set of input scores is somewhat contradictory, because some of them are really low and some of them are really high. If we consider false negatives to be worse than false positives, it make sense to return a high similarity score as aggregation output, because the two considered entities are high similar in more than the half of the possible worlds and therefore we cannot exclude that they are true duplicates. In contrast, if we consider false positives to be worse than false negatives, it make sense to return a low similarity score as aggregation output, because the two considered entities are only less similar in almost a half of the possible worlds and therefore we cannot exclude that they are true non-duplicates. Finally, if we consider false positives to be equally poor than false negatives, it seems to be most meaningful to return an average similarity as aggregation output because the likelihood that both entities have a high similarity (and hence are likely true duplicates) is roughly as large as the likelihood that these entities have a low similarity (and hence are likely true non-duplicates). Thus, the most meaningful aggregation output can vary a lot and strongly depends on the considered quality requirements.*

Since the given quality requirements depend on the considered applications, we need aggregation methods that can be configured to individual needs. Such a configuration, however, cannot be made by any of the methods presented above.

A simple mediating aggregation method that has been used by Lian and Chen in the context of similarity join computation [LC10] is to compute the accumulative weight of all scores that are above a given threshold θ, i.e.

$$agg = \sum_{v \in dom^*, v \geq \theta} relW(v) \tag{7.5}$$

By changing the threshold θ, we can incorporate quality requirements into the aggregation method. Using a low threshold corresponds to an audacious method (less false negatives) and using a high threshold corresponds to a cautious method (less false positives) . The shortcoming of this method is that it does not differentiate between the actual scores and can return same aggregation outputs for quite different

input sets. This may not be a disadvantage in similarity join computation, but can become a crucial problem in duplicate detection because the resultant similarity scores are often required in the duplicate clustering phase where we have to distinguish between more clear MATCHES and less clear MATCHES. For instance, if we use a threshold $\theta = 0.4$ the input set depicted in Figure 7.14(b) is aggregated to a similarity score of 1.0 which is the same aggregation output that would result for an input set that only contains the single similarity score 1.0. Nonetheless, whereas the latter set is an high indication for a duplicate, the former set is only an average indication for a duplicate. By using the aggregation method discussed above, however, we lose this information and thus cannot use it in the subsequent duplicate detection phases.

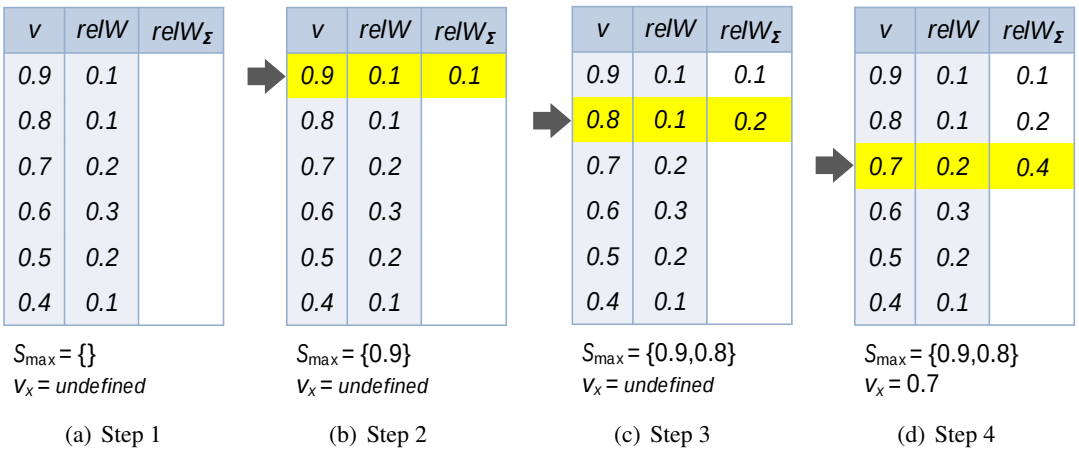
For that reason, we introduce two aggregation methods that are based on the idea of using the minimal (maximal) similarity score of the input set as aggregation output. Returning the minimal input score corresponds to the most audacious of all meaningful methods and returning the maximal input score corresponds to the most cautious of all meaningful methods. Nevertheless, the weights of the minimal (maximal) input score can be close to zero. Thence, we additionally introduce the threshold θ and do not consider the maximal possible similarity score as aggregation output, but compute the maximal weighted average similarity of all subsets of the input set that have an accumulative weight equal to θ instead. Due to the weights usually do not exactly sum up to θ and considering weights greater than θ can distort the aggregation output (see example below), the weight of the lowest selected similarity score is restricted to the difference between the accumulative weight of the other scores and θ.

The maximal-θ similarity can be computed as follows: First we sort all possible similarity scores in a descendent order. Then we select a score from the top of the sorted list as long as the accumulative weight of all selected scores is lower than θ and add it to the initially empty set S_{max}. After creating S_{max}, we select the next score from the top the sorted list that has a weight greater than zero and denote it as v_x. Note, since the weight of v_x is greater than zero, the weight $relW(S_{max} \cup \{v_x\})$ must be greater than θ, because otherwise v_x would have been added to S_{max} in the previous computation step. Finally, we compute the maximal-θ similarity as the weighted average of all selected similarity scores whereby the weight of v_x is only taken into account by the portion that is required to get an accumulative weight of exact θ, i.e.:

$$MaxSim_\theta = \frac{1}{\theta}\left(\sum\nolimits_{v \in S_{max}} relW(v) \times v + \big(\theta - relW(S_{max})\big) \times v_x\right) \tag{7.6}$$

The minimal-θ similarity is defined analogously.

Example 115 *For illustration, we compute the maximal-θ similarity for the second scenario from Figure 7.14(b) by using $\theta = 0.3$. In the first step, all similarity scores are sorted in an descending order. The sorting result is depicted in Figure 7.15(a). In the second step (see Figure 7.15(b)), we select the top score on the sorted list, set the accumulative weight $relW_\Sigma$ as the weight of the selected score, and add it to the set S_{max}. As presented in Figure 7.15(c), we then select the next score on the list and increase the accumulative weight by the score's weight in Step 3. Since the accumulative weight is still lower than θ, we add the selected score to S_{max}. In the fourth step, we select 0.7 as the next score on the list and again increase the accumulative weight (see Figure 7.15(d)). Because the accumulative weight becomes greater than θ, we do not add this score to S_{max}, but consider that score*



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

Figure 7.15.: Computation of the maximal-0.3 similarity on the second sample scenario

as v_x and stop the iteration. Finally, we compute the maximal-0.3 similarity by Equation 7.6, i.e.: $MaxSim_{0.3} = \frac{1}{0.3}\big(0.1 \times 0.9 + 0.1 \times 0.8 + (0.3 - 0.2) \times 0.7\big) = 0.8$.

Note, it is important that we take the score 0.7 only into account with the weight $\theta - relW(S_{max}$ instead of the weight $relW(0.7)$, because otherwise an input set with lower weights for the greatest values, e.g. one with $relW(0.9) = relW(0.8) = relW(0.7) = 1$, could result in a greater aggregation output.

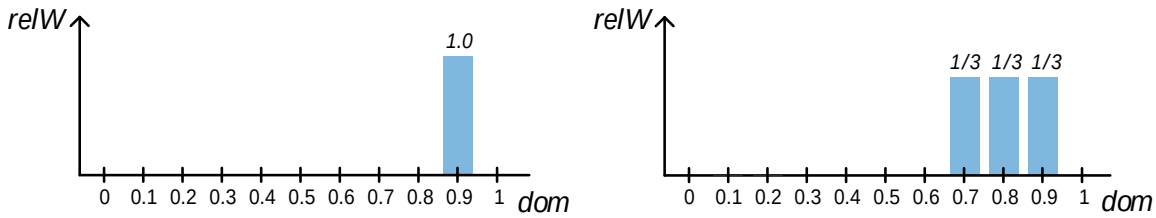
If we consider the first scenario from Figure 7.14(a), the maximal-0.3 similarity results in 0.9, because the weight of the largest element is already greater than θ and therefore is directly considered as v_x. As a consequence, the maximal-0.3 similarity is computed as $MaxSim_{0.3} = \frac{1}{0.3}\big(0 + (0.3 - 0) \times 0.9\big) = 0.9$.

Obviously, if false negatives are considered to be worse than false positives, we use the minimal-θ similarity as aggregation method and if false positives are considered to be worse than false negatives, we aggregate by using the maximal-θ similarity. The larger the used threshold, the more input values are considered for computing the aggregation output and therefore the more we approach the badness of both error classes. For instance by using $\theta = 1.0$, all input scores are taken into account and the aggregation output corresponds to the weighted average of all input scores, i.e. false negatives are considered to be equally poor than false positives. As a consequence, the threshold can be utilized to incorporate quality requirements into the aggregation method.

Interestingly, the minimal-θ similarity as well as the maximal-θ similarity can be considered from another point of view. Both methods are conform to a simple principle that first transforms the originally given input set into another set and then applies a conventional aggregation method to the transformed set. In the input transformation, we compute the set S_{max} and the value v_x and then define a new weighting function $relW_t$ that is defined as follows:

$$relW_t(v) = \begin{cases} relW(v)/\theta, & \text{if } v \in S_{max}, \\ (\theta - relW(v))/\theta, & \text{if } v = v_x, \\ 0, & \text{else.} \end{cases} \tag{7.7}$$

In the final score computation, we then consider the new input set $dom_t^* = \{v \mid v \in dom, relW_t(v) > 0\}$ and use an aggregation method that considered false positives and false negatives to be equally poor (e.g.



(a) Scenario 1 (b) Scenario 2

Figure 7.16.: Two input sets of weighted similarity scores

the weighted average). This principle can be illustrated on our two sample input sets from Figure 7.14(a) and Figure 7.14(b). Computing the maximal-0.3 similarity on these two sets corresponds to computing the weighted average similarity on the two input sets presented in Figure 7.16(a) and Figure 7.16(b) respectively.

As we will present in the following sections, this interpretation of the minimal-θ similarity and the maximal-θ similarity can be adopted to other domains. For that reason, we sometimes generalize their names to minimal-θ aggregation or the maximal-θ aggregation respectively.

7.2.7. Aggregation of Feature Scores

Each feature score consists of four vectors: (i) the attribute comparison vector, (ii) the attribute impact vector, (iii) the relationship comparison vector, and (iv) the relationship impact vector.

As for any other domain, we can distinguish between aggregation methods that are based on deciding strategies and aggregation methods that are based on mediating strategies. Of course, a convenient deciding method is to select the most weighted feature score as aggregation output. If several comparison vectors share the same values at the same position it can make sense to compute the most weighted value per vector position as aggregation output instead of the most weighted feature score.

If we use a mediating aggregation method or if we want to compute distances between feature scores we have to differentiate between the comparison vectors and the impact vectors because the first contain the actual similarities between the entities' features and the latter only model the significance of these similarity scores. It is obvious that an impact value should only contribute to the aggregation of a set of impact values, if the similarity score that is described by this impact value contributes to the aggregated similarity score that should be described by the aggregated impact value. For that reason, we aggregate the comparison vectors and the impact vectors separately where the aggregation of the impact vectors depends on the used aggregation of the comparison vectors.

A meaningful aggregation of impact values is debatable. In our work, we aggregate the impact values of a specific vector position by computing the weighted average of all impact values its corresponding similarity scores contributes to the aggregation output of the corresponding comparison vector position. For instance, if we use the first two of five similarity scores in the first position of the input attribute comparison vectors to compute the aggregated score that fill the first position of the output attribute comparison vector, we compute the aggregated impact value of the first position of the output attribute impact vector by the weighted average of the first two impact values of the input attribute impact vectors.

feature score	*attribute comparison vector*	*attribute impact vector*	*relW*
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.21
fs2	$\vec{c}_A$=[0.0,0.2]	$\vec{i}_A$=[0.7,0.8]	0.24
fs3	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]	0.15
fs4	$\vec{c}_A$=[0.33,0.2]	$\vec{i}_A$=[0.5,0.8]	0.14
fs5	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.16
fs6	$\vec{c}_A$=[0.5,0.5]	$\vec{i}_A$=[0.4,0.9]	0.10

Figure 7.17.: Sample input set of feature scores

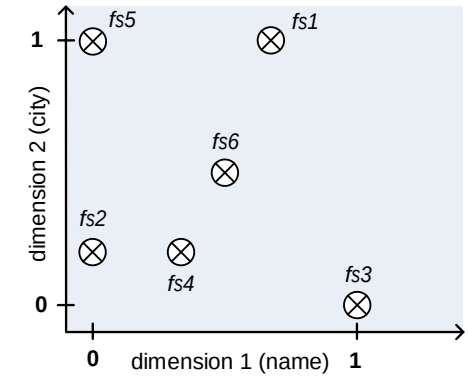
The values of the two comparison vectors represent the similarity of the two corresponding entities and can be considered as coordinate values in a coordinate space. Therefore, aggregation can be done by first creating a high dimensional space that has one dimension per position of any of the comparison vectors, then by mapping each feature score as a point into that space, and lastly by computing the centroid of all points. Finally, either the centroid (mediating method) or the feature score that is nearest to the centroid (deciding method) is used as aggregation output.

Since the individual attribute pairs and relationship pairs can have a difference significance for the detection process, the individual dimensions can be stretched or compressed by user defined weights. In that case, it make sense to use the same weights that would be used in a distance-based decision model (see Section 4.3.6.1).

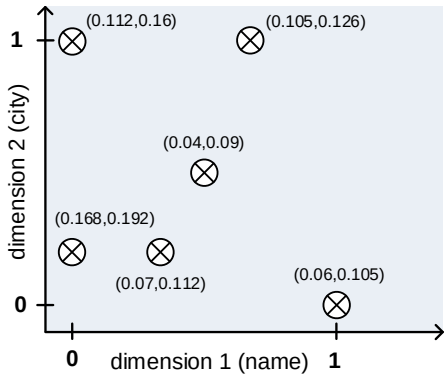
Moreover, we have the two impact vectors and the weights of the individual input feature scores. All these values describe the significance of the whole feature score or describe the significance of a similarity score in any of the comparison vectors respectively. Thus, we can compute an individual weighting scheme (one weight per dimension) for each input feature score.

Example 116 *To demonstrate this aggregation method, we consider the six feature scores presented in Figure 7.17. For ease of presentation but without loss of generality, we restrict our consideration on the attribute comparison vector and the attribute impact vector and therefore omit the relationship comparison vector and the relationship impact vector from the presented feature scores. Since the vectors have a length of two, we create the 2-dimensional space shown in Figure 7.18(a). For illustration purposes we assume that the first vector positions refer to an attribute* 'name' *and the second vector positions refer to an attribute* 'city'. *Each of the six feature scores is represented by a point in that space. In the next step, we compute the dimension weights for the individual feature scores. These weights are represented in Figure 7.18(b) by brackets above/below the corresponding points where the first value is the weight of the first dimension and the second value is the weight of the second dimension. For instance, the weight of feature score fs1 for the attribute* 'name' *is computed by multiplying the first value of the attribute impact vector with the feature score's weight, i.e.* $w(fs1, \text{'name'}) = \vec{i}_A(fs1)[0] \times relW(fs1) = 0.5 \times 0.21 = 0.105$. *Finally, we compute the weighted mean (red point) that has the coordinates* $(0.305, 0.518)$ *(see Figure 7.18(c)).*

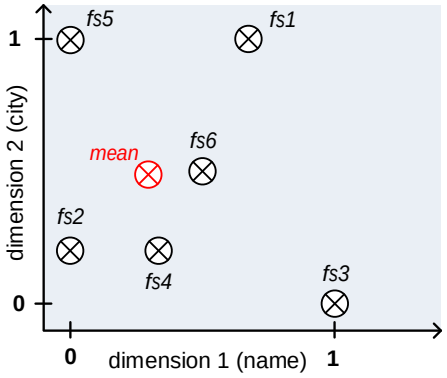
In the mediating method, we select the mean point as the representative feature score. In the deciding method, we select the feature score that has the shortest distance to this mean. Due to both attributes



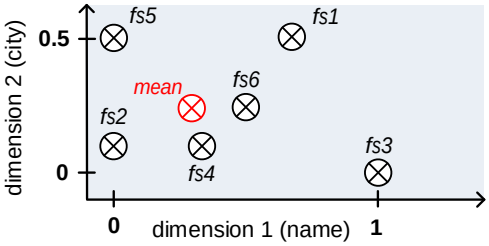
(a) 2-dimensional space with one point per feature score



(b) Dimension weights per feature score



(c) The weighted mean (red point)



(d) Space after compressing dimension 2

Figure 7.18.: Sample for aggregating feature scores

are different important for the detection process (in our example we assume the weight 1.0 *for attribute* 'name' *and assume the weight* 0.5 *for attribute* 'city'*), we initially need to compress the second dimension that represents the attribute*'city' *before we compute the distances. As we can see in Figure 7.18(d), the feature score with the shortest distance to the mean is* fs_4*. Interestingly, without compressing the second dimension, the feature score with the shortest mean distance is* fs_6*.*

In the 1-dimensional space, this aggregation method is equivalent to compute the weighted average.

7.2.7.1. Incorporation of Quality Requirements

Of course, in the aggregation of feature scores we need to utilize the same trade-off as in the aggregation of similarity scores. For that reason, we adopt the underlying ideas of the minimal-θ aggregation and the maximal-θ aggregation from the domain of similarity scores to the domain of feature scores. For that purpose, we require an ordering relation on the domain of feature scores. Let n be the number of considered dimensions, the point $\mathbf{0_n} = (0, 0, \ldots, 0)$ represents the lowest possible similarity between two entities and the point $\mathbf{1_n} = (1, 1, \ldots, 1)$ represents the highest possible similarity between two

(a) Step 1 (b) Step 2 (c) Step 3

Figure 7.19.: Computation of the maximal-0.5 aggregation

feature score	attribute comparison vector	attribute impact vector	$relW_t$
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.42
fs3	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]	0.30
fs5	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.08
fs6	$\vec{c}_A$=[0.5,0.5]	$\vec{i}_A$=[0.4,0.9]	0.20

Figure 7.20.: Transformed set of input feature scores

entities. As a consequence, we can sort the feature scores by their distance to one of these two points. For distance computation several measures are conceivable as for example the Euclidean Distance or the Manhattan Distance. If we select the point $\mathbf{0_n}$ as reference, a feature score fs_r is sorted in front of a second feature score fs_s if the distance between fs_r and $\mathbf{0_n}$ is larger than the distance between fs_s and $\mathbf{0_n}$, i.e. $fs_r > fs_s \Rightarrow dst(fs_r, \mathbf{0_n}) > dst(fs_s, \mathbf{0_n})$. Of course, different points in the coordinate space (and hence different feature scores) can have the same distance. In that case we need a tie breaker as for example a method that sort feature scores by the value at the first positions of their attribute comparison vectors.

By using this ordering relation, the maximal-θ aggregation (and hence the minimal-θ aggregation) of feature scores can be computed accordingly to the domain of similarity scores. We first sort all feature scores by the aforementioned ordering relation in descending order (or ascending order respectively). Then we compute the set S_{max} and the value v_x by selecting scores from the top of the sorted list until the accumulative weight of the selected scores exceeds the given threshold. Finally, we transform the given weight function as defined in Equation 7.7 and use a conventional feature score aggregation method (e.g. the space-based aggregation method presented above) to compute the aggregation output. As for the domain of similarity scores, we can utilize the required trade-off between recall and precision by changing the used threshold θ.

Example 117 *For illustration, we reconsider the six feature scores from Figure 7.17 and assume that we have an application that considers false negatives to be much worse than false positives. For that*

reason, we use the maximal-0.5 *aggregation, i.e.* $\theta = 0.5$*. First we sort all six feature scores by their distance to the point* $\mathbf{0_2}$ *(note that we use the compressed space for distance computation). The sorted list is presented in Figure 7.19(a). Then we select the feature score* fs_3 *from the top of the list, add it to the set* S_{max} *and set the accumulative weight to the weight of* fs_3*, i.e. to the value* 0.15*. Afterward, we do the same with the feature scores* fs_1 *and* fs_6 *so that the accumulative weight increases to* 0.46*. The result is presented in Figure 7.19(b). The next feature score on the sorted list is* fs_5*. Since its weight is* 0.16*, the newly computed accumulative weight exceeds the given threshold. Thus, we set the value* v_x *to* fs_5 *and stop the iteration (see Figure 7.19(c)). The transformed weighting function as well as the feature scores with a non-zero weight are presented in Figure 7.20. Finally, we use the space-based aggregation method to compute the mean of all scores by using the transformed weighting function. In this case, the mean results in the coordinates* $(0.645, 0.584)$ *and is used as the attribute comparison vector of the aggregation output (mediating strategy). The impact vector of the aggregation output is computed by the weighted average of the impact values of the feature scores* fs_1*,* fs_3*,* fs_5*, and* fs_6 *using the transformed weighting function. By doing so, the impact value of the first vector position results in* 0.466 *and the impact value of the second vector position results in* 0.77*. As a consequence, the aggregation output is a feature score with an attribute comparison vector* $\vec{c}_A = \langle 0.645, 0.584 \rangle$ *and an attribute impact vector* $\vec{i}_A = \langle 0.466, 0.77 \rangle$*.*

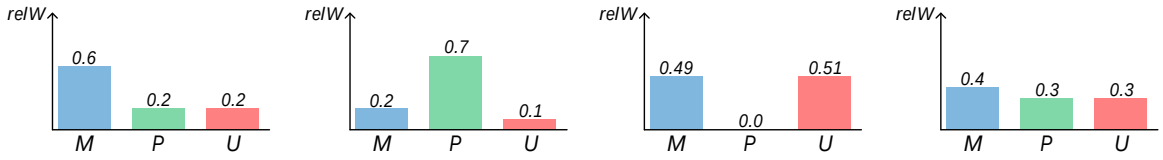
In Example 116, we automatically used the standard quality requirements (false negatives are equally poor as false positives). In this case, the aggregation output was a feature score with the attribute comparison vector $\vec{c}_A = \langle 0.305, 0.518 \rangle$*. Thus, in our example using the maximal-*0.5 *aggregation considerably increases the similarity scores of the comparison vector compared to the standard aggregation method. Since we consider false negatives to be worse than false positives this was our actual intention for using the maximal-*0.5 *aggregation.*

7.2.8. Aggregation of Matching Classes

We consider three matching classes in this thesis, namely MATCH (short M), POSSIBLE MATCH (short P) and UNMATCH (short U). Thus, methods for aggregating matching classes are defined on the domain $dom = \{M, P, U\}$. Sometimes the subsequently utilized phases of the detection process cannot work on POSSIBLE MATCHES. In that case, the the output domain is considered to only consists of MATCH and UNMATCH.

As in the previous two sections, we first consider aggregation methods by assuming the standard quality requirement (false positives are considered to be equally poor as false negatives) and then consider an incorporation of specific quality requirements into the aggregation methods afterwards.

In contrast to similarity scores and feature scores, we cannot compute a kind of weighted average in the domain of matching classes, but require for other aggregation approaches. Although this domain only consists of three elements, a meaningful aggregation is not simply to define because we have to consider a second trade-off, i.e. the compromise between the number of ambiguous made decisions and the number of clerical reviews. If we want to reduce the number of ambiguous made decisions, i.e. pairs that are classified as MATCHES or UNMATCHES with only less confidence, we automatically increase the number of POSSIBLE MATCHES and thus usually increase the number of clerical reviews required.



(a) Input set S_1: Likely M (b) Input set S_2: P if possible (c) Input set S_3: U or P? (d) Input set S_4: M or P?

Figure 7.21.: Four sample input sets for matching class aggregation

Clerical reviews, however, are expensive (monetary costs as well as time) and hence often need to be avoided as much as possible.

An aggregation of matching classes without specific quality requirements can be done by a fixed formula or can be done by a learned classifier (Decision Tree, Support Vector Machine, etc.) as we have presented in Section 4.3.6.4.

Example 118 *To illustrate the resemblance between the proposed aggregation methods and the aggregation output that we intuitively consider to be most meaningful if false negatives and false positives are considered equally poor, we consider the four sample scenarios depicted in Figure 7.21. In the first scenario, the most meaningful aggregation output is somewhat clear, because the weight of* MATCH *is much greater than these of* POSSIBLE MATCH *and* UNMATCH*. In the second scenario, the result is similar, because the weight of* POSSIBLE MATCH *dominates the weights of* MATCH *and* UNMATCH*. Thus, if* POSSIBLE MATCH *is an option for the aggregation output it should be used and* MATCH *(more audacious) or* UNMATCH *(more cautious) otherwise. In the third scenario,* UNMATCH *has a higher weight than* MATCH*, but the difference is really close. Thus, taking* UNMATCH *as aggregation output would imply a high risk of producing a false negative. For that reason, taking* POSSIBLE MATCH *as aggregation output is maybe the better option, even if the weight of* POSSIBLE MATCH *is zero. In the fourth scenario, although the relative weight of* MATCH *is the greatest one, the weights of all three matching classes are close to each other. Thus, taking* MATCH *as aggregation output would imply a high risk of producing a false positive and* POSSIBLE MATCH *is maybe be the better option.*

In the following, we present three methods for aggregating matching classes. We start with a simple an intuitive one and then proceed with some extensions. To explain the ideas behind these methods, we compare the output they produce on the four sample input sets from Figure 7.21 with the outputs we intuitively think they are best.

- The most intuitive aggregation is to classify two entities as a MATCH, if the weight of MATCH is greater than the weight of UNMATCH. In order to avoid a certain decision in ambiguous situations where the two weights are nearly equivalent and to enable a trade-off between recall and precision, we additionally introduce the two fixed values $\alpha_1, \alpha_2 \in \mathbb{R}$ so that $\alpha_1, \alpha_2 \geq 1$.

$$agg = \begin{cases} M, & \text{if } relW(M) \geq \alpha_1 \times relW(U), \\ U, & \text{else if } relW(U) > \alpha_2 \times relW(M), \\ P, & \text{else.} \end{cases} \tag{7.8}$$

Increasing α_1 implies to decrease the number of MATCHES (and hence likely implies to decrease the number of false positives) to the price of additional POSSIBLE MATCHES. Similarly, increasing α_2 implies to decrease the number of UNMATCHES (and hence the number of false negatives), but increases the number of POSSIBLE MATCHES. Using the setting $\alpha_1 = 1$ and $\alpha_2 = 1$ excludes the set of POSSIBLE MATCHES and the aggregation output is always either MATCH or UNMATCH respectively.

- As presented in our second sample scenario, it seems critical to classify two entities as a MATCH or UNMATCH, if the weight of POSSIBLE MATCH is much larger than the accumulative weight of the other two classes. For that purpose it can make sense to extent the formula from Equation 7.8 by the additional condition that the accumulative weight of MATCH and UNMATCH need to be greater than a specific threshold $\theta_{M+U} \in [0, 1]$ in order to produce MATCH or UNMATCH as aggregation output, i.e.:

$$agg = \begin{cases} M, & \text{if } \mathit{relW}(M) \geq \alpha_1 \times \mathit{relW}(U) \wedge \mathit{relW}(\{M, U\}) \geq \theta_{M+U}, \\ U, & \text{if } \mathit{relW}(U) > \alpha_2 \times \mathit{relW}(M) \wedge \mathit{relW}(\{M, U\}) \geq \theta_{M+U} \\ P, & \text{else.} \end{cases} \tag{7.9}$$

  Obviously, this aggregation method can be specialized to the method presented in Equation 7.8 by setting $\theta_{M+U} = 0$.

- Sometimes the set of POSSIBLE MATCHES does not need to be solved manually, but the duplicate detection process uses a duplicate clustering approach that can deal with POSSIBLE MATCHES as input. In that case, the number of POSSIBLE MATCHES is only less strongly restricted and it can make sense to classify two entities only as a MATCH or an UNMATCH, if the corresponding weight represents the absolute majority. This means that even by assuming the worst case, i.e. each POSSIBLE MATCH is actually an UNMATCH (or MATCH respectively), the weight of MATCH (or UNMATCH respectively) is greater than its opposite:

$$agg = \begin{cases} M, & \text{if } \mathit{relW}(M) \geq \alpha_1 \times \mathit{relW}(\{U, P\}), \\ U, & \text{if } \mathit{relW}(U) > \alpha_2 \times \mathit{relW}(\{M, P\}), \\ P, & \text{else.} \end{cases} \tag{7.10}$$

  In this case, instead of setting α_1 and α_2 to two fix values, we can compute these values individually based on the relative amounts of the weights. For instance, it make sense to compute α_1 based on the amount of $\mathit{relW}(U)$ to $\mathit{relW}(\{U, P\})$, Note, since $\mathit{relW}(\{M, U, P\}) = 1$, $\alpha_1 \geq 1$, and $\alpha_2 \geq 1$, the additional condition $\mathit{relW}(\{M, U\}) \geq \theta_{M+U}$ from Equation 7.9 can be omitted for the most thresholds θ_{M+U}, because if $\mathit{relW}(M) \geq \mathit{relW}(\{U, P\})$ it follows that $\mathit{relW}(M) \geq 0.5$ and $\mathit{relW}(P)$ cannot be the dominating weight.

Example 119 *Now, we consider the presented aggregation methods with respect to the four sample input sets presented in Figure 7.21. For a configuration setting $\alpha_1 \leq 1.5$ and $\theta_{M+U} \leq 0.8$ all three*

methods compute a MATCH *for the first set. By using the same configuration settings, the first aggregation method computes a* MATCH*, but the second and the third aggregation methods each produces a* POSSIBLE MATCH *for set S_2. Considering set S_3, all methods produce a* POSSIBLE MATCH *by using a configuration setting with $\alpha_2 > 1$. For a configuration setting $\alpha_1 \leq 1.3$ and $\theta_{M+U} \leq 0.7$ the first two methods compute a* MATCH*, but the third method produces a* POSSIBLE MATCH *for the fourth sample set. As we can see, the first method is most careless, because it prefers a* MATCH *or an* UNMATCH *to a* POSSIBLE MATCH *already for low indications. In contrast, the second method is more carefully and produces a* MATCH *or an* UNMATCH *as aggregation output less often. Finally, the third method is the most carefully of all the three methods, because it returns a* MATCH *or an* UNMATCH *only in clear cases.*

As mentioned above, machine learning techniques can be used to learn an appropriate aggregation method. In this case, the input to the learning model are 3-dimensional vectors where each of the vectors' positions contains the weight of one of the three matching classes, e.g. the first position represents the weight of MATCH, the second position represents the weight of POSSIBLE MATCH, and the third positions represents the weight of UNMATCH.

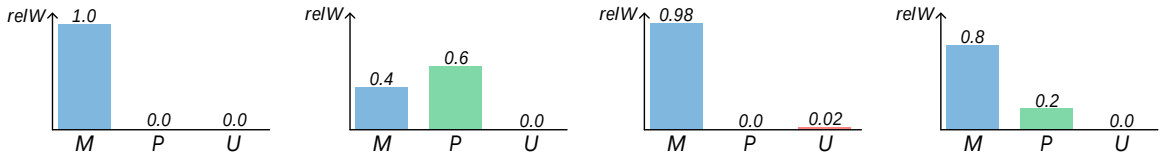
Besides aggregating several matching classes to a single matching class, it can be useful to initially derive a similarity score from the given matching classes and then to use one or two thresholds to determine the matching class that serves as aggregation output. A simple approach to derive a single similarity score from a set of weighted matching classes is to use the accumulative weight of MATCH and POSSIBLE MATCH as similarity where the influence of POSSIBLE MATCH is restricted by the factor $\beta \in [0, 1]$.

$$sim = \textit{relW}(M) + \beta \times \textit{relW}(P) \tag{7.11}$$

In general, by initially mapping each matching class to a similarity score, each of the methods that we have presented for aggregating similarity scores in Section 7.2.6 can be used for aggregating matching classes as well. For instance, the above presented method for similarity computation (Equation 7.11) can be considered as computing the weighted average similarity (see Equation 7.4), by initially mapping $M \mapsto 1$, $P \mapsto \beta$ and $U \mapsto 0$.

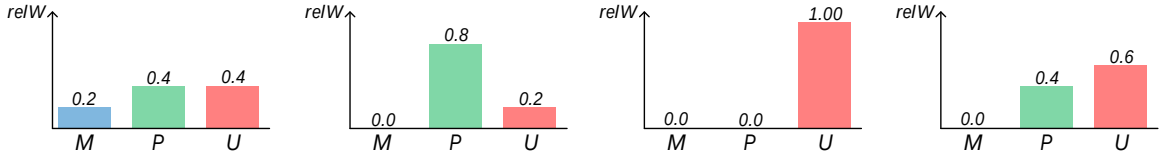
7.2.8.1. Incorporation of Quality Requirements

In the previous part of this section, we have discussed several methods for aggregating matching classes that consider false positives to be equally poor than false negatives. Now, we consider an incorporation of quality requirements into the aggregation process. The most audacious aggregation method is to return a MATCH if any of the input values is a MATCH, i.e. if $\textit{relW}(M) > 0$, and the most cautious method is to return an UNMATCH if any of the input values is an UNMATCH, i.e. if $\textit{relW}(U) > 0$. To enable an incorporation of a trade-off between these two extremes into the aggregation process, we adopt the minimal-θ aggregation and the maximal-θ aggregation that we have already used for the domain of similarity scores and the domain of feature scores. For that purpose, we sort all matching classes (descendant or ascendant), select some of these classes from the top of the list, recompute the weights of all matching classes based on the weights of the selected ones and finally perform any of the aggregation method that we have presented above by using the recomputed weights. The ordering relation on the domain of matching classes is obviously defined as MATCH $>$ POSSIBLE MATCH $>$ UNMATCH. Since



(a) Transformed input set S_1 (b) Transformed input set S_2 (c) Transformed input set S_3 (d) Transformed input set S_4

Figure 7.22.: Transformations of the sample input sets S_1 to S_4 from Figure 7.21 with the maximal-0.5 aggregation



(a) Transformed input set S_1 (b) Transformed input set S_2 (c) Transformed input set S_3 (d) Transformed input set S_4

Figure 7.23.: Transformations of the sample input sets S_1 to S_4 from Figure 7.21 with the minimal-0.5 aggregation

we illustrate the execution process of the maximal-θ aggregation already for the domains of similarity scores and feature scores, we eschew such a presentation for the domain of matching classes, but instead show some input set transformations that result from using the maximal-θ aggregation and the minimal-θ aggregation with different thresholds.

Example 120 *For demonstrating the effect of the maximal-θ aggregation and the minimal-θ aggregation in the domain of matching classes, we depict the sets that result from transforming the sample input sets S_1 to S_4 from Figure 7.21 by using the maximal-0.5 aggregation in Figure 7.22 and by using the minimal-0.5 aggregation in Figure 7.23.*

It is simple to see that the maximal-0.5 aggregation extracts the positive part of the input sets and the minimal-0.5 aggregation extracts the negative part of the input sets. This difference is especially significant for input set S_3. In the original input set, the evidence that the considered entity pair is a MATCH *is equal to the evidence that this pair is an* UNMATCH*. Thus, from an audacious point of view the evidence that this pair is a* MATCH *is sufficient to take* MATCH *as an aggregation output. In contrast, from a cautious point of view the evidence that this pair is an* UNMATCH *is large enough to take* UNMATCH *as aggregation output.*

7.2.9. Aggregation of Duplicate Decisions

A duplicate decision is a triple that consists of a matching class, a similarity score, and an impact value. We aggregate a set of duplicate decisions by aggregating their matching classes, similarity scores, and impact values separately. Of course, it make sense to use aggregation methods for each of the three domains that comply similar aggregation pattern (e.g. methods that assume the same quality requirements).

As for feature scores, we aggregate the impact values by computing the weighted average of all values its corresponding matching classes and similarity scores contribute to the aggregated matching class or the aggregated similarity score respectively.

For incorporating quality requirements into the aggregation method we adopt the concepts of the minimal-θ aggregation and the maximal-θ aggregation. For that purpose, we order duplicate decisions by their matching classes (first order criterion), their similarity score (second order criterion) and their impact value (third order criterion). For instance, the duplicate decision $(M, 0.8, 1.0)$ is ordered in front of the duplicate decision $(M, 0.6, 1.0)$ which in turn is ordered in front of the duplicate decision $(P, 0.8, 1.0)$.

A special resolution method for a probabilistic duplicate decision is to compute the probability that the corresponding entity pair $\{e_r, e_s\}$ is a MATCH by simply summing up the probabilities of all alternatives that have a MATCH as matching class. By doing so we actually do not resolve uncertainty completely, but quantify it by a simple number (instead of a probabilistic value). This so called matching probability $Pr(\{e_i, e_j\} \in \text{MATCH})$ can then be used as input to a duplicate clustering approach that we present in Section 8.7. Note, by utilizing aggregation methods for uncertainty resolution, computing the matching probability corresponds to an aggregation method that maps a set of duplicate decisions to a normalized real value by summing up the relative weights of all input decisions that have a MATCH as matching class.

7.2.10. Aggregation of Clusterings

The last moment of uncertainty resolution is to aggregate a set of duplicate clusterings that are each cluster-disjoint. Aggregating a set of cluster-disjoint clusterings with a varying numbers of clusters has been considered in several works [FS03, GMT07, WWL09]. A simple deciding strategy is to select the clustering with the largest weight as aggregation output. However, it is simply to see that this approach can produce an aggregation output of a low quality in many scenarios. Moreover, it can happen that all input clusterings are weighted equally.

A more promising aggregation method is to compute a clustering that has the shortest distance to all input clusterings. Gionis et al. [GMT07] consider a distance measure that computes the number of pairwise disagreements, i.e. pairs whose elements are placed in the same cluster by one clustering and are placed in different clusters by the other clustering. In contrast, Wang et al. [WWL09] uses the Bregman divergence as distance measure. Nevertheless, there are many other distance measures (or similarity measures respectively) that can be used for the purpose of computing a distance between two clusterings. Examples are the Basic Merge Distance [AKE04] and the Rand Index [Ran71] that we have both presented in Section 4.3.8 in the context of evaluating duplicate detection quality.

Of course, this approach can be stretched to the whole input domain or can be restricted to the set of input clusterings. In the first case, the output clustering must not be part of the aggregation input and hence the aggregation method is mediating. However, computing the clustering of the considered range that has the minimal distance to a set of input clusterings of the same range is a problem that is known to be NP-complete [FS03]. In contrast, if we restrict the possible aggregation outcomes to the input clusterings, the aggregation method becomes deciding and its complexity is only quadratic in the number of input clusterings.

Example 121 *For illustration, we consider the three input clusterings $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ as well as their corresponding weights as presented in Figure 7.24(a). Figure 7.24(b) shows six plausible aggregation*

Input Clustering	***relW***
$\mathcal{C}_1 = \{\langle e_1, e_2\rangle, \langle e_3\rangle, \langle e_4, e_5\rangle\}$	0.4
$\mathcal{C}_2 = \{\langle e_1, e_2\rangle, \langle e_3, e_4\rangle, \langle e_5\rangle\}$	0.35
$\mathcal{C}_3 = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3, e_4, e_5\rangle\}$	0.25

(a) Input clusterings

Plausible Aggregation Output	**Distance to $\mathcal{C}_1$**	**Distance to $\mathcal{C}_2$**	**Distance to $\mathcal{C}_3$**	**Weighted Distance**
$\mathcal{C}_{A1} = \{\langle e_1, e_2\rangle, \langle e_3\rangle, \langle e_4, e_5\rangle\}$	0	1 *merge* + 1 *split*	1 *merge* + 1 *split*	$2 \times 0.35 + 2 \times 0.25 = 1.2$
$\mathcal{C}_{A2} = \{\langle e_1, e_2\rangle, \langle e_3, e_4\rangle, \langle e_5\rangle\}$	1 *merge* + 1 *split*	0	1 *merge* + 1 *split*	$2 \times 0.4 + 2 \times 0.25 = 1.3$
$\boldsymbol{\mathcal{C}_{A3} = \{\langle e_1, e_2\rangle, \langle e_3, e_4, e_5\rangle\}}$	**1 *merge***	**1 *merge***	**1 *merge***	**0.4+0.35 +0.25 = 1.0**
$\mathcal{C}_{A4} = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3\rangle, \langle e_4, e_5\rangle\}$	1 *split*	1 *merge* + 2 *split*	1 *split*	$0.4 + 3 \times 0.35 + 0.25 = 1.65$
$\mathcal{C}_{A5} = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3, e_4\rangle, \langle e_5\rangle\}$	1 *merge* + 2 *split*	1 *split*	1 *split*	$3 \times 0.4 + 0.35 + 0.25 = 1.8$
$\mathcal{C}_{A6} = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3, e_4, e_5\rangle\}$	1 *merge* + 1 *split*	1 *merge* + 1 *split*	0	$2 \times 0.4 + 2 \times 0.35 = 1.5$

(b) The merge distances of the plausible aggregations outputs

Figure 7.24.: Sample for aggregating a set of three clusterings

output clusterings along with their basic merge distances (presented in the number of merge operations and the number of split operations that are required to transform this output clustering into the considered input clustering) to the three input clusterings. Clustering $\mathcal{C}_{A3}$ has the minimal weighted distance to all three input clusterings and therefore seems to be the most appropriate aggregation output. Since this clustering does not belong to the input set, choosing $\mathcal{C}_{A3}$ as aggregation output corresponds to a mediating strategy. In contrast, if we want to restrict the aggregation output to the three input clusterings (maybe because of the aforementioned processing reasons), the clustering $\mathcal{C}_{A1} = \mathcal{C}_1$ has the lowest weighted distance to the other input clusterings and therefore likely represents the set of input clusterings best.

7.2.10.1. Incorporation of Quality Requirements

Of course, quality requirements can be incorporated into the aggregation of clusterings in the same way as in the four previously discussed domains, i.e. by adopting the concepts of the minimal-θ aggregation and the maximal-θ aggregation. Thus, we first select a subset of all input clusterings whose accumulative weight is at least as large as the given threshold and then compute a new weight function from the selected clusterings. An aggregation method is most audacious if it selects the clusterings with the most MATCHES and is most cautious if it selects the clusterings with the least MATCHES. Therefore, for computing the maximal-θ aggregation (or minimal-θ aggregation respectively) we sort the input clusterings by their number of MATCHES in decreasing order (or ascending order respectively). The number of MATCHES modeled within a duplicate clustering can be either exactly computed as presented in Section 4.3.8 or the number of clusters can be used as an indication for it.

Another way for incorporating quality requirements into clustering aggregation is to use distance measure that can be adjusted to individual needs. Many distance measures as the Basic Merge Distance or the the number of pairwise disagreements do not distinguish between false positives and false negatives

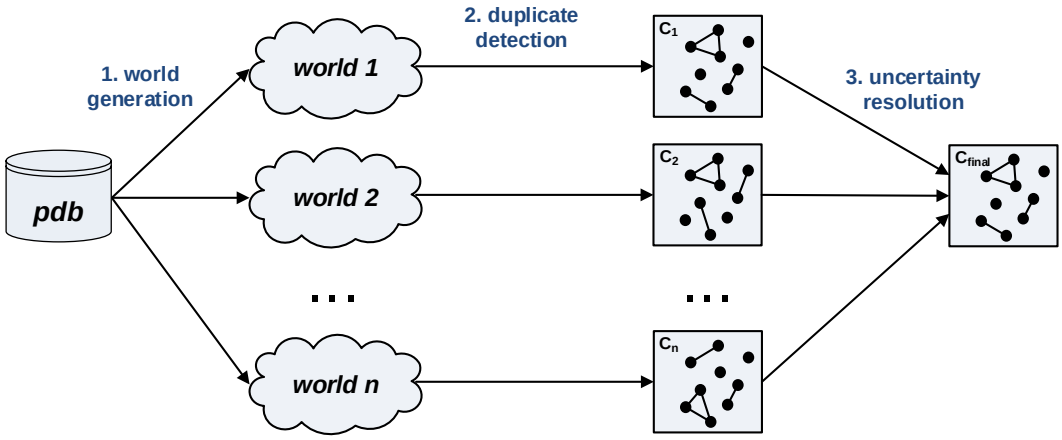


Figure 7.25.: The principle idea of the world-based approach for duplicate detection in uncertain data

and therefore cannot be adjusted to a specific quality requirement. However, some distance measures are adjustable. For instance, the Generalized Merge Distance can be adjusted to specific quality requirements by utilizing the two cost functions f_m (costs of a cluster merge) and f_s (cost of a cluster split).

7.3. World-based Approach

With the world-based approach we follow the traditional idea of the possible worlds semantics and hence consider uncertainty on database level by performing a certain duplicate detection approach in each of the database's possible worlds separately.

The principle idea of the world-based approach is depicted in Figure 7.25. First from the uncertain database, the possible worlds representation is generated. Then in each of the possible worlds the same process for duplicate detection in certain data is performed separately. Since we require a single duplicate clustering as a result, remaining uncertainty is finally resolved by aggregating the duplicate clusterings that have been computed from the individual worlds.

Thus, let f_{UR} be the function that is used for resolving an uncertain clustering and let δ_{det} be a deterministic duplicate detection process that is defined for certain data, the world-based approach can be considered as a function δ^U_{det} that maps a probabilistic database $pdb = (\mathbf{W}, Pr)$ to a cluster-disjoint clustering, i.e.:

$$
\begin{aligned}
& \delta^U_{\text{det}}(pdb) = f_{UR}(\mathbf{W}_\delta, Pr_\delta) \\
\text{where} \quad & \mathbf{W}_\delta = \{\delta_{\text{det}}(W) | W \in \mathbf{W}\} \\
\text{and} \quad & \forall W \in \mathbf{W}_\delta \colon Pr_\delta(W) = \sum\nolimits_{W' \in \mathbf{W}, \delta_{\text{det}}(W') = W} Pr(W')
\end{aligned}
$$

The challenge for accomplishing this idea is twofold:

(a) an aggregation of whole clusterings is often not useful for reasons of efficiency as well as effectiveness and resolving uncertainty at an earlier moment in the detection process is therefore more suitable.

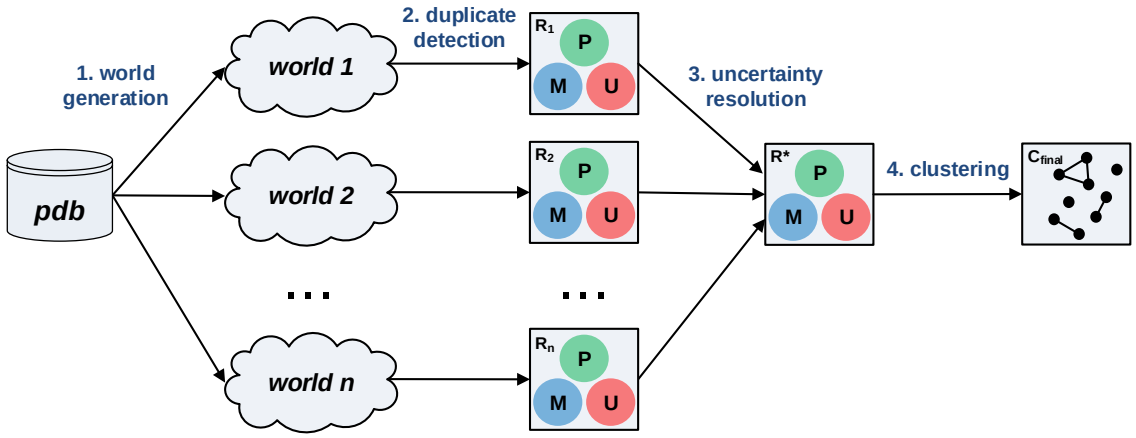


Figure 7.26.: World-based approach with uncertainty resolution after the decision model phase

(b) due to the number of possible worlds can be considerably large, performing duplicate detection in each world separately is computational intractable,

We address both challenges in depth in the following two subsections.

7.3.1. Uncertainty Resolution

The underlying idea of the world-based detection approach is to perform the complete detection process in each of the worlds separately and finally to aggregate the produced clusterings to a single one. Nevertheless, it can be useful to resolve uncertainty at an earlier moment of the detection process.

As a consequence, we can generalize the original idea of the world-based detection approach into a three step procedure. First duplicate detection is processed in all worlds separately until a predefined moment, then the results of all these worlds are aggregated, and finally we perform the remaining detection phases on the aggregation output by using a conventional certain data approach. This principle of generalization is exemplary presented for an aggregation of duplicate decisions in Figure 7.26. First, in each world the duplicate decisions for all entity pairs are computed (Step 1 and Step 2), these decisions are then aggregated for each entity pair individually (Step 3). The result of this resolution process are a single set of MATCHES, a single set of POSSIBLE MATCHES, and a single set of UNMATCHES. Since we require a duplicate clustering as output, a conventional clustering phase is lastly performed to produce a final detection result (Step 4).

In summary, the uncertainty of the input data can be resolved at six different moments of the detection process.

- **World Aggregation:** The earliest moment for uncertainty resolution is to aggregate all possible worlds of the probabilistic input database to a single world and hence to transform the probabilistic database into a certain database. Since the transformed input data is certain, we can use any conventional duplicate detection approach to compute the output clustering. As we have discussed in Section 7.2.4, an aggregation of worlds can be accomplished by selecting one of the existing worlds or to compute a new world that does not belong to the input set, but represents these worlds best.

Whereas a duplicate detection process that resolves uncertainty at this moment is extremely efficient, because it is only marginally more complex than a conventional duplicate detection process performed on a certain database, its effectiveness is questionable, because any kind of uncertainty is ignored in making the individual duplicate decisions.

- **Entity Description Aggregation:** The second earliest moment of uncertainty resolution is to generate all possible worlds, to extract a description for each entity per world and then to aggregate all the descriptions of one entity. Finally, we use all entities along with their aggregated descriptions as input to a conventional duplicate detection process. Recall, we presents methods for aggregating entity descriptions in Section 7.2.5.

- **Feature Score Aggregation:** The next moment of uncertainty resolution is after the feature matching phase and hence to aggregate the feature scores of each entity pair. For the purpose of uncertainty resolution, we can utilize any of the aggregation methods that we have proposed in Section 7.2.7. In this case, the world-based computation of the duplicate detection process stops after the similarity computation phase, we aggregate the similarity scores for each candidate pair to a single one, and then proceed with conventional approaches for similarity-based classification and clustering.

- **Similarity Score Aggregation:** The fourth moment of uncertainty resolution is to aggregate all the similarity scores that are computed for each entity pair by using any of the aggregation methods discussed in Section 7.2.6. In that case, the feature matching phase and the similarity computation phase are performed in each world separately. After aggregating the similarity scores of each entity pair, we use a conventional classification phase and a conventional duplicate clustering phase to compute the final clustering.

- **Duplicate Decision Aggregation:** The second last moment of resolution is the one we have depicted in Figure 7.26. First we perform the detection process in each world separately until the end of the classification phase, aggregate the duplicate decisions of each entity pair to a single one, and then use a conventional duplicate clustering phase to compute the final detection result based on the aggregated decisions. Recall, methods for aggregating duplicate decisions are presented in Section 7.2.9.

  Since each clustering can be transformed into a set of pairwise matching classes, functions for aggregating duplicate decisions can be also applied to the final detection results. In that case, however, we need a method that computes similarity scores from the set of possible clusterings. One option is to use the similarity scores from the duplicate decisions that have been served as input to the clustering phases. Another option is to use the accumulative weight of all clusterings that provide a MATCH for the considered entity pair as its entity similarity.

- **Clustering Aggregation:** The last possible moment of uncertainty resolution is the originally intended one. Thus, we completely perform the duplicate detection approach in each world separately and then use one of the aggregation methods presented in Section 7.2.10 to aggregate the clusterings that have been computed from the individual worlds to a single one.

SELECT	$t1.DEI, t2.DEI$
FROM	*db* $t1$, *db* $t2$
WHERE	$t1.RK < t2.RK$
AND	$t1.DEI \neq t2.DEI$
AND	$sim(t1, t2) > \theta$;

Figure 7.27.: A sample query representing an attribute-based duplicate detection on a single-table database *db* by using a tuple similarity measure sim and a user defined threshold θ

Compared to an aggregation of pairwise matching results or an aggregation of entity descriptions, an aggregation of whole clusterings has two shortcomings. First, an aggregation of clusterings can become very expensive. Second, we compute only one world impact vector for the set of all database entities. In contrast, if we aggregate pairwise matching results we can compute an individual world impact vector per entity pair. Thus, we can react on the specific membership information of both entities and therefore can better compensate the unintended influence of membership information than we can do it in an aggregation of whole clusterings.

Of course, the earlier uncertainty is resolved the less computational complex becomes the world-based detection approach because the less detection phases need to be computed in several worlds simultaneously. Nonetheless, because uncertainty resolution always implicates a loss of information, it intuitively seems that the earlier the resolution is performed the higher is the risk of decreasing the effectiveness of the detection approach.

7.3.2. Possible World Generation

There are two intuitive solutions to avoid a separate processing of all possible worlds. The first solution is to use a mechanism that processes all the possible worlds collectively as we know it from the principles of intensional and extensional query evaluation that we have presented in Section 3.5. The second solution is to approximate the detection result by processing only a sample set of worlds instead of the whole world space.

Processing a duplicate detection process in each of the possible worlds separately is similar to evaluating an aggregate query under the distribution semantics (see Section 3.5.4) because theoretically from each world another duplicate clustering can result. Thus, an exact computation of the complete detection process under the possible worlds semantics is per se intractable for large world spaces. In contrast, if we restrict the world-based evaluated part of the detection process to the pairwise matching phases, e.g. if we resolve uncertainty by aggregating feature scores, similarity scores, or duplicate decisions, duplicate detection within a single table is similar to processing a similarity join query where two database tuples belong to the query answer if their similarity is above a given threshold. Such a similarity join query is presented in Figure 7.27 where sim is a similarity measure for database tuples and θ is a threshold that is used to demarcate the set of MATCHES from the set of UNMATCHES. Notice, this query is safe if the input database is a BID-database, because we do not compare tuples with same representation key values and two tuples of the join result that share the same *DEI*s are always mutual exclusive because they originate from different alternatives of the same x-tuples (thus we can utilize an extensional imple-

mentation of the projection operator that always assumes a mutual exclusion between combined tuples). As a consequence, we can construct and execute an extensional query plan to compute the query answer.

However, world impact values cannot be computed in this way. Moreover, in complex databases, database entities are not restricted to single tuples, but can be represented by tuples from several tables. In that case, a matching of database entities cannot be reduced to a matching of tuples. Nevertheless, if we first extract a probabilistic description per entity, we can consider a pairwise matching of entity descriptions instead of tuples. Furthermore, if the input database is entity independent we can reduce the matching of two entities in all possible worlds to a matching of all possible combinations of the entities' description alternatives. Consequently, we do not need to compute any lineage condition. In contrast, the query answer can be considered as a BID-database that contains one x-tuple per entity pair, where every x-tuples represents the probabilistic matching result (feature score, similarity score, or duplicate decision) of its corresponding entity pair and a pair's x-tuple is computed by matching the probabilistic descriptions of the pair's entities. Note, this approach is similar to the approach that has been used by Lian and Chen [LC10] in order to perform a similarity join query on mutual independent probabilistic sets.

Although the input database is not entity-independent, the pairwise matching phases process each entity pair independently. Thus, the matching result is only influenced by the correlations between the two compared entities. Such correlations in turn can be simply incorporated into the pairwise matching of the probabilistic entity descriptions as we will present in Section 7.4.5. Nevertheless, since this detection approach has not much in common with the principles of intensional query evaluation or the principles of extensional query evaluation, we consider it as an own approach, i.e. the aforementioned description-based detection approach, and therefore consider it separately in Section 7.4.

For all these reasons, we restrict the world-based detection approach to an approximation that is based on processing a sample set of worlds. Note that the evaluation requirements for conventional database queries are much harder than for duplicate detection processes because detecting duplicates is heuristic by nature and a perfect solution can generally not be guaranteed, i.e. even an exact computation of the world-based detection approach would not return a perfect detection result in the majority of use cases. For that reason, we can relax the query principles of the possible worlds semantics and using an approximation does not necessarily change the detection result to the worse.

For generating a sample set of possible worlds we use the Monte-Carlo Simulation based approach that we have described in Section 3.5.3. A disadvantage of this sampling approach is its non-deterministic nature, i.e. it does not always produce the same set of worlds even if same input data and same configurations are given. Thus, if we perform this approach for k times it is high likely that it returns k different set of worlds. This property cumbers experimental evaluations, because an experiment can only be repeated with the same world space if that world space is intermediately stored. Nevertheless, a sampling of worlds that is based on the Monte-Carlo Simulation has to be proven to be effective and efficient in many application scenarios as for example for stochastic analyses [JXW+11]. For that reason, we adopt it for our purpose.

7.4. Description-based Approach

In contrast to the world-based approach, in the description-based approach we compare two entities only once and consider uncertainty not by generating multiple worlds, but by extracting and matching uncertain entity descriptions.

7.4.1. Modeling Uncertain Entity Descriptions

By using uncertain value theory, a probabilistic entity description is defined as a probabilistic value of the domain of all entity descriptions that are conform to the same description type.

Definition 58 ***(Probabilistic Entity Description:)*** *Let possD($\mathfrak{T}$) bet the set of all entity descriptions that are conform to description type $\mathfrak{T}$, a probabilistic entity description $\mathfrak{d} = (\mathfrak{d}, Pr)$ that is conform to $\mathfrak{T}$ is a probabilistic value $\mathfrak{d} \in^{[\rho]}$ possD($\mathfrak{T}$). The function $\mathfrak{d}_{PED}(e)$ maps a database entity e to its probabilistic entity description.*

An incomplete entity description is defined accordingly.

In the rest of this section, we restrict our consideration to probabilistic entity descriptions. Nevertheless, an adaptation to incomplete entity descriptions is straightforward.

7.4.2. Extracting Uncertain Entity Descriptions

Theoretically, the probabilistic entity description of a database entity needs to be extracted from a probabilistic database by extracting the entity's description in each of the database's possible worlds separately. Let $pdb = (\mathbf{W}, Pr)$ be a probabilistic database and let f_{ED} be a function that is used to extract entity descriptions from a certain database, a corresponding function f_{PED} for extracting the probabilistic entity description of a database entity $e \in Ext_{Poss}(pdb)$ is defined:

$$
\begin{aligned}
& f_{PED}(e, pdb) = \mathfrak{d} = (\mathfrak{d}, Pr_{\mathfrak{d}}) \\
\text{where} \quad & \mathfrak{d} = \{f_{ED}(e, W) \mid W \in \mathbf{W}\} \\
\text{and} \quad & \forall d \in \mathfrak{d}\colon Pr_{\mathfrak{d}}(d) = \sum\nolimits_{W \in \mathbf{W}, f_{ED}(e,W)=d} Pr(W)
\end{aligned}
$$

This way of extraction is illustrated in Figure 7.28. First the database's worlds are generated (Step 1). Then, an entity description is extracted in each world (Step 2) and all extracted certain descriptions are combined to a probabilistic one. The probability of a description alternative results in the accumulative probability of all worlds from which this description is extracted.

Of course, extracting an entity description from each possible world is impractical if the world space is large. Anyway, extracting descriptions from all worlds of the initial database is an unessecary overhead, because the considered entity is represented with same data in different worlds and hence from different worlds the same description are extracted. For that reason, the size of the database as well as the size of its world space can be extremely reduced by initially projecting on only the data that concerns entity e. This approach is presented in Figure 7.29 where database pdb^e represents the part of database *pdb* that concerns entity e.

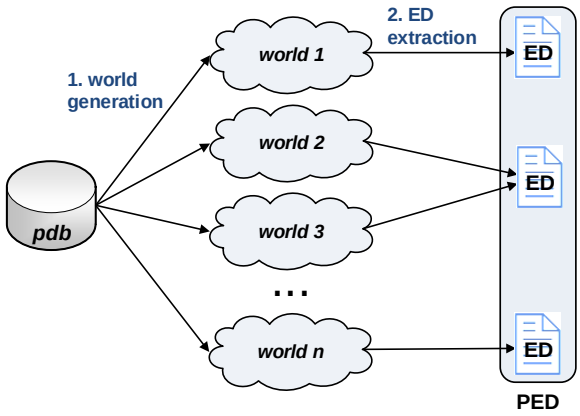


Figure 7.28.: Naive approach for extracting probabilistic entity descriptions by processing all possible worlds of *pdb*

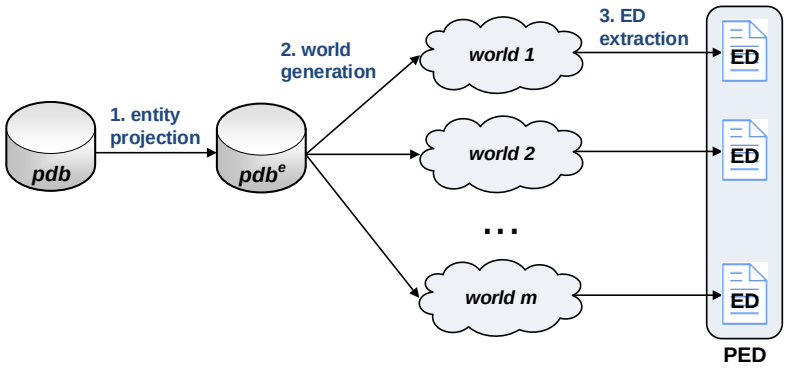
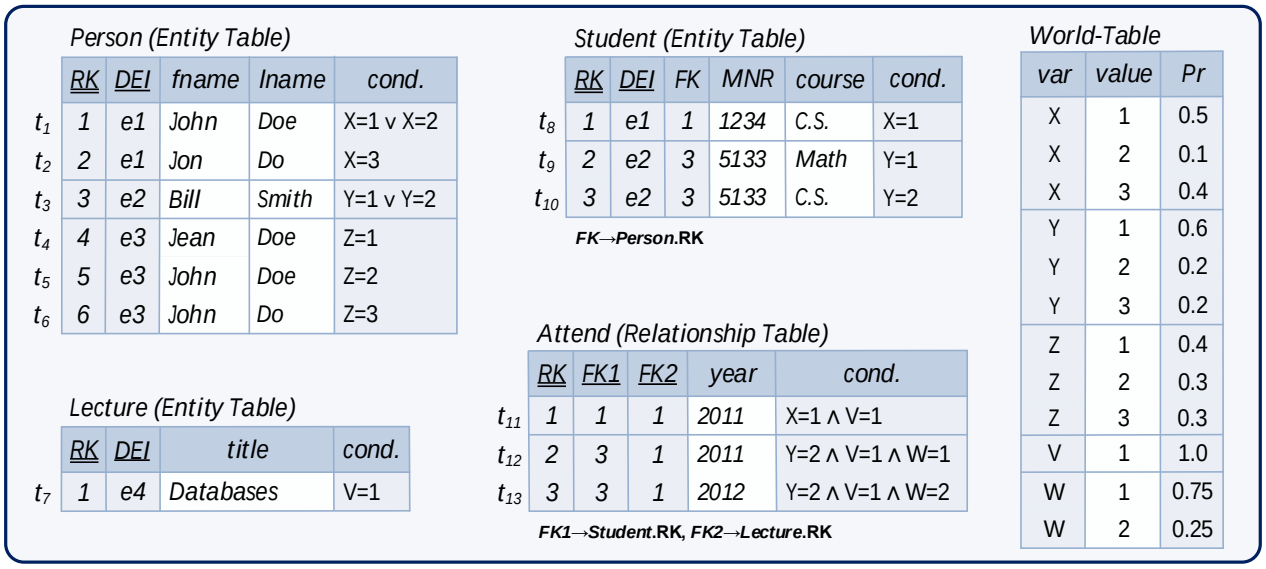


Figure 7.29.: Approach for extracting probabilistic entity descriptions by processing all possible worlds of pdb^e

The part of the database that concerns a specific entity e contains all tuples of entity tables that represent e, i.e. tuples that have the value e in the attribute *DEI*, and contains all tuples of relationship tables and all tuples of entity tables that reference to a tuple that represents e, i.e. tuples that have a foreign key value that references to the representation key value of a tuple that has the value e in the attribute *DEI*. Whereas the first is required for extracting the attribute descriptions, the latter is required for extracting relationship descriptions. Of course, as presented in Section 5.1, the idea of relationship descriptions is not restricted to direct relationships, but can be specified by an arbitrary database query. Thus, if the relationship description should also contain information on indirect relationships, tuples that do not reference to the considered entity need to remain in the reduced database as well. However, which of the input tuples actually need to remain in the reduced database depends on the database queries that are used for extracting the indirect relationship information. Note, for simplifying the reduced database, it make sense to resolve the foreign key references to other database entities, i.e. a reference to the representation key of a tuple is replaced by the *DEI* of this tuple.

To reduce complexity further on, all attributes and relationships roles that are not required for the detection process can be discarded as well. The number of worlds of the reduced database pdb^e is usually much lower than the number of worlds of the initial database *pdb*. Therefore, the possible worlds of pdb^e can be generated more easily than these of *pdb* and from most of these worlds different entity descriptions result. Due to the number of possible worlds can be even extreme large for databases containing



Person (Entity Table)

	RK	DEI	fname	lname	cond.
t_1	1	e1	John	Doe	X=1 ∨ X=2
t_2	2	e1	Jon	Do	X=3
t_3	3	e2	Bill	Smith	Y=1 ∨ Y=2
t_4	4	e3	Jean	Doe	Z=1
t_5	5	e3	John	Doe	Z=2
t_6	6	e3	John	Do	Z=3

Lecture (Entity Table)

	RK	DEI	title	cond.
t_7	1	e4	Databases	V=1

Student (Entity Table)

	RK	DEI	FK	MNR	course	cond.
t_8	1	e1	1	1234	C.S.	X=1
t_9	2	e2	3	5133	Math	Y=1
t_{10}	3	e2	3	5133	C.S.	Y=2

FK→Person.RK

Attend (Relationship Table)

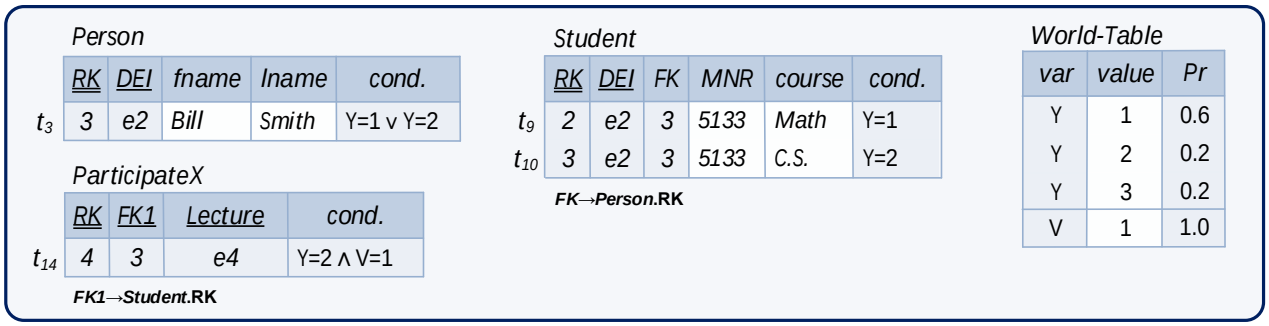
	RK	FK1	FK2	year	cond.
t_{11}	1	1	1	2011	X=1 ∧ V=1
t_{12}	2	3	1	2011	Y=2 ∧ V=1 ∧ W=1
t_{13}	3	3	1	2012	Y=2 ∧ V=1 ∧ W=2

FK1→Student.RK, FK2→Lecture.RK

World-Table

var	value	Pr
X	1	0.5
X	2	0.1
X	3	0.4
Y	1	0.6
Y	2	0.2
Y	3	0.2
Z	1	0.4
Z	2	0.3
Z	3	0.3
V	1	1.0
W	1	0.75
W	2	0.25

Figure 7.30.: The initial probabilistic sample database *pdb*



Person

	RK	DEI	fname	lname	cond.
t_3	3	e2	Bill	Smith	Y=1 ∨ Y=2

ParticipateX

	RK	FK1	Lecture	cond.
t_{14}	4	3	e4	Y=2 ∧ V=1

FK1→Student.RK

Student

	RK	DEI	FK	MNR	course	cond.
t_9	2	e2	3	5133	Math	Y=1
t_{10}	3	e2	3	5133	C.S.	Y=2

FK→Person.RK

World-Table

var	value	Pr
Y	1	0.6
Y	2	0.2
Y	3	0.2
V	1	1.0

Figure 7.31.: The reduced probabilistic sample database pdb^{e_2}

information on a single entity, we discuss the extraction of compactly represented probabilistic entity descriptions without constructing the possible world space first in Section 11.2.1.

Example 122 *To illustrate the extraction process, we consider the pc-database pdb presented in Figure 7.30. This database has a schema with four tables (three entity tables and one relationship table) and has an instance with four database entities and some relationships between them. As we can simply derive from the world-table, the initial database compactly represent* $3 \times 3 \times 3 \times 1 \times 2 = 54$ *possible worlds.*

Since we do not want to enumerate all these possible worlds for extracting the probabilistic entity description of entity e_2*, we first derive the probabilistic database* pdb^{e_2} *from pdb by projecting on the data concerning* e_2*. Due to we model relationship information by a set of related entities, we transform the table* 'Attend' *by projecting out the attribute* 'year' *and by resolving the foreign key reference to table* 'Lecture'*, i.e. we remove the foreign key and directly store the DEI of the referenced lecture in this table. The derived database is presented in Figure 7.31 and represents the three possible worlds listed in Figure 7.32. Note that the table* 'Lecture' *is empty in all possible worlds and therefore is omitted from the reduced pc-database as well as the possible worlds representation.*

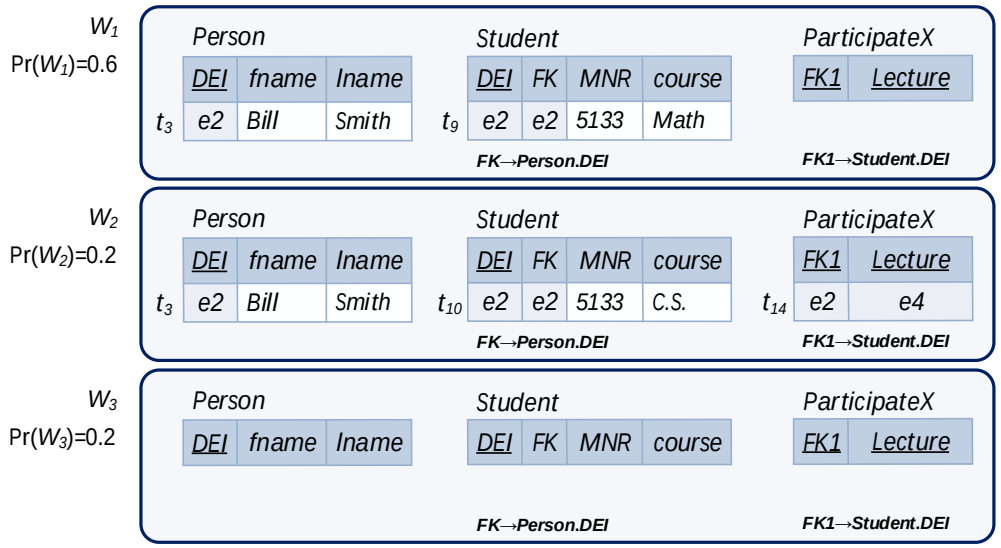


Figure 7.32.: The possible world representation of pdb^{e_2}

	isPerson	fname	lname	isStudent	MNR	course	related Lectures	Pr
$d_1 = f_{ED}(e_2, W_1)$	true	Bill	Smith	true	5133	Math	{}	0.6
$d_2 = f_{ED}(e_2, W_2)$	true	Bill	Smith	true	5133	C.S.	{e4}	0.2
$d_3 = f_{ED}(e_2, W_3)$	false	⊥	⊥	false	⊥	⊥	{}	0.2

Figure 7.33.: The probabilistic entity description of e_2 that is extracted from pdb^{e_2} (and hence from *pdb*)

In the next step from each of these worlds an entity description is extracted. Due to e_2 *is only maybe a person, it does not exists in world* W_3*. Consequently, the entity description that is extracted from this world is an empty description, i.e. the attribute description does not contain an attribute value and the set of related entities of the relationship description is empty*[3] *Each of the extracted entity descriptions serves then as an alternative of the probabilistic entity description of* e_2*. In our sample from each world another description result. Thus, the probabilities of the alternative descriptions can be simply adopted from their corresponding worlds. The resultant probabilistic entity description is depicted in tabular form in Figure 7.33.*

In general, projecting on a single database entity can violate existing database constraints. For the purpose of extraction, however, such violations do not matter and can be simply ignored.

7.4.3. Uncertain Description-based Entity Matching: An Overview

The process of description-based matching of database entities in uncertain data is illustrated in Figure 7.34 and only slightly differs from the process of description-based matching of database entities that we have presented for certain data in Section 5.2. First, for efficiency reasons the search space need to be reduced by filtering out clear UNMATCHES. For that purpose, we adapt existing blocking methods that can be used for constructing the candidate pair space (short *CPS*) in certain data (see Section 4.3.4) to

[3] Actually the description is not completely empty because we can infer the value 'false' for the membership attributes *'isPerson'* and *'isStud'*.

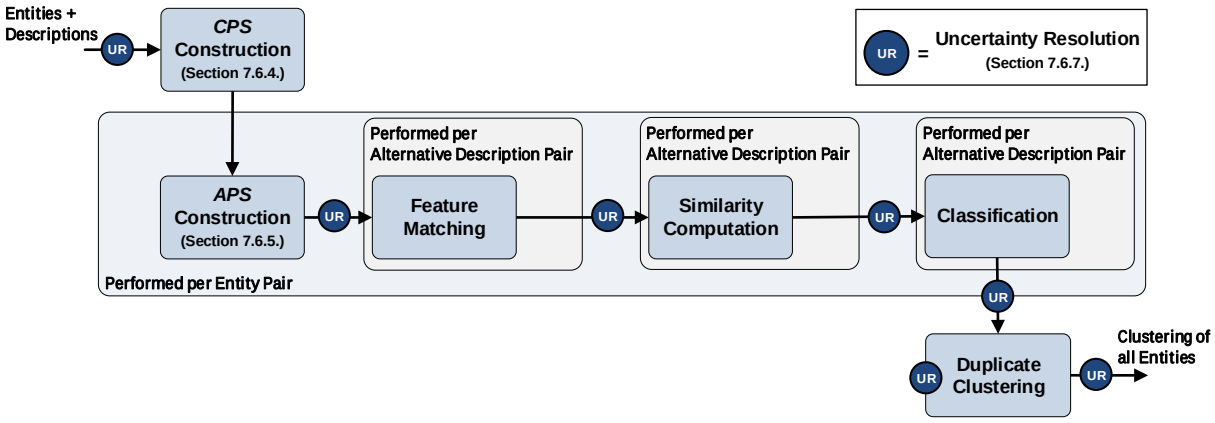


Figure 7.34.: Execution model for matching database entities based on uncertain entity descriptions

uncertain data in Section 7.4.4. Two database entities are then matched by matching their uncertain entity descriptions and two uncertain entity descriptions are matched by pairwise matching their alternatives. Thus, the first step in matching two uncertain entity descriptions is to construct an *alternative (description) pair space* (short *APS*) that contains all the pairs of description alternatives that need to be matched. Since each uncertain description alternative corresponds to a certain entity description, alternative description pairs can be compared by utilizing the phases of feature matching, similarity computation, and classification as described for certain data in Section 5.2.3. The matching result is then passed to the phase of duplicate clustering.

To produce a deterministic detection result, uncertainty need to be resolved in the matching process. As in the world-based approach, resolution (see dark blue colored circles in Figure 7.34) can be performed at different moments of the matching process (between phases or within phases). We will discuss uncertainty resolution in Section 7.4.7.

7.4.4. Candidate Pair Space Construction

As in certain data, a pairwise matching of all database entities is impractical for large databases, but a smaller candidate pair space need to be constructed instead. As presented in Section 4.3.4, candidate pair space construction in certain data is typically realized by blocking methods as for example the Sorted Neighborhood Method. The methods' underlying ideas for rejecting clear UNMATCHES from the candidate pair space are based on different indications for real-world non-equivalence and hence applies to uncertain data as well. For that reason, it makes sense to rather adapt existing blocking methods to the handling of uncertain input data than to develop new methods from scratch. Of course, if additional context information is available pruning methods can be adopted to uncertain databases, too.

Recall, the most of existing blocking methods are based on the use of blocking keys (*bk*) that are utilized to extract blocking key values (*bkv*s) from the data (original database or the entity descriptions). In uncertain databases, however, the data from which the *bkv*s are extracted can be uncertain and hence applying a conventional *bk* to this data can produce uncertain *bkv*s. Since existing blocking methods are not designed to deal with uncertain *bkv*s, we cannot use them for uncertain databases without any adaptation. Adaptation can be realized in two ways: (a) by resolving data uncertainty during key value

DEI	AID	fname	lname	YoB	Pr	
e1	1	John	Doell	1974	0.56	?
e1	2	Jon	Doel	1975	0.34	
e2	1	Bill	Smith	1991	1.00	
e3	1	Jon	Yoel	1975	0.35	
e3	2	Jean	Hoel	1975	0.25	
e3	3	Jonny	Doel	1972	0.20	
e3	4	John	Doel	1972	0.20	
e4	1	Will	Smith	1990	0.60	
e4	2	William	Smith	1993	0.40	
e5	1	John	Hoel	1972	0.80	?

Figure 7.35.: Sample set of database entities along with their probabilistic attribute descriptions

extraction and hence by extracting certain *bkv*s from uncertain data, or (b) by adapting the core functionality of the respective blocking method to uncertain *bkv*s. Note, a probabilistic candidate pair space is completely pointless because we construct it for efficiency reasons and an entity pair either need to be matched, i.e. it belongs to the candidate pair space, or not, i.e. it belongs not to the candidate pair space. For that reason, we need to resolve uncertainty within the blocking method.

The advantage of extracting certain *bkv*s from uncertain data is obvious. Due to the *bkv*s are still certain, each blocking method that has been conceptualized for certain data can be used for uncertain data without any adaptation. In contrast, an adaptation to uncertain *bkv*s depends on the individual blocking method and an adaptation that has been developed for one method cannot be simply used for another. For that reason, we focus on adaptations that are based on extracting certain *bkv*s and consider an adaptation to uncertain *bkv*s only exemplary for the Sorted Neighborhood Method in Section 7.4.4.6.

Because it is intuitively not clear how to derive certain *bkv*s from uncertain data best, we proposed and evaluated four adaptation strategies in [PWFR12]. In the strategy *Multi-Pass over Possible Worlds* (Section 7.4.4.1), we perform a separate blocking pass to some of the database's possible worlds that are each a certain database. In *Key-per-Entity*[4] (Section 7.4.4.2), we extract a certain *bkv* for each entity by applying a conventional *bk* to a certain representative of its uncertain description. In *Key-per-Alternative* (Section 7.4.4.3), we extract a certain *bkv* per alternative of the uncertain entity descriptions. In *Key-per-Representative* (Section 7.4.4.4), we first compute a set of certain representatives for each uncertain entity description and then derive a certain *bkv* per representative. Consequently, this strategy combines the underlying ideas of *key-per-entity* and *key-per-alternative*. Some of these four strategies can be also applied to immediately extracted uncertain *bkv*s instead of the original descriptions. We call this concept *Uncertain Keys First* and consider it in Section 7.4.4.5. Note, since most blocking techniques are based on the use of only attribute values, the blocking input can usually be restricted to the entities' attribute descriptions.

Example 123 *To illustrate the different adaptation strategies that we will introduce in the rest of this section, we use the sample set of five database entities that is presented in Figure 7.35. Each entity is described by the three attributes* 'fname'*,* 'lname'*, and* 'YoB' *(year of birth). For reason of simplicity we*

[4] In [PWFR12] this strategy is denoted as *Key-per-Tuple*.

present the entities' probabilistic attribute descriptions by x-tuples where the attribute 'AID' *enumerates the alternative descriptions per entity and the combination of the DEI and the* 'AID' *serves as representation key, i.e. each description alternative can be clearly identified by its values in these two attributes. Note, we cannot extract bkvs from an empty description. For that reason, we ignore these alternatives in the following consideration and represent an uncertain entity description that has the empty attribute description as an alternative by a maybe x-tuple instead.*

7.4.4.1. Multi-Pass over Possible Worlds

The idea of the multi-pass over possible worlds is that each of the database's worlds is a certain database on that a conventional blocking method can be applied as usual. As a consequence, the underlying idea of this strategy is principally the same as this of the world-based approach for duplicate detection presented in Section 7.3. First we select a set of possible worlds, then we perform a conventional blocking method in each of these worlds separately, and finally aggregate the resultant candidate pair spaces to a single one.

Computing the k most probable instances of a specific entity from the original database can become inefficient, if each entity is represented by several tuples. Moreover, the important information of these instances have been already extracted in the form of the probabilistic entity descriptions. For that reason, we construct worlds from the attribute descriptions instead of the original database. Consequently, each world corresponds to a set of certain attribute descriptions (each one per entity) rather than a certain database instance.

- **World Selection:** For several reasons, an adequate strategy for world selection in the context of candidate pair space construction essentially differs from the Monte-Carlo based approach that we have used for our world-based approach for duplicate detection in Section 7.3. First it does not make sense to consider worlds in which some entities are not present (have an empty attribute description), because these entities cannot be paired with any other entity. As we discuss in Section 7.7, the input database is commonly prepared to an entity independent database. In that case, for each possible world with missing entities there is another possible world that includes this world and has all entities present. In general, even if the input database is not entity independent, we assume an independence between the individual attribute descriptions in the candidate pair space construction phase for the aforementioned reason of missing entities.

  Obviously, the first pass has to be performed on the most probable world that contains all entities. However, as intuitively thought and experimentally proved in [PWFR12], processing the next most probable worlds does not bring any improvement in pairs completeness[5], because the most probable worlds are usually very similar to each other and the candidate pair spaces that result from these worlds are nearly identical to this of the most probable world. For that reason, we should not only consider worlds that are high probable, but rather should select worlds that are also dissimilar to each other, because in this case each world is assumed to produce a different candidate pair space and each pass improves pairs completeness further on.

[5] Recall, pairs completeness measures the amount of true duplicate pairs that belong to the candidate pair space.

DEI	fname	lname	YoB
e1	John	Doell	1974
e2	Bill	Smith	1991
e3	Jon	Yoel	1975
e4	Will	Smith	1990
e5	John	Hoel	1972

world W_1 (*first alternative*)

DEI	fname	lname	YoB
e1	Jon	Doel	1975
e2	Bill	Smith	1991
e3	Jean	Hoel	1975
e4	William	Smith	1993
e5	John	Hoel	1972

world W_2 (*second alternative*)

DEI	fname	lname	YoB
e1	John	Doell	1974
e2	Bill	Smith	1991
e3	Jonny	Doel	1972
e4	Will	Smith	1990
e5	John	Hoel	1972

world W_3 (*third alternative*)

DEI	fname	lname	YoB
e1	John	Doell	1974
e2	Bill	Smith	1991
e3	John	Doel	1972
e4	Will	Smith	1990
e5	John	Hoel	1972

world W_4 (*fourth alternative*)

Figure 7.36.: The four worlds constructed by our developed strategy for selecting dissimilar worlds

Since measuring similarity of worlds can be extremely expensive in time, we proposed a strategy for world selection in [PWFR12] that does not use any measure of similarity, but accomplish a high dissimilarity between worlds by choosing completely different sets of description alternatives for each constructed world. In the first step, the most probable world is computed by selecting the most probable alternative per description. Afterwards, a second world is constructed by taking the second most probable alternative per description, a third world is constructed by taking the third most probable alternative per description and so on. This procedure is repeated until all alternatives have been used or a user-defined threshold is reached. If for any constructed world a description has no more new alternative, its most probable alternative is used for the remaining worlds. If some alternatives of one description are equally probable a strict order is achieved by using a tie breaker as for example a random function. Since the algorithm stops when each alternative of each description has been used at least once, the number of constructed worlds is rather small, i.e. at most as large as the maximal number of alternatives the considered attribute descriptions have.

An algorithm that computes the k most probable worlds of an entity independent database as well as an algorithm that utilizes the above presented selecting strategy to produce a world space with k dissimilar worlds are both presented in [PWFR12]. As validated by our experiments presented in [PWFR12], selecting a sample set of worlds by using the latter algorithm produces a candidate pair space with a much higher pairs completeness than the pair space that result from selecting the k most probable worlds.

Example 124 *For illustration, we consider the set of sample attribute descriptions from Figure 7.35 and compute a set of dissimilar worlds by utilizing the above presented selecting strategy. The four resultant worlds are presented in Figure 7.36. The first world is constructed by selecting the most probable alternative from each attribute description. Then, for the second world the second most probable alternative from each attribute description is selected. Since the entities* e_2

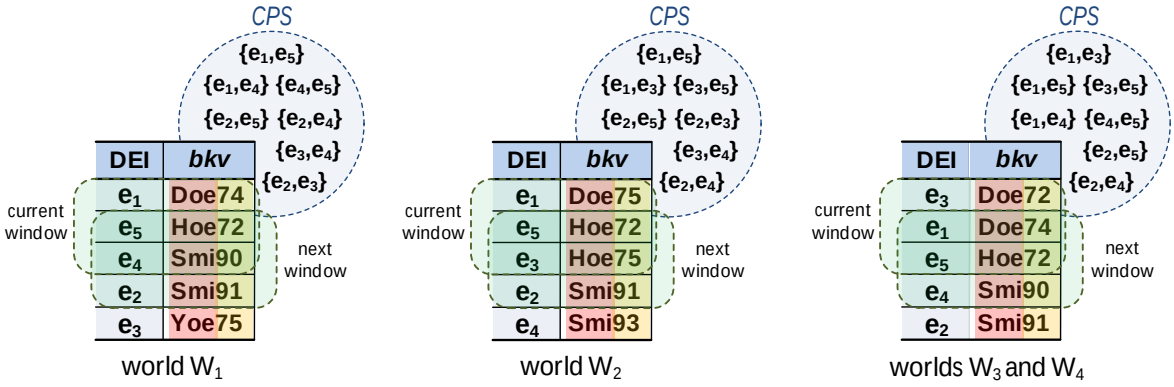


Figure 7.37.: The candidate pair spaces that result from applying the Sorted Neighborhood Method to the sample worlds W_1 to W_4 from Figure 7.36

and e_5 each have only one alternative, these alternatives are chosen for all remaining worlds. The third world and the fourth world are constructed accordingly.

For blocking illustration, we use a blocking key that append the last two digits of the year of birth to the first three letters of the lastname and apply the Sorted Neighborhood Method with a window size of three. The sorted lists of entities as well as the candidate pair spaces that result from the individual four worlds are depicted in Figure 7.37. Note that from the two possible worlds W_3 and W_4 the same bkvs and hence the same candidate pair spaces result.

- **Candidate Pair Space Aggregation:** After performing blocking in each pass separately, we need to combine the final candidate pair spaces. A conventional strategy is to simply use the set union operator, i.e. an entity pair belongs to the candidate pair space of the uncertain database, if it belongs to the candidate pair space of at least one of the processed worlds. Another strategy is to apply a voting approach, i.e. an entity pair belongs to the candidate pair space of the uncertain database, if it belongs to the candidate pair space of at least k of the processed worlds. The latter strategy is more restrict than the first and hence tends to produce a candidate pair space that is less pair complete than the space that is produced by the first strategy. In contrast the space that is produced by the voting strategy is lower in size and this strategy tends to produce a greater reduction ratio and a greater pairs quality than the first. Since the voting strategy is equivalent to the strategy of performing a set union if we set $k = 1$, the trade-off between pairs completeness and pairs quality remains in selecting an appropriate setting for k.

- **Efficiency:** The biggest disadvantage of the multi-pass over possible worlds strategy is its execution time. Nevertheless, because all passes are performed independently, execution time can theoretically be massively reduced by using a parallel computation approach. Such a parallelization can be accomplished by using the MapReduce Framework [DG04]. Nevertheless, we do not discuss parallelization in this thesis, but consider it as a part of future research (see Section 11.2.7).

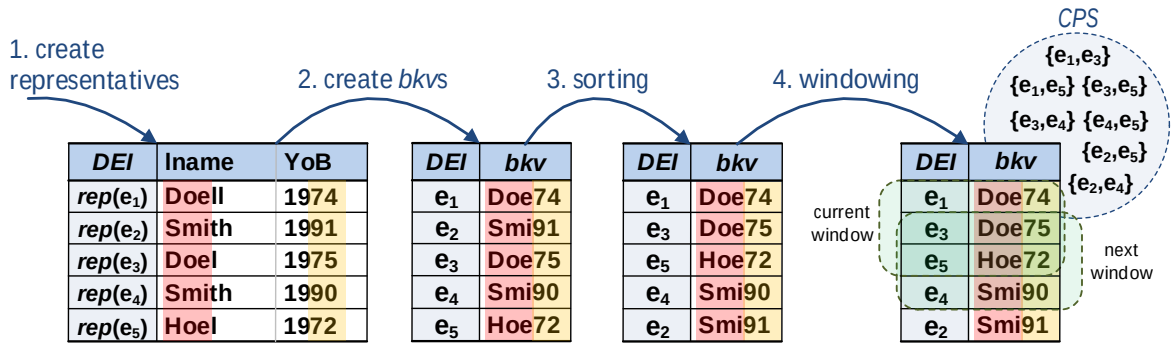


Figure 7.38.: Sample procedure of key-per-entity with the Sorted Neighborhood Method

7.4.4.2. Key per Entity

The strategy key-per-entity resolves data uncertainty in two steps: First, for each database entity a certain attribute description is computed as a representative of the entity's uncertain attribute description. Second, a certain *bkv* is extracted from each of the description representatives. For computing description representatives we only need to consider data that is used by the blocking key. Computing a representative from a set of description alternatives (each a certain attribute description) is equivalent to resolve an uncertain attribute description by the use of aggregation methods. As already discussed in Section 7.2.5, an uncertain attribute description can be resolved in two ways: First by selecting one of the description alternatives (deciding strategy) or by combining the alternatives to a new description (mediating strategy). Nevertheless, not all the methods that we have discussed in Section 7.2.5 are useful in the context of blocking. For instance, whereas a concatenation of several attribute values can be useful for feature matching, it is meaningless in the context of blocking, because a blocking key typically uses letters from fixed positions. Obviously, the most intuitive deciding strategy is to select one of the most probable alternatives as the representative or to match the alternatives pairwise and then to use the alternative that has the shortest distance to all other alternatives as the representative. An intuitive mediating strategy is to create a description that gets the most probable value per attribute.

Example 125 *For illustration, we apply key-per-entity combined with the Sorted Neighborhood Method to the set of sample attribute descriptions from Figure 7.35. This procedure is graphically presented in Figure 7.38. In the first step, we compute a single representative per entity by selecting the most probable value per attribute. Since the blocking key only uses proportions of the attributes* lname *(first three letters) and* YoB *(last two digits), the attribute* fname *does not need to be considered for the representatives. In the second step, we derive a bkv from each representative and sort the entities by these key values in descendant order. Finally, a window of size three slides over the sorted list and each two entities that are in the window at the same time are added to the candidate pair space.*

In this example, we can see the difference between a deciding strategy and a mediating strategy that uses deciding functions for conflict resolution, because the description representative that has been computed for entity e_3 is not an alternative of the uncertain attribute description of e_3.

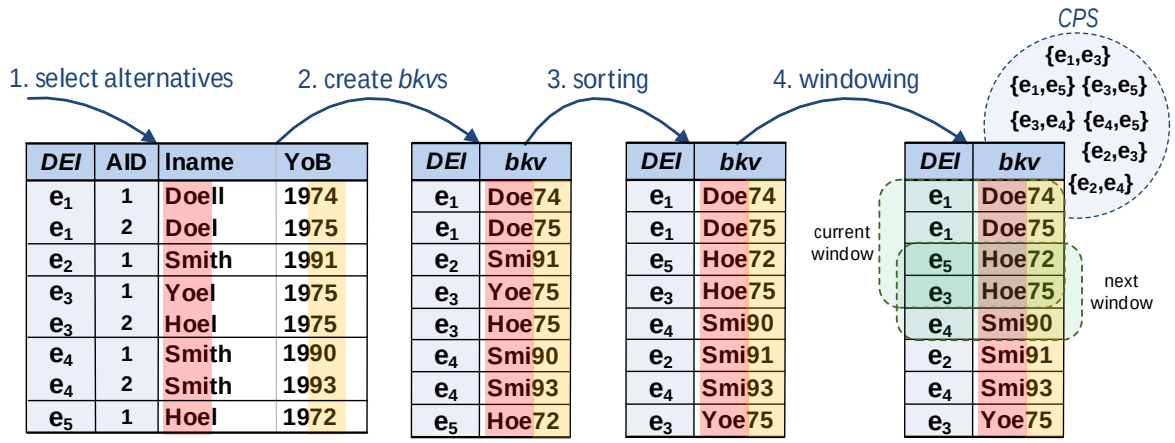


Figure 7.39.: Sample procedure of the *top-2-variant* of key-per-alternative with the Sorted Neighborhood Method

7.4.4.3. Key per Alternative

In key-per-alternative, we do not extract a single *bkv* for each entity, but extract a *bkv* for some of the alternatives of the entity's corresponding description. As a consequence, each entity is associated with multiple *bkv*s. By combining key-per-alternative with the Sorted Neighborhood Method each entity can appear in the sorted list for several times. Since an entity can appear in one window for multiple times, we redefine the window size as the number of different entities per window instead of the number of *bkv*-entity pairs per window.

The main challenge in this strategy is to decide which of the description alternatives should be used for extracting *bkv*s, because using all alternatives could crash the trade-off between costs and benefits to the worse if some descriptions have a large number of alternatives. For that purpose, we introduce the *top-k-variant* in [PWFR12] that extracts a *bkv* from the k most probable alternatives per attribute description.

Example 126 *For illustration, we apply the top-2-variant of key-per-alternative combined with the Sorted Neighborhood Method to the set of sample attribute descriptions from Figure 7.35. This procedure is graphically presented in Figure 7.39. In the first step, we select the two most probable description alternatives per entity. Then, we create a bkv from each alternative and sort the entities by the created key values in descendant order. Finally, a window of size three slides over the sorted list and each two entities that are in the window at the same time are added to the candidate pair space. Note, due to entity* e_1 *appears in the sorted list at the first position and the third position, the size of the window is four instead of three in the first sliding step.*

7.4.4.4. Key per Representative

Our fourth strategy combines the ideas of key-per-entity and key-per-alternatives by first computing a set of description representatives per entity and then by extracting a *bkv* from each of these representatives. Key-per-representative can be specialized to key-per-entity by computing a single description representative per entity and can be specialized to key-per-alternative by computing description repre-

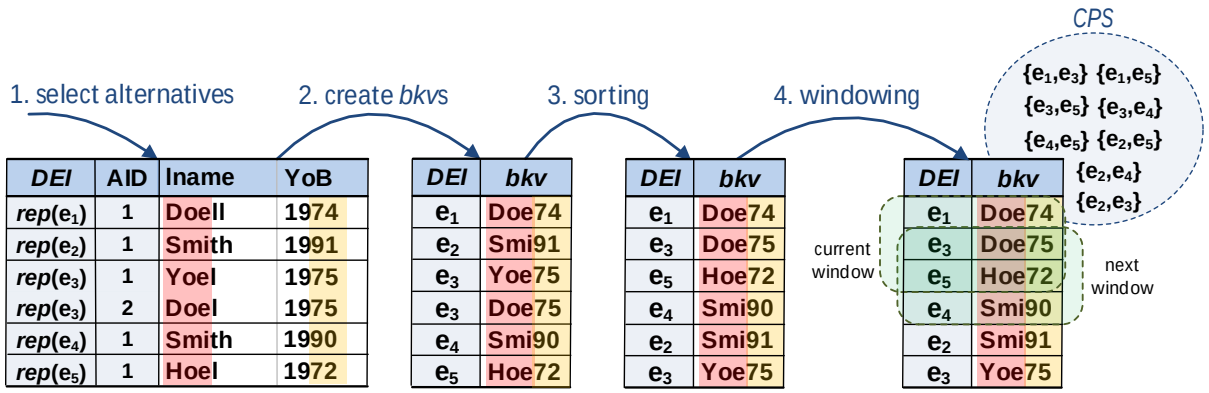


Figure 7.40.: Sample procedure of key-per-representative with the Sorted Neighborhood Method

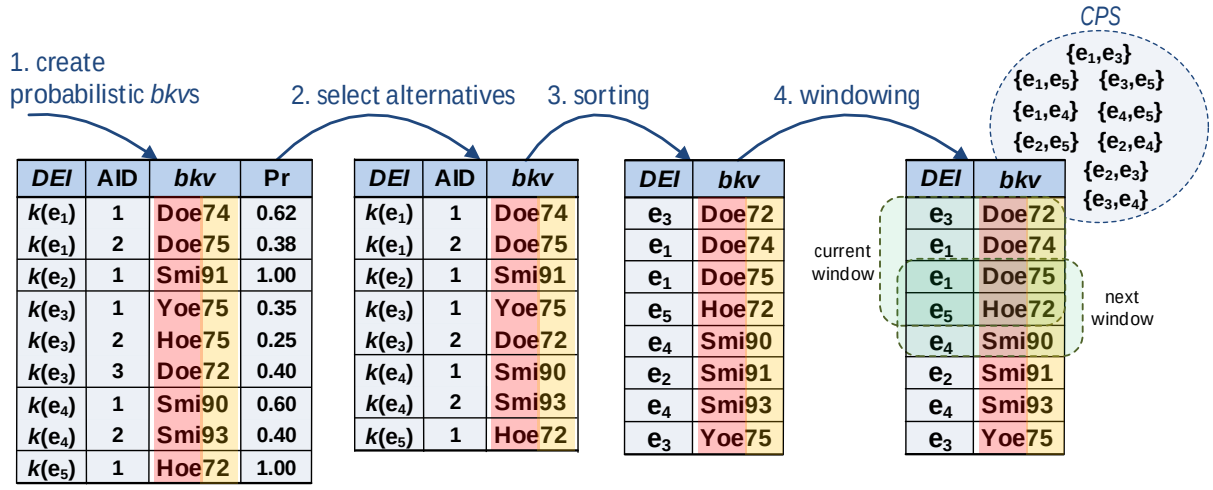
sentatives only by deciding strategies, i.e. each representative corresponds to one of the given description alternatives.

Example 127 *To illustrate the idea of key-per-representative, we apply it in combination with the Sorted Neighborhood Method to the set of sample attribute descriptions from Figure 7.35. This procedure is presented in Figure 7.40. In the first step, we compute two representatives per entity. Whereas the most probable alternative is selected as the first description representative, the second description representative is computed by selecting the most probable value per attribute. Due to both description representatives are identical for the entities e_1, e_2, e_4, and e_5, they are only represented by a single description representative in Figure 7.40. In contrast, entity e_3 is represented by two different description representatives. In the second step, we extract a bkv from each description representative and sort the entities by the extracted key values in descendant order. Finally, a window of size three slides over the sorted list and each two entities that are in the window at the same time are added to the candidate pair space.*

7.4.4.5. Concept of *Uncertain Keys First*

In the concept of *Uncertain Keys First*, we do not extract certain *bkv*s from the original attribute descriptions, but first compute a probabilistic *bkv* per entity and then apply the proposed methods for certain key value extraction to the intermediately produced probabilistic *bkv*s that each corresponds to a probabilistic attribute description with a single attribute. Using uncertain value theory, a probabilistic *bkv* is a probabilistic value of the set of all possible *bkv*s for a given blocking key. According to the possible worlds semantics, a probabilistic *bkv* is extracted from a probabilistic attribute description by applying the used blocking key to each of the description's alternative separately. Since *bkv*s that only maybe belong to the set of probabilistic *bkv*s do not make sense, we finally normalize the probabilities of each probabilistic *bkv*.

Due to from several but less probable description alternatives the same *bkv* can be extracted, the most probable alternative of an entity's probabilistic *bkv* can differ from the *bkv* of the most probable alternative of the entity's probabilistic attribute description (for illustration see Example 128). As a consequence, the *bkv*s computed by using the concept of *Uncertain Keys First* can be more representative



DEI	AID	bkv	Pr
k(e1)	1	Doe74	0.62
k(e1)	2	Doe75	0.38
k(e2)	1	Smi91	1.00
k(e3)	1	Yoe75	0.35
k(e3)	2	Hoe75	0.25
k(e3)	3	Doe72	0.40
k(e4)	1	Smi90	0.60
k(e4)	2	Smi93	0.40
k(e5)	1	Hoe72	1.00

DEI	AID	bkv
k(e1)	1	Doe74
k(e1)	2	Doe75
k(e2)	1	Smi91
k(e3)	1	Yoe75
k(e3)	2	Doe72
k(e4)	1	Smi90
k(e4)	2	Smi93
k(e5)	1	Hoe72

DEI	bkv
e3	Doe72
e1	Doe74
e1	Doe75
e5	Hoe72
e4	Smi90
e2	Smi91
e4	Smi93
e3	Yoe75

DEI	bkv
e3	Doe72
e1	Doe74
e1	Doe75
e5	Hoe72
e4	Smi90
e2	Smi91
e4	Smi93
e3	Yoe75

Figure 7.41.: Sample procedure of the *top-2-variant* of key-per-alternative with the Sorted Neighborhood Method and *Uncertain Keys First*

for the considered entities than the *bkv*s that are extracted from the description alternatives directly. We experimentally show in [PWFR12] that the concept of *Uncertain Keys First* improved blocking quality the more, the less *bkv*s per entity were extracted. Consequently, it is especially useful in the *top-k-variant* of key-per-alternative if k is set to a low value. For key-per-entity this concept is only useful if description representatives are computed by deciding strategies, because *bkv*s do not have an inherent semantics and mediating them is usually meaningless.

Example 128 *To demonstrate the concept of* Uncertain Keys First *we apply it in combination with the top-2-variant of key-per-alternative and the Sorted Neighborhood Method to the set of sample attribute descriptions from Figure 7.35. This procedure is presented in Figure 7.41. In the first step, we compute a probabilistic bkv per entity. Since the third alternative and the fourth alternative of the attribute description of entity e_3 result in the same bkv, the probabilistic bkv of e_3 has only three alternatives instead of four (note that the probability of the bkv that is extracted from several alternatives results in the sum of the their probabilities). Then, we select the two most probable alternatives per probabilistic bkv and sort the entities by their corresponding key values in descendant order. Finally, a window of size three slides over the sorted list and each two entities that are in the window at the same time are added to the candidate pair space.*

To illustrate the differences to the standard approach, we take a closer look at entity e_3. Whereas the bkvs from the two most probable alternatives of its attribute description are 'Yoe75' and 'Hoe75' (see Figure 7.39), the most probable alternatives of its probabilistic bkv are 'Yoe75' and 'Doe72'.

7.4.4.6. Probabilistic Key-based Adaptations

In contrast to the certain key-based adaptations that we have presented in the previous section, probabilistic key-based adaptations need to be tailor-made for a specific blocking method. In this section we illustrate the concept of probabilistic key-based adaptations for the Sorted Neighborhood Method. After extracting *bkv*s, the Sorted Neighborhood Method processes two phases, the sorting phase and the

windowing phase. Consequently, the uncertainty of the data need to be resolved in any of these two phases.

Sorting entities by their *bkv*s corresponds to a rank scenario where the *bkv*s are used as ranking scores and the lexicographic order is used as ranking order. For that reason, we can consider existing techniques for ranking probabilistic tuples [IS11] to resolve the uncertainty of the probabilistic *bkv*s during the sorting phase.

We distinguish between *single-ranking approaches* where a single ranking is derived from the probabilistic *bkv*s and *multi-ranking approaches* where a set of possible rankings is derived from the probabilistic *bkv*s. Whereas single-ranking approaches resolve uncertainty in the sorting phase, multi-ranking approaches resolve remaining uncertainty in the windowing phase.

Single-ranking approaches are based on the idea to compute a single ranking as a representative for all possible rankings and then to perform windowing only on this ranking. Conceivable single ranking approaches are:

- **Most Probable Ranking:** The most intuitive idea is to use the most probabilistic ranking as a sorting result. If we use a deterministic tie breaker in cases of equal *bkv*s, the number of ways n entities can be ranked is equal to the number of permutations of n distinct objects and hence is $n!$ [DB07]. The shortcoming of this approach is that computing the most probable ranking is not straightforward, because the most probable ranking can differ from the ranking of the most probable world. Nevertheless, this ranking can be approximated in polynomial time by using the Monte-Carlo Simulation.

- **Most Probable Pair Ranking:** The underlying idea of this approach is to rank the entities by the most probable pairwise order of their *bkv*s. For each pair of entities we first compute the order of all pairs of their key alternatives. Then an entity e_r is ranked in front of an entity e_s, if the probability that the *bkv* of e_r is lower than the *bkv* of e_s is higher than the probability of the contrary case.

  For that purpose, we introduce the two ordering relations $<_p$ and $=_p$ that are defined on probabilistic *bkv*s. By using a blocking key κ, these relations are defined as:

  $$\begin{aligned} e_r <_p e_s &\Leftrightarrow Prob(\kappa(e_r) < \kappa(e_s)) > Prob(\kappa(e_r) > \kappa(e_s)) \\ e_r =_p e_s &\Leftrightarrow \neg(e_r <_p e_s) \wedge \neg(e_s <_p e_r) \end{aligned}$$

  Note that the probability $Prob(\kappa(e_r) < \kappa(e_s))$ corresponds to the accumulative probability of all possible rankings in which entity e_r is sorted in front of entity e_s.

  Due to we simply use another relation for ordering, computation complexity is still dominated by the sorting time and is of class $\mathcal{O}(n \times \log(n))$.

  Notice, since we compute the ordering of each pair independently, the most probable pair ranking does not correspond to the most probable ranking.

- **Expected Position Ranking:** This approach is based on the idea to first compute the expected rank position per entity and then to rank all entities by this position. Due to the number of alternatives per *bkv* is finite, the expected rank position can be computed in $\mathcal{O}(n \times \log(n))$ [IS11].

Pos.	R_1	R_2	R_3	R_4	R_5	R_6
1	e1	e1	e1	e1	e3	e3
2	e5	e5	e5	e5	e1	e1
3	e4	e2	e3	e3	e5	e5
4	e2	e4	e4	e2	e4	e2
5	e3	e3	e2	e4	e2	e4
Prob→	0.21	0.14	0.15	0.10	0.24	0.16

(a) The six possible rankings $R_1, \ldots, R_6$

e1	1	2	1	2	1	2	1	2	1	2	1	2
e2	1	1	1	1	1	1	1	1	1	1	1	1
e3	1	1	1	1	2	2	2	2	3	3	3	3
e4	1	1	2	2	1	1	2	2	1	1	2	2
e5	1	1	1	1	1	1	1	1	1	1	1	1
	R_1	R_1	R_2	R_2	R_3	R_3	R_4	R_4	R_5	R_5	R_6	R_6

(b) Mapping from worlds to rankings

Figure 7.42.: The six possible rankings of the entities by using the probabilistic key values from Figure 7.41

- **Expected Score Ranking:** The underlying idea of this approach is to use a function that maps the *bkv*s to numerical values, to consider these values as ranking scores, and finally rank the entities by their expected scores.

  For instance, an order preserving mapping from non-numerical *bkv*s to numerical *bkv*s is to define the ranking score of a certain *bkv* v as the number of certain *bkv*s that belong to the domain of the used blocking key and that are lexicographically smaller than v. Let κ be the used blocking key and let $dom(\kappa)$ be the set of all *bkv*s that are conform to κ, the numerical score of a *bkv* $v \in dom(\kappa)$ is then defined as:

  $$score(v) = |\{v' \mid v' \in dom(\kappa), v' < v\}| \tag{7.12}$$

  For simple mappings as the one presented above computation is still dominated by the ranking time and hence has a complexity of class $\mathcal{O}(n \times \log(n))$.

- **Uncertain Rank Aggregation:** The idea of the uncertain rank aggregation is to compute a single ranking that has the minimal average (or expected) distance to all possible rankings. Whereas such a computation is known to be NP-Hard by using the Kendall Tau distance [KMS07, IS11], it can be computed in polynomial time by using the Spearman footrule distance [DKNS01] if uncertainty is modeled only on attribute level (no dependencies between attribute values or entities) [SI09, IS11] which is satisfied in our case, because each *bkv* corresponds to an x-tuple with a single attribute.

Example 129 *For illustration, we consider the probabilistic bkvs from Figure 7.41. Five entities can be ranked in* 125 *different ways, but in our case the given bkvs make only six of these rankings possible. These six rankings* R_1 *to* R_6 *are presented in Figure 7.42(a). Figure 7.42(b) depicts the selection of key value alternatives that leads to the individual rankings.*

<	k(e1)	k(e2)	k(e3)	k(e4)	k(e5)
k(e1)	-	1	0.6	1	1
k(e2)	0	-	0.35	0.4	0
k(e3)	0.4	0.65	-	0.65	0.4
k(e4)	0	0.6	0.35	-	0
k(e5)	0	1	0.6	1	-

(a) Most probable pair ranking (probability matrix)

DEI	Pos.1	Pos.2	Pos.3	Pos.4	Pos.5	exp. Pos.
e1	0.6	0.4	0	0	0	= 1.4
e2	0	0	0.14	0.47	0.39	= 4.25
e3	0.4	0	0.25	0	0.35	= 2.9
e4	0	0	0.21	0.53	0.26	= 4.05
e5	0	0.6	0.4	0	0	= 2.4

(b) Expected position

bkv	26^2×100	26×100	100	10	1	Score
Doe74	4	15	5	7	4	= 309974
Smi91	19	13	9	9	1	= 1324391
Yoe75	25	15	5	7	5	= 1729575
...	...	...	...	...	...	= ...
Hoe72	8	15	5	7	2	= 580372

(c) Score computation

DEI	exp. Score
e1	309974.8
e2	1324391
e3	874433.8
e4	1324391.2
e5	580372

(d) Expected scores

Figure 7.43.: Intermediate results for our sample scenario using the different single ranking approaches

The most probable ranking is R_5. In contrast, ranking the entities by the most probable pairwise order of their bkvs results in ranking R_3. The same ranking results if we rank the entities by their expected rank positions. The pairwise probabilities that are used for the most probable pair ranking are shown in Figure 7.43(a), and the computation of the expected rank positions are presented in Figure 7.43(b).

In our sample, the bkvs are defined by three letters in the front and two digits at the end. Thus, if we assume that all names only contain letters from the latin alphabet, the domain of all possible bkvs can be defined by the regular expression $[a-z]^3[0-9]^2$. Consequently, the number of bkvs from this domain that are lower than a certain bkv can be computed as exemplarily presented in Figure 7.43(c). By using this mapping, sorting the entities by their expected scores results in ranking R_4 (the expected scores of the individual entities are presented in Figure 7.43(d)).

By using the Spearman footrule distance to compute the distance between two rankings, R_3 is the ranking that has the minimal expected distance to all possible rankings.

Interestingly, although we only have five entities in this example, different single-ranking approaches lead to different rankings that are considered to be most representative. This demonstrates that finding a ranking that represents the given set of possible rankings best is not straightforward, because the meaning of 'best' is already vague and can be interpreted in several ways.

A simple but yet not evaluated multi-ranking approach is to select some rankings, as for example the k most probable rankings, and then to slide the window on all these rankings in parallel. This idea is similar to the multi-pass over possible worlds that we have presented in Section 7.4.4.1. The difference is that it can be useful to utilize a ranking that is not possible, but similar to the most probable rankings (recall that from Uncertain Rank Aggregation an impossible ranking can result). As for worlds, it makes sense to use rankings that are somewhat dissimilar. However, in contrast to possible worlds, the set of possible rankings cannot be directly derived from the alternatives of the *bkv*s, because selecting different alternatives per description lead to different worlds, but can lead to same rankings. A simple option for

selecting a set of rankings is to utilize several of the single-ranking approaches that we have presented above. For example, we could take the rankings that are produced by the Expected Position Ranking, the Expected Score Ranking, the Uncertain Rank Aggregation, and the Most Probable Pair Ranking as input to the multi-pass approach.

Example 130 *To illustrate the underlying idea of the multi-ranking approach, we consider the sample scenario with the probabilistic bkvs presented in Figure 7.41. As mentioned above, a set of rankings can be selected by applying different single-ranking approaches to the probabilistic bkvs. The Most Probable Pair Ranking and the Expected Position Ranking result in ranking R_3, the Expected Score Ranking results in ranking R_4, and the Most Probable Ranking results in ranking R_5. Therefore, we can take all four rankings (R_3 twice) and add two entities to the candidate pair space, if they are paired by the windowing step in two of the four rankings. By doing so and by using a window of size three, the constructed candidate pair space contains the entity pairs $\{e_1, e_3\}$, $\{e_1, e_5\}$, $\{e_2, e_3\}$, $\{e_2, e_4\}$, $\{e_2, e_5\}$, $\{e_3, e_4\}$, $\{e_3, e_5\}$, and $\{e_4, e_5\}$.*

7.4.4.7. Blocking Quality

We provide an extensive evaluation of the certain key-based adaptation strategies that we have presented in this section in [PWFR12]. The experimental results show that using multiple *bkv*s per entity (multi-pass over possible worlds and key-per-alternative) led to considerably better results in pairs completeness than only using a single *bkv* (e.g. key-per-entity). This observation especially held for the Sorted Neighborhood Method, because using a single *bkv* implicates that all the entities that are paired with the considered entity need to have similar *bkv*s as well. As a consequence, an entity whose description has two (or more) highly dissimilar alternatives can either be sorted next to entities that have description alternatives that are similar to the first or can be sorted next to entities that have description alternatives that are similar to the second, but cannot be sorted close to both.

As assumed, performing a multi-pass over several dissimilar worlds was more rewarding than performing passes on the most probable worlds, because the most probable worlds were too similar and hence constructed candidate pair spaces that were nearly identical. In general, pairs completeness of key-per-alternative and the multi-pass over dissimilar worlds was quite similar, but the first had a better pairs quality than the second and was faster if more than five worlds were selected by the multi-pass approach. Moreover, both adaptation strategies were similar robust to a varying quality of the source data.

The quality of the candidate pair space produced by key-per-alternative only decreased slightly if the number of *bkv*s per entity was reduced from ten to three, but decreased notable if less than three *bkv*s were used per entity (note that a variant that extracts a *bkv* only from one description alternative corresponds to the strategy of key-per-entity). This difference in quality increases the more alternatives the considered entity descriptions had.

The concept of *Uncertain Keys First* was profitable in strategies that select key values based on the alternatives' probabilities. In general, the smaller the amount of the provided alternatives that were selected for key value extraction the more valuable was the *Uncertain Keys First* concept.

	name	residence	Pr
d_{11}	John Doe	Rome	0.35
d_{12}	John Smith	Paris	0.40
d_{13}	Jon Do	Prague	0.25

(a) Probabilistic entity description $\mathfrak{d}_1 = \mathfrak{d}_{PED}(e_1)$

	name	residence	Pr
d_{21}	Jon Do	Rome	0.60
d_{22}	John Ho	Paris	0.40

(b) Probabilistic entity description $\mathfrak{d}_2 = \mathfrak{d}_{PED}(e_2)$

alternative pair	probability
(d_{11}, d_{21})	0.21
(d_{12}, d_{21})	0.24
(d_{13}, d_{21})	0.15
(d_{11}, d_{22})	0.14
(d_{12}, d_{22})	0.16
(d_{13}, d_{22})	0.1

(c) The naive alternative pair space

alternative pair	probability
(d_{11}, d_{21})	0.35
(d_{12}, d_{22})	0.4
(d_{13}, d_{21})	0.25

(d) The modified alternative pair space

Figure 7.44.: Sample for alternative pair space construction with and without correlations

As intuitively clear and validated by first experiments, the single-ranking approaches for handling probabilistic key values within the Sorted Neighborhood Method bear the same shortcoming as key-per-entity, because each entity is placed in the ranked list only once and hence cannot be ranked closed to all other entities that have a similar description alternative. For that reason, these strategies usually cannot compete with key-per-alternative or the multi-pass over dissimilar worlds. In general, the probabilistic key-based approaches seem to be less useful than the certain key-based approaches because they are tailor-made for a specific blocking method and could not compensate this shortcoming with any remarkable improvement in blocking quality. For the sake of completeness, however, an extensive evaluation of different single-ranking approaches and multi-ranking approaches is planned in future research.

7.4.5. Alternative Pair Space Construction

In the candidate pair space construction phase, we compute all entity pairs that are potential duplicates and hence require a more detailed consideration in order to decide whether or not they are MATCHES or UNMATCHES. Thus, as in certain data, each candidate pair serves as input to the pairwise matching phases in which a duplicate decision is computed based on the entities' descriptions. Naturally, the similarity between two uncertain descriptions is based on the similarity between their alternatives. For that reason, the matching of two uncertain descriptions starts with the construction of an alternative (description) pair space (short *APS*). In the naive case, we assume an independence between the two compared entities and hence assume an independence between their probabilistic entity descriptions. Thus, the naive alternative pair space of two probabilistic entity descriptions is constructed by building the cross product between their sets of alternatives, i.e. each alternative of the first probabilistic entity description is paired with each alternative of the second probabilistic entity description. Logically, by assuming independence the probability of an alternative description pair is defined as the product of

the alternatives' probabilities. In contrast to entity pairs, alternative description pairs are ordered pairs because both probabilistic descriptions can share same alternatives and it is important to know from which of these descriptions the individual alternatives originate. In conclusion, let $\mathfrak{d}_r = (\mathfrak{d}_r, Pr_r)$ and $\mathfrak{d}_s = (\mathfrak{d}_s, Pr_s)$ be two probabilistic entity descriptions, the naive alternative pair space $APS(\mathfrak{d}_r, \mathfrak{d}_s)$ is defined as:

$$\begin{aligned} & APS(\mathfrak{d}_r, \mathfrak{d}_s) = (W_{APS}, Pr_{APS}) \\ \text{where} \quad & W_{APS} = \mathfrak{d}_r \times \mathfrak{d}_s = \{(d_r, d_s) \mid d_r \in \mathfrak{d}_r, d_s \in \mathfrak{d}_s\} \\ \text{and} \quad & \forall (d_r, d_s) \in W_{APS} \colon Pr_{APS}(d_r, d_s) = Pr_r(d_r) \times Pr_s(d_s) \end{aligned}$$

Sometimes the compared entities are not independent and we need to incorporate dependencies between the individual description alternatives into the alternative description pair space (see Section 7.7.1). In this case, the set of possible alternative description pairs is restricted ($W_{APS} \subset \mathfrak{d}_r \times \mathfrak{d}_s$) or at least the probability distribution over the alternative description pairs is changed. Nevertheless, the accumulative probability of all alternative description pairs always have to sum up to 1, because all these pairs are jointly exhaustive and mutually exclusive events.

Example 131 *Let us consider the two probabilistic entity descriptions $\mathfrak{d}_1 = \mathfrak{d}_{PED}(e_1)$ and $\mathfrak{d}_2 = \mathfrak{d}_{PED}(e_2)$ that are depicted in Figure 7.44(a) and Figure 7.44(b) respectively. The description $\mathfrak{d}_1$ has three alternatives and the description $\mathfrak{d}_2$ has two alternatives. Thus, the naive alternative pair space results in the six alternative description pairs as listed in Table 7.44(c). Now let us assume that either e_1 and e_2 live both in Paris or none of them. In this case, the alternative description pairs (d_{11}, d_{22}), (d_{13}, d_{22}), and (d_{12}, d_{21}) can be excluded for sure. This dependency is incorporated into the matching process by changing the alternative pair space as presented in Table 7.44(d).*

For reasons of efficiency it can be useful to reduce the alternative pair space by conventional blocking methods or by merging some alternative description pairs. We will discuss both aspects in Section 7.6.2.

7.4.6. Pairwise Matching of Uncertain Entity Descriptions

The alternative pair space serves as input to the pairwise matching phases. Since each alternative description pair is associated with a probability and all pairs are jointly exhaustive and mutually exclusive events, the alternative pair space can be considered as a probabilistic value defined on the domain of all possible description pairs. Moreover, each matching phase corresponds to a certain function. According to the uncertain value theory presented in Chapter 6, a certain function that is applied to a probabilistic input value produces a probabilistic output value (see Section 6.7). As a consequence, applying the feature matching phase to the alternative pair space produces a probabilistic feature score as output. Thus, let $APS(\mathfrak{d}_r, \mathfrak{d}_s) = (W_{APS}, Pr_{APS})$ be an alternative pair space and let f_{FMP} be an implementation of the feature matching phase, the result of applying f_{FMP} to $APS(\mathfrak{d}_r, \mathfrak{d}_s)$ is the probabilistic feature score *fs* that is defined as:

$$\begin{aligned} & fs = (W_{fs}, Pr_{fs}) \\ \text{where} \quad & W_{fs} = \{f_{FMP}(d_r, d_s) \mid (d_r, d_s) \in W_{APS}\} \\ \text{and} \quad & \forall x \in W_{fs} \colon Pr_{fs}(x) = \sum\nolimits_{(d_r, d_s) \in W_{APS}, f_{FMP}(d_r, d_s) = x} Pr_{APS}(d_r, d_s) \end{aligned}$$

alt. pair	ACVector	AIVector
(d_{11},d_{21})	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]
(d_{12},d_{21})	$\vec{c}_A$=[0.0,0.2]	$\vec{i}_A$=[0.7,0.8]
(d_{13},d_{21})	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]
(d_{11},d_{22})	$\vec{c}_A$=[0.33,0.2]	$\vec{i}_A$=[0.5,0.8]
(d_{12},d_{22})	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]
(d_{13},d_{22})	$\vec{c}_A$=[0.5,0.5]	$\vec{i}_A$=[0.4,0.9]

(a) Feature scores of the alternative description pairs

alt.	ACVector	AIVector	Pr
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.21
fs2	$\vec{c}_A$=[0.0,0.2]	$\vec{i}_A$=[0.7,0.8]	0.24
fs3	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]	0.15
fs4	$\vec{c}_A$=[0.33,0.2]	$\vec{i}_A$=[0.5,0.8]	0.14
fs5	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.16
fs6	$\vec{c}_A$=[0.5,0.5]	$\vec{i}_A$=[0.4,0.9]	0.10

(b) Probabilistic feature score 1

alt.	ACVector	AIVector	Pr
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.21
fs2	$\vec{c}_A$=[0.0,0.2]	$\vec{i}_A$=[0.7,0.8]	0.38
fs3	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]	0.15
fs4	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.16
fs5	$\vec{c}_A$=[0.5,0.5]	$\vec{i}_A$=[0.4,0.9]	0.10

(c) Probabilistic feature score 2

Figure 7.45.: Sample for applying the feature matching phase to an alternative pair space

This score in turn serves as input to the similarity computation phase that produces a probabilistic similarity score as output. This probabilistic similarity score is then used as input to the classification phase that computes a probabilistic duplicate decision as output. Both phases are adopted to probabilistic input values in the same way as the feature matching phase presented above, i.e. they correspond to the probabilistic versions of their certain equivalents as defined in Definition 47.

The probabilistic duplicate decisions are then finally used as input to the duplicate clustering phase. Nevertheless, since we strive for a deterministic detection process, uncertainty need to be resolved at any moment of the detection process. Of course, if uncertainty has not been resolved until the end of the classification phases, we require a clustering approach that can deal with uncertain duplicate decisions as input.

Example 132 *For illustration, we consider the naive alternative pair space from Figure 7.44(c) and use a feature matching phase that maps each of the alternative description pairs to a feature score as presented in Figure 7.45(a) Note, since we do not have relationship information in this example, we restrict the feature score to the attribute comparison vector (short* ACVector*) and the attribute impact vector (short* AIVector*). The resultant feature scores are now considered as alternatives of a probabilistic feature score that is presented in Figure 7.45(b). In this example, each alternative description pair is mapped to another feature score. As a consequence, the number of alternatives of the probabilistic feature score is equal to the number of alternative description pairs.*

Nevertheless, it can happen that from different pairs the same feature score is computed. Thus, let us assume that the alternative description pair (d_{11}, d_{22}) *is mapped to the same feature score as pair* (d_{12}, d_{21})*. In this case, the probabilistic feature score has only five alternatives instead of six (see*

Figure 7.45(c)) and the probability of alternative fs2 is set to the accumulative probability of these two pairs.

7.4.7. Uncertainty Resolution

To enable a deterministic detection result the uncertainty that is introduced by the probabilistic descriptions need to be resolved during the detection process. As in the world-based approach, uncertainty resolution can be applied at different moments of the detection process. Note that in the description-based approach we do not construct any possible world of the input database and an aggregation of these worlds is therefore not an option.

- **Aggregation of Description Alternatives:** The first moment for resolving uncertainty is to aggregate the alternatives of each probabilistic entity description to a single representative. Since each entity is then represented by a certain entity description, entity matching can be performed as described for certain data. Recall, methods for aggregating entity descriptions are discussed in Section 7.2.5.

- **Aggregation of Alternative Description Pairs:** The second moment of resolution is to aggregate the alternative pair space to a single pair. Mediating alternative description pairs seems less meaningful. Nevertheless, aggregation can be realized by a deciding strategy as for example by selecting the most probable pair as aggregation output.

  Due to each entity pair is then represented by a single pair of entity descriptions, matching can be performed as in certain data. The advantage compared to aggregating the alternatives of the probabilistic entity descriptions is that correlations between the compared entities can be taken into account. For example, the most probable alternative description pair must not be the one that contains the most probable alternative of each probabilistic entity description.

- **Aggregation of Feature Scores:** The third moment of uncertainty resolution is to perform feature matching for each alternative description pair separately and then to aggregate the resultant probabilistic feature score to a certain one. Methods for aggregating a weighted set of feature scores has been elaborated in Section 7.2.7.

- **Aggregation of Similarity Scores:** The next moment is to resolve the result of the similarity computation phase and hence to aggregate a weighted set of similarity scores. Recall, methods for aggregating similarity scores are presented in Section 7.2.6.

- **Aggregation of Duplicate Decisions:** The fifth moment for uncertainty resolution is after the classification phase and hence to aggregate a weighted set of duplicate decisions based on the discussion made in Section 7.2.9.

- **Resolving Decision Uncertainty in the Clustering Process:** If uncertainty is not resolved during the pairwise entity matching phases, the input to the duplicate clustering phase is a probabilistic duplicate decision. The most clustering phases cannot deal with probabilistic duplicate decisions as input. Nevertheless, as discussed in Section 7.2.9, we can transform each probabilistic duplicate

decision into a matching probability. These probabilities can be in turn used as input to the clustering approach that we propose in Section 8.7. The clustering phase can handle the given input uncertainty in two ways. First it either produce a probabilistic duplicate clustering or it resolves uncertainty by itself, i.e. it get probabilistic duplicate decisions as input, but produces a single clustering as output.

- **Aggregation of Duplicate Clusterings:** The last moment of uncertainty resolution is to aggregate the set of possible clusterings that are produced by the duplicate clustering phase. Aggregation of clusterings has been considered in Section 7.2.10.

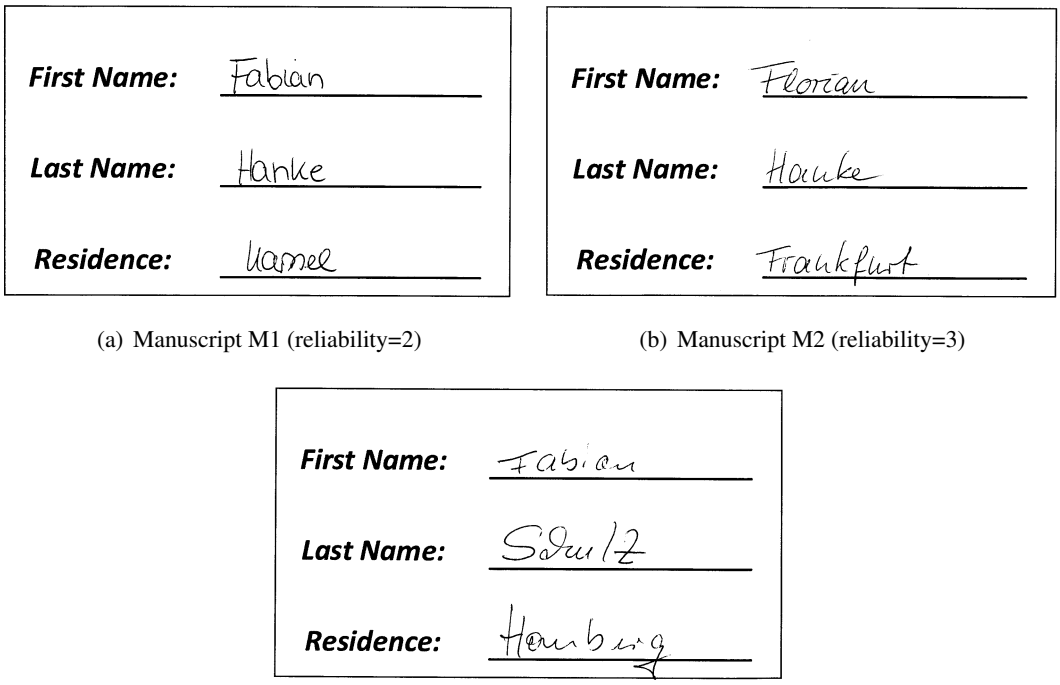
7.5. Detecting Scattered Duplicates

As we have illustrated in Example 133, the detection of scattered duplicates is a particular challenge of deterministic duplicate detection in uncertain databases, because scattered entities are highly dissimilar in the majority of possible worlds even if they represent the same information. A probabilistic entity description can be considered as a probability mass function that is defined on the domain of all possible entity descriptions. Moreover, two probabilistic entity descriptions are data equivalent if their probability mass functions are equivalent. As a consequence, it seems plausible to reduce the problem of detecting scattered duplicates (and maybe the detection of duplicates in probabilistic databases in general) to the problem of detecting similar probability mass functions, i.e. we use a similarity measure for probability mass functions to resolve the uncertainty of a probabilistic similarity score. Although a variety of similarity measures have been proposed for probability mass functions in the past, the most of these measures are not suitable for our purpose because they take only differences in the probabilities into account, but do not consider similarities between domain elements. Duplicates, however, can be corrupted by typos, transformation errors, etc. and therefore do not only differ in their assigned probabilities.

The rest of this section is structured as follows. For illustration and motivation, we first present a simple example for the emergence of scattered duplicates in Section 7.5.1. Then we elaborate a list of discrepancies that can considerably exacerbate the detection of scattered duplicates in Section 7.5.2. Afterwards, we present existing research on measuring similarity between two probability mass function in Section 7.5.3 and introduce a new similarity measure that is based on the Monge-Elkan Similarity in Section 7.5.4. Then, we extend the newly introduced measure to correlations between the compared probability mass functions in Section 7.5.4.1, discuss an incorporation of quality requirements into this measure in Section 7.5.4.3, and describe in which way this measure can be used within the description-based detection approach and the world-based detection approach in Section 7.5.4.4 and Section 7.5.4.5. Finally, we shortly discuss a compensation of discrepancies that cannot be compensated by a similarity measure for probability mass functions in Section 7.5.5 and shortly deliberate about the effects of the uncertainty of two compared probabilistic entity descriptions on the divergence between their similarity and the concluded indication for their real-world equivalence in Section 7.5.6.

7.5.1. Illustrative Example

As we will demonstrate throughout this section, a detection of scattered duplicates is simple if their probabilistic descriptions are equivalent, i.e. they assign the same probability mass to the same alternatives,



(a) Manuscript M1 (reliability=2)

(b) Manuscript M2 (reliability=3)

(c) Manuscript M3 (reliability=5)

Figure 7.46.: Three manuscripts that contain information on the same person

but can become complicated if these descriptions are different. In general, the data of scattered duplicates can differ in many aspects. For illustration, we consider the following example.

Example 133 *Let us consider the three manuscripts presented in Figure 7.46 and let us assume that it is proven that all manuscripts contain information on the same person, but the currency of them is not completely known and hence each of them is associated with a numerical score that represents its reliability. The differences between the data values of the individual manuscripts can be caused by several reasons. For instance, the considered person could have changed his address for two times so that each manuscript contains another value for residence. The difference in the last name could be explained by a marriage (nowadays it is not untypical that the bridegroom change his name instead of the bride) and the difference in the first name can be caused by a simple confusion*[6]*.*

The data of the first manuscript can be correctly read with only less doubt because it is written with a somewhat neat handwriting. In contrast, in the second manuscript the correct reading of the last name is not clear because the third letter can be interpreted as an 'n', i.e. the name 'Hanke', or can be interpreted as an 'u', i.e. the name 'Hauke'. In the third manuscript the reading of the residence is ambiguous, i.e. it can be read as 'Homberg', but can also be read as 'Hamburg' or 'Homburg'.

Let us assume a scenario where the three manuscripts are provided to several parties that each use a software for handprint character recognition to extract the handwritten data into an x-tuple of a BID-database. Moreover, some of these parties provide their extracted data so that other parties can copy the data from these parties.

[6] According to our experience it actually happens very often that the first name 'Fabian' is confused with the first name 'Florian'.

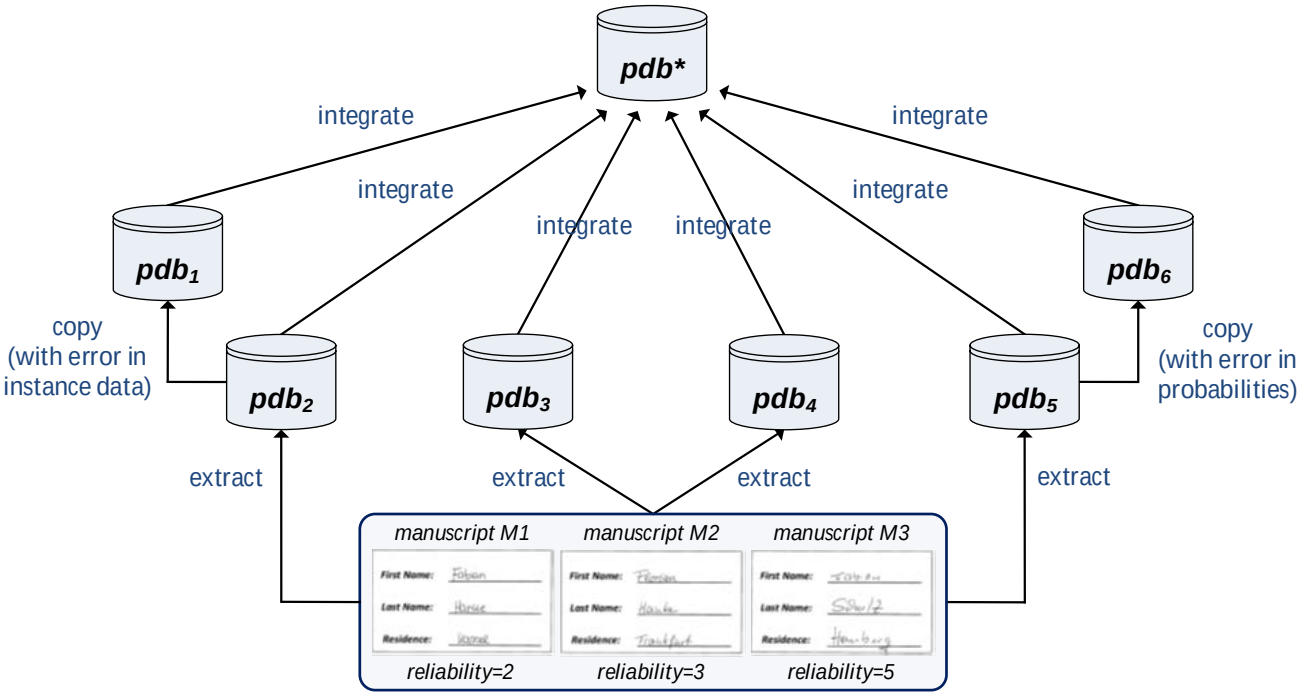


Figure 7.47.: Sample scenario for the emergence of scattered duplicates

In our example, we consider seven parties (in the following we identify the parties by their databases). Four parties, namely pdb_2, pdb_3, pdb_4, and pdb_5, extract the data directly from the original manuscripts. The two parties pdb_1 and pdb_6 copy the data from two of the former parties, i.e. pdb_1 copies the data from pdb_2 and pdb_6 copies the data from pdb_5. The seventh party pdb^* integrates the data that is provided by all the other parties into a single BID-database and therefore contains duplicate x-tuples. The data flow between the seven parties is depicted in Figure 7.47.

The concrete database instances produced by the individual parties are presented in Figure 7.48. Party pdb_2 extracts the data from the original source and produces an x-tuple with three alternatives whereby it uses the normalizes reliability scores as probabilities. Let us assume that the software that is used by this party actually extracts the value 'Hauke' from the last name of the second manuscript, but a person finally revises the data and this person recognizes that the last name likely has to be read as 'Hanke' instead of 'Hauke' because the first manuscript provides 'Hanke' as well.

Party pdb_5 uses the same handprint character recognition than pdb_2, but in contrast to pdb_2 it only got the first and the second manuscript as input. For that reason, it only produces two x-tuple alternatives instead of three. Moreover, since the extracted data is not finally revised by a person, the reading 'Hauke' finds its way into the database.

Party pdb_3 uses another software for handprint character recognition than pdb_2 and pdb_5. In contrast to the software used by these parties, the software used from pdb_3 cannot clearly read the residence value of the third manuscript and therefore create two x-tuple alternatives for this value, i.e. one with the residence 'Homberg' and one with the residence 'Hamburg'.

The fourth party uses an handprint character recognition system that extracts the same four alternatives than pdb_3, but uses another method for probability computation and therefore redistribute the

pdb_1

RK	DEI	fname	lname	residence	p
1	e1	Fabian	Hanke	Kassel	0.2
2	e1	Florian	Hanke	Frankfurt	0.3
3	e1	Fabian	Schulz	Homberg	0.5

pdb_2

RK	DEI	fname	lname	residence	p
1	e2	Fabian	Hanke	Kassel	0.2
2	e2	Florian	Schulz	Frankfurt	0.3
3	e2	Fabian	Hanke	Homberg	0.5

pdb_3

RK	DEI	fname	lname	residence	p
1	e3	Fabian	Hanke	Kassel	0.2
2	e3	Florian	Hanke	Frankfurt	0.3
3	e3	Fabian	Schulz	Hamburg	0.4
4	e3	Fabian	Schulz	Homberg	0.1

pdb_4

RK	DEI	fname	lname	residence	p
1	e4	Fabian	Hanke	Kassel	0.2
2	e4	Florian	Hanke	Frankfurt	0.3
3	e4	Fabian	Schulz	Hamburg	0.2
4	e4	Fabian	Schulz	Homberg	0.3

pdb_5

RK	DEI	fname	lname	residence	p
1	e5	Fabian	Hanke	Kassel	0.4
2	e5	Florian	Hauke	Frankfurt	0.6

pdb_6

RK	DEI	fname	lname	residence	p
1	e6	Fabian	Hanke	Kassel	0.6
2	e6	Florian	Hauke	Frankfurt	0.4

Figure 7.48.: Databases with duplicate scattered database entities

probability mass assigned to the third manuscript in another way on its two alternatives than the system of the third party does. As a consequence, the parties pdb_3 and pdb_4 extract the same set of alternatives, but assign different probabilities to them.

As mentioned above, party pdb_1 copies the data that was extracted from pdb_2. If the copying process would not make an error, the two databases pdb_1 and pdb_2 are identical. However, in our example, we assume that the copying process introduce an error by confusing the values in the attribute 'lname' *of the last two alternatives.*

The sixth party copies the data from pdb_5, but produces an error during the copy process by transposing the probabilities of both alternatives. As a consequence, pdb_5 and pdb_6 have the same two alternatives but ascribed the probabilities to these alternatives in the opposite way.

The seventh database pdb^* contains all these x-tuples and try to detect them as duplicates. As we can see in Figure 7.48 this endeavor is not easy made because the duplicate x-tuples have different number of alternatives, have different values in their alternatives and have different probabilities assigned to their alternatives.

Of course, in the example presented above, a detection of all these duplicates would be simple by classifying each pair of x-tuples as a MATCH if they have at least one similar pair of description alternatives. This approach, however, is often not suitable in practice, because it tends to produce too many false positives. Moreover, if the two compared x-tuples have hundred or more alternatives, the probability that they have high similar description alternative is close to zero. For all these reasons we strive for a more sophisticated detection approach.

7.5.2. Types of Description Discrepancies

As we have demonstrated by the previous example, the descriptions of scattered duplicates can differ from each other in several ways. Thus, before considering existing measures for computing the simi-

larity between two probability mass functions in detail, we take a closer look on the different types of discrepancies[7] that make a detection of scattered duplicates a real challenge. Detecting scattered duplicates becomes especially difficult if

- the two compared descriptions have **different numbers of alternatives** so that an 1:1 mapping between both sets of alternatives cannot be constructed.

  Recall, we depict several of such discrepancies in Example 133. For instance, whereas the description of entity e_1 has three alternatives, the description of entity e_3 has four alternatives even they consume their data from the same source. There exists various reasons for such discrepancies as for example as the usage of different (numbers of) sources or the usage of different methods for data collections.

- all the alternatives of the two compared descriptions have **differences in their attribute values or related entity sets**. As a consequence, computing an 1:1 mapping between both sets of alternatives can become difficult even in cases both descriptions have the same number of alternatives, because some alternatives of one description do not have a perfect match in the other description.

  For instance, in Example 133, the second alternative of x-tuple e_1 has another value in the attribute *'lname'* than its corresponding alternative of x-tuple e_5. Such discrepancies can be caused by different naming conventions, typos, value confusions, or data obsolesces.

- one of the two compared descriptions has a **confusion of features** (attribute values or related entities) between two of its alternatives. Recall, such a situation is presented by the two entities e_1 and e_2 in Example 133 because in e_2 the values in the attribute *'lname'* of the second alternative and the third alternative have been confused during the copying process. Such a discrepancy can be caused by errors in manual database insertions or can be caused by erroneous programming code.

- the two compared descriptions have assigned **different probability masses** to the same alternative. Such a discrepancy can be caused by meta data obsolescence, the usage of different methods for probability computation, etc.. In general, probabilities are often computed based on subjective perceptions and therefore it is not uncommon that independently performed computation processes produce different probability distributions. We present such a discrepancy in Example 133 by the two database entities e_3 and e_4. Both x-tuples share the same alternatives, but have distinct probability masses distributed among these alternatives because they have been produced by different extraction processes.

- one of the compared descriptions has **errors in their probabilities**. Such discrepancies can result from failures that are introduced by manual insertions or incorrect programming code. In general, errors in probabilities can be categorized into three classes:

[7] It is important to note that these discrepancies do not necessarily need to result from data errors, but they can be consequences of the data's inherent uncertainty. For instance, if the first description has the two alternatives 'A' and 'B' and the other description has the two alternatives 'A' and 'C', they differ from each other but none of them is incorrect if 'A' is the true alternative.

- ▷ confusion of alternatives, i.e. correct probability scores are assigned to the incorrect alternatives. We illustrated such an error in Example 133 where the copying process that creates entity e_6 confuses the probabilities of the two copied alternatives.
- ▷ confusion of digits, i.e. the digits within a probability score are confused during manual insertion. For instance, the probability 0.21 is mistakenly inserted as 0.12.
- ▷ typos within probability scores, i.e. wrong digits are inserted. For instance, the probability 0.21 is mistakenly inserted as 0.021 or 0.31.

Of course, the probabilities of all alternatives of a probabilistic entity description always sum up to one. However, the errors are usually introduced in the original data and in this case the created x-tuple can be maybe. Thus, a probability 0.21 can often be inserted as 0.12 without violating any of the database's constraints. Recall, for each database entity that only maybe belongs to the database an empty description alternative is constructed. Therefore, in the presence of such an error, the probability of the empty description alternative changes as well.

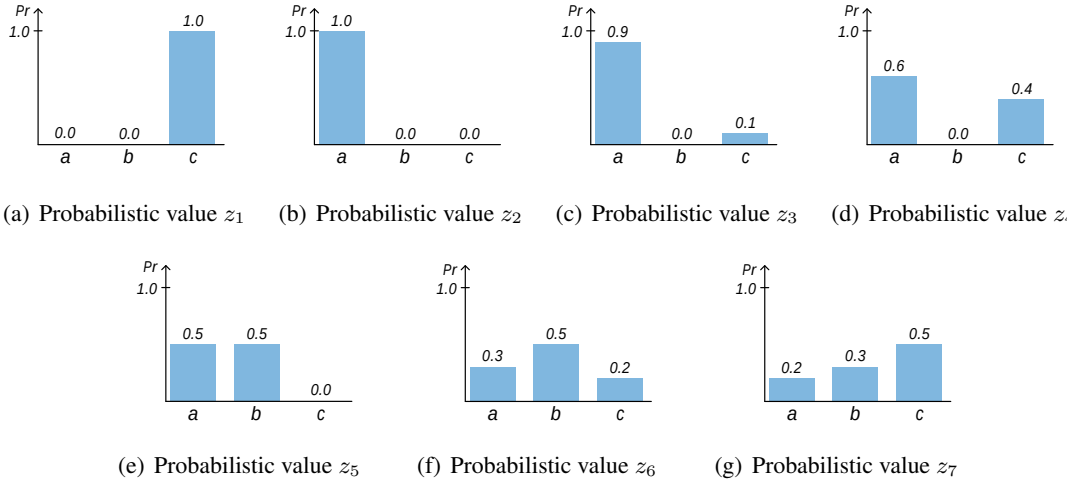
Whereas, the last two types of discrepancies are rather rare because their emergence requires specific errors in data collection, the first three types represent common discrepancies because they can already emerge if the data is collected from different sources, if the data is collected at different times, or if the data is collected by the use of different methods. Moreover, a confusion of attribute values (or entity sets) between different alternatives of the same description is a discrepancy that is specific for the domain of entity descriptions. As a consequence, such discrepancies are not considered in existing research on measuring similarity between probability mass functions. For that reason, we primarily focus on first three types in the rest of this section and shortly discuss an handling of the last two types separately in Section 7.5.5.

7.5.3. Measuring the Similarity between Probability Mass Functions

Intuitively, the similarity between two incomplete values can be measured by using a set-based similarity measure, because two incomplete values are the more similar the more alternatives they share. Accordingly, similarity of probabilistic values can be measured by using a similarity measure for probability mass functions (short *pmf*), because two probabilistic values are the more similar the more they assign a similar probability mass to same alternatives. The challenge of measuring similarity between uncertain values is to incorporate the different types of discrepancies we have presented in Section 7.5.2 into the measuring process.

In the case of incomplete values, besides a different number of alternatives only the resemblance of domain elements need to be taken into account and hybrid similarity measures as the Monge-Elkan Similarity (see Section 4.3.5.6) can be used to solve the given problem. In contrast, two probabilistic values cannot only contain discrepancies in their alternatives, but can also contain dissimilarities in their probability distributions. For that reason, measuring similarity is a much harder challenge for probabilistic values than for incomplete values.

The most distance/similarity measures for *pmf*s (or histograms) like the *Minkowski Distance* [Cha07], the *Intersection Distance* [DHA01], the *Cosine Similarity* [DD06], the *Bhattacharyya Distance* [Bha43],



(a) Probabilistic value z_1 (b) Probabilistic value z_2 (c) Probabilistic value z_3 (d) Probabilistic value z_4

(e) Probabilistic value z_5 (f) Probabilistic value z_6 (g) Probabilistic value z_7

Figure 7.49.: Sample probabilistic values of the domain $dom = \{a, b, c\}$

the *Pearson* χ^2 *Divergence* [Pea00], or the *Kullback-Leibler Distance* [KL51] (for an comprehensive overview see [Cha07]) do not consider similarities between domain elements and hence are only useful to detect scattered duplicates if their probability mass is distributed on same domain elements in different ways. Measures for comparing two *pmf*s that additionally consider the similarity between individual domain elements are the *Earth Mover's Distance* [RTG00], the *Quadratic Form Distance* [NBE+93], the *Match Distance* [SW83, WPR85], and the *Kolmogorov-Smirnov Distance* [RTG00]. Rubner et al. [RTG00] show that the Quadratic Form Distance provides counterintuitive similarity scores in many cases. Moreover, the Match Distance as well as the Kolmogorov Smirnov Distance are based on the use of cumulative histograms which require a total ordering of the underlying domain. Since the space of possible description alternatives is not ordered, we cannot use them for our purpose. For that reason, we focus on the Earth Mover's Distance and shortly present it in Section 7.5.3.1. Furthermore, we additionally introduce a probabilistic version of the Monge-Elkan Similarity in Section 7.5.4.

For illustrating the different approaches for measuring probabilistic value similarity that we will discuss in this section, we use seven probabilistic values that are all defined on the simple domain $dom = \{a, b, c\}$ and are graphically depicted in Figure 7.49. Moreover, we assume a symmetric similarity measure sim_{dom} that is defined on the elements of dom. The similarity of one domain element to itself is always 1 and the remaining similarities are defined as $sim_{dom}(a, b) = sim_{dom}(b, a) = 0.6$, $sim_{dom}(a, c) = sim_{dom}(c, a) = 0.1$, and $sim_{dom}(b, c) = sim_{dom}(c, b) = 0.4$. Sometimes, the notion of distances is more intuitive for the considered measures than the notion of similarities. In that case, we define the distance between two domain elements $v_i, v_j \in dom$ as $dst(v_i, v_j) = 1 - sim_{dom}(v_i, v_j)$.

Since we work on probabilistic entity description, we use the notion of probability values instead of probability mass functions or signatures. Nevertheless, because each probabilistic value corresponds to a *pmf* we can use this notion without any loss of generality.

7.5.3.1. Earth Mover's Distance

First introduced by Rubner et al. [RTG00], the Earth Mover's Distance (short *EMD*) is a measure to quantify the similarity between signatures in pattern recognition. Signatures are generic descriptions of distributions that assign masses to features (e.g. clusters, textures, or colors) and can be considered as a generalization of histograms (and hence of *pmf*s). The *EMD* is equivalent to the *Mallows Distance* [Mal72] (also known as *Wasserstein Distance* [Rac84]) if the compared signatures include the same total amount of mass as it is given by comparing two *pmf*s [LB01].

The underlying idea of the *EMD* is to transform one signature into another by moving mass between the signatures' features. For illustration purposes, the source signature can be considered as a mass of earth that is spread at different locations and the target signature can be considered as a collection of holes that need to be filled with earth. The *EMD* measures the least amount of work that is required to fill the earth into the holes where a unit of work is considered as transporting a specific mass of earth by a specific ground distance.

Computing the *EMD* of two given signatures can be reduced to the transportation problem [Hit41] that is also known as the Monge-Kantorovich problem [Rac84]. It is a well known linear programming problem that assumes that a set of suppliers, each equipped with an amount of the same goods, are required to supply a set of consumers, each having a specific demand on that goods. For each pair of supplier and consumer, the transportation costs for a single unit of goods is given. The transportation problem is now to find the minimal cost that is required to transport the goods from the suppliers to the consumers so that the consumers' demands are satisfied.

The original problem definition of computing the *EMD* given by Rubner et al. [RTG00] is not constrained to cases where the total amount of goods owned by the suppliers is equal to the total amount of goods demanded by the consumers. Moreover, the resultant costs are not normalized.

By considering signatures as probabilistic values, however, the total mass of each signature is 1, i.e. both signatures have same mass and the overall costs is upper bounded by 1. In this case, the problem formulation can be simplified as described in [Cha08].

Let dom be an arbitrary domain and let $z_s = (y_s, Pr_s)$ and $z_c = (y_c, Pr_c)$ be two probabilistic values of that domain, i.e. $z_s, z_c \in^{[\rho]} dom$. The alternatives $y_s = \{s_1, \ldots, s_k\} \subseteq dom$ are considered as supplier and the alternatives $y_c = \{c_1, \ldots, c_l\} \subseteq dom$ are considered as consumer. Let d_{ij} be the distance between s_i and c_j and let m_{ij} be the probability mass that is transported from s_i to c_j. Computing the *EMD* between z_s and z_c corresponds to solve the hard optimization problem of finding the mass transportation flow $f = \langle m_{11}, \ldots, m_{kl} \rangle$ that minimizes the overall costs and hence results in:

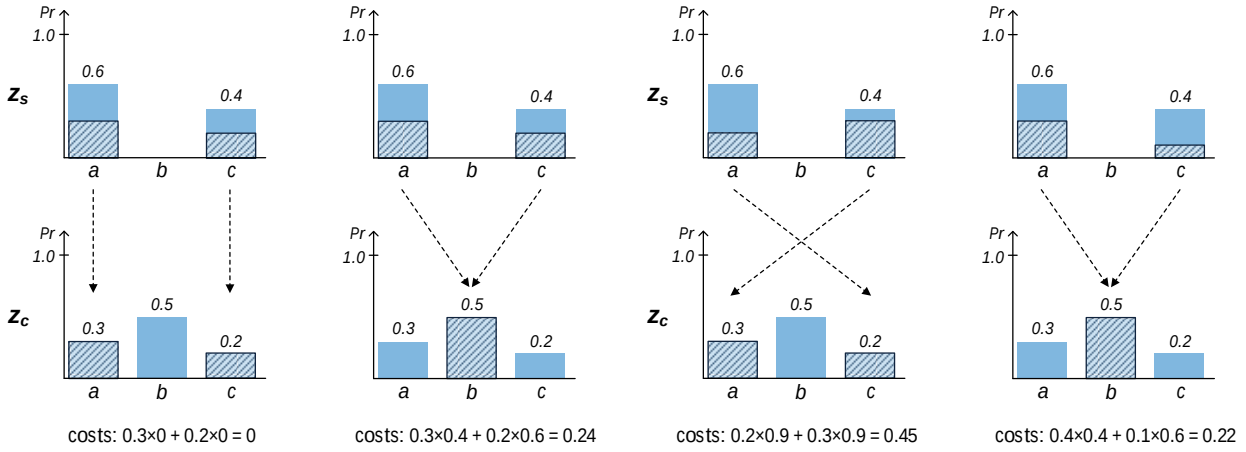
$$EMD(z_s, z_c) = \mathbf{min}_f \left(\sum_{i=1}^{k} \sum_{j=1}^{l} d_{ij} \times m_{ij} \right)$$

so that the following conditions are satisfied:

(i) mass is only unidirectionally transported from z_s to z_c: $\forall i \in \{1, \ldots, k\}, j \in \{1, \ldots, l\} : m_{ij} \geq 0$

(ii) each $s \in y_s$ supplies exact the mass it has: $\forall i \in \{1, \ldots, k\} : \sum_{j=1}^{l} m_{ij} = Pr_s(s_i)$

(iii) each $c \in y_c$ receives exact the mass it demands: $\forall j \in \{1, \ldots, l\} : \sum_{i=1}^{k} m_{ij} = Pr_c(c_j)$



(a) Optimal mass transportation flow f_1

(b) Non-optimal mass transportation flow f_2

$$D = \begin{bmatrix} dst(a,a) & dst(b,a) & dst(a,c) \\ dst(b,a) & dst(b,b) & dst(b,c) \\ dst(c,a) & dst(c,b) & dst(c,c) \end{bmatrix} = \begin{bmatrix} 0 & 0.4 & 0.9 \\ 0.4 & 0 & 0.6 \\ 0.9 & 0.6 & 0 \end{bmatrix}$$

(c) Distance matrix

$$F_1 = \begin{bmatrix} 0.3 & 0.3 & 0 \\ 0 & 0 & 0 \\ 0 & 0.2 & 0.2 \end{bmatrix} \quad F_2 = \begin{bmatrix} 0.0 & 0.4 & 0.2 \\ 0 & 0 & 0 \\ 0.3 & 0.1 & 0 \end{bmatrix}$$

(d) Mass transportation matrices of the flows f_1 and f_2

Figure 7.50.: Sample of an optimal mass transportation and an non-optimal mass transportation from v_s to v_c

The distances between suppliers and consumers correspond to the distances between individual elements of the domain dom and are usually given by a distance matrix D. The *EMD* is symmetric if the used distance matrix is symmetric.

Example 134 *For illustrating the EMD we consider the two probabilistic values z_4 and z_6 from Figure 7.49 and use the distance matrix presented in Figure 7.50(c). Value z_3 can be transformed into z_6 in several ways. Obviously, the cheapest transformation is to transport the mass from $z_s = z_4$ to $z_c = z_6$ as shown in Figure 7.50(a). Since transporting mass from a domain element to itself does not cost anything, the first transportation step is cost-free. Thus, the overall costs only result from transporting mass from $Pr_s(a)$ to $Pr_c(b)$ and from transporting mass from $Pr_s(c)$ to $Pr_c(b)$. In contrast, the transformation presented in Figure 7.50(b) is non-optimal because mass is unnecessarily transported from $Pr_s(a)$ to $Pr_c(c)$ and from $Pr_s(c)$ to $Pr_c(a)$. The two resultant mass transportation flows are additionally presented by the two matrices F_1 and F_2 in Figure 7.50(d).*

The standard algorithms for solving the transportation problem such as the *Transportation Simplex Method* [HL90] or the *Hungarian Algorithm* [Kuh55], are known to have polynomial time complexity, i.e. $\mathcal{O}(n^3 log\ n)$ (Simplex Method) [AMO93] and $\mathcal{O}(n^3)$ (Hungarian Algorithm) [EK72] where $n = |y_s \cup y_c|$. Although complexity is usually far away from the worst case, computing the *EMD* by using one of these algorithms can become extremely inefficient if it is applied to large histograms (and hence to probabilistic values with many alternatives) [RTG00]. In general, the complexity essentially depends on the used distance matrices, because for some type of matrices it has been shown that the problem can

be solved in linear time [CS02, Cha08]. However, due to our distance matrices will in most cases neither be nominal, ordinal, modulo, nor shuffle-ordinal, none of the algorithm presented by Cha [Cha08] can be used for our purpose.

A linear time approximation for computing the *EMD* of low dimensional histograms has been proposed by Shirdhonkar and Jacobs [SJ08]. Due to each probabilistic value corresponds to a one dimensional histogram, it can be used for matching entities with probabilistic descriptions. Pele and Werman [PW09] proposed a fast and robust algorithm that is based on the use of saturated distances. In theory, this algorithm is still in $\mathcal{O}(n^3 log\ n)$, but in practice it computes the *EMD* by an order of magnitude faster than the algorithm used by Rubner et al. [RTG00]. Moreover, Pele and Taskar have recently proposed two improved variants of the *EMD* in [PW08] and [PT13]. Note, in the experimental evaluations that are presented in Section 9.4 we use the open available java library *JFastEMD*[8] that is based on the algorithm of Pele and Werman [PW09].

7.5.4. Probabilistic Monge-Elkan Similarity

The Earth Mover's Distance has two shortcomings. First it can become computational expensive even if one of its fast implementations is used and second it is not straightforward to incorporate correlations between the two compared distributions into the matching process. In matching probabilistic entity descriptions, however, correlations can be introduced into in the alternative pair space. For that reason, we propose a new measure in this section that we call the *Probabilistic Monge-Elkan Similarity* (short *PME-Sim*).

The Probabilistic Monge-Elkan Similarity is based on the idea of the Monge-Elkan Similarity [ME96] that is a hybrid measure for set similarity (see Section 4.3.5.6). Let $y_i, y_j \in^{[\iota]} dom$ be two incomplete values of an arbitrary domain dom, the Monge-Elkan Similarity from y_i to y_j[9] computes the average maximal similarity from an alternative of y_i to the alternatives of y_j. For computing the similarity between two alternatives $x, x' \in dom$, the Monge-Elkan Similarity utilizes an internal similarity measure sim_{dom} that is defined on the elements of dom.

In the case of probabilistic values, we also need to take probabilities into account. We accomplish this purpose by using a similar principle has we have utilized in the maximal-θ similarity (see Section 7.2.6). Let $z_i, z_j \in^{[\rho]} dom$ be two probabilistic values of the domain dom, instead of computing the maximal similarity from x to the alternatives in z_j for each alternative $x \in alt(z_i)$, we select the probability mass equivalent part of z_j that is most similar to x and compute the expected similarity between x and the selected part of z_j. The Probabilistic Monge-Elkan Similarity from z_i to z_j is then defined as the expected value of the similarity scores that have been computed for the alternatives of z_i.

Consequently, the computation can be divided into two parts. In the first part, for each alternative of z_i a single similarity score is computed by comparing it with the alternatives of z_j. In the second part, the similarity scores that are computed for the alternatives of z_i are combined by computing their expectation value. The first part is similar to the maximal-θ similarity. Thus, let $z_i = (y_i, Pr_i), z_j =$

[8] https://github.com/telmomenezes/JFastEMD

[9] Note, the Monge-Elkan Similarity (and thus the Probabilistic Monge-Elkan Similarity) is asymmetric. For that reason we always write '*the similarity from x to y*' instead of '*the similarity between x and y*' if the similarity is only computed unidirectional.

$(y_j, Pr_j) \in^{[\rho]} dom$ be two probabilistic values of the domain dom, the similarity from a single alternative $x \in y_i$ to the probabilistic value z_j (in the following denoted as $sim(z_i, z_j, x)$) is computed in the following four steps:

1. first, we initialize the accumulative probability $p_\sum$ and the to computed similarity score $sim(z_i, z_j, x)$ to zero,
2. second, we sort the alternatives of z_j by their similarity to x in descendant order,
3. then we iterate over each alternative $x' \in y_j$ of the sorted list, compute the probability $p_{x'}$ for this alternative, and add this probability to the accumulative probability $p_\sum$. If the resultant accumulative probability is then lower than the probability of x, i.e. $p_\sum < Pr_i(x)$, we increase $sim(z_i, z_j, x)$ by
$$p_{x'} \times sim_{dom}(x, x')$$
and continue with the next iteration step. Otherwise, we increase $sim(z_i, z_j, x)$ by
$$(Pr_i(x) + p_{x'} - p_\sum) \times sim_{dom}(x, x')$$
and stop the iteration.

   In the standard approach, we define the probability $p_{x'}$ as the probability of x' in z_j, i.e. $p_{x'} = Pr_j(x')$. Nonetheless, because in some cases (for example see Section 7.5.4.1) another assignment can be more useful, we introduce a more generalized notion by utilizing $p_{x'}$ instead of $Pr_j(x')$.
4. finally, we divide the score $sim(z_i, z_j, x)$ by $Pr_i(x)$ so stretch its possible output range from $[0, Pr_i(x)]$ to $[0, 1]$.

Let S be the set of all alternatives of z_i that have been selected in any iteration step, let x_{min} be the alternative that has been selected in the last iteration, i.e. $x_{min} = \textbf{argmin}_{x' \in S} sim_{dom}(x, x')$, let $S_{max} = S - \{x_{min}\}$ and let $p_{min} = Pr_i(x) - \sum_{x' \in S_{max}} p_{x'}$ (the amount of the probability from x_{min} that is taken into account for computing $sim(z_i, z_j, x)$), the above described computation process can be formulated as:

$$sim(z_i, z_j, x) = \frac{1}{Pr_i(x)} \left(\sum_{x' \in S_{max}} p_{x'} \times sim_{dom}(x, x') + p_{min} \times sim_{dom}(x, x_{min}) \right) \tag{7.13}$$

The high resemblance of $sim(z_i, z_j, x)$ to the maximal-θ similarity can be simply seen by comparing Equation 7.13 with Equation 7.6. Nevertheless, as we will see in the remainder of this section there are some refined distinctions between both measures.

By using the above presented measure for computing the similarity from an alternative of one probabilistic value to another probabilistic value, the Probabilistic Monge-Elkan Similarity from $z_i = (y_i, Pr_i)$ to $z_j = (y_j, Pr_j)$ is then defined as:

$$\textit{PME-Sim}(z_i, z_j) = \sum_{x \in y_i} Pr_i(x) \times sim(z_i, z_j, x) \tag{7.14}$$

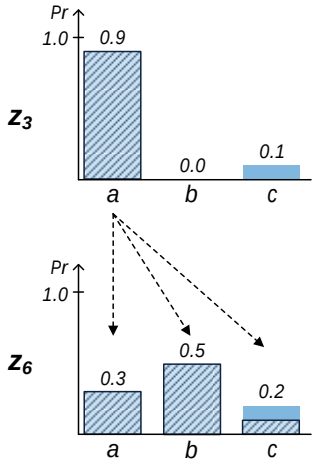
Like the traditional Monge-Elkan Similarity, the Probabilistic Monge-Elkan Similarity is asymmetric, i.e. $\textit{PME-Sim}(z_i, z_j) \neq \textit{PME-Sim}(z_j, z_i)$ in some cases. To neglect that disadvantage, we can calculate the similarity in both directions and then aggregate the two resultant scores by computing their minimum, their maximum, or their average.

Pos.	x'	$sim(a,x')$	$p_{x'}$	p_Σ	$sim(z_3,z_6,a)$
1	-	-	-	0	0
2	a	1.0	0.3	0.3	+ 0.3 × 1.0
3	b	0.6	0.5	0.8	+ 0.5 × 0.6
4	c	0.1	0.2	1.0	+ (0.9-1.0+0.2) × 0.1
					= 0.61/0.9 = 0.678

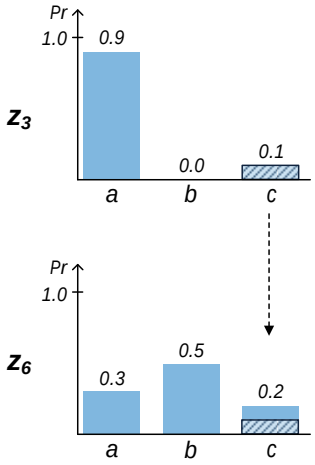
(a) Computation of the similarity score $sim(z_3, z_6, a)$

Pos.	x'	$sim(c,x')$	$p_{x'}$	p_Σ	$sim(z_3,z_6,c)$
1	-	-	-	0	0
2	c	1.0	0.2	0.2	+ (0.1-0.2+0.2) × 1.0
3	b	0.4	0.5	-	-
4	a	0.1	0.3	-	-
					= 0.1/0.1 = 1.0

(b) Computation of the similarity score $sim(z_3, z_6, c)$



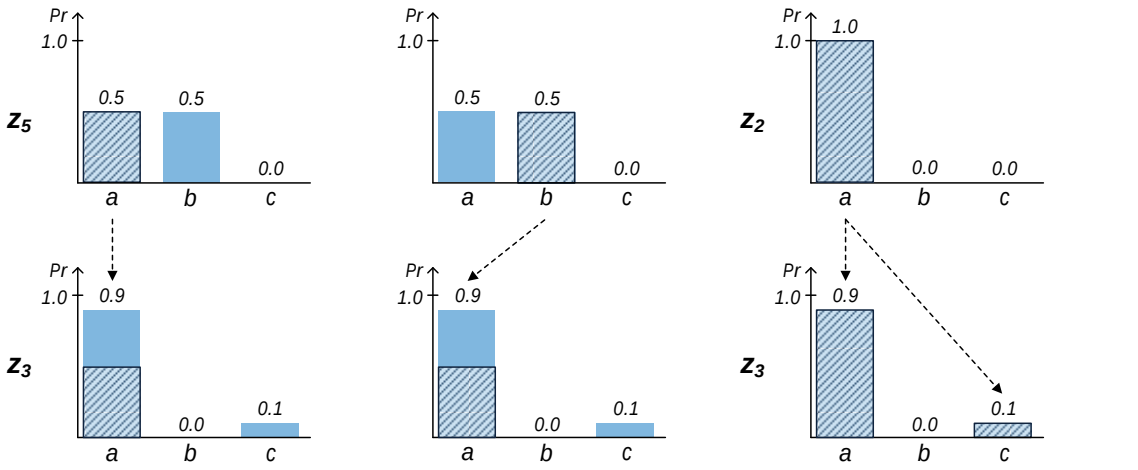
(c) Probability mass mapping from $Pr_3(a)$ to z_6



(d) Probability mass mapping from $Pr_3(c)$ to z_6

Figure 7.51.: Sample for the computing the similarities from alternatives to probabilistic values

Example 135 *For illustration, we consider the computation of the Probabilistic Monge-Elkan Similarity from the probabilistic value z_3 to the probabilistic value z_6 from our example presented in Figure 7.49. For each of the two alternatives of z_3, we compute the probability mass equivalent part of z_6 that is most similar to it. The process for computing the similarity from $a \in alt(z_3)$ to z_6 is presented in Figure 7.51(a) and works as follows: First we set the accumulative probability and the computed similarity score to zero. Then, a, b, and c are sorted by their similarity to a. Next, we take a, compute its probability as $p_a = Pr_6(a) = 0.3$, and add that probability to the accumulative probability p_Σ. Due to this probability is lower than $Pr_3(a) = 0.9$, we increase the similarity score by $p_a \times sim_{dom}(a, a) = 0.3 \times 1.0 = 0.3$ and continue with the next iteration step. In this iteration step, we take b, compute its probability as $p_b = Pr_6(b) = 0.5$, and add it to p_Σ. Since the accumulative probability is still lower than 0.9, we increase the similarity score by $p_b \times sim_{dom}(a, b) = 0.5 \times 0.6 = 0.3$ and start the next iteration step. In that step, we take c, compute its probability as $p_c = Pr_6(c) = 0.2$ and add it to p_Σ. Now, the accumulative probability is greater than 0.9. Thus, we increase $sim(z_3, z_6, a)$ only by considering the still missing amount of mass instead by the considering the complete probability mass p_c, i.e. we increase the similarity score by $0.2 \times sim_{dom}(a, c) = 0.1 \times 0.1 = 0.01$ and stops the iteration process. Finally, the computed similarity score $sim(z_3, z_6, a) = 0.3 + 0.3 + 0.01 = 0.61$ is divided it by $Pr_3(a) = 0.9$ and hence results in $sim(z_3, z_6, a) = 0.678$.*



(a) Probability mass mapping from z_5 to z_3

(b) Probability mass mapping from z_2 to z_3

Figure 7.52.: Probability mass mappings between z_2 and z_3, and between z_5 and z_3

As it is graphically presented in Figure 7.51(c), the probability mass $Pr_3(a)$ *is mapped to all three alternatives of* z_6*. In contrast, the probability mass* $Pr_3(c)$ *is only mapped to* $c \in alt(z_6)$ *(see Figure 7.51(d)). The process of computing* $sim(z_3, z_6, c)$ *is presented in Figure 7.51(b). In this case, the computation stops after the first iteration step and produces the similarity score* $sim(z_3, z_6, c) = 0.1 \times sim_{dom}(c, c)/0.1 = 1.0$*.*

Using the two similarity scores computed above, the Probabilistic Monge-Elkan Similarity between z_3 *and* z_6 *results in PME-Sim*$(z_3, z_6) = 0.9 \times 0.678 + 0.1 \times 1.0 = 0.71$*. Note, since each probability mass of* z_3 *is mapped to another probability mass of* z_6*, the Probabilistic Monge-Elkan Similarity is symmetric in this example, i.e. PME-Sim*$(z_3, z_6) =$ *PME-Sim*(z_6, z_3)*. Moreover, it is equivalent to the inverse of the Earth Mover's Distance which is EMD*$(z_3, z_6) = 0.29$*.*

Of course, the more uncertain two probabilistic values are, the less likely is the chance that they actually represent the same real-world property (for more details see Section 7.5.6). Nevertheless, similarity measures (of what domain ever) generally focus on syntactical or semantical resemblance instead on the above described likelihood. For instance, two equivalent string values describing the first names of persons are usually considered to be maximal similar even if this name is commonly used and an agreement in this name is not a large indication for real-world equivalence. For that reason, we make the same assumption for probabilistic values and consider it to be meaningful that a probabilistic value is equal (and hence maximal similar) to itself.

Theorem 12 *The Probabilistic Monge-Elkan Similarity returns the maximal possible similarity score* 1.0 *if a probabilistic value is matched with itself. Thus, let* z *be a probabilistic value of an arbitrary domain, it holds that PME-Sim*$(z, z) = 1.0$*.*

Theorem 12 can be proved as follows:

Proof 8 *Let* $z_i = (y_i, Pr_i)$ *and* $z_j = (y_j, Pr_j)$ *be two probabilistic values of the same domain. For calculating the similarity PME-Sim*(z_i, z_j)*, we compute for each alternative* $x \in y_i$ *the probabil-*

ity mass equivalent part of z_j that is most similar to x. If both probabilistic values are equal, i.e. $\forall x \in y_i \colon Pr_i(x) = Pr_j(x)$, it holds that for each alternative $x \in y_i$ its most similar probability mass equivalent part of z_j is x itself. Thus, the similarity $sim(z_i, z_j, x)$ is one for each alternative $x \in y_i$ and therefore the similarity PME-Sim(z_i, z_j) is one as well.

Moreover, the concept of probabilistic values is a generalization of the concept of crisp values. Thus, a meaningful similarity measure for probabilistic values should be conform to the internally used measure for domain element similarity, if the compared probabilistic values are both crisp, i.e. have a single alternative.

Theorem 13 *The Probabilistic Monge-Elkan Similarity returns the same similarity as its internally used similarity measure if the two matched values are crisp. Thus, let sim_{dom} be the internally used similarity measure that is defined on domain dom and let $z_i, z_j \in^{[\rho]} dom$ be two probabilistic values of that domain, if $z_i = \{(x_1 : 1.0)\}$ and $z_j = \{(x_2 : 1.0)\}$ it follows that PME-Sim(z_i, z_j) = PME-Sim(z_j, z_i) = $sim_{dom}(x_1, x_2)$.*

Theorem 13 can be proved as follows:

Proof 9 *Let $z_i = (y_i, Pr_i)$ and let $z_j = (y_j, Pr_j)$. Because z_i has only the single alternative x_1, it holds that PME-Sim$(z_i, z_j) = sim(z_i, z_j, x_1)$. Since z_j has only the single alternative x_2, x_1 can only be mapped to x_2. Thus it follows that $sim(z_i, z_j, x_1) = \frac{1}{Pr_i(x_1)} \times Pr_j(x_2) \times sim_{dom}(x_1, x_2)$. Due to $Pr_i(x_1) = Pr_j(x_2) = 1.0$ it follows that $sim(z_i, z_j, a) = sim_{dom}(x_1, x_2)$ and therefore follows that PME-Sim$(z_i, z_j) = sim_{dom}(x_1, x_2)$.*

The symmetry of PME-Sim(z_i, z_j) can be simply proven by exchanging x_1 and x_2.

In general, if each probability mass of the probabilistic value z_i is mapped to another probability mass of the probabilistic value z_j, the Probabilistic Monge-Elkan Similarity performs an 1:1 mapping between both probability distributions, i.e. it is symmetric if the internally used similarity measure is symmetric, and its produced mapping is maximum weighted. Thus, in this case the Probabilistic Monge-Elkan Similarity is equivalent to the inverse of the Earth Mover's Distance, i.e. in this case it holds PME-Sim(z_i, z_j) = PME-Sim$(z_j, z_i) = 1 -$ EMD(z_i, z_j).

The Probabilistic Monge-Elkan Similarity suffers from a similar problem as the conventional Monge-Elkan Similarity. Because it does not necessarily produce an 1:1 mapping between the alternatives of both values, it is not only asymmetric, but sometimes it also produces similarity scores that are odd with respect to intuitive perceptions.

Example 136 *We want to illustrate this effect by comparing the Probabilistic Monge-Elkan Similarity from z_5 to z_3 with the Probabilistic Monge-Elkan Similarity from z_2 to z_3. The probability mass mappings of the first similarity are presented in Figure 7.52(a) and the probability mass mappings of the second similarity are presented in Figure 7.52(b). Note, because both mass mappings are considered independently and $Pr_3(a)$ is greater than $Pr_5(a)$ and $Pr_3(a)$ is greater than $Pr_5(b)$, the probability mass $Pr_5(a)$ as well as the probability mass $Pr_5(b)$ are both completely mapped to the probability mass $Pr_3(a)$ although their accumulative mass is greater than $Pr_3(a)$. In contrast, the probability mass $Pr_2(a)$ is not only mapped to the probability mass $Pr_3(a)$, but is also mapped to the mass $Pr_3(c)$. The*

crucial point in this example is that in the case the similarity between a and b is very high ($a \simeq b$) and the similarity between a and c is very low, the two probabilistic values z_2 and z_5 are very similar to each other (in both directions), but have a complete different similarity to z_3. Whereas the similarity from z_5 to z_3 is high, the similarity from z_2 to z_3 is rather low and that only because the probability mass of element a in z_2 is distributed on the similar elements a and b in z_5.

By considering our initially defined element similarities, i.e. $sim_{dom}(a,b) = 0.6$, $sim_{dom}(a,c) = 0.1$, and $sim_{dom}(b,c) = 0.4$, the Probabilistic Monge-Elkan Similarity from z_5 to z_3 is 0.8, from z_3 to z_5 is 0.78, and from z_2 to z_3 (as well as from z_3 to z_2) is 0.91. Thus, the independently produced mapping from z_5 to z_3 causes that the Probabilistic Monge-Elkan Similarity becomes asymmetric. For comparison, the Earth Mover's Distance between z_3 and z_5 is 0.22 and the Earth Mover's Distance between z_2 and z_3 is 0.09. As a consequence, even in the asymmetric case of matching z_3 and z_5 the correlation between the Probabilistic Monge-Elkan Similarity and the inverse of the Earth Mover's Distance is considerably high.

The disadvantage of odd similarity scores can be mitigated by computing the Probabilistic Monge-Elkan Similarity in both directions, but cannot be completely erased. Nevertheless, as we will experimentally evaluate in Section 9.4, the Probabilistic Monge-Elkan Similarity most often provides scores that are highly similar to the scores that are produced by the Earth Mover's Distance. Moreover, the Probabilistic Monge-Elkan Similarity is particularly suitable for our purpose, because (a) its computation is usually an order of magnitude faster than this of the Earth Mover's Distance (see Section 9.4), (b) correlations between the two compared probabilistic values can be simply incorporated (see Section 7.5.4.1), and (c) it can be computed incrementally which makes it especially useful for the optimization techniques that we present in Section 7.6.

Let us assume that we match a probabilistic value z_i with m alternatives with a probabilistic values z_j with n alternatives. First, we have to match each alternative of z_i with each alternative of z_j, i.e. $\mathcal{O}(n \times m)$. Then, for each alternative of z_i we need to sort the alternatives of z_j, i.e.$m \times \mathcal{O}(n \times \log n)$, and then we have to scan the sorted list once, i.e. $m \times \mathcal{O}(n)$. Therefore, the complexity of the presented algorithm is of class $\mathcal{O}(m \times n \times \log n)$.

Up to this point we consider the matching of two independent probabilistic values. In the description-based detection approach, however, we do not simply match two independent probabilistic entity descriptions, but work on an alternative pair space that can include correlations instead. Moreover, in order to decompose the strong influence of non-membership on the matching result, we have to deal with the impact values that have been computed for the individual alternative description pairs.

7.5.4.1. Incorporation of Value Correlations

The aggregation methods presented in Section 7.2 are defined on combining pairwise matching results. As a consequence, they do not need to be adapted to dependencies between description alternatives because these correlations are already modeled within the input data. In contrast, the Probabilistic Monge-Elkan Similarity from one probabilistic value to another probabilistic value is based on matching each alternative of the first value with all alternatives of the second value at a single blow and therefore does

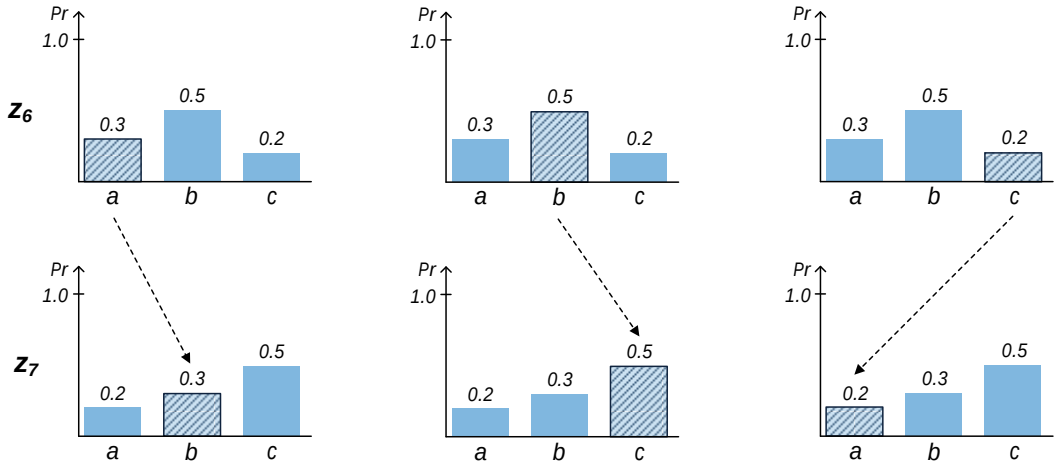


Figure 7.53.: The 1:1 probability mass mappings between the alternatives of z_6 and the alternatives of z_7

not take correlations into account. Therefore, an adaptation to dependencies between description alternatives is necessary, but is not straightforward.

Correlations between two probabilistic values are modeled by a joint probability distribution. A joint probability distribution is in turn a a generalization of two independent probability distributions, because each pair of independent probability distributions can be represented by a joint probability distribution. Thus, the adapted variant of the Probabilistic Monge-Elkan Similarity should produce the same similarity score as the original variant in cases of independence. Moreover, it seems plausible that an alternative of the second value should only contribute to the similarity from an alternative of the first value to the second value if the combination of the two alternatives is possible. For instance, if the dependencies already imply an 1:1 mapping between the alternatives of both values, the Probabilistic Monge-Elkan Similarity should be equivalent to the expected similarity of the corresponding alternative description pairs because it represents the expected similarity of the only possible mapping between the given set of description alternatives.

Example 137 *For illustration, we consider the two probabilistic values* $z_6 = (y_6, Pr_6)$ *and* $z_7 = (y_7, Pr_7)$ *that are depicted in Figure 7.49. A situation of an 1:1 mapping between both values is illustrated by the joint probability distribution* $Pr^B_{6,7}$ *presented in Figure 7.54(b) (each row corresponds to an alternative of* z_6 *and each column corresponds to an alternative of* z_7*). For example, element* a *can only be the true alternative of* z_6 *if element* b *is the true alternative of* z_7 *and vice versa. Thus, for computing the similarity* $sim(z_6, z_7, a)$ *only the similarity from* a *to* b *should be taken into account. The consequential probability mass mapping is depicted in Figure 7.53. Recall, in such a case of an 1:1 mapping between the alternatives, the probability mass mapping from* z_7 *to* z_6 *is the inverted one and the Probabilistic Monge-Elkan Similarity is symmetric.*

Let $z_i = (y_i, Pr_i)$ and $z_j = (y_j, Pr_j)$ be two probabilistic values. From the observations above we can conclude three hard conditions for matching an alternative $x \in y_i$ with z_j if a joint probability distribution $Pr_{i,j}$ from z_i and z_j is given:

- in cases of independence the adapted variant has to produce the same result for the similarity $sim(z_i, z_j, x)$ as the original variant of the Probabilistic Monge-Elkan Similarity,
- an alternative of $x' \in y_j$ that is not possible in combination with x cannot contribute to the similarity $sim(z_i, z_j, x)$, i.e. we can change the similarity between x' and x without changing $sim(z_i, z_j, x)$,
- to guarantee that for each $x \in y_i$ there exists a probability mass equivalent part of z_j, the accumulative probability $p_\Sigma = \sum_{x' \in y_j, Pr_{i,j}(x,x')>0} p_{x'}$ has to be always greater than or equal to the probability $Pr_i(x)$.

Note, the last two conditions ensure the result that we expect in the presence of an 1:1 mapping between the alternatives of the compared probabilistic values.

By dividing the joint probability distribution $Pr_{i,j}$ with Pr_j (0 divided by 0 is counted as zero), we get a matrix where the values of one row can be considered as the priority the different alternatives of z_j have for a specific alternative of z_i. The priority scores can be normalized by dividing them with the maximal priority score that belongs to the same row.

Let x be an alternative of value z_i and let x' be an alternative of value z_j, the unnormalized priority score of x' to x (short $UPM_{i,j}(x, x')$) and the normalized priority score of x' to x (short $NPM_{i,j}(x, x')$) are therefore defined as:

$$UPM_{i,j}(x, x') = \frac{Pr_{i,j}(x, x')}{Pr_j(x')} \tag{7.15}$$

$$NPM_{i,j}(x, x') = \frac{UPM_{i,j}(x, x')}{\mathbf{max}_{x^* \in y_j} UPM_{i,j}(x, x^*)} = \frac{\frac{Pr_{i,j}(x,x')}{Pr_j(x')}}{\mathbf{max}_{x^* \in y_j} \frac{Pr_{i,j}(x,x^*)}{Pr_j(x^*)}} \tag{7.16}$$

Example 138 *Figure 7.54 presents the computation of the normalized priority scores for three different joint probability distributions of the two probabilistic values z_6 and z_7 from Figure 7.49. The first distribution models a case of independence ($Pr^A_{6,7}$ in Figure 7.54(a)), the second distribution represents an 1:1 mapping between the alternatives of z_6 and the alternatives of z_7 ($Pr^B_{6,7}$ in Figure 7.54(b)), and the last distribution represents a 2:2 mapping between the alternatives of z_6 and the alternatives of z_7 ($Pr^C_{6,7}$ in Figure 7.54(c)). The three matrices $UPM^A_{6,7}$, $UPM^B_{6,7}$, and $UPM^C_{6,7}$ contain the unnormalized priority scores that result from dividing the joint probabilities by the probabilities from Pr_7. The three matrices $NPM^A_{6,7}$, $NPM^B_{6,7}$, and $NPM^C_{6,7}$ contain the corresponding normalized priority scores. As we can see the normalized priority scores are all one in the case of independence. That is intuitive, because no alternative is preferred to another. In the case of the 1:1 mapping, each row has only a single value that is greater than zero, i.e. the priority score of the alternative that is the only possible for the alternative of the corresponding row. This result is intuitive as well because for each alternative of z_6 only one alternative of z_7 has to be considered and this alternative has therefore the maximal possible priority. In the third example, the priority scores are mixed, but some alternatives of z_7 have a higher priority score to a specific alternative of z_6 than the others.*

Since the second part of the Probabilistic Monge-Elkan Similarity that combines the similarity scores of the individual alternatives by computing the expectation value only depends on the probability distribution of the first probabilistic value, it does not need to be adapted to correlations between the values'

$$Pr^A_{6,7} = \begin{bmatrix} 0.06 & 0.09 & 0.15 \\ 0.1 & 0.15 & 0.25 \\ 0.04 & 0.06 & 0.1 \end{bmatrix} \qquad \mathit{UPM}^A_{6,7} = \begin{bmatrix} 0.3 & 0.3 & 0.3 \\ 0.5 & 0.5 & 0.5 \\ 0.2 & 0.2 & 0.2 \end{bmatrix} \qquad \mathit{NPM}^A_{6,7} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(a) Priority score computation in the case of independence

$$Pr^B_{6,7} = \begin{bmatrix} 0 & 0.3 & 0 \\ 0 & 0 & 0.5 \\ 0.2 & 0 & 0 \end{bmatrix} \qquad \mathit{UPM}^B_{6,7} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad \mathit{NPM}^B_{6,7} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

(b) Priority score computation in the case of an 1:1 mapping between the set of alternatives

$$Pr^C_{6,7} = \begin{bmatrix} 0.1 & 0.2 & 0 \\ 0 & 0.1 & 0.4 \\ 0.1 & 0 & 0.1 \end{bmatrix} \qquad \mathit{UPM}^C_{6,7} = \begin{bmatrix} 0.5 & 0.67 & 0 \\ 0 & 0.33 & 0.8 \\ 0.5 & 0 & 0.2 \end{bmatrix} \qquad \mathit{NPM}^C_{6,7} = \begin{bmatrix} 0.75 & 1 & 0 \\ 0 & 0.42 & 1 \\ 1 & 0 & 0.4 \end{bmatrix}$$

(c) Priority score computation in the case of a 2:2 mapping between the set of alternatives

Figure 7.54.: Three samples for mapping joint probability distributions to normalized priority scores

alternatives. As a consequence, we only need to adapt the first part that match a single alternative of the first probabilistic value with all the alternatives of the second probabilistic value to the aforementioned correlations.

We developed two strategies for incorporating the normalized priority scores into the process of matching an alternative x of z_i with the probabilistic value z_j. These strategies can be applied individually, but can be also combined.

1. In the original variant from Section 7.5.4, the alternatives of z_j are sorted by their similarity to x. Thus, we already used a kind of priority score, i.e. the similarity $sim_{dom}(x, \bullet)$. A logical consequence seems to extend the similarity based priority scores by the priority scores that are computed from the joint probability distribution and hence by incorporating them into the sorting process. By doing so we first compute a single rank score per alternative of z_j, then sort these alternatives by their scores and finally compute the similarity from x to z_j as described in Section 7.5.4.

   To avoid a consideration of alternatives that are impossible in the combination with x, we have to ensure that these alternatives are sorted behind all alternatives that are possible in combination with x even if the latter alternatives are totally dissimilar to x, i.e. $sim_{dom}(x, \bullet) = 0$. Therefore, we compute the rank score of alternative x' from z_j to x as follows:

$$\mathit{rank}(x, x') = \begin{cases} \mathit{NPM}_{i,j}(x, x') \times sim_{dom}(x, x'), & \text{if } \mathit{NPM}_{i,j}(x, x') > 0, \\ -1, & \text{else.} \end{cases} \tag{7.17}$$

   This variant of the Probabilistic Monge-Elkan Similarity is in the following denoted as *PME-Sim*$^{\ominus}$.

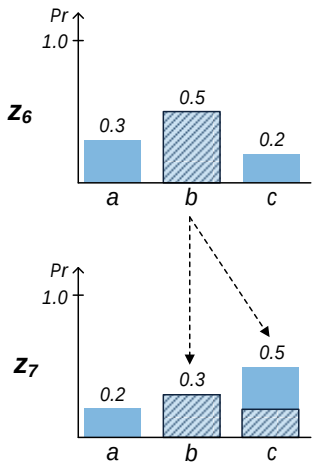
2. In the second strategy we use the similarity between the alternatives of z_j and x as the only sorting criterion, but restrict the amount of probability that is taken into account for similarity computation

Pos.	x'	$rank(b,x')$	$p_{x'}$	p_Σ	$sim(z_6,z_7,b)$
1	-	-	-	0	0
2	b	0.42 × 1.0	0.3	0.3	+ 0.3 × 1
3	c	1 × 0.4	0.5	0.8	+ (0.5-0.8+0.5) × 0.4
4	a	-1	0.2	-	-
					= 0.38/0.5 = 0.76

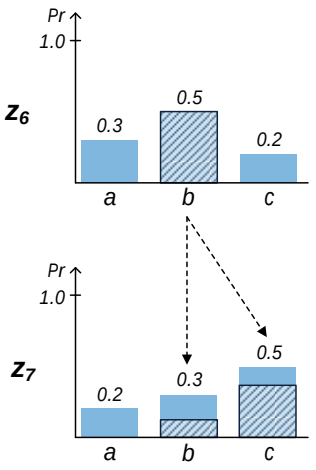
(a) Computing $sim(z_6, z_7, b)$ with *PME-Sim*$^{\ominus}$

Pos.	x'	$sim(b,x')$	$p_{x'}$	p_Σ	$sim(z_6,z_7,b)$
1	-	-	-	0	0
2	b	1.0	0.42 × 0.3	0.126	+ 0.126 × 1
3	a	0.6	0.0	0.126	+ 0 × 0.6
4	c	0.4	1.0 × 0.5	0.626	+(0.5-0.626+0.5) × 0.4
					= 0.2756/0.5 = 0.5512

(b) Computing $sim(z_6, z_7, b)$ with *PME-Sim*$^{\oslash}$



(c) Probability mass mapping with *PME-Sim*$^{\ominus}$



(d) Probability mass mapping with *PME-Sim*$^{\oslash}$

Figure 7.55.: Computing the similarities from alternative b of z_6 to z_7 with *PME-Sim*$^{\ominus}$ and *PME-Sim*$^{\oslash}$

from a selected alternative by the alternative's priority score. Here we use the priority score as a factor. Therefore, the probability is only not restricted if the alternative has the maximum possible priority score 1 and is only completely restricted if it has the lowest possible priority score 0. As a consequence, we define the probability $p_{x'}$ of a selected alternative x' in a different way as we do in the original variant:

$$p_{x'} = NPM_{i,j}(x,x') \times Pr_j(x') = \frac{\frac{Pr_{i,j}(x,x')}{Pr_j(I')}}{\mathbf{max}_{x^* \in y_j}\left(\frac{Pr_{i,j}(x,x^*)}{Pr_j(x^*)}\right)} \times Pr_j(x')$$
$$= \frac{Pr_{i,j}(x,x')}{\mathbf{max}_{x^* \in y_j}\left(\frac{Pr_{i,j}(x,x^*)}{Pr_j(x^*)}\right)} \quad (7.18)$$

This variant of the Probabilistic Monge-Elkan Similarity is in the following denoted as *PME-Sim*$^{\oslash}$.

The variant of the Probabilistic Monge-Elkan Similarity that combines both strategies, i.e. a variant that sorts the alternatives of z_j by scores that are computed by combining similarity and priority and that restricts the probability of each selected alternative by its priority, is in the following denoted as *PME-Sim*$^{\otimes}$.

Example 139 *The functionality of the two variants PME-Sim*$^{\ominus}$ *and PME-Sim*$^{\oslash}$ *as well as their differences are illustrated in Figure 7.55. In this example, we consider the computation of the similarity*

$sim(z_6, z_7, b)$ where we assume the joint probability distribution $Pr^C_{6,7}$ from Figure 7.54. Alternative $a \in y_7$ is impossible in combination with alternative $b \in y_6$ and hence has the normalized priority score NPM$_{6,7}(b, a) = 0$. Because alternative $b \in y_6$ is most probable in combination with alternative $c \in y_7$, the normalized priority score of c and b is NPM$_{6,7}(b, c) = 1$ and the normalized priority score of of b and b is NPM$_{6,7}(b, b) = 0.42$.

Figure 7.55(a) shows the computation process by using PME-Sim$^{\ominus}$. The corresponding mapping of probability mass is presented in Figure 7.55(c). In contrast, Figure 7.55(b) shows the process of computing the same similarity, but now by using PME-Sim$^{\oslash}$. The corresponding probability mass mapping is depicted in Figure 7.55(d).

As we can see, both variants produce somewhat different similarity scores for that sample situation, but whereas PME-Sim$^{\ominus}$ only hardly consider the differences in the priority scores (note, element b would be sorted in front of element c with or without considering dependencies), PME-Sim$^{\oslash}$ does.

This sample shows a general observation: Due to *PME-Sim$^{\ominus}$* considers the priority scores only for sorting and different scores can lead to the same sorting (and hence to the same computation result), the variants *PME-Sim$^{\oslash}$* and *PME-Sim$^{\otimes}$* are more sensitive to differences in the priority scores than *PME-Sim$^{\ominus}$*. Nevertheless, all three variants satisfy our initially proposed set of hard conditions.

Theorem 14 *The variants PME-Sim$^{\ominus}$, PME-Sim$^{\oslash}$, and PME-Sim$^{\otimes}$ all satisfy the first hard condition and hence each supply the same results as PME-Sim in cases of independence.*

Theorem 14 can be proved as follows:

Proof 10 *In the case of independence, each alternative x' of z_j has the same unnormalized priority score to x, because $Pr_{i,j}(x, x') = Pr_i(x) \times Pr_j(x')$ and dividing $Pr_{i,j}(x, x')$ by $Pr_j(x')$ results in $Pr_i(x)$. Thus, each alternative of x_2 has the normalized priority score 1. Consequently, neither the sorting order is affected by this score (rank$(x, x') = sim_{dom}(x, x')$) nor any probability is restricted by this score ($p_{x'} = Pr_j(x')$). Therefore, in cases of independence PME-Sim$^{\ominus}$, PME-Sim$^{\oslash}$, as well as PME-Sim$^{\otimes}$ produce the same results as PME-Sim.*

Theorem 15 *The variants PME-Sim$^{\ominus}$, PME-Sim$^{\oslash}$, and PME-Sim$^{\otimes}$ all satisfy the second hard condition that prescribes that an alternative of z_j that is not possible in combination with x cannot contribute to the similarity $sim(z_i, z_j, x)$.*

Theorem 15 can be proved as follows:

Proof 11 *In the case of PME-Sim$^{\oslash}$ an 'impossible' alternative $x' \in y_j$ can be selected in the iteration over the sorted list, but because its priority score is zero, its probability $p_{x'}$ is zero as well and it does not contribute to the computed similarity score.*

In the case of PME-Sim$^{\ominus}$ and PME-Sim$^{\otimes}$ such alternatives are sorted at the end of the list and could only be selected if the accumulative probability of all alternatives of z_j that are possible in combination with x (and hence are sorted in front of the 'impossible' alternatives) is lower than the probability $Pr_i(x)$. This, however, would violate the third hard condition that is proved by Theorem 16.

Theorem 16 *The variants PME-Sim$^{\ominus}$, PME-Sim$^{\oslash}$, and PME-Sim$^{\otimes}$ all satisify the third hard condition that prescribes that the accumulative probability $Pr_{\Sigma} = \sum_{I' \in alt(v_j), Pr_{i,j}(I,I')>0} Pr_{I'}$ is always greater than or equal to the probability $Pr_i(I)$.*

Theorem 16 can be proved as follows:

Proof 12 *In the case of PME-Sim$^{\ominus}$ that prove is straightforward because the original probabilities are used:*

$$\begin{aligned}
Pr_i(x) &= \sum\nolimits_{x' \in y_j} Pr_{i,j}(x, x') \\
\textit{(because } Pr_{i,j}(x,x') = 0 \Rightarrow p_{x'} = 0\textit{)} \quad &= \sum\nolimits_{x' \in y_j, Pr_{i,j}(x,x')>0} Pr_{i,j}(I, I') \\
&\leq \sum\nolimits_{x' \in y_j, Pr_{i,j}(x,x')>0} \sum\nolimits_{I^* \in y_i} Pr_{i,j}(x^*, x') \\
&= \sum\nolimits_{x' \in y_j, Pr_{i,j}(x,x')>0} Pr_j(x') \\
&= \sum\nolimits_{x' \in y_j, Pr_{i,j}(x,x')>0} p_{x'}
\end{aligned}$$

In the case of PME-Sim$^{\oslash}$ and PME-Sim$^{\otimes}$, the probability $p_{x'}$ corresponds to the product of the probability $Pr_j(x')$ and the normalized priority score $NPM_{i,j}(x, x')$. Thus, the satisfaction of the third condition can be proven as:

$$\begin{aligned}
Pr_i(x) &= \sum\nolimits_{x' \in y_j} Pr_{i,j}(x, x') \\
\textit{(because } \mathbf{max}(\bullet) \leq 1\textit{)} \quad &\leq \sum\nolimits_{x' \in y_j} \frac{Pr_{i,j}(x, x')}{\mathbf{max}_{x^* \in y_j} \left(\frac{Pr_{i,j}(x,x^*)}{Pr_j(x^*)} \right)} \\
&= \sum\nolimits_{x' \in alt(v_j)} p_{x'} \\
\textit{(because } Pr_{i,j}(x,x') = 0 \Rightarrow p_{x'} = 0\textit{)} \quad &= \sum\nolimits_{x' \in y_j, Pr_{i,j}(x,x')>0} p_{x'}
\end{aligned}$$

Since we only change the sorting criterion and the computation of the probability $p_{x'}$, the complexity of the extended variants of the Probabilistic Monge-Elkan Similarity is the same as the original variant and hence is in class $\mathcal{O}(m \times n \times \log n)$. The only overhead is that the normalized priority scores need to be computed once.

Intuitively, correlations between alternatives can be incorporated into the Earth Mover's Distance by assigning each supplier with an individual priority per consumer. Then for each consumer the supplier are not only selected by their distance and the considered mass of earth, but also by their priority. In other words, the priority of a supplier for a consumer is incorporated into the cost computation.

At a first glance, it is not clear if this modification completely change the underlying optimization problem and if it can be still reduced to the transportation problem anymore. Moreover, even this reduction is still valid, it is not clear in which way an efficient computation method of the Earth Mover's Distance need to be adapted to this new cost factor. For that reason, such an adaptation is out of the scope of this thesis and we consider it for future research. Consequently, we restrict the usage of the Earth Mover's Distance to matching entities that do not have any correlations between their description alternatives.

$$Pr^{B}_{6,7} = \begin{bmatrix} 0 & 0.3 & 0 \\ 0 & 0 & 0.5 \\ 0.2 & 0 & 0 \end{bmatrix} \quad IM_{6,7} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0.4 \\ 0.8 & 0 & 0 \end{bmatrix} \quad WM = \begin{bmatrix} 0 & 0.3 & 0 \\ 0 & 0 & 0.2 \\ 0.16 & 0 & 0 \end{bmatrix} \quad Pr^{B\times I}_{6,7} = \begin{bmatrix} 0 & 0.46 & 0 \\ 0 & 0 & 0.3 \\ 0.24 & 0 & 0 \end{bmatrix}$$

Figure 7.56.: Sample incorporation of impact values into a joint probability distribution

7.5.4.2. Incorporation of Impact Values

To restrict the strong influence of non-membership information on the matching result, we compute the world impact values. In Section 7.2, we incorporate impact values into uncertainty resolution by combining them with the probability distribution. In the Probabilistic Monge-Elkan Similarity, we can use the same principle and therefore can incorporate the impact values into the joint probability distribution, i.e. for each alternative description pair a new probability mass is computed based on the original mass and its impact value[10]. Due to the impact values do not sum up to one a final normalization is required. Since different pairs can have different impact values, this incorporation can introduce value dependencies.

Example 140 *For illustration, we consider the two probabilistic values z_6 and z_7 from Figure 7.49 and assume the joint probability distribution $Pr^{B}_{6,7}$ from Figure 7.54(b). Moreover, we assume the impact values that are presented by the impact matrix $IM_{6,7}$ in Figure 7.56. We incorporate them into the probability distribution by multiplying the probabilities with the impact values. The resultant values are shown by the weight matrix WM. Since the values do not sum up to 1 anymore, we need to divide them by 0.66 to regain the valid probability distribution $Pr^{B\times I}_{6,7}$. If we compare distribution $Pr^{B}_{6,7}$ with distribution $Pr^{B\times I}_{6,7}$ we see that the incorporation of the impact values redistributes the probability mass of the 1:1 mapping, i.e. the alternative pair (b, c) loses a lot of probability mass and the alternative pair (a, b) gains a lot of probability mass. Thus, by using $Pr^{B\times I}_{6,7}$ as input to any of the adapted variants, the Monge-Elkan Similarity from z_6 to z_7 results in:*

$$\begin{aligned} \textit{PME-Sim}(z_6, z_7) &= 0.46 \times sim_{dom}(a, b) + 0.3 \times sim_{dom}(b, c) + 0.24 \times sim_{dom}(c, a) \\ &= 0.46 \times 0.6 + 0.3 \times 0.4 + 0.24 \times 0.1 = 0.42 \end{aligned}$$

An incorporation of impact values into the *EMD* is difficult, because it requires the handling of correlation between the matched probabilistic values.

7.5.4.3. Incorporation of Quality Requirements

Since we adapt the Probabilistic Monge-Elkan Similarity to correlations between the two compared probabilistic values, quality requirements can be incorporated into this measure in the same way as we do it for any other method for aggregating similarity scores. That is by utilizing the concepts of the minimal-θ aggregation and the maximal-θ aggregation. Thus, in the first step, we compute all similarities

[10] Actually, the resultant distribution models weights instead of probabilities as we know it from the aggregation methods presented in Section 7.2. However, since we use the notion of probabilities in the definitions of the different variants of the Probabilistic Monge-Elkan Similarity and we want to stay conform with these definitions, we consider these weights as probabilities.

alt.	ACVector	AIVector	Pr
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.21
fs2	$\vec{c}_A$=[0.0,0.2]	$\vec{i}_A$=[0.7,0.8]	0.24
fs3	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]	0.15
fs4	$\vec{c}_A$=[0.33,0.2]	$\vec{i}_A$=[0.5,0.8]	0.14
fs5	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.16
fs6	$\vec{c}_A$=[0.5,0.5]	$\vec{i}_A$=[0.4,0.9]	0.10

(a) Probabilistic feature score 1

alt. pair	prob.	feature score
(d_{11},d_{21})	0.21	fs1
(d_{12},d_{21})	0.24	fs2
(d_{13},d_{21})	0.15	fs3
(d_{11},d_{22})	0.14	fs4
(d_{12},d_{22})	0.16	fs5
(d_{13},d_{22})	0.10	fs6

(b) Provenance table 1

alt.	ACVector	AIVector	Pr
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.21
fs2	$\vec{c}_A$=[0.0,0.2]	$\vec{i}_A$=[0.7,0.8]	0.38
fs3	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]	0.15
fs4	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.16
fs5	$\vec{c}_A$=[0.5,0.5]	$\vec{i}_A$=[0.4,0.9]	0.10

(c) Probabilistic feature score 2

alt. pair	prob.	feature score
(d_{11},d_{21})	0.21	fs1
(d_{12},d_{21})	0.24	fs2
(d_{13},d_{21})	0.15	fs3
(d_{11},d_{22})	0.14	fs2
(d_{12},d_{22})	0.16	fs4
(d_{13},d_{22})	0.10	fs5

(d) Provenance table 2

Figure 7.57.: Two samples for mapping alternative description pairs to matching result alternatives

between the individual description alternatives. In the second step we either select the most similar pairs (maximal-θ aggregation) or the most dissimilar pairs (minimal-θ aggregation). Finally, we transform the alternative pair space based on the selected pairs (all non-selected pairs become impossible) and compute the Probabilistic Monge-Elkan Similarity. Such a modification of the alternative description pair space including a corresponding modification of the provenance table is demonstrated in Section 7.6 in the context of matching cost optimization.

7.5.4.4. Using the *PME-Sim* in the Description-based Detection Approach

In the description-based approach we compute a probabilistic matching result in each of the matching phases. Nevertheless, for computing the Probabilistic Monge-Elkan Similarity (or the Earth Mover's Distance) we need to know which possible matching result has been produced by processing which pair of description alternatives. For that purpose, we additional manage a provenance table per matching phase that maps each alternative description pair to one alternative of the the phase's probabilistic matching result.

For runtime optimizing purposes (see Section 7.6.2), it can be required to change the probabilities of the individual alternative description pairs subsequently. By doing so the joint probability distribution on the alternative description pairs that has been initially computed in the *APS* construction phase is not valid anymore, but need to be recomputed from the provenance table instead. For that reason and because several alternative description pairs can refer to the same matching result alternative, we additionally store the probability for each alternative description pair in the provenance table.

Example 141 *For illustration we consider the two probabilistic entity descriptions $\mathfrak{d}_1$ and $\mathfrak{d}_2$ from Figure 7.44(a) and Figure 7.44(b), and assume their naive alternative description pair space $APS(\mathfrak{d}_1, \mathfrak{d}_2)$ that is depicted in Figure 7.44(c). Recall, in the feature matching phase, for each of the alternative description pairs a feature score is computed. As presented in Section 7.4.6, these scores serves as alternatives of a probabilistic feature score where the probability of a possible feature score corresponds to the accumulative probability of all alternative description pairs from which this score has been computed by the matching phase.*

An exemplary probabilistic feature score that could result from processing the sample pair space $APS(\mathfrak{d}_1, \mathfrak{d}_2)$ is shown in Figure 7.45(b) and is depicted once again in Figure 7.57(a). Figure 7.57(b) shows the corresponding provenance table that maps each of the six alternative description pairs to a specific probability mass of one of the possible feature scores.

A second exemplary probabilistic feature score is presented in Figure 7.45(c) and is depicted once again in Figure 7.57(c). The corresponding provenance table is shown in Figure 7.57(d). In contrast to the first example, in this case two alternative description pairs are mapped to the same feature score alternative and a separate storage of the probability mass in the provenance table becomes necessary.

Of course, the probabilistic matching results actually do not need to be stored separately, but can be derived from their corresponding provenance table instead. In that case, we need to store the alternatives' values in the provenance table instead of their identifier. Nevertheless, to save computation time it is commonly useful to maintain the probabilistic matching result separately and to take care that the redundantly stored data does not become inconsistent, i.e. the probability of an alternative of a matching result must always be equal to the accumulative probability of all provenance table rows that refer to that alternative.

7.5.4.5. Using the *PME-Sim* in the World-based Detection Approach

An incorporation of the Probabilistic Monge-Elkan Similarity into the world-based detection approach is complicated. First, we need to know the entity descriptions by which two database entities have been matched in each world or we have to know by which possible instances these database entities where represented in these worlds. In the description-based approach we match two database entities by matching all their possible combinations of description alternatives. In contrast, in the world-based approach, we process a sampled set of worlds and it can happen that the most database entities are represented by another possible instance in each of the sampled worlds and hence each entity has a different entity description in each world. In that case, each two probabilistic entity descriptions are highly correlated and we have an 1:1 mapping between their alternatives. Thus, the Probabilistic Monge-Elkan Similarity degenerates to a simple expected value aggregation.

7.5.5. Compensation of Other Types of Description Discrepancies

The Earth Mover's Distance as well as the Probabilistic Monge-Elkan Similarity are able to compensate discrepancies that result from different numbers of alternatives, different probability assignments, or dissimilarities in attribute values (or entity sets), but cannot compensate a confusion of attribute values (or related entities) between different alternatives of the same uncertain entity description (see Section 7.5.2)

Pos.	x'	$sim(a,x')$	$p_{x'}$	p_Σ	$sim(z_3,z_6,a)$
1	-	-	-	0	0
2	*Florian*	1.0	0.3	0.3	+ 0.3 × 1.0
3	*Fabian*	0.7	0.7	-	-

= 0.3/0.3 = 1.0

(a) Computation of $sim(\mathfrak{d}_{PED}(e_1), \mathfrak{d}_{PED}(e_2), d_2, \text{'fname'})$

Pos.	x'	$sim(a,x')$	$p_{x'}$	p_Σ	$sim(z_3,z_6,a)$
1	-	-	-	0	0
2	*Hanke*	1.0	0.7	0.7	+ 0.3 × 1.0
3	*Schulz*	0.0	0.3	-	-

= 0.3/0.3 = 1.0

(b) Computation of $sim(\mathfrak{d}_{PED}(e_1), \mathfrak{d}_{PED}(e_2), d_2, \text{'lname'})$

Pos.	x'	$sim(a,x')$	$p_{x'}$	p_Σ	$sim(z_3,z_6,a)$
1	-	-	-	0	0
2	*Frankfurt*	1.0	0.3	0.3	+ 0.3 × 1.0
3	*Kassel*	0.1	0.2	-	-
4	*Homberg*	0.0	0.5	-	-

= 0.3/0.3 = 1.0

(c) Computation of $sim(\mathfrak{d}_{PED}(e_1), \mathfrak{d}_{PED}(e_2), d_2, \text{'residence'})$

Figure 7.58.: Computation of the similarity score $sim(\mathfrak{d}_{PED}(e_1), \mathfrak{d}_{PED}(e_2), d_2)$ for the different attributes

and cannot compensate discrepancies that result from errors within single probability scores. For that reason, we shortly discuss an appropriate handling of both types in this section and sketch a simple extension of the Probabilistic Monge-Elkan Similarity for an additional compensation of feature value confusions. Nonetheless, the made discussions only introduce some intuitive ideas and should not be considered as a detailed study. In contrast, a deeper consideration of both types of description discrepancies remains to future research.

7.5.5.1. Compensating Feature Value Confusions

The Earth Mover's Distance as well as the Probabilistic Monge-Elkan Similarity consider the similarity (or distance) between two domain elements always independently from the similarities (or distances) to the remaining elements. However, due to the features are confused between some alternatives of the same uncertain description, this confusion cannot be considered in the comparison of a single alternative description pair, but instead requires a collective consideration of all alternatives of the description that contains the confusion.

Nevertheless, an intuitive idea to incorporate such a consideration into the Probabilistic Monge-Elkan Similarity is to modify the definition of the similarity score $sim(\mathfrak{d}_1, \mathfrak{d}_2, d)$ that computes the similarity from the alternative d of the probabilistic description $\mathfrak{d}_1$ to the probabilistic description $\mathfrak{d}_2$. For that purpose, we change the considered granularity of this similarity measure from instance level to feature level and hence compute it for every feature separately. Finally, we combine the similarity scores of the individual features to a single one. Thus, let $\mathfrak{d}_1$ and $\mathfrak{d}_2$ be two probabilistic entity descriptions, let d be an alternative of $\mathfrak{d}_1$, let *Feature* be the set of all considered features, and let w be a weight function that is defined on these features. We redefine the similarity score from d to $\mathfrak{d}_2$ as:

$$sim(\mathfrak{d}_1, \mathfrak{d}_2, d) = \sum_{F \in Feature} sim(\mathfrak{d}_1, \mathfrak{d}_2, d, F) \tag{7.19}$$

where $sim(\mathfrak{d}_1, \mathfrak{d}_2, d, F)$ is $sim(\pi_F(\mathfrak{d}_1), \pi_F(\mathfrak{d}_2), \pi_F(d))$ and $\pi_F(x)$ corresponds to the value of x in feature F. We call the variant of the Probabilistic Monge-Elkan Similarity that utilizes the internal similarity measure at feature granularity as *PME-Sim*$_F$.

Example 142 *For illustration, we consider the probabilistic entity descriptions of the two database entities* e_1 *and* e_2 *from Figure 7.48 and consider the computation of the similarity from the second alternative (denoted as* d *in this example) of* $\mathfrak{d}_{PED}(e_1)$ *to the probabilistic description* $\mathfrak{d}_{PED}(e_2)$*. Both descriptions contain three attributes. As a consequence, we split the computation of this similarity into the three computations that are presented in Figure 7.58(a) (attribute* 'fname'*), Figure 7.58(b) (attribute* 'lname'*), and Figure 7.58(c) (attribute* 'residence'*). Since all the attribute values of* d *belong to a set of alternatives of* $\mathfrak{d}_{PED}(e_2)$ *that has an accumulative probability of* 0.3 *or more, the similarity is* 1.0 *in all three cases. Consequently, the similarity* $sim(\mathfrak{d}_{PED}(e_1), \mathfrak{d}_{PED}(e_2), d)$ *is* 1.0 *as well and we successfully compensate the discrepancy that was caused by the confusion from the values 'Hanke' and 'Schulz' in the second alternative and the third alternative of* e_2 *(and hence* $\mathfrak{d}_{PED}(e_2)$*).*

Of course, the similarities $sim(\pi_F(\mathfrak{d}_1), \pi_F(\mathfrak{d}_2), \pi_F(d))$ of the individual features $F \in$ *Feature* corresponds to the comparison vectors of a feature score. Moreover, the definition made in Equation 7.19 corresponds to a distance-based decision model that uses the weighted average to compute an entity similarity score from the feature score. Thus, we can replace Equation 7.19 by any other method for computing entity similarity as for example a Decision Tree or a set of rules.

The problem with such discrepancies is that a confusion must not be the consequence of an error, but can be also an indication for a non-duplicate, i.e. we actually do not have a confusion, but the values are assigned to different alternatives by purpose. Moreover, each of the compared probabilistic entity descriptions can contain more than one confusions and the more features are confused, the less likely is a confusion by mistake. However, the adaptation of the Probabilistic Monge-Elkan Similarity presented above completely ignore correlations between features and therefore can produce the same similarity score regardless one or more features have been confused.

Due to both reasons it makes sense to penalize the resultant score of the similarity score that is computed by utilizing *PME-Sim*$_F$ with a factor $x \in [0, 1]$. In general, the less confusions exists, the less description alternatives are involved in such confusions and the more alternative description pairs are assumed to have an high similarity in the original variant of the Probabilistic Monge-Elkan Similarity. thus, one option to calculate an appropriate penalization score is to compute the original variant of the Probabilistic Monge-Elkan Similarity as well and then to us its difference to the feature based variant in order to determine the penalization. For instance, if no value is confused, the two variants *PME-Sim* and *PME-Sim*$_F$ produce the same similarity score (of course, only if in both variants the same method for entity similarity computation is used) and we set the penalization to zero. In general, as we think the greater the difference between both scores, the greater should be the used penalization. Indeed, such assumptions should be evaluated in future research.

7.5.5.2. Compensating Errors in Probabilities

A compensation of probability errors is especially difficult if the probabilities have been normalized in any of the previous detection phases as for example the preparation phase (see Section 7.7.2), because af-

fname	lname	residence
Florian	Hanke	Frankfurt
Florian	Schulz	Homberg

(a) Incomplete entity description d_1

fname	lname	residence
F*	H*	Frankfurt

(b) Incomplete entity description d_2

fname	lname	residence
Florian	Hanke	Frankfurt

(c) Incomplete entity description d_3

Figure 7.59.: Three incomplete entity descriptions with different degrees of uncertainty

ter normalization an erroneous insertion of an additional digit or a confusion of digits cannot be detected anymore.

An intuitive idea for the non-normalized case, i.e. original probabilities are given, is to separately process the used similarity measure (Earth Mover's Distance or Probabilistic Monge-Elkan Similarity) once for the alternatives of both descriptions and once for the probabilities of both descriptions, and finally to combine both scores to a single one.

7.5.6. Divergence between Description Similarity and Real-World Equivalence

In duplicate detection, the similarity between two probabilistic entity descriptions serves as an indication for the real-world equivalence of their corresponding entities. If we use a similarity measure for probability mass functions, however, we introduce a new aspect in this consideration, because the similarity between two probability mass functions theoretically becomes a less indication for real-world equivalence the more uncertain the two compared probability mass functions are. For instance, two probabilistic values that each have a uniform probability distribution on the underlying domain are equivalent, but each of these values do not contain much information. As a consequence, both values only agree in the fact that all domain elements are equally probable. Thus, in theory two non-equal probabilistic entity descriptions can be an higher indication for a duplicate that two equivalent descriptions.

Example 143 *This should be illustrated by a simple example. For that purpose, we consider the incomplete description d_1 presented in Figure 7.59(a) and consider the incomplete description d_2 presented in Figure 7.59(b) where we assume that the second description is modeled in a compact way and actually contains one alternative per possible combination of first names and last names where the first name starts with the letter 'F' and the last name starts with the letter 'H'.*

We start with description d_1 and assume that we compare that description with itself, i.e. we compare two database entities that both have this descriptions. In this case, the equivalence is an high indication for a duplicate, because it is unlikely that we have collect the same information on two different persons. Now we consider description d_2 and assume again that we compare that description with itself. In this case, the equivalence is only a low indication for a duplicate, because it is not uncommon that two persons living in Frankfurt have a first name that starts with 'F' and have a last name that starts with 'H'. Thus, although both comparisons return the same similarity score, the first is much more significant than the second.

Moreover, let us assume that we compare the first description with the description d_3 from Figure 7.59(c). In this case, the similarity is lower than one. However, it is still an higher indication for a duplicate than the perfect similarity score that result from comparing description d_2 with itself.

One solution to this problem is to penalize the resultant similarity based on the uncertainty of the compared descriptions or to set the similarity score's impact value based on this uncertainty.

Nevertheless, especially in the context of entity descriptions such a penalization is less important because the domain of all possible entity descriptions is typically extremely large and each probabilistic entity descriptions contains only a small subset of all the domain's elements as alternatives. Moreover, if the number of alternatives per description is not tremendous in size, such an uncertainty can even increase the indication for real-world equivalence in many applications scenarios. For instance, it is very unlikely that two persons changed their names and residences in the same way.

Example 144 *For illustration we reconsider the entity description d_3 that presented in Figure 7.59(c). This entity description is certain. Thus, if we assume a comparison of two entities that both have this description, we can conclude an high indication that both entities are duplicates. Nevertheless, as we think, under some circumstances this equivalence can be a lower indication for a duplicate than the equivalence between the incomplete entity description d_1 to itself. For instance, let us reconsider the scenario presented in Example 133 and assume that the two description alternatives of d_1 result from correct data that has been collected at different times and we simply do not know which of the alternatives is obsolete and which not. In that case, the likelihood that two different persons have the first name 'Florian', have the last name 'Hanke', and both live in Frankfurt is lower than the likelihood that two different persons having the first name 'Florian' each lives once in Frankfurt while having the last name 'Hanke' and each lives once in Hamburg while having the last name 'Schulz'.*

As a consequence, increasing the number of alternatives can also increase the significance of the computed similarity score. Thus, whether the uncertainty of the two compared values should influence the indication for real-world equivalence in a positive way or should influence it in a negative way depends on the considered use case. For that reason, it is up to the user to define in which way the number of alternatives influences the computed matching result. However, as presented above such an influence can be simply incorporated into our detection approach by penalizing the final similarity score or by changing its impact value.

7.6. Matching Cost Optimization

Matching two entity descriptions (and the feature matching phase in particular) is a cost, time as well as space, consuming task. Therefore, an optimization of matching costs in as important aspect for duplicate detection in large databases and is especially important for duplicate detection in uncertain databases that contain entities having hundreds of possible instances. In general, matching cost optimization in the world-based detection approach corresponds to reduce matching costs in certain databases. For that reason, we focus on matching cost optimization in the description-based detection approach in this section.

The costs of matching two probabilistic entity descriptions can be decreased in several ways. First by reducing the number of attribute value similarities and by reducing the number of related entity set similarities that need to be computed for matching two description alternatives. Second by reducing the number of alternative description pairs that need to be matched for matching two probabilistic descriptions.

We discuss several techniques for matching cost optimization in this section. In general, optimization techniques can be distinguished into hard optimization techniques, i.e. techniques that compute the matching result efficiently without changing it, and soft optimization techniques, i.e. techniques that improve efficiency, but can slightly change the matching result. We start with hard optimization techniques in Section 7.6.1 and continue with soft optimization techniques in Section 7.6.2.

Of course, optimization techniques that are defined for matching certain entity descriptions (e.g. omitting a match of relationship information, if the attribute description similarity is too low) can be used for matching two probabilistic description alternatives as well.

The optimization techniques proposed by Lian et al. [LC10] cannot be used for our purpose because they are based on the Jaccard Similarity and we cannot utilize this measure for matching entity descriptions.

7.6.1. Hard Optimization Techniques

As mentioned above, hard optimization techniques are characterized by improving efficiency without changing the matching result.

7.6.1.1. Hard Attribute Match Pruning

Computing attribute value similarity is one of the most costly task in duplicate detection. Moreover, different alternatives of the same probabilistic attribute description often have one or more attribute values in common or some of their attribute values share same tokens. To avoid multiple computations of same similarity scores we utilize a look-up table (for illustration see Figure 7.60) that stores previously computed similarities between attribute values or tokens. For each new similarity computation the look-up table is checked first. If the required similarity is not already stored in the table, it is computed and inserted. Of course, since the process can match same values with different similarity measures, the key of the look-up table consists of two values and one measure. If the measure is symmetric (marked by the extra attribute *'isSymmetric'*, the similarity is always searched in both directions.

Value v_1	**Value v_2**	**Similarity Measure**	**isSymmetric**	$sim(v_1, v_2)$
'John'	'Jon'	*LevSim*	1	0.75
'Smith'	'Do'	*LevSim*	1	0.0
'John Smith'	'Jon Do'	*MongeElkanSim*	0	0.2
'Jon Do'	'John Smith'	*MongeElkanSim*	0	0.5
...	...	...	...	...

Figure 7.60.: Sample look-up table

Example 145 *For example, by matching all alternatives of the two probabilistic entity descriptions $\mathfrak{d}_1 = \mathfrak{d}_{PED}(e_1)$ and $\mathfrak{d}_2 = \mathfrak{d}_{PED}(e_2)$ from Figure 7.44 without attribute match pruning, the similarity between the two tokens 'John' and 'Jon' is computed for three times. Altogether, we perform* 30 *pairwise matchings of tokens and hence need to compute* 30 *similarity scores. In contrast, by using the look-up table we can reduce that number from* 30 *to* 22 *which is a saving of* 26.7%.

Of course, attribute match pruning can be also utilized for matching entity descriptions in certain data, but is particularly useful in matching probabilistic entity descriptions because different alternatives of the same description usually have many attribute values or at least tokens in common.

7.6.1.2. Hard Alternative Match Pruning

In Section 7.2 and Section 7.5 we present different methods for resolving uncertainty in matching results. The output of such an uncertainty resolution can often be computed without matching all elements of the alternative pair space. Thus, matching costs can be reduced by pruning as many of the unnecessary alternative description pair matchings as possible. In meaningful resolution methods, the most probable alternative pairs have the strongest impact on the resolution result. As a consequence, it make sense to match the most probable alternative pairs first. For that purpose, we sort the alternative pair space by the pairs' probabilities and then start to match the most probable pairs one after another. Since the hard alternative match pruning is realized by processing all phases of description matching pair by pair we also denote it as *In-Depth Pruning*.

To stop the resolution method (and hence the matching process) as early as possible, we compute the resolution output of the *Worst Match Scenario* (short *WMS*) and the *Best Match Scenario* (short *BMS*) for every newly matched pair of description alternatives. If from both match scenarios the same consequences result, we do not need to match the remaining alternative pairs and the matching process can be stopped.

In the Worst Match Scenario, all alternative pairs that have not been matched so far are assumed to produce the most negative matching result. In contrast, in the Best Match Scenario, all alternative pairs that have not been matched so far are assumed to produce the most positive matching result. The definition of the most negative matching result and the most positive matching result depends on the considered matching phase. For instance, the most negative duplicate decision is the triple $(\text{UNMATCH}, 0, 1)$ because the corresponding entities have no similarity and are classified as an UNMATCH with the maximal possible impact. In contrast, the most positive duplicate decision is the triple $(\text{MATCH}, 1, 1)$ because the corresponding entities have the maximal possible similarity and are classified as a MATCH with the maximal possible impact. The most negative feature score, the most positive feature score, the most negative similarity score, and the most positive similarity score are defined accordingly.

The definition of same consequences depends on the considered process configuration. For example, let us assume a detection process that resolves uncertainty after the classification phase, and then uses the method of *partitioning based on connected components* (see Section 4.3.7) to compute the final duplicate clustering. Since the similarity between both descriptions is not required in the used clustering method, the process of matching two probabilistic descriptions can be stopped if the matching class computed in the Worst Match Scenario and the matching class computed in the Best Match Scenario are equal.

In contrast, if the detection process utilizes a clustering method that gets the duplicate pair graph as input, it needs to compute the exact similarity scores of all MATCHES. Therefore, the matching of two probabilistic entity descriptions can be immediately stopped if both match scenarios produces an UNMATCH because we do not require their exact similarity, but cannot be stopped if both match scenarios produces a MATCH. Nevertheless, the number of UNMATCHES typically dominates the candidate pair

$$WMS_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad BMS_0 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

(a) Initialized similarity matrices

$$WMS_1 = \begin{bmatrix} 0 & 0 \\ 0.2 & 0 \\ 0 & 0 \end{bmatrix} \quad BMS_1 = \begin{bmatrix} 1 & 1 \\ 0.2 & 1 \\ 1 & 1 \end{bmatrix}$$

(b) Similarity matrices after iteration 1

$$WMS_2 = \begin{bmatrix} 0.8 & 0 \\ 0.2 & 0 \\ 0 & 0 \end{bmatrix} \quad BMS_2 = \begin{bmatrix} 0.8 & 1 \\ 0.2 & 1 \\ 1 & 1 \end{bmatrix}$$

(c) Similarity matrices after iteration 2

$$WMS_3 = \begin{bmatrix} 0.8 & 0 \\ 0.2 & 0.5 \\ 0 & 0 \end{bmatrix} \quad BMS_3 = \begin{bmatrix} 0.8 & 1 \\ 0.2 & 0.5 \\ 1 & 1 \end{bmatrix}$$

(d) Similarity matrices after iteration 3

Iteration	Similarity Score	Matching Class
1.	$0.24 \times 0.2 + 0.76 \times 0 = 0.048$	UNMATCH
2.	$0.24 \times 0.2 + 0.21 \times 0.8 + 0.55 \times 0 = 0.216$	UNMATCH
3.	$0.24 \times 0.2 + 0.21 \times 0.8 + 0.16 \times 0.5 + 0.39 \times 0 = 0.296$	UNMATCH

(e) Similarity scores and matching classes for *Worst Match Scenario*

Iteration	Similarity Score	Matching Class
1.	$0.24 \times 0.2 + 0.76 \times 1 = 0.808$	MATCH
2.	$0.24 \times 0.2 + 0.21 \times 0.8 + 0.55 \times 1 = 0.766$	MATCH
3.	$0.24 \times 0.2 + 0.21 \times 0.8 + 0.16 \times 0.5 + 0.39 \times 1 = 0.686$	UNMATCH

(f) Similarity scores and matching classes for *Best Match Scenario*

Figure 7.61.: The similarity matrices, the similarity scores and the matching classes of the different iterations for the Worst Case Scenario and the Best Case Scenario in the sample matching process

space by far. Thus, this pruning technique improves matching efficiency enormously even if it makes an impact only in the case of UNMATCHES.

In general, this pruning technique is especially useful in the case of UNMATCHES because matching one or two pairs of description alternatives is often sufficient to satisfy the stop criterion. In contrast, in the case of MATCHES the process will usually only stop after matching the majority of pairs because in the most resolution methods a description pair is only classified as a MATCH, if most of their alternative pairs are similar or at least for each alternative of one description there is a high similar alternative of the other description.

Example 146 *To illustrate the principle of hard match pruning, we match the two probabilistic entity descriptions* $\mathfrak{d}_1$ *and* $\mathfrak{d}_2$ *from Figure 7.44 and assume that these descriptions are independent, i.e. we construct the naive alternative pair space as presented in Figure 7.44(c). We consider a simple resolution method that computes the expected similarity of all alternative pairs after the similarity computation phase and then applies a threshold* $\theta_{U/M} = 0.7$ *to classify the considered entity pair as a* MATCH *or an* UNMATCH. *Moreover, we assume that the exact similarity score is not required in the duplicate*

clustering phase. Therefore, the matching process can be stopped, if both match scenarios supply the same matching class.

We manage two similarity matrices (or distance matrices respectively), each one for the Best Match Scenario and the Best Case Scenario. Because none of the alternative pairs has been matched so far, all cells of the matrix for the WMS are initially set to zero and all cells of the matrix for the BMS are initially set to one (see Figure 7.61(a)). By matching the alternative pairs, we fill the cells of the two matrices step by step (see Figure 7.61(b) to Figure 7.61(d)). In the first step, we compute the similarity between d_{12} *and* d_{21} *(probability* $0.4 \times 0.6 = 0.24$*), store the resultant score in both matrices and compute the expected similarity score for each of the both matrices. By assuming a similarity of* 0.2 *the expected scores results in* 0.048 *(WMS) and* 0.808 *(BMS). Since in the WMS both entities are classified as an* UNMATCH *and in the BMS they are classified as a* MATCH*, we need to continue by matching the next probable alternative pair which is* $\{d_{11}, d_{21}\}$ *(probability* $0.35 \times 0.6 = 0.21$*). The computed similarity (here we assume* 0.8*) is stored in both matrices and the expected scores are computed again, i.e.* 0.216 *(WMS) and* 0.766 *(BMS). In both match scenarios, the entity pair is classified into the same matching class as in the iteration before. Thus a further alternative pair, i.e.* $\{d_{12}, d_{22}\}$ *(probability* $0.4 \times 0.4 = 0.16$*), need to be matched. By assuming a similarity of* 0.5*, the two expected scores result in:* 0.296 *(WMS) and* 0.686 *(BMS). Now, the entities are classified as an* UNMATCH *in both match scenarios. Thus, whatever similarity scores the remaining alternative pairs have, the matching class of* e_1 *and* e_2 *will always be an* UNMATCH *and we can stop the matching process. By doing so, we save the matching of three alternative pairs which corresponds to a cost reduction of* 50%*. The matching results (similarity score as well as matching class) as well as the computations for each of the three iterations are presented in Figure 7.61(e) (WMS) and Figure 7.61(f) (BMS).*

It can be intuitively seen for this resolution method that we do not need to compute the whole expectation value in each iteration, but can compute the expectation value of one iteration based on the expectation value of its previous iteration. Let $\mathfrak{d}_r$ and $\mathfrak{d}_s$ be two probabilistic descriptions where Pr_r and Pr_s are their respective probability distributions and $Pr_{r,s}$ is their joint probability distribution. Moreover, let $exp^i_X(\mathfrak{d}_r, \mathfrak{d}_s)$ be the expectation value of the similarity between $\mathfrak{d}_r$ and $\mathfrak{d}_s$ that is computed for match scenario $X \in \{WMS, BMS\}$ in the ith iteration, and let $\{d_r, d_s\}$ be the alternative pair that has been matched in iteration i, the value $exp^i_X(\mathfrak{d}_r, \mathfrak{d}_s)$ can be computed from the value $exp^{i-1}_X(\mathfrak{d}_r, \mathfrak{d}_s)$ as follows:

$$\begin{aligned} exp^i_{WMS}(\mathfrak{d}_r, \mathfrak{d}_s) &= exp^{i-1}_{WMS}(\mathfrak{d}_r, \mathfrak{d}_s) + Pr_{r,s}(d_r, d_s) \times sim(d_r, d_s) \\ exp^i_{BMS}(\mathfrak{d}_r, \mathfrak{d}_s) &= exp^{i-1}_{BMS}(\mathfrak{d}_r, \mathfrak{d}_s) - Pr_{r,s}(d_r, d_s) + Pr_{r,s}(d_r, d_s) \times sim(d_r, d_s) \end{aligned}$$

Such an iterative computation approach can save a lot of computation time, especially if a more complex resolution method is used. To the best of our knowledge, the Earth Mover's Distance cannot be computed in an iterative fashion, because the optimization problem cannot be divided into subproblems. Nevertheless, for the Probabilistic Monge-Elkan Similarity such an approach of divide and conquer fits very well because finding the maximal value of *PME-Sim*$(\mathfrak{d}_r, \mathfrak{d}_s)$ corresponds to finding the maximal value of $sim(\mathfrak{d}_r, \mathfrak{d}_s, d)$ for all $d \in alt(\mathfrak{d}_r)$ where the maximal values of the individual similarities are independent to each other, i.e. $sim(\mathfrak{d}_r, \mathfrak{d}_s, d)$ is independent to $sim(\mathfrak{d}_r, \mathfrak{d}_s, d')$ for each $d, d' \in alt(\mathfrak{d}_r)$.

		Worst Match Scenario				Best Match Scenario			
	matched alternative pair	$sim(d_2,d_1,d_{21})$	$sim(d_2,d_1,d_{22})$	$PMESim(d_2,d_1)$	Match. Class ($\theta_{U/M}$=0.85)	$sim(d_2,d_1,d_{21})$	$sim(d_2,d_1,d_{22})$	$PMESim(d_2,d_1)$	Match. Class ($\theta_{U/M}$=0.85)
i=0	-	0	0	0	UNMATCH	1	1	1	MATCH
i=1	$sim(d_{21},d_{12})$=0.2	0.13	0	0.078	UNMATCH	1	1	1	MATCH
i=2	$sim(d_{21},d_{11})$=0.8	0.55	0	0.33	UNMATCH	0.88	1	0.928	MATCH
i=3	$sim(d_{22},d_{12})$=0.5	0.55	0.5	0.53	UNMATCH	0.88	1	0.928	MATCH
i=4	$sim(d_{21},d_{13})$=0.6	0.72	0.5	0.632	UNMATCH	0.72	1	0.832	UNMATCH
i=5	$sim(d_{22},d_{11})$=0.6	0.72	0.5875	0.667	UNMATCH	0.72	0.85	0.772	UNMATCH
i=6	$sim(d_{22},d_{13})$=0.4	0.72	0.5875	0.667	UNMATCH	0.72	0.5875	0.667	UNMATCH

Figure 7.62.: The incremental computation of *PME-Sim*$(\mathfrak{d}_2, \mathfrak{d}_1)$

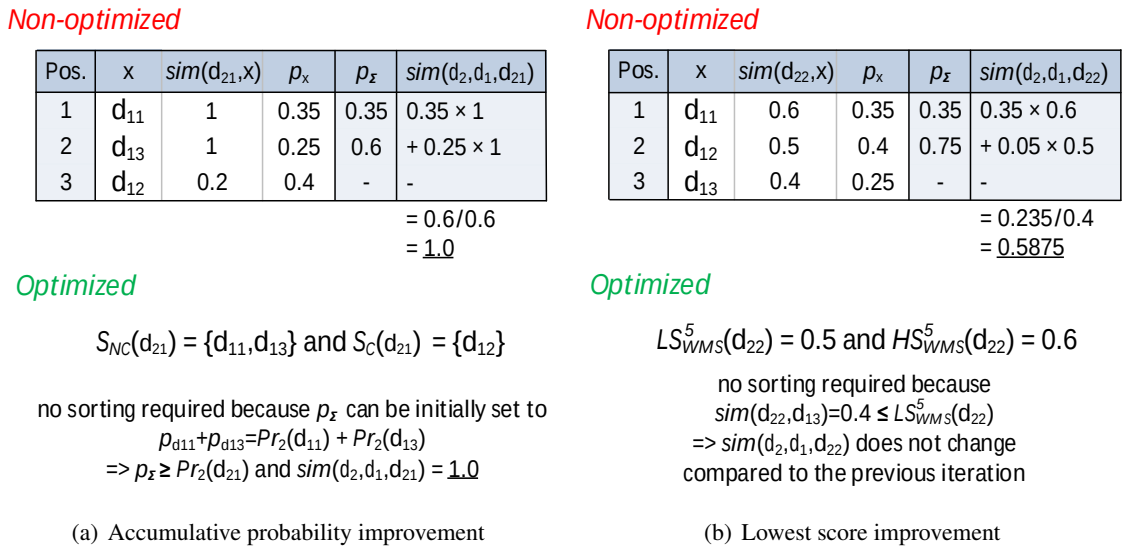
Therefore, let $\{d_r, d_s\}$ be the alternative pair that has been matched in iteration i and let $sim^i_X(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ be the similarity from d_r to $\mathfrak{d}_s$ that has been computed for scenario $X \in \{WMS, BMS\}$ in iteration i, we only need to recompute the similarity scores $sim^i_{WMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ and $sim^i_{BMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ and then can derive the Probabilistic Monge-Elkan Similarity between $\mathfrak{d}_r$ and $\mathfrak{d}_s$ for scenario $X \in \{WMS, BMS\}$ based on the result of the previous iteration by using the newly computed scores as:

$$PME\text{-}Sim^i_X(\mathfrak{d}_r, \mathfrak{d}_s) = PME\text{-}Sim^{i-1}_X(\mathfrak{d}_r, \mathfrak{d}_s) - sim^{i-1}_X(\mathfrak{d}_r, \mathfrak{d}_s, d_r) + sim^i_X(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$$

Example 147 *For illustration, we consider the two probabilistic entity description* $\mathfrak{d}1$ *and* $\mathfrak{d}_2$ *from Figure 7.44 and assume the alternative pair space that is constructed without dependencies (see Figure 7.44(c)). The incremental computation of the Probabilistic Monge-Elkan Similarity PME-Sim*$(\mathfrak{d}_2, \mathfrak{d}_1)$ *is depicted in Figure 7.62 (note that PME-Sim is directed and we compute the similarity from* $\mathfrak{d}_2$ *to* $\mathfrak{d}_1$ *instead to the contrary direction because of illustration reasons). To classify the two corresponding entities* e_1 *and* e_2 *as a* MATCH *or an* UNMATCH*, we use the threshold* $\theta_{U/M} = 0.85$*. To match all alternative pairs six iterations are required (the cells of all completely recomputed similarity scores are yellow colored). Nevertheless, if we use a clustering method that does not need the similarity scores of the* UNMATCHES *as input, the process of incremental computation can be stopped after the fourth iteration and we save two alternative pair matchings.*

The computation of $sim^i_{WMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ and $sim^i_{BMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ can be improved in several ways as well. A first improvement can be done by storing the lowest similarity score ($LS^{i-1}_X(d_r)$) and the highest similarity score ($HS^{i-1}_X(d_r)$) from any alternative of $\mathfrak{d}_s$ to d_r (or ranking score in the case of *PME-Sim*$^{\ominus}$ or *PME-Sim*$^{\otimes}$) that was considered for computing $sim^{i-1}_X(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ in the previous iteration. If the similarity score $sim(d_r, d_s)$ that is newly computed in iteration i is lower than $LS^{i-1}_{WMS}(d_r)$, we do not need to recompute $sim^i_{WMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ because it cannot change, i.e. $sim^i_{WMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r) = sim^{i-1}_{WMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$. Moreover, if $sim(d_r, d_s)$ is lower than $LS^{i-1}_{BMS}(d_r)$, we only need to recompute $sim^i_{BMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ if $HS^{i-1}_{BMS}(d_r) = 1$, because in that situation it could be the case that any score in the front of the sorted list is replaced by the newly computed one.

If we actually need to recompute $sim^i_{BMS}(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$, we can improve this computation further on. First we divide the set of alternatives of $\mathfrak{d}_s$ into two disjoints sets $S_C(d_r)$ and $S_{NC}(d_r)$. The first set



(a) Accumulative probability improvement

(b) Lowest score improvement

Figure 7.63.: Two samples for improving the computation of the similarity from an alternative of $\mathfrak{d}_2$ to $\mathfrak{d}_1$

contains all alternatives d_s for which we already have computed the similarity from d_r to d_s and $S_{NC}(d_r)$ contains the rest. In the original variant of the Probabilistic Monge-Elkan Similarity (see Section 7.5.4), we initialize the accumulative probability $p_\sum$ and the similarity score $sim(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$ by zero, but now we initialize them with $p_\sum = \sum_{d^* \in S_{NC}(d_r)} Pr_{r,s}(d_r, d^*)$ and $sim(\mathfrak{d}_r, \mathfrak{d}_s, d_r) = p_\sum$. If this initial value of $p_\sum$ is already greater than $Pr_r(d_r)$, the algorithm of *PME-Sim* already stops before its sorting step and returns $sim(\mathfrak{d}_r, \mathfrak{d}_s, d_r) = 1$. Otherwise we sort the alternatives of $S_C(d_r)$ and start the iterative step of the algorithm. If we utilize one of the variants *PME-Sim*$^\ominus$ or *PME-Sim*$^\otimes$, this optimization technique becomes invalid, because alternatives are not only sorted by their similarity, but are also sorted by their priority score. These scores can be computed without matching all alternative pairs. Nevertheless, we cannot assume anymore that all alternatives of $S_{NC}(d_r)$ would be sorted in front of all alternatives of $S_C(d_r)$. Note that the sets $S_C(d_r)$ and $S_{NC}(d_r)$ can be computed incrementally as well. Initially, we set $S_C(d_r) = \emptyset$ and $S_{NC}(d_r) = alt(d_s)$ and then for each newly computed similarity score that concerns d_r (and hence for each recomputation of $sim(\mathfrak{d}_r, \mathfrak{d}_s, d_r)$) we shift an element from $S_{NC}(d_r)$ to $S_C(d_r)$. Note, for the Worst Match Scenario, such an optimization technique cannot be used, because the alternatives of $S_{NC}(d_r)$ are sorted at the end of the list.

Example 148 *For illustration, we consider the sample process from Figure 7.62 and take a closer look on the computations of* $sim_{BMS}(\mathfrak{d}_2, \mathfrak{d}_1, d_{21})$ *in iteration* 1 *and* $sim_{WMS}(\mathfrak{d}_2, \mathfrak{d}_1, d_{22})$ *in iteration* 6*. The computation of the first score (see Figure 7.63(a)) can be improved because the accumulative probability of all non-matched pairs is equal to the probability* $Pr_2(d_{21})$*. Thus, we can conclude that the similarity score is* 1 *without computing it. The computation of the second score (see Figure 7.63(b)) can be improved because the newly computed similarity score* $sim(d_{22}, d_{13}) = 0.4$ *is lower than the lowest score that has been used for computing* $sim_{WMS}(\mathfrak{d}_2, \mathfrak{d}_1, d_{22})$ *in the previous iteration, i.e.* $0.5 < LS^5_{WMS}(d_{22})$*. Thus, the similarity score will not change and its re-computation can be avoided.*

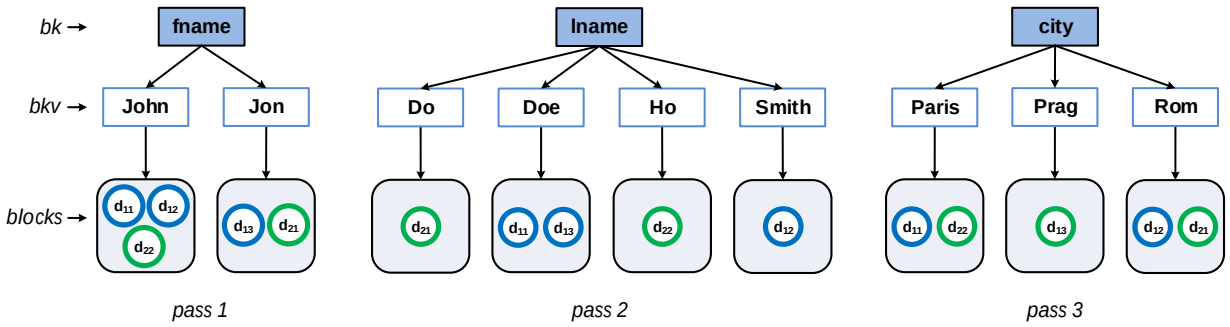


Figure 7.64.: Multi-pass alternative blocking of the entity descriptions $\mathfrak{d}_1$ and $\mathfrak{d}_2$

Of course, the hard pruning of matching alternative description pairs becomes less effective if a specific quality requirement is used, because in this case we usually need to compute the similarity of all description alternative pairs in advance in order to decide which of them should contribute to the final aggregation output (see concepts of minimal-θ aggregation and maximal-θ aggregation in Section 7.2.6).

7.6.2. Soft Optimization Techniques

In contrast to hard optimization techniques, soft optimization techniques can change the matching result. Nevertheless, entity matching is heuristic by nature and an absolutely correct result can usually not be guaranteed. For that reason, a change in the matching result is acceptable if the decrease in matching quality remains low.

7.6.2.1. Alternative Pair Space Reduction

The simplest way to prune the number of alternative pair matchings is to use a conventional blocking method as we know it from constructing the candidate pair space for duplicate detection in certain data and to consider both probabilistic values as duplicate-free sources, i.e. alternatives of the same description do not need to be matched.

Of course, if we reject some alternative description pairs from the pair space we cannot compute a matching result for these pairs in any of the matching phases. Nevertheless, because uncertainty resolution can be performed at any moment of the detection process, we need a matching result for these pair for every matching phase. One possible solution is to assume the most negative feature score for these pairs. As a consequence, for each matching phase all rejected pairs are represented by the same alternative of the probabilistic matching result. It seems intuitive than a matching phase always produces the most negative value of its output domain if its input value was the most negative value of its input domain. For that reason, we save computation time whilst we do not actually process any of the matching phases for these values, but automatically assume the most negative value of the phase's output domain as matching result.

Example 149 *For illustration, we consider the two probabilistic entity descriptions* $\mathfrak{d}_1$ *and* $\mathfrak{d}_2$ *from Figure 7.44 and apply a multi-pass approach of standard blocking to their alternatives. For that purpose, we use the attribute* 'fname'*, the attribute* 'lname'*, and the attribute* 'city' *each as a blocking key in a separate pass. The resultant blocks are presented in Figure 7.64. All alternative description pairs from*

the alternative pair space that are not in the same block, i.e. $\{d_{11}, d_{21}\}$ *and* $\{d_{13}, d_{22}\}$*, are automatically associated with the most negative feature score, are associated with a similarity score of zero, and are classified as an* UNMATCH *thereby having the maximal impact value one.*

If we use a blocking method to initially reject clear UNMATCHES it is useful to compute the resolution result already before we start to match the remaining alternative pairs, because the rejections made by the blocking method can be sufficient to clearly classify the considered entity pair as an UNMATCH.

7.6.2.2. Soft Alternative Match Pruning

The underlying idea of soft match pruning is to process matching phase by matching phase for all alternative description pairs collectively and then to reduce uncertainty between two phases. Recall, if all alternative description pairs are processed collectively, the output of a matching phase can be interpreted as a probabilistic matching result, e.g. probabilistic feature score, that is then passed to the next matching phase or is passed to an uncertainty resolution method respectively. Since the soft alternative match pruning is realized by processing a matching phase for all alternative pairs collectively we also call it *In-Width pruning*.

The actual purpose for reducing the uncertainty of a matching result between two matching phases is to reduce computation effort. As a consequence, a reduction method is primarily useful if it decreases the number of alternatives[11]. Recall from Section 6.7.2.2 that such uncertainty reduction functions are either deciding or mediating. We adopt these distinction for the methods we use for uncertainty reduction in this thesis.

- **Deciding Reduction Methods:** Deciding reduction methods select some alternatives of the probabilistic input value as representatives and discard the other alternatives. Since the probabilities of the selected alternatives do not sum up to one anymore, they need to be normalized. The simplest selection approach is to only retain the k most probable alternatives.

- **Mediating Reduction Methods:** In general mediating reduction methods can be defined in several ways. In this thesis, however, we only consider mediating reduction methods that cluster all alternatives of the probabilistic input value and then aggregate the alternatives of each cluster to a single representative. The probability of a representative is defined as the accumulative probability of all the alternatives of its corresponding cluster. For aggregation we can use any of the methods that we have discussed in Section 7.2. The clustering process can be constrained by the number of clusters, can be constrained by a minimal threshold for the accumulative probability per cluster, or can be constrained by a minimal pairwise distance between two alternatives of the same cluster.

  In contrast to deciding reduction methods, the probabilities of the computed representatives still sum up to one. Thus, on one hand deciding reduction methods require a normalization and mediating reduction methods do not, but on the other hand computing a clustering is much more time

[11] If we combine In-Width pruning with In-Depth pruning (see the hybrid pruning approach presented in the end of this section), uncertainty reduction can be already useful without reducing the number of alternatives. For example, if we use an uncertainty reduction method that redistributes a large amount of probability mass from several alternatives to a single alternative, this alternative becomes more probable and the resolution process can be stopped earlier.

consuming than simply selecting some of the given alternatives. However, the clustering effort can be reduced by selecting the most probable alternatives as they are and then to perform clustering only on the remaining alternatives.

Since we only use methods that shift probability mass from less probable alternatives to more probable alternatives, these methods are uncertainty reducing with respect to the most measures of certainty (see Section 6.10).

Uncertainty reduction turns into uncertainty resolution, if only one alternative is selected or all alternatives are grouped within a single cluster.

As uncertainty resolution, uncertainty reduction can be conducted at any moment of the detection process. Thus, the earliest moment of reduction is to reduce the number of alternatives each of the probabilistic entity descriptions has (before data preparation or after data preparation). The second moment is to reduce the alternative pair space, the third moment is to reduce the uncertainty of the probabilistic feature score that has been computed for the corresponding descriptions, and so on.

Example 150 *The principle of soft matching is illustrated in Figure 7.65. In the depicted sample scenario, we start with two probabilistic entity descriptions* $\mathfrak{d}_1$ *and* $\mathfrak{d}_2$ *that have five description alternatives (description* $\mathfrak{d}_1$*) or six (description* $\mathfrak{d}_2$*) description alternatives respectively (Moment 1). As demonstrated at Moment 2, the earliest moment of uncertainty reduction is to reduce the uncertainty of the given entity descriptions by discarding some description alternatives (from entity description* $\mathfrak{d}_1$ *two alternatives are discarded) or by aggregating some description alternatives (for entity description* $\mathfrak{d}_2$*, the last three alternatives are aggregated to a single one). Recall, if we would additionally perform a data preparation phase, this kind of reduction can theoretically performed for a second time (one time for the preparation phase and one time after the preparation phase). From the uncertainty reduced entity descriptions, the alternative pair space is constructed (Moment 3). Then, uncertainty reduction is applied to the pair space by aggregating pairs of description alternatives (Moment 4). Afterwards to each of the remaining alternative pairs the feature matching phase is applied (Moment 5). Note, in this example two alternative pairs produce the same probabilistic feature score alternative. The probabilistic feature score is then uncertainty reduced by applying an aggregation method at Moment 6. The uncertainty reduced feature score serves then as input to the similarity computation phase (Moment 7). Subsequently, the uncertainty of the computed probabilistic similarity score is reduced by discarding one of its alternatives (Moment 8). The reduced similarity score is then passed to the classification phase that computes a probabilistic duplicate decision based on the probabilistic similarity score (Moment 9). Finally (Moment 10) a certain duplicate decision is derived from the probabilistic duplicate decision by any uncertainty resolution method (see Section 7.2). Obviously, if uncertainty resolution would be performed earlier in the matching process, performing uncertainty reduction between the following phases is useless because the phases' matching results are already certain.*

Without uncertainty reduction, we would perform each of the matching phases for $5 \times 6 = 30$ *times. Thus, in overall we would apply a matching phase to an alternative pair for* $3 \times 30 = 90$ *times (of course, because of same pairs result in same intermediate results, the number of actually performed matching phases is a little bit lower). In contrast, by using uncertainty reduction the number of times a matching phase need to be performed is reduced to* 7 *(feature matching),* 4 *(similarity computation),*

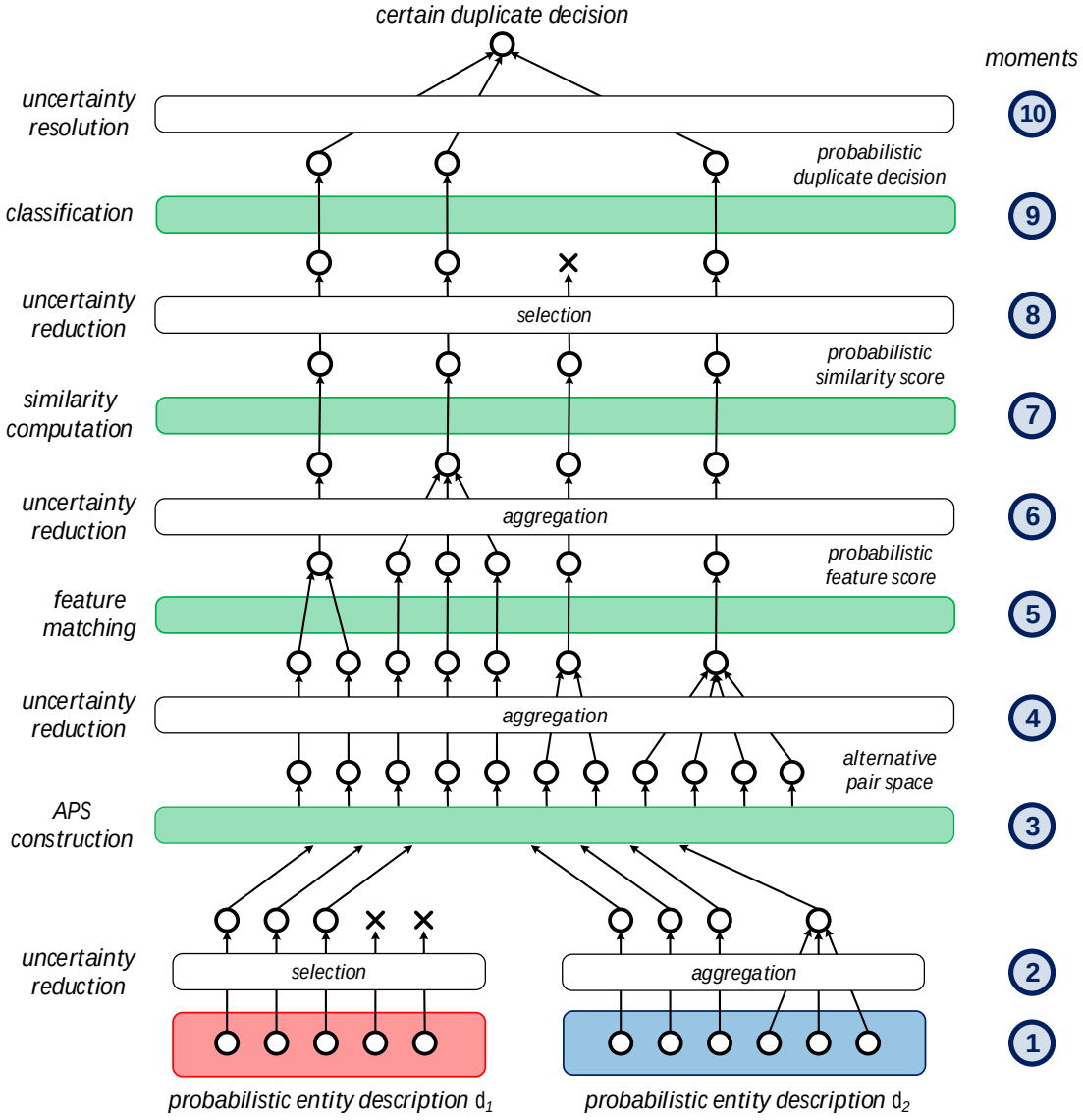
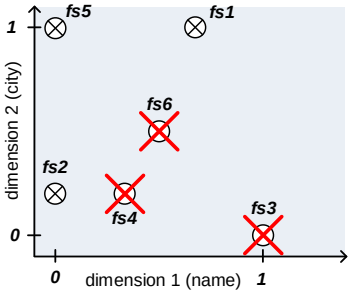


Figure 7.65.: A sample execution process of the In-Width pruning

and 3 *(classification). Consequently, the overall matching effort has been decreased by* $\frac{(90-14)\times 100}{90}\% = 84\%$.

As presented in Section 7.5, several resolution methods as the Probabilistic Monge-Elkan Similarity require a map from the pairs of the alternative pair space to the alternatives of the probabilistic matching result. Discarding alternatives from the probabilistic matching result, however, leads to the case that an alternative description pair cannot be mapped to an alternative of the matching result anymore. Nonetheless, this shortcoming is automatically corrected by the normalization, because the probability mass of that alternative description pair is redistributed to other pairs and the pair itself becomes impossible. It is important to note that a redistribution of probability mass introduces correlations into the alternative pair space and we therefore have to utilize one of the extended versions of the Probabilistic Monge-Elkan Similarity that are described in Section 7.5.4.1. Moreover, the original probabilities of the individual



(a) Reduction Process

alt.	ACVector	AIVector	Pr
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.35
fs2	$\vec{c}_A$=[0.0,0.2]	$\vec{i}_A$=[0.7,0.8]	0.39
fs3	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.26

(b) Probabilistic feature score

alt.	Pr_1
d_{11}	0.35
d_{12}	0.65
d_{13}	0.0

alt.	Pr_2
d_{21}	0.74
d_{22}	0.26

(c) Updated probability distributions

alt. pair	prob.	feature score
(d_{11},d_{21})	0.35	fs1
(d_{12},d_{21})	0.39	fs2
(d_{13},d_{21})	0.0	-
(d_{11},d_{22})	0.0	-
(d_{12},d_{22})	0.26	fs3
(d_{13},d_{22})	0.0	-

(d) Provenance Table

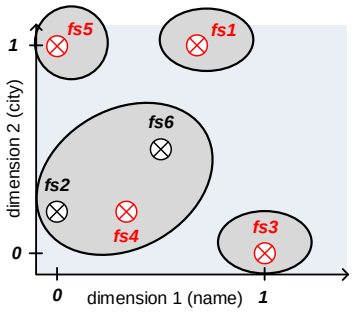
Figure 7.66.: Uncertainty reduction by using a deciding reduction method

entity description alternatives are not valid anymore, but need to be recomputed from the modified joint probability distribution that can be derived from the provenance table instead.

Example 151 *For illustration, we consider the same situation that is depicted in Example 141 and therefore consider the output of the feature matching phase that result from processing the alternative pair space from Figure 7.44(c). Moreover, we assume the probabilistic feature score and the corresponding provenance table that are presented in Figure 7.57(a) and Figure 7.57(b) respectively. Now, we reduce the resultant probabilistic feature score by discarding its three least probable alternatives, i.e. fs_3, fs_4, and fs_6, and normalize the probabilities of the remaining alternatives. The reduction process is graphically demonstrated in Figure 7.66(a) and the reduced probabilistic feature score is presented in Figure 7.66(b). The provenance table is now modified by replacing all references to one of the discarded alternatives to a null pointer nil. The modified provenance table is presented in Figure 7.66(d). Since the provenance table changes, the joint probability distribution on the considered description alternatives changes as well. As a consequence, the probability of a single description alternative need to be updated to the accumulative probability of all alternative description pairs that contains this description alternative. The updated probability distributions of the two considered probabilistic entity descriptions $\mathfrak{d}_1 = (d_1, Pr_1)$ and $\mathfrak{d}_2 = (d_2, Pr_2)$ are shown in Figure 7.66(c).*

In contrast, if we perform a mediating reduction method the probabilities of the alternative description pairs do not change, but the pairs are mapped to another matching result.

Example 152 *For illustration, we consider the same alternative description pair space, the same probabilistic feature score, and the same provenance table as in the previous example. First we group the alternatives of the probabilistic feature score into four clusters and select the alternative that is closest to the cluster's centroid as the cluster's representative (see Figure 7.67(a)). By doing so, the alternatives fs_2, fs_4, and fs_6 are aggregated to the alternative fs_4. Since fs_4 represents the two other feature score alternatives in the reduction result, their appearances in the provenance table are replaced by fs_4. Moreover, the probability of fs_4 is increased by the accumulative probability of fs_2 and fs_6 and therefore results in $0.24 + 0.14 + 0.1 = 0.48$. The reduced probabilistic feature score and the modified provenance table are presented in Figure 7.67(b) or Figure 7.67(c) respectively.*



(a) Reduction Process

alt.	ACVector	AIVector	Pr
fs1	$\vec{c}_A$=[0.67,1.0]	$\vec{i}_A$=[0.5,0.6]	0.21
fs3	$\vec{c}_A$=[1.0,0.0]	$\vec{i}_A$=[0.4,0.7]	0.15
fs4	$\vec{c}_A$=[0.33,0.2]	$\vec{i}_A$=[0.5,0.8]	0.48
fs5	$\vec{c}_A$=[0.0,1.0]	$\vec{i}_A$=[0.7,1.0]	0.16

(b) Probabilistic feature score

alt. pair	prob.	feature score
(d_{11},d_{21})	0.21	fs1
(d_{12},d_{21})	0.24	fs4
(d_{13},d_{21})	0.15	fs3
(d_{11},d_{22})	0.14	fs4
(d_{12},d_{22})	0.16	fs5
(d_{13},d_{22})	0.10	fs4

(c) Provenance table

Figure 7.67.: Uncertainty reduction by using a mediating reduction method

7.6.2.3. Hybrid Alternative Match Pruning

Whereas by using in-depth pruning we perform all matching phases for one alternative pair one after another, by using in-width pruning we perform each matching phase for all alternative pairs at once. Thus, the underlying ideas of both pruning concepts are contrary. Nevertheless, they can be combined by considering the alternative description pairs in a block-wise fashion. For that purpose, we divide the alternative description pair space into several disjoint blocks (these blocks should not be confused with the blocks that are created by a blocking method for rejecting pairs from the matching process) and then perform matching with in-width pruning for each of these blocks subsequently until the resolution result is deterministic, i.e. until the whole matching process can be stopped because from Worst Match Scenario and Best Match Scenario the same consequences result (see Section 7.6.1.2).

A high probable alternative description pair is often sufficient to compute a deterministic resolution result. Therefore, adding a prescribed number of pairs into one block can produce an unnecessary overhead in many situations and the pruning becomes contra productive. For that reason, we use the accumulative probability of the pairs rather than the number of pairs for building blocks. Thus, we initially sort the alternative description pairs by their probabilities and then start with adding the most probable pair to an empty block. The each next probable pair is then added to this block until the accumulative probability of all the pairs in this block is above a given threshold θ_{block}. After the block is full, the matching process is performed for all pairs of that block with in-width pruning, i.e. all these pairs are processed phase by phase. If the pairs of the processed block are not sufficient to compute a deterministic resolution result, the next block is built and so on until the resolution result is deterministic or all alternative description pairs have been processed. Due to most probable alternative pairs are matched first, the blocks of the latter match iterations are usually larger than these of the first.

In cases where no method of uncertainty reduction is initially applied to the probabilistic descriptions, using the hybrid pruning technique is equivalent to using in-width pruning if $\theta_{block} = 1$ because all pairs are grouped into one block. In contrast, by setting $\theta_{block} = 0$, using hybrid pruning is equivalent to using in-depth pruning because each pair is grouped into a separate block.

Example 153 *For illustration, we consider a sample matching process that is performed with hybrid match pruning and that is presented in Figure 7.68. For ease of presentation, we consider uncertainty*

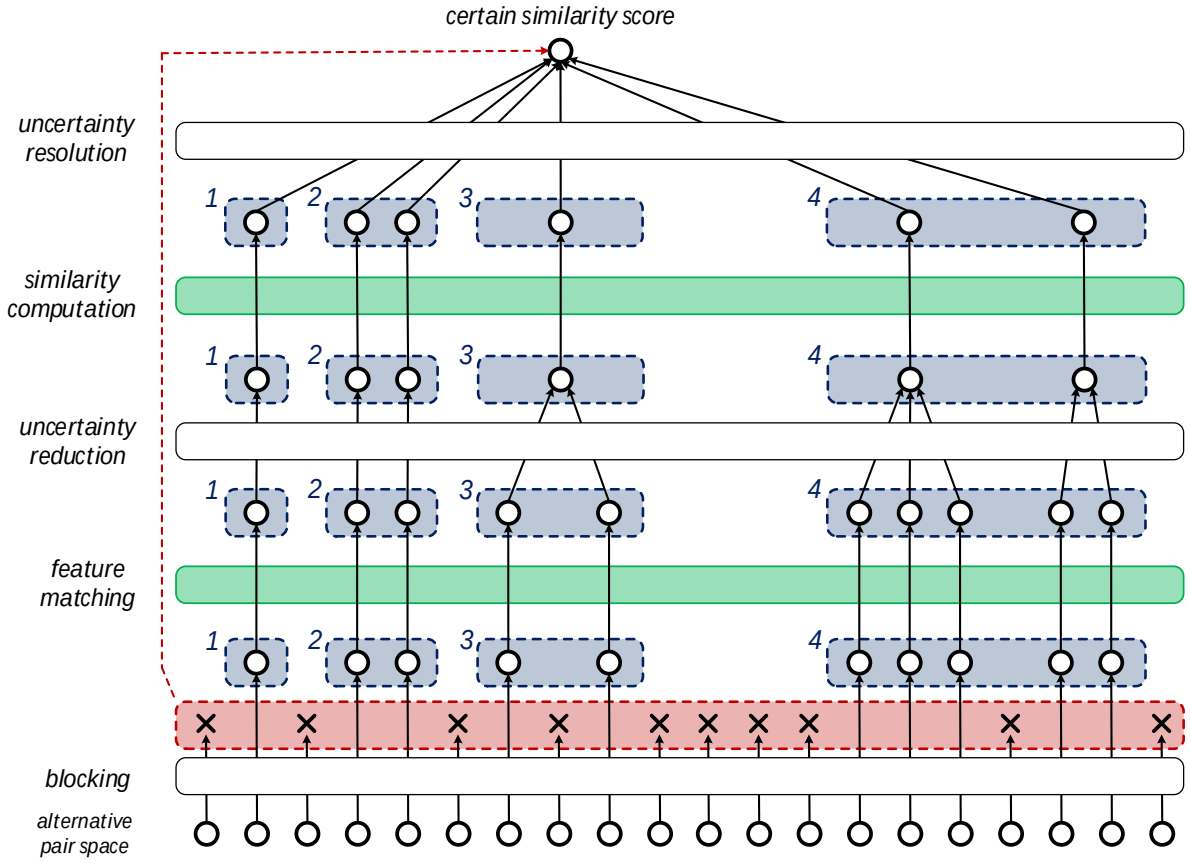


Figure 7.68.: Sample matching process with hybrid match optimization

reduction only after the feature matching phase and assume uncertainty resolution to be performed after the similarity computation phase.

We start from the alternative description pair space that consists of twenty pairs. After pair space construction, we assume that a conventional blocking method is used to reject clear UNMATCHES. *Because ten pairs are already classified as* UNMATCHES, *we compute the Worst Match Scenario and the Best Match Scenario of the resolution result for a first time and assume that both scenarios do not have same consequences.*

Then, the remaining alternative description pairs are grouped into four blocks. Whereas the first block only contains a single pair, the second block and the third block contain each two pairs and the fourth block contains five pairs. After block building, the single alternative description pair of the first block is passed to the feature matching phase and the similarity computation phase. The resultant similarity score is considered in the resolution method. Since this score is not sufficient to compute a deterministic resolution result, the pairs of the next block are passed to the feature matching phase and a probabilistic feature score is computed. After feature matching a method of uncertainty reduction is applied to that score, but does not reduce its number of alternatives. Therefore a probabilistic similarity score is computed and incorporated into the resolution result. As in the previous iteration, the resolution result is not deterministic and we need to process another block. Thus, the pairs of third block are passed

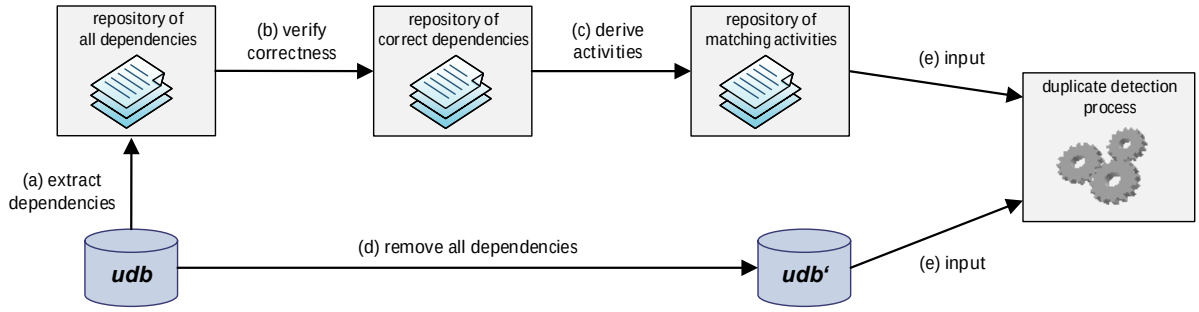


Figure 7.69.: The five steps of cleaning the database from entity dependencies

to the feature matching phase. Again, a reduction method is applied to the resultant probabilistic feature score and in this case both feature score alternatives are aggregated to a single one. This feature score is then passed to the similarity computation phase and the resultant similarity score is incorporated into the resolution method. If the resolution result is still not deterministic, we need to process the last block as well. By doing so a feature score with five alternatives is computed by the feature matching phase. These alternatives are then aggregated to two alternatives by the uncertainty reduction method. The reduced feature score is then passed to the similarity computation phase and its resultant similarity score is passed to the resolution method. Since we have computed a similarity score for all alternative description pairs, the resolution result must now be deterministic. In summary, we compute the feature matching phase for ten times and compute the similarity computation phase for six times. In contrast, without pruning we need to compute both phases for twenty times. Thus, even we needed to process all blocks we save $\frac{(40-16)\times 100}{40}\% = 60\%$ *of matching effort.*

7.7. Uncertain Data Preparation

Data preparation of uncertain data generally encompasses the same preparation activities as we have presented for certain data in Section 4.3.3. Nevertheless, in the presence of data uncertainty additional preparation activities can help to improve detection quality further on. As demonstrated in Section 7.1, the effectiveness of a duplicate detection process can be diminished if the database contains incorrect entity dependencies and/or incorrect ARM-dependencies. Moreover, the intended affections of entity dependencies often differ from their actual affections if we let them unprepared in the database. For that reason, we introduce two new preparation activities in this section. We start with a cleaning of entity dependencies in Section 7.7.1 and then proceed with a removal of incorrect ARM-dependencies in Section 7.7.2.

7.7.1. Cleaning of Entity Dependencies

As discussed in Section 7.1.5, incorrect entity dependencies can produce inter-world duplicates that in turn cannot be detected if we restrict detection to single worlds. Moreover, they can change the detection of intra-world duplicates to the worse if the set of possible worlds is reduced for an incorrect reason.

To neglect these negative affects we have to differentiate between the dependencies themselves and the affections that are caused by these dependencies. We realize this differentiation in the five steps illustrated in Figure 7.69.

(a) First all entity dependencies are extracted from the uncertain input database *udb* and stored in a dependency repository.
(b) Second the correctness of all extracted dependencies is verified.
(c) Then we derive specific matching activities from the correct dependencies and store them in an activity repository.
(d) In the fourth step, we remove all entity dependencies from the input database *udb* and hence create the entity-independent database *udb*$'$.
(e) Finally, the prepared database *udb*$'$ as well as the repository of the derived matching activities serves as input to the duplicate detection process.

We will discuss these five steps in the following subsections. Extracting dependencies and removing dependencies is quite similar. For that reason we start with a collective consideration of both.

7.7.1.1. Extracting/Removing Entity Dependencies

Removing dependencies from a possible worlds representation implies to extend the world space by additional worlds and/or to recompute some of the worlds' probabilities. For illustration we consider the possible world space $\mathbf{W} = \{W_2, W_3\}$ from Example 110 that is presented in Figure 7.11(b). Removing the given dependencies between the entities e_1 and e_2 means to transform $\mathbf{W}$ into the possible world space $\mathbf{W}' = \{W_1, W_2, W_3, W_4\}$ that is presented in Figure 7.11(a). For large databases with many entities and many worlds, this process can be very complex and hard to define. Fortunately, common representation systems model dependencies explicitly. For instance, in pc-databases dependencies between two entities are modeled by using the same variables in the conditions of the entities' tuples. This fact makes a removal of dependencies from a compactly represented database much simpler than performing it on the possible worlds representation.

In general, methods for extracting and removing entity dependencies depend on the used representation system. TI-databases, AOR-databases, and AOR?-databases are not powerful enough to model dependencies between different entities. The same holds for BID-databases if we consider the *DEI* as block number, i.e. each tuple of one block represents the same databas entity. For that reason, we restrict our considerations to pc-databases in this Section.

Example 154 *To demonstrate the different natures of entity dependencies we start with the single pc-table that is presented in Figure 7.70(a). Recall, the marginal probability of a tuple in a pc-database is actually implicated by its condition and does not need to be stored separately. Nevertheless, for demonstration purposes we additionally store the marginal probability of each tuple by a separate attribute in this example. The considered pc-table contains four database entities each represented by a single x-tuple that has one or two alternatives. Whereas* e_1 *and* e_4 *certainly belong to the database,* e_2 *and* e_3 *only maybe belong to it. Since each database entity is represented by a single x-tuple, we consider entities and x-tuples interchangeable in this example. Moreover, to better illustrate the correlations between the individual entities, we use the notion* $e_{i,j}$ *to refer to the* j*th tuple from entity* e_i*, e.g. tuple* t_3

Person

	RK	DEI	name	residence	p	condition
t_1	1	e1	John Doe	New York	0.6	X=1 v X=2
t_2	2	e1	John Doe	Detroit	0.4	X=3 v X=4
t_3	3	e2	Janett Ho	Detroit	0.4	Y=1
t_4	4	e3	Jane Doe	New York	0.36	(X=1 v X=2) ∧ Y=2
t_5	5	e3	Jane Do	Detroit	0.24	(X=3 v X=4) ∧ Y=2
t_6	6	e4	Bill Smith	Chicago	0.8	X=1 v X=3
t_7	7	e4	William Smith	Chicago	0.2	X=2 v X=4

World-Table

var	value	Pr
X	1	0.5
X	2	0.1
X	3	0.3
X	4	0.1
Y	1	0.4
Y	2	0.6

(a) Database with entity dependencies

Prob →	0.2	0.3	0.04	0.06	0.12	0.18	0.04	0.06
Setting	X=1 Y=1	X=1 Y=2	X=2 Y=1	X=2 Y=2	X=3 Y=1	X=3 Y=2	X=4 Y=1	X=4 Y=2
	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8
$e_{1,1}$	■	■	■	■				
$e_{1,2}$					■	■	■	■
$e_{2,1}$	■		■		■		■	
$e_{3,1}$		■		■				
$e_{3,2}$						■		■
$e_{4,1}$	■	■			■	■		
$e_{4,2}$			■	■			■	■

(b) Possible worlds representation with dependencies

Figure 7.70.: Sample database with entity dependencies and its corresponding possible worlds representation

is therefore also denoted as alternative $e_{2,1}$ *in the remainder of this section (note, since we consider a single pc-table this notion is unique for all tuples).*

The considered pc-table contains different kinds of dependencies that are modeled by defining the tuple conditions on the two variables X *and* Y*. First, the memberships of entity* e_2 *and entity* e_3 *to the table* 'Person' *(and hence the whole database) are mutually exclusive. This is modeled by conditioning their tuples with different assignments of variable* Y*. Second, the alternatives of* e_3 *are strongly correlated with the alternatives of* e_1*, because the first alternative of* e_3 *(tuple* t_4*) only exists, if the first alternative of* e_1 *(tuple* t_1*) exists. Similarly, the existence of alternative* $e_{3,2}$ *depends on the existence of alternative* $e_{1,2}$*. This dependency is modeled in the database by using the variable* X *in the conditions of the aforementioned alternatives. Finally, the database contains a weak dependency between* e_1 *and* e_4*. This dependency does not exclude any combination of alternatives as for example the dependency between* e_1 *and* e_3 *does, but redistributes the probabilities to which the individual alternatives coexist. For example, in the case of independence, the alternatives* $e_{1,1}$ *and* $e_{4,1}$ *would coexist with a probability of* $0.6 \times 0.8 = 0.48$*, but in our database this probability is* 0.5 *instead. This dependency is modeled by the variable* X

	$e_{4,1}$	$e_{4,2}$
$e_{3,1}$	$X = 1 \wedge Y = 2$	$X = 2 \wedge Y = 2$
$e_{3,2}$	$X = 3 \wedge Y = 2$	$X = 4 \wedge Y = 2$
$\emptyset$	$(X = 1 \vee X = 3) \wedge Y = 1$	$(X = 2 \vee X = 4) \wedge Y = 1$

(a) Dependency represented by conditions

	$e_{4,1}$	$e_{4,2}$	$\sum$
$e_{3,1}$	0.3	0.06	**0.36**
$e_{3,2}$	0.18	0.06	**0.24**
$\emptyset$	0.32	0.08	**0.4**
$\sum$	**0.8**	**0.2**	**1.0**

(b) Joint probability distribution

Figure 7.71.: The dependency between e_3 and e_4 represented by conditions and probabilities

that has four assignments, each one for the possible combinations of the alternatives of e_1 and e_4. Since e_4 and e_3 both correlates with e_1, there is also a correlation between e_4 and e_3.

The eight possible worlds that are modeled by this database are depicted in Figure 7.70(b). Each row corresponds to an alternative of the considered entities and each column represents a possible world. A blue colored cell means that the alternative of the corresponding row belongs to the world of the corresponding column. The probability as well as setting of the variables X and Y leading to a specific world are listed on the top of the world's column.

A dependency can be identified by one or more same variables that are used in the conditions of tuples that represent different entities. For extracting such a dependency, we select all entities that refer to these variables. If we are only interested on dependencies between two entities, we additionally restrict the selection to the considered entities. In cases of strong dependencies we can then represent the extracted dependency by a logical expression. For example, the mutual exclusion between e_2 and e_3 can be expressed as $\forall W \in \mathbf{W}\colon e_{2,1} \in W \Rightarrow e_{3,1}, e_{3,2} \notin W$. Such a compact way of representation is not possible for weak dependencies. In this case, we have to explicitly store the joint probability distribution of the involved entities or have to use a graphical model (e.g. a bayesian network) instead.

Example 155 *For illustration we consider the dependency between e_3 and e_4 that is caused by the variables X and Y. The conditions that lead to the coexistence of different pairs of x-tuple alternatives are presented in Figure 7.71(a) (note, the symbol $\emptyset$ represents the case that e_3 does not belong to the considered pc-table at all). The joint probability distribution on both entities is shown in Figure 7.71(b). Since each alternative of x-tuple e_3 can coexists with each alternative of x-tuple e_4, the dependency between these two entities is not strong, but weak.*

In the Description-based Approach, the removal of entity correlations is automatically performed by the extraction of the probabilistic entity descriptions and a removal of entity dependencies in the preparation phase is therefore not required. Nevertheless, in the World-based approach such a removal is necessary if we do not use a specific algorithm for world construction.

If we consider a single pc-table, entity dependencies can be simply removed in three steps. First we compute the marginal probability of each tuple, then we delete the existent variables, and finally we create a new variable per entity (each a single x-tuple) in order to model the mutual exclusion between its alternatives. The number of possible assignments of a newly inserted variable is equal to the entity's number of alternatives if that entity certainly belongs to the table or is one more else. The probability of

Person

	RK	DEI	name	residence	p	condition
t_1	1	e1	John Doe	New York	0.6	V=1
t_2	2	e1	John Doe	Detroit	0.4	V=2
t_3	3	e2	Janett Ho	Detroit	0.4	X=1
t_4	4	e3	Jane Doe	New York	0.36	Y=1
t_5	5	e3	Jane Do	Detroit	0.24	Y=2
t_6	6	e4	Bill Smith	Chicago	0.8	Z=1
t_7	7	e4	William Smith	Chicago	0.2	Z=2

World-Table

var	value	Pr
V	1	0.6
V	2	0.4
X	1	0.4
X	2	0.6
Y	1	0.36
Y	2	0.24
Y	3	0.4
Z	1	0.8
Z	2	0.2

(a) PC-database with removed entity dependencies

Prob →	0.069	0.017	0.046	0.012	0.077	0.019	0.104	0.026	0.069	0.012	0.115	0.029	0.046	0.011	0.031	0.008	0.051	0.013	0.069	0.017	0.046	0.008	0.077	0.019
Setting	V=1	V=1	V=1	V=1	V=1	V=1	V=1	V=1	V=1	V=1	V=1	V=1	V=2	V=2	V=2	V=2	V=2	V=2	V=2	V=2	V=2	V=2	V=2	V=2
	X=1	X=1	X=1	X=1	X=1	X=1	X=2	X=2	X=2	X=2	X=2	X=2	X=1	X=1	X=1	X=1	X=1	X=1	X=2	X=2	X=2	X=2	X=2	X=2
	Y=1	Y=1	Y=2	Y=2	Y=3	Y=3	Y=1	Y=1	Y=2	Y=2	Y=3	Y=3	Y=1	Y=1	Y=2	Y=2	Y=3	Y=3	Y=1	Y=1	Y=2	Y=2	Y=3	Y=3
	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2	Z=1	Z=2
	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}	W_{17}	W_{18}	W_{19}	W_{20}	W_{21}	W_{22}	W_{23}	W_{24}
$e_{1,1}$	■	■	■	■	■	■	■	■	■	■	■	■												
$e_{1,2}$													■	■	■	■	■	■	■	■	■	■	■	■
$e_{2,1}$	■	■	■	■	■	■							■	■	■	■	■	■						
$e_{3,1}$	■	■					■	■					■	■					■	■				
$e_{3,2}$			■	■					■	■					■	■					■	■		
$e_{4,1}$	■		■		■		■		■		■		■		■		■		■		■		■	
$e_{4,2}$		■		■		■		■		■		■		■		■		■		■		■		■

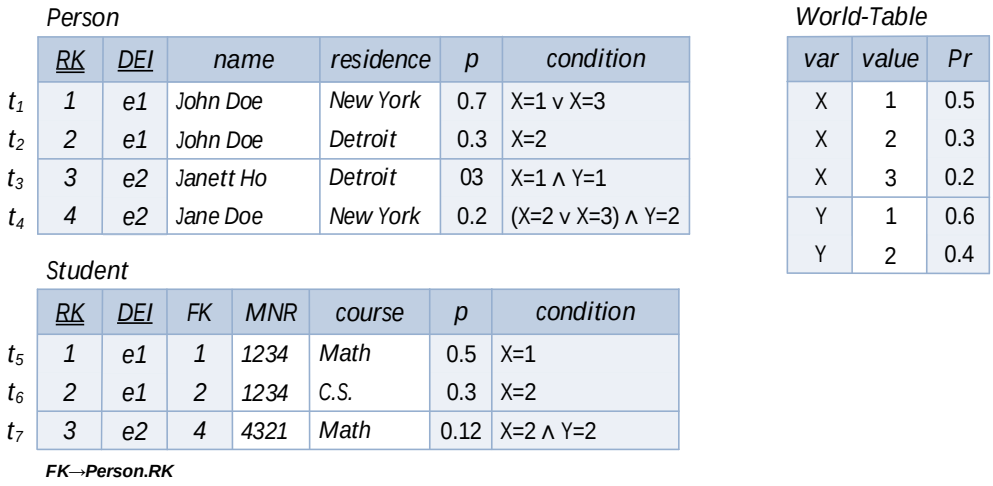
(b) Possible worlds representation without dependencies

Figure 7.72.: Sample database with removed entity dependencies and its corresponding possible worlds representation

each variable assignment is set to the marginal probability of its corresponding alternative or is set to the probability of the entity's non-membership respectively.

Example 156 *Removing all entity dependencies from the pc-table from Figure 7.70(a) result in the pc-table presented in Figure 7.72(a). In this sample, the dependencies are removed by deleting the original variables X and Y and by introducing the four new variables V, X, Y, and Z. Whereas V, X, and Z each have two possible assignments, Y has three possible assignments (one for each alternative of e_3 and one for the case of the non-existence of e_3). The possible worlds representation of that database is shown in Figure 7.72(b) and contains twenty four possible worlds.*

If we consider a larger pc-database with multiple tables where a single database entity can be represented by tuples in several tables, we cannot simply delete the old variables because they can also model



Person

	RK	DEI	name	residence	p	condition
t_1	1	e1	John Doe	New York	0.7	X=1 ∨ X=3
t_2	2	e1	John Doe	Detroit	0.3	X=2
t_3	3	e2	Janett Ho	Detroit	03	X=1 ∧ Y=1
t_4	4	e2	Jane Doe	New York	0.2	(X=2 ∨ X=3) ∧ Y=2

World-Table

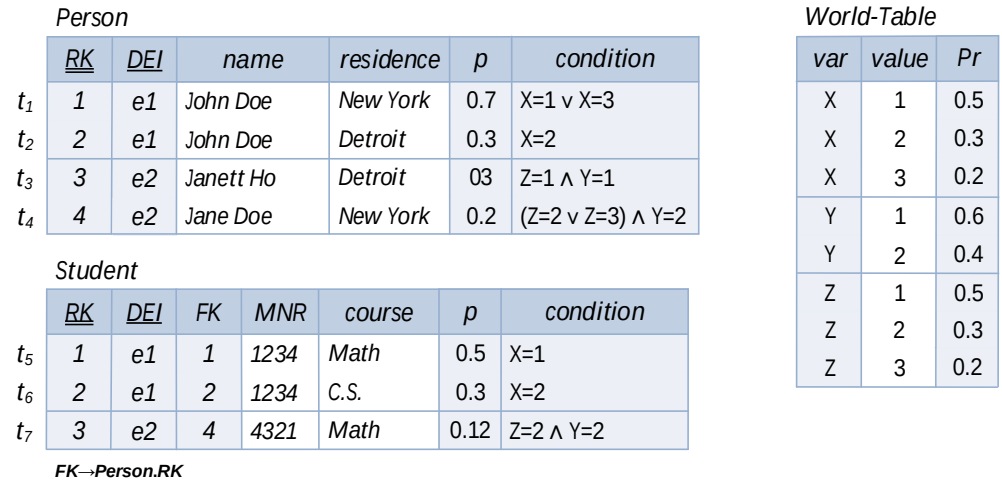
var	value	Pr
X	1	0.5
X	2	0.3
X	3	0.2
Y	1	0.6
Y	2	0.4

Student

	RK	DEI	FK	MNR	course	p	condition
t_5	1	e1	1	1234	Math	0.5	X=1
t_6	2	e1	2	1234	C.S.	0.3	X=2
t_7	3	e2	4	4321	Math	0.12	X=2 ∧ Y=2

FK→Person.RK

(a) PC-database with multi table memberships and entity dependencies



Person

	RK	DEI	name	residence	p	condition
t_1	1	e1	John Doe	New York	0.7	X=1 ∨ X=3
t_2	2	e1	John Doe	Detroit	0.3	X=2
t_3	3	e2	Janett Ho	Detroit	03	Z=1 ∧ Y=1
t_4	4	e2	Jane Doe	New York	0.2	(Z=2 ∨ Z=3) ∧ Y=2

World-Table

var	value	Pr
X	1	0.5
X	2	0.3
X	3	0.2
Y	1	0.6
Y	2	0.4
Z	1	0.5
Z	2	0.3
Z	3	0.2

Student

	RK	DEI	FK	MNR	course	p	condition
t_5	1	e1	1	1234	Math	0.5	X=1
t_6	2	e1	2	1234	C.S.	0.3	X=2
t_7	3	e2	4	4321	Math	0.12	Z=2 ∧ Y=2

FK→Person.RK

(b) PC-database with multi table memberships and removed entity dependencies

Figure 7.73.: Removal of entity dependencies from a pc-database with several tables

correlations between the tuples that represent the same entity and we do not want to remove these dependencies from the database. In such a case, we identify each variable that is used in the conditions of tuples from different entities and then clone this variable (one clone per entity). By doing so, we introduce independence between tuples from different entities without changing the correlations between the tuples from the same entity.

Example 157 *For illustration, we consider the pc-database presented in Figure 7.73(a) that contains two pc-tables. Besides the exclusion to tuple t_2, tuple t_1 is correlated with tuple t_3 and is also correlated with tuple t_6. Whereas t_1 and t_3 represent different entities, t_1 and t_6 represent the same entity. For that reason, we want to remove the first dependency, but do not want to remove the second. This can be done by cloning variable X and then by replacing X with the cloned variable in the condition of the tuples of one of the two concerned entities . In our example, we select entity e_2 and replace all appearances of the variable X in the conditions of tuples representing e_2, i.e. the tuples t_3, t_4, t_7, with the cloned*

variable Z. Note that cloning means that the cloned variable Z has the same probability distribution over the same possible assignments than X. By doing so the input pc-database from Figure 7.73(a) is transformed into the pc-database that is presented in Figure 7.73(b).

It is obvious that after removing entity dependencies the complex conditions of some tuples can be simplified and some of the resultant variables can be combined or their sets of possible assignments can be reduced. For instance, in the previous example, the six combinations of the assignments of the two variables Y and Z can be combined to a single variable with four possible assignments: One for $Z = 1 \wedge Y = 1$, one for $Z = 2 \wedge Y = 2$, one for $Z = 3 \wedge Y = 2$, and one for the remaining cases. Such a simplification, however, is out of the scope of this thesis.

7.7.1.2. Verifying the Correctness of Entity Dependencies

The correctness of an entity dependency usually cannot be verified by the dependency itself, but depends on the dependency's origin. Therefore, the verification whether or not a dependency is correct can usually only made with additional domain knowledge or meta data available. In Section 8.4.1, we will present a kind of entity dependency that is modeled in the database by using additional meta data that is called the *Merge Base*. In that case, the reason of the dependency can be automatically extracted from the data and verification can be performed automatically as well. In general, useful meta data can be of different natures. For instance, it can be a log file that stores the history of the performed database transaction so that we can automatically detect what dependency was initiated by which data operation and hence can automatically made a judgment on the correctness (or trustability) of each dependency. If such meta data is not available, verification need to be made manually by domain experts or all dependencies are simply rated as incorrect or less-trustable. Whereas from incorrect dependencies no matching activity is derived, from less-trustable dependencies such activities are derived and associated with a low confidence score so that the duplicate detection process can consider them as possible indications rather than certain evidences.

7.7.1.3. Deriving Matching Activities from Dependencies

Before we can discuss how matching activities can be derived from the extracted dependencies, we have to point out what types of matching activities are meaningful to a duplicate detection process. Principally, a matching activity can concern more than two entities. Since we consider duplicate detection processes that are based on pairwise comparisons of database entities, we primarily focus on pairwise activities in this thesis. Meaningful types of matching activities are:

- **Matching Class Indication:** Activities of the first type directly classify an entity pair as a MATCH or an UNMATCH. Since the trustability of a dependency is often restricted or a dependency is rather an indication that two entities are a MATCH or an UNMATCH than a certain evidence, we generalize activities of that type by adding confidence scores to the matching class. Thus, an activity of this type is a triple $(\{e_r, e_s\}, \mathit{Class}, \mathit{Confidence})$ where $\mathit{Confidence} \in [0, 1]$ is the confidence that the entity pair $\{e_r, e_s\}$ belongs to the matching class $\mathit{Class} \in \{M, U\}$.

- **Matching Exclusion:** Matching exclusion are used to exclude an entity pair from the matching process because the two entities are not comparable, even their descriptions are conform to comparable description types. The reason for non-comparability could be the fact that the matching of both entities is already implicated by the matching of some other entities. For instance, if we consider indeterministic deduplication results as duplicate detection input, the database can contain an entity that represents a possible merge of two other entities of that database, i.e. the former represents the possible world in which the latter have been classified as duplicates by the indeterministic deduplication process. In this case, a matching of the latter two entities can be needless because their merge is already represented in the database.

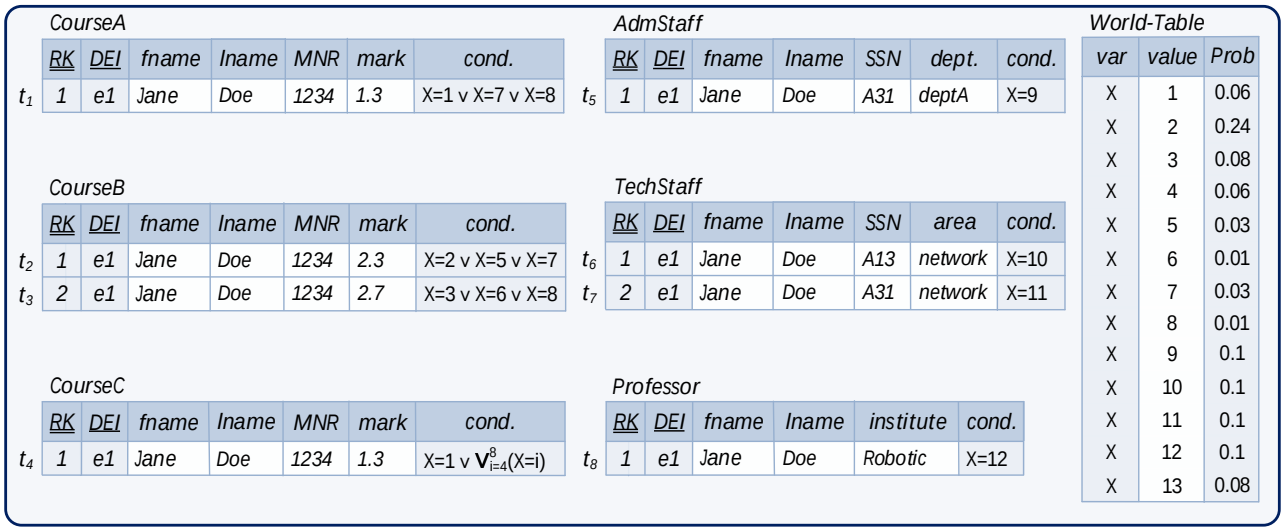
- **Matching Space Modification:** Activities of the last type do not concern the matching class, but the input of the matching process. They define a probability distribution on the coexistence of the possible instances of the considered entities and hence implicate which of the later extracted description alternatives should be matched and which not. For illustration we consider the data that result from the scientific experiment described in Example 111. In this scenario, it was only meaningful to match those x-tuple alternatives that are obtained from same conditions/configurations. As a consequence, we only match specific pairs of description alternatives instead all of them.

  If the derived modification is the same as the original entity dependency, applying this activity is semantically equivalent to consider the original dependency for generating the database's possible worlds representation. Nevertheless, the modification performed by the activity must not be equivalent to the affection of the dependency it is derived from.

As correctness, the most meaningful matching activity of an entity dependency strongly depends on the dependency's origin and can differ from case to case. Thus, meta data or domain knowledge is usually required for deriving activities. For that reason, it is not possible to propose a general approach for deriving matching activities.

7.7.1.4. Incorporation of Matching Activities into the Detection Process

The way a matching activity has to be incorporated into the detection process depends on the type of activity and the used approach for duplicate detection. Matching exclusions are used to prune entity pairs from the candidate pair space. Using HaDDeF, matching class indication can be incorporated into the detection process by utilizing the belief map. The incorporation of matching space modifications differs from one duplicate detection approach to the other. In the world-based approach, we incorporate them into the process of world selection or consider them in computing the world impact vector. Note, an incorporation into the world selection is equivalent to consider the original dependencies in possible world construction, if the derived matching activity is equal to the affection that is implicated by the corresponding dependency. Thus, by choosing this way of incorporation, it can be useful to left some dependencies unchanged in the prepared database. In contrast, in the description-based approach matching space modifications are incorporated into the alternative pair space construction by explicitly defining a joint probability distribution on the possible alternative description pairs . For instance, we already present such a matching space modifications in Example 131 where we restrict the naive



CourseA

	RK	DEI	fname	lname	MNR	mark	cond.
t_1	1	e1	Jane	Doe	1234	1.3	X=1 v X=7 v X=8

AdmStaff

	RK	DEI	fname	lname	SSN	dept.	cond.
t_5	1	e1	Jane	Doe	A31	deptA	X=9

CourseB

	RK	DEI	fname	lname	MNR	mark	cond.
t_2	1	e1	Jane	Doe	1234	2.3	X=2 v X=5 v X=7
t_3	2	e1	Jane	Doe	1234	2.7	X=3 v X=6 v X=8

TechStaff

	RK	DEI	fname	lname	SSN	area	cond.
t_6	1	e1	Jane	Doe	A13	network	X=10
t_7	2	e1	Jane	Doe	A31	network	X=11

CourseC

	RK	DEI	fname	lname	MNR	mark	cond.
t_4	1	e1	Jane	Doe	1234	1.3	X=1 v $\mathbf{V}_{i=4}^{8}$(X=i)

Professor

	RK	DEI	fname	lname	institute	cond.
t_8	1	e1	Jane	Doe	Robotic	X=12

World-Table

var	value	Prob
X	1	0.06
X	2	0.24
X	3	0.08
X	4	0.06
X	5	0.03
X	6	0.01
X	7	0.03
X	8	0.01
X	9	0.1
X	10	0.1
X	11	0.1
X	12	0.1
X	13	0.08

Figure 7.74.: Complex pc-database with incorrect ARM-dependencies

alternative pair space from six alternative description pairs to three alternative description pairs because of the given dependency that either both considered persons live in Paris or none of them.

7.7.2. Resolving Incorrect ARM-Dependencies

As we have shown in Section 7.1, incorrect ARM-dependencies can change a duplicate detection result to the worse. For that reason, it can be useful to remove such dependencies from the data before starting the actual duplicate detection process. The main reason for incorrect ARM-dependencies is a missing modeling of inheritance hierarchies within the relational database schema due to poor schema designs, simple and less powerful representation systems, or because the considered data has been integrated from different sources.

In this section, we propose a data preparation activity that removes incorrect ARM-dependencies by transforming the input database into a database that models all the present inheritance hierarchies that have been missed in the input database. For illustration reasons, we use the Vertical Partitioning Approach to model inheritance within the relational data model. Since this approach requires a modeling of complex correlations between the tuples of one database entity, we present a transformation approach that produces a pc-database as output. Because such a transformation requires knowledge on the *is-a*-relationships between the entity tables and requires the knowledge to which entity table an attribute actually belongs to, this step of data preparation is only optional.

Recall, each probabilistic representation system can be transformed into a pc-database. Moreover, the transformed database is only used for detection purposes. As a consequence, the presented approach is adoptable for all other representation systems.

7.7.2.1. Schema Transformation and Data Exportation

Transforming an already modeled inheritance hierarchy into the Vertical Partitioning Approach depends on the originally used modeling approach. For that reason, we restrict our consideration to the case where a modeling of inheritance is completely missing in the input database. In the first transformation step all

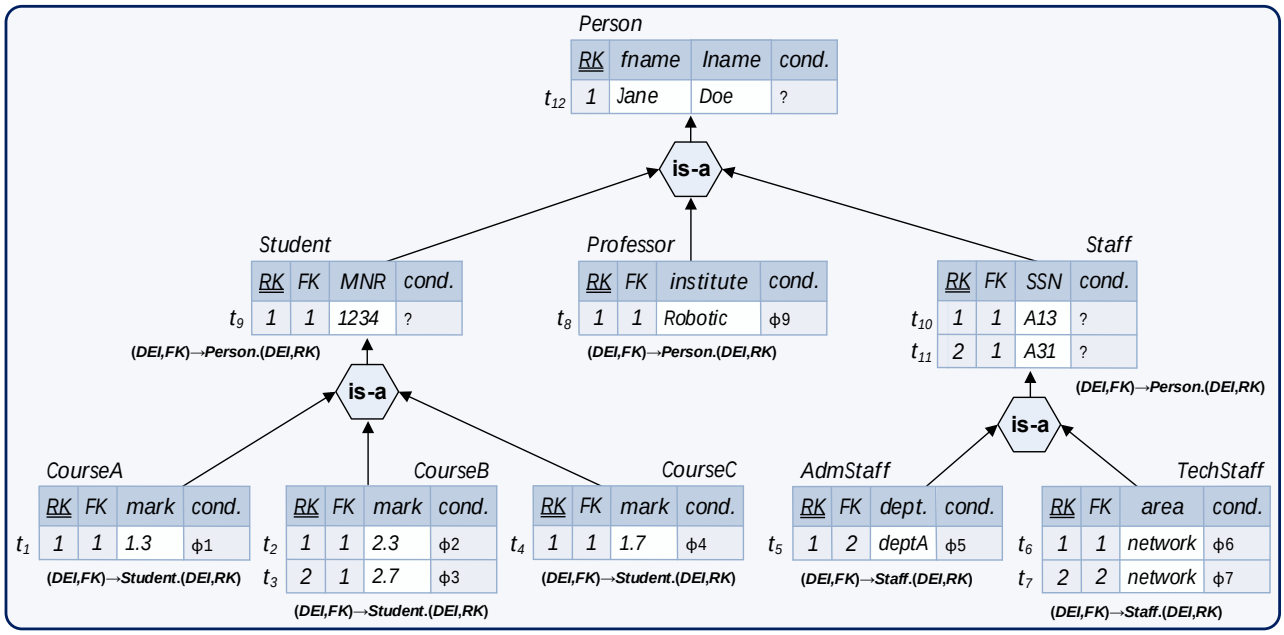


Figure 7.75.: Transformed pc-database with correct ARM-dependencies

entity tables need to be grouped by their semantical resemblance. For each group we detect the set of attributes and the set of relationship roles that are shared by these tables and therefore incorrectly depend on the memberships to these tables. Then we create a new entity table containing the shared attributes as well as the shared relationship roles, remove them from the original tables and connect the hold tables to the new table by the use of foreign keys. Since semantical resemblance can be nested, the newly created tables serve as input for another grouping step. The process finishes if no new table has been created. After transforming the schema, the instance data need to be exported from the old to the new schema. Exportation is done by generating new database tuples. Exact algorithm for the transformation process as well as the exportation process depend on the system the input database is represented in. In our example, we consider a pc-database and hence create new pc-tables (schema transformation) and create a new set of condition annotated tuples (data exportation). After exporting the data, the conditions of the newly created tuples need to be computed. Due to ambiguous interpretations of membership uncertainty (see Section 7.1.3.4) this computation is often not straightforward. For that reason, we consider it separately in Section 7.7.2.2.

Example 158 *We start with an example of schema transformation. For ease of presentation but without loss of generality we do not consider relationship roles in this example. We consider the pc-database that is depicted in Figure 7.74. At the moment, we ignore the instance of this database and only consider its schema. This database has the six entity tables* 'CourseA', 'CourseB', 'CourseC', 'TechStaff', 'AdmStaff', *and* 'Professor'. *Whereas the tables* 'CourseA', 'CourseB', 'CourseC' *share some attributes that are specific for persons and students, the tables* 'TechStaff' *and* 'AdmStaff' *share some attributes that are specific for persons and staff members. Moreover, the table* 'Professor' *has some attributes that are specific for persons. Using the attributes shared by the three course tables, we can create the table* 'Student' *having the attributes* 'fname', 'lname', *and* 'MNR' *(matriculation number). Note that the all three tables also own the attribute* 'mark', *but this attribute is specifically defined for each course, so that they*

are not semantically equivalent (of course, we require context information to capture these differences). Moreover, we can use the shared attributes of 'TechStaff' *and* 'AdmStaff' *to create a new table* 'Staff' *having the attributes* 'fname', 'lname', *and* 'SSN' *(social security number). Then, considering the attributes shared by the newly created tables* 'Student' *and* 'Staff' *as well as the original table* 'Professor', *we can create the new table* 'Person' *having the two attributes* 'fname' *and* 'lname'. *Since the new table* 'Person' *does not share any attribute with any other table, the process of schema transformation stops. Due to we use the Vertical Partitioning Approach, we need to remove attributes from some of the formerly created tables every time we create a new table, e.g. the attributes* 'fname' *and* 'lname' *need to be removed from table* 'Student' *after creating table* 'Person'. *Moreover, foreign key constraints need to be added. For that purpose we introduce the attribute* 'FK'. *Since the input database can contain complex correlations between the attribute values of each tuple and because these values can belong to different tuples in the transformed database, the foreign keys need to reference to the representation key of the corresponding entity table. The resultant schema is presented in Figure 7.75. For space reasons, we omit the attribute DEI in each table (note, all tuples of the presented database instance belong to the same entity e_1 and therefore have the same DEI). As we can see, in the transformed schema each attribute belongs to exact one entity table. Moreover, no attribute has a semantics that is more general than the semantics of the entity table it belongs to. Thus, an instance of the transformed database cannot contain incorrect ARM-dependencies.*

In the second step, we export the data. For that purpose, we shift the attribute values from the old tables to the new tables and set the foreign key references.

Example 159 *For illustration we consider the input database instance from Figure 7.74 and its corresponding output instance presented in Figure 7.75. For ease of presentation, we consider the database instance to only consists tuples that represent the same database entity e_1. Nevertheless, in order to present the nuances of data exportation and condition computation we define these tuples to have complex correlations. In the considered example, the two tables* 'AdmStaff' *and* 'TechStaff' *are membership exclusive. Therefore the existence of tuple t_5 implicates the non-existence of the tuples t_7 and t_8. Besides correlations that are implicated by the semantics of the corresponding tables, entity-specific correlations can exist as well. For instance, e_1 is only represented by a tuple in table* 'CourseA' *if it is represented by a tuple in table* 'CourseC' *although both tables are membership independent to each other. Thus, the existence of tuple t_1 implicates the existence of tuple t_4. In the considered example, we consider a single variable X. Thus, we have an 1:1 mapping between the possible instances of e_1 and the assignments of X. For a better understanding, Table 7.2 describes the possible instances of e_1 that are implicated by the individual assignments of X. It is important to note that e_1 maybe does not belong to any of the database tables with a probability of 0.08 (assignment $X \mapsto 13$).*

Now, we consider the exportation process. The four course tuples in the input instance share the same matriculation number, i.e. 'MNR' $= 1234$. *As a consequence, in the instance of the output database we require only a single tuple in table* 'Student' *that is referenced by all four modified course tuples. In contrast, the three tuples of the tables* 'AdmStaff' *and* 'TechStaff' *in the input instance have two different values in the attribute* 'SSN', *i.e. t_5 and t_7 have value 'A31' and t_6 has value 'A13'. For that reason, we need to create two tuples in table* 'Staff' *in the output instance. Whereas the modified tuple t_6*

Variable Assignment	Description
$X \mapsto 1$	e_1 belongs to *'CourseA'* and *'CourseC'*
$X \mapsto 2$	e_1 belongs to *'CourseB'* only and has the mark '2.3'
$X \mapsto 3$	e_1 belongs to *'CourseB'* only and has the mark '2.7'
$X \mapsto 4$	e_1 belongs to *'CourseC'* only
$X \mapsto 5$	e_1 belongs to *'CourseB'* (mark '2.3') and *'CourseC'*
$X \mapsto 6$	e_1 belongs to *'CourseB'* (mark '2.7') and *'CourseC'*
$X \mapsto 7$	e_1 belongs to *'CourseB'* (mark '2.3'), *'CourseA'* and *'CourseC'*
$X \mapsto 8$	e_1 belongs to *'CourseB'* (mark '2.7'), *'CourseA'* and *'CourseC'*
$X \mapsto 9$	e_1 belongs to *'AdmStaff'* and has the SSN 'A31'
$X \mapsto 10$	e_1 belongs to *'TechStaff'* and has the SSN 'A13'
$X \mapsto 11$	e_1 belongs to *'TechStaff'* and has the SSN 'A31'
$X \mapsto 12$	e_1 belongs to *'Professor'*
$X \mapsto 13$	e_1 belongs to none of the tables

Table 7.2.: Descriptions of the individual assignments of variable X in the sample database from Figure 7.74

references to the first, the modified tuples t_5 and t_7 references to the second. Since all input tuples share the same first name and the same last name, we only need to create a single tuple in table 'Person' *and introduce a foreign key reference to this tuple in the tuples t_8, t_9, t_{10}, and t_{11}.*

As we illustrate by the question marks in Figure 7.75, the conditions of the newly created tuples are not clear. Therefore, the main challenge of data exportation is the computation of the conditions that need to be assigned to the newly created tuples. We will consider a solution to that challenge in the following section.

7.7.2.2. Computing Tuple Conditions

After transforming the schema and exporting the attribute data, the conditions of the tuples in the new tables need to be computed. In situations of ambiguous interpretations of membership uncertainty this computation is not predetermined. Nevertheless, each new tuple condition can be restricted by the conditions of the input tuples.

Before we can compute the new tuple conditions we additionally need context information, namely the membership dependencies between the involved entity tables. Since inclusion is already modeled by the concept of inheritance itself, we restrict our consideration to membership exclusion. Then, we form a tree (called *Reference Tree*) per database entity where the set of nodes corresponds to the set of all output tuples that represent this entity. A tuple is a child node of another tuple if the first tuple's value in the foreign key attribute *'FK'* references to the representation key value of the latter. Two sibling nodes are exclusive, i.e. they cannot coexist in the same world, if they belong to the same entity table or if they belong to entity tables that are membership exclusive. Moreover, the child nodes of a tuple t can form a total specialization, i.e. at least one of these tuples must exists in every possible world that contains t. Since t only exists if one of its child nodes exists as well, we denote such nodes to be abstract.

Starting from the reference tree, the new tuple conditions are computed in three steps. First we compute lower bounds (the minimal required conditions) bottom up, then we compute upper bounds (the maximal possible conditions) top down, and finally select a condition for each tuple based on these two bounds. Since the conditions of the tuples that belong to the input database are already given, the minimal required conditions and the maximal possible conditions of these tuples are predefined and equivalent to each other.

1. **Lower Bound Computation:** In the first step, we compute the minimal required conditions (short Φ^{MinRC}) for each non-leaf node bottom up. Since a tuple references to its parent node, it can only exist if the parent node exist (otherwise referential integrity is violated). Thus, the condition of a non-leaf node N must cover all the conditions of its child nodes, i.e. the condition of N must be satisfied by every variable assignment that satisfies the condition of any of its child nodes. As a consequence, the minimal required condition of node N results in the disjunction of the minimal required conditions of its child nodes, i.e.:

$$\Phi^{MinRC}(N) = \bigvee_{N_c \in child(N)} \Phi^{MinRC}(N_c) \tag{7.20}$$

   Note that correlations between the individual child nodes are automatically considered in the condition of N if we compute it by the disjunction operator. For instance, let N_1 and N_2 be the two only child nodes of N and let the existence of N_1 to be implicated by the existence of N_2, i.e. every variable assignment that satisfies the condition of N_2 also satisfies the condition of N_1. In that case, computing the disjunction of their conditions results in the condition of N_1 and therefore the condition of N is always satisfied if the conditions of N_1 or N_2 are satisfied.

2. **Upper Bound Computation:** In the second step, we compute the maximal possible condition (short Φ^{MaxPC}) per non-leaf node in a top down fashion. The maximal possible condition of a non-root node is influenced by the maximal possible condition of it parent node and is influenced by the minimal required conditions of its exclusive siblings because they cannot coexist.

   An exception occurs if the child nodes form a total specialization and hence if the considered node is abstract. In that case the node cannot exists if none of its child nodes exists and we have to set the maximal possible condition of that node to its the minimal required condition.

   In conclusion, let N be a non-root node that has the parent node P, its maximal possible condition is computed as (recall $N \rightleftharpoons^M N'$ means that N and N' are membership exclusive):

$$\Phi^{MaxPC}(N) = \begin{cases} \Phi^{MinRC}(N), & \text{if } N \text{ is abstract} \\ \Phi^{MaxPC}(P) \wedge \neg(\bigvee_{N' \in child(P), N \rightleftharpoons^M N'} \Phi^{MinRC}(N')), & \text{else.} \end{cases} \tag{7.21}$$

   The root node does not have a parent node nor have sibling nodes. Thus, the maximal possible condition of a root node is set to the boolean constant *true* if it is not abstract.

3. **Condition Selection:** Because we do not use interval probabilities (and hence interval conditions) we finally need to select a single condition per node. This selection is done by traversing the reference tree top down. Given the finally selected condition Φ_P of node P, the final conditions of the child nodes of P are selected in a way that satisfies the following requirements:

- The selected condition of P must cover all the selected conditions of its child nodes. Thus, it must hold that:

$$\forall N \in \mathit{child}(P)\colon \varkappa(\Phi_N) \subseteq \varkappa(\Phi_P) \tag{7.22}$$

  Recall, $\varkappa(\Phi)$ is the set of all variable assignments that satisfy the condition Φ.

- The conditions that are selected for two exclusive children of P must be contradictory, i.e. their conjunction must be a condition that can never be satisfied. Thus, it must hold that:

$$\forall N, N' \in \mathit{child}(P)\colon N \rightleftharpoons^{M} N' \Rightarrow \Phi_N \wedge \Phi_{N'} = \mathit{false} \tag{7.23}$$

- The finally selected condition of each child node N must cover its minimal required condition and must be covered by its maximal possible condition, i.e.:

$$\forall N \in \mathit{child}(P)\colon \varkappa(\Phi^{\mathit{MinRC}}(N)) \subseteq \varkappa(\Phi_N) \subseteq \varkappa(\Phi^{\mathit{MaxPC}}(N)) \tag{7.24}$$

- The finally selected condition of node P must be equal to the disjunction of the finally selected conditions of its child nodes if P is abstract, i.e.:

$$\varkappa(\bigvee\nolimits_{N \in \mathit{child}(P)} \Phi_N) = \bigcup\nolimits_{N \in \mathit{child}(P)} \varkappa(\Phi_N) = \varkappa(\Phi_P) \tag{7.25}$$

The simplest selection strategy is to select the minimal required condition for each node. Notice, if the input database is consistent with respect to the membership exclusions of the entity tables, i.e. two tuples from different membership exclusive tables that represent the same database entity are mutual exclusive, selecting the minimal required conditions for each node automatically satisfies all the aforementioned requirements.

The most suitable selection strategy, however, always depends on the considered use case. We think that the likelihood of belonging to a supertype but not belonging to a subtype is typically greater than the likelihood of not belonging to the supertype at all. For example, it is more likely that an entity that is only maybe a student is certainly a person instead of only maybe a person. For that reason, it seems logical to select the least restricted condition that is possible for the considered node if no other selecting strategy is explicitly defined. Moreover, it seems meaningful to select conditions that are fairly distributed on a set of sibling nodes if several of them are not predetermined and hence can be selected at the user's own discretion. Nevertheless, if such a selection is too computational expensive, the aforementioned simplest selection strategy can be used instead.

We will illustrate this three step approach on the transformed pc-database from Figure 7.75.

Example 160 *After schema transformation and data exportation, entity e_1 is represented by at least one tuple in all nine entity tables, but the conditions of its corresponding tuples in the tables* 'Student'*,* 'Staff'*, and* 'Person' *are missing. Note, for space reasons we abbreviate the condition of the original tuple t_i by Φ_i.*

The tables 'CourseA'*,* 'CourseB'*,* 'CourseC' *are neither membership inclusive nor membership exclusive, i.e. a student can participate at each of them independently. The tables* 'TechStaff' *and* 'AdmStaff'

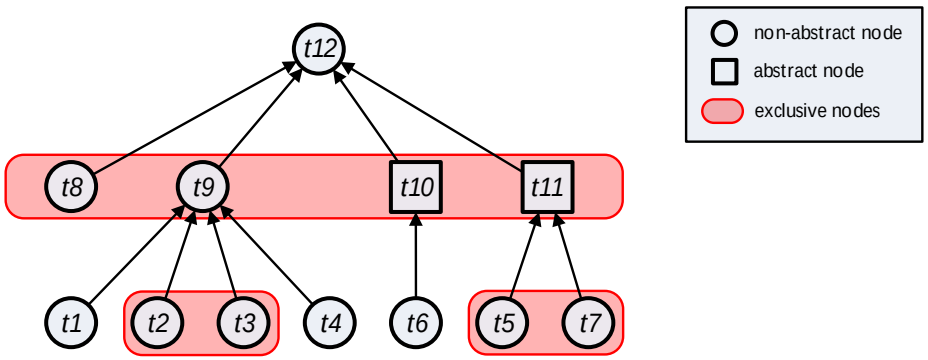


Figure 7.76.: Reference tree of the sample database instance from Figure 7.75

are membership exclusive. Moreover, a person can be either a student or a staff member or a professor and hence the three tables 'Student', 'Staff', *and* 'Professor' *are mutually membership exclusive. Each staff member is considered to be either technical or administrative and hence the two specializations of staff member are total. From the given instance and the additional information on the membership correlations of the individual tables, we can derive the reference tree as presented in Figure 7.76. Note, two nodes that are surrounded by the same red colored box are exclusive, e.g. the tuples* t_2 *and* t_3 *are exclusive because they belong to the same table, and abstract nodes are represented by rectangles instead of circles, e.g. the nodes* t_{10} *and* t_{11} *are abstract.*

In the first step, we start to compute the minimal required conditions of the non-leaf nodes bottom up. Entity e_1 *must belong to the table* 'Student' *if it belongs to any of the course tables. Thus, the minimal required condition of tuple* t_9 *is the disjunction of all conditions of the tuples in the course tables of the input database instance that share the same value in the attribute* 'MNR' *(in our case the tuples* t_1*,* t_2*,* t_3*, and* t_4*). Therefore,the minimal required condition of* t_9 *results in* $\Phi^{MinRC}(t_9) = \Phi_1 \vee \Phi_2 \vee \Phi_3 \vee \Phi_4 = \bigvee_{i=1}^{8}(X = i)$*. The minimal required condition of tuple* t_{10} *results in the condition of tuple* t_6 *and hence results in* $\Phi^{MinRC}(t_{10}) = \Phi_6 = (X = 10)$*. The minimal required condition of tuple* t_{11} *results in the disjunction of the conditions of the tuples* t_5 *and* t_7 *as therefore is computed as* $\Phi^{MinRC}(t_{11}) = \Phi_5 \vee \Phi_7 = (X = 9) \vee (X = 11)$*. Entity* e_1 *is a person (and therefore represented by tuple* t_{12}*) if she is a student, if she is a staff member, or if she is a professor. Consequently, the minimal required condition of tuple* t_{12} *results in* $\Phi^{MinRC}(t_{12}) = \Phi^{MinRC}(t_9) \vee \Phi^{MinRC}(t_{10}) \vee \Phi^{MinRC}(t_{11}) \vee \Phi_8 = \bigvee_{i=1}^{12}(X = i)$.

In the second step, we compute the maximal possible conditions of the non-leaf nodes top down. Since the set of specializations from table 'Person' *to the tables* 'Student', 'Staff', *and* 'Professor' *is not total, i.e. not every person must be a student, staff member, or professor, the maximal possible condition of tuple* t_{12} *is not restricted by an upper bound and therefore results in* $\Phi^{MaxPC}(t_{12}) = true$*. In contrast, a student cannot be a staff member and cannot be a professor. For that reason, the maximal possible condition of tuple* t_9 *is bounded by the minimal required conditions of the tuples* t_8*,* t_{10}*, and* t_{11} *and thus is computed as:* $\Phi^{MaxPC}(t_9) = \neg(\Phi_8 \vee \Phi^{MinRC}(t_{10}) \vee \Phi^{MinRC}(t_{11})) = \neg(\bigvee_{i=9}^{12}(X = i))$*. Tuple* t_{10} *can only exists if none of the tuples* t_8*,* t_9*, and* t_{11} *exist. As a consequence, the maximal possible condition of tuple* t_{10} *would actually be computed as:* $\Phi^{MaxPC}(t_{10}) = \neg(\Phi_8 \vee \Phi^{MinRC}(t_9) \vee \Phi^{MinRC}(t_{11})) = \neg(\bigvee_{i=1}^{9}(X = i) \vee (X = 11) \vee (X = 12))$*. Nevertheless, this node is only abstract (each staff member either belongs*

Node (tuple)	Minimal Required Condition	Maximal Possible Condition	Finally Selected Condition
t_9	$\bigvee_{i=1}^{4} \Phi_i$	$\neg(\bigvee_{i=5}^{8} \Phi_i)$	$\bigvee_{i=1}^{4} \Phi_i \vee (X = 14)$
t_{10}	Φ_6	Φ_6	Φ_6
t_{11}	$\Phi_5 \vee \Phi_7$	$\Phi_5 \vee \Phi_7$	$\Phi_5 \vee \Phi_7$
t_{12}	$\bigvee_{i=1}^{8} \Phi_i$	*true*	*true*

Table 7.3.: The minimal required conditions, the maximal possible conditions, and the finally selected conditions of the newly created tuples

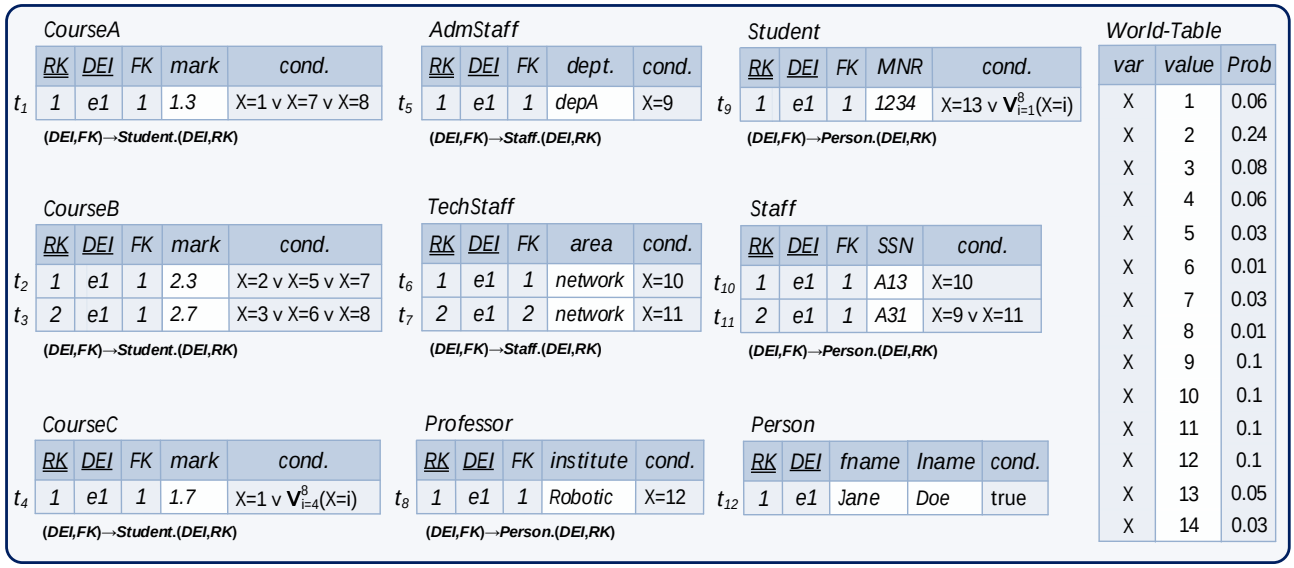


Figure 7.77.: Complete representation of the transformed pc-database without incorrect ARM-dependencies

to the administration staff or belongs to the technical staff) and the maximal possible condition of t_{10} *is therefore set to its minimal required condition. The same holds for tuple* t_{11}.

In the last step we select the final condition for each none-leaf node based on the previously computed minimal required conditions and maximal possible conditions. We start with tuple t_{12} *and select the least restricted possible condition* $\Phi_{12} = true$. *Note, by doing this the meaning of the variable assignment* $X \rightarrow 13$ *changes from* entity e_1 belongs to none of the tables *to* entity e_1 belongs to the table *'Person'* only. *The conditions of the tuples* t_{10} *and* t_{11} *are predefined as* $\Phi_{10} = \Phi_6$ *and* $\Phi_{11} = \Phi_5 \vee \Phi_7$ *because the maximal possible conditions and the minimal required conditions of these tuples are equivalent. Finally, we have to select a condition for tuple* t_9. *Since we select the minimal required conditions for the tuples* t_{10} *and* t_{11} *and since we select the maximal possible condition for tuple* t_{12}, *for tuple* t_9 *we can select any condition that covers the tuple's minimal required condition and that is covered by the tuple's maximal possible condition. For illustration purposes, we assume that entity* e_1 *is a student that does not participate at any course with probability* 0.05. *This assumption can be realized by splitting the variable assignment* $X \rightarrow 13$ *(*entity e_1 belongs to the table *'Person'* only*) into the two assignments* $X \rightarrow 13$ *(*entity e_1 belongs to the table *'Person'* only*) with probability* 0.03 *and* $X \rightarrow 14$ *(*entity e_1 belongs to the tables *'Person'* and *'Student', i.e.* e_1 *is a student that does not participate at any course) with probability* 0.05 *and then select the condition of tuple* t_9 *as* $\Phi_9 = \Phi^{MinRC}(t_9) \vee (X = 14) = \bigvee_{i=1}^{8}(X = i) \vee (X =$

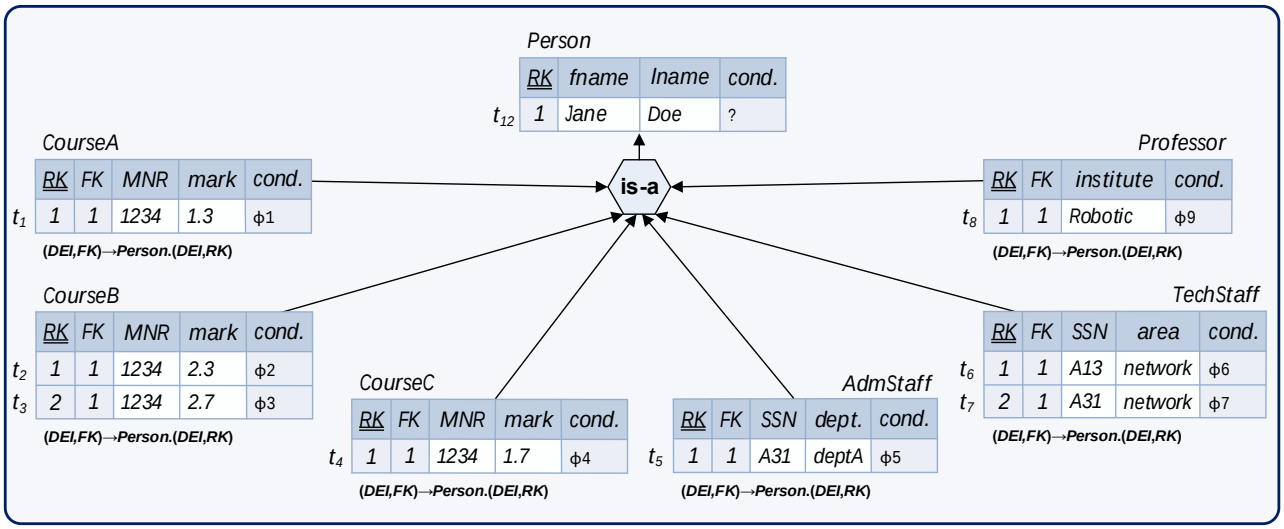


Figure 7.78.: Output database with still existing, but less important incorrect ARM-dependencies

14). *Table 7.3 gives an overview on the minimal required conditions, the maximal possible conditions, and the finally selected conditions of the individual tuples. The output database with the selected tuple conditions and the extended world table is presented in Figure 7.77.*

7.7.2.3. Cost-Benefit Trade Off

In the previous part of this section, we propose an exact approach for removing incorrect ARM-dependencies by transforming the input database into a pc-database. On one hand, the transformation approach solves any kind of incorrect ARM-dependencies and hence could massively improve the effectiveness of the subsequently applied duplicate detection process. On the other hand, this transformation can become computational expensive if some of the newly modeled inheritance hierarchies consist of many tables.

Nevertheless, a removal of incorrect ARM-dependencies is only for detection purposes and an exact removal of all of these dependencies is thus not mandatory. In contrast, preparation is always a trade-off between costs and benefits. The costs of the transformation approach can be reduced in several ways. For instance, in many cases it is already useful to only resolve some of the incorrect ARM-dependencies instead all of them.

Example 161 *As an example we reconsider the initial sample schema from Figure 7.74. A resolution of the incorrect dependencies of the attributes* fname *and* lname *would result in the schema that is presented in Figure 7.78 (note that we again omit the attribute DEI from the presentation for space reasons). This solution is much easier to achieve than the solution presented in Figure 7.75. The loss of computation complexity especially concerns the step of computing the conditions of the newly created tuples. If we set the conditions of all tuples in the table* 'Person' *to true we do not need to compute a single condition. Even if we use a strategy that is more conservative than selecting the maximal possible condition, we only need to compute the minimal required conditions, because the maximal possible conditions of the root node is predefined as true and we do not have any other new table except the root node. Moreover,*

although we resolve only some of the incorrect ARM-dependencies, the solution can improve detection effectiveness already a lot, especially in a duplicate detection processes that considers the attributes MNR *and* SSN *to be less useful for detecting real-world equivalence.*

7.7.2.4. Inconsistent Preparation Results

A consistent removal of incorrect ARM-dependencies becomes tricky, if the membership to a subtype is correlated with the values in some attributes of the newly created supertype.

Example 162 *For illustration we consider an entity table* 'Senior' *having among others an attribute* 'age' *and assume that this table is intended to contain only persons older than sixty years. Moreover, we assume an entity* e_1 *that is represented in this table by a an x-tuple with a single alternative that has the value* 65 *in the attribute* 'age' *and has the probability* 0.7*. Since the attribute* 'age' *is not exclusive for seniors, but is applicable to persons in general, we generate a table* 'Person' *and shift the attribute* 'age' *from* 'Senior' *to* 'Person'*. If we now select the condition true for the single x-tuple alternative that represent entity* e_1 *in* 'Person'*, we introduce an inconsistency into the database, because if this entity is certainly a person of age* 65 *it would be certainly a senior. In contrast, in the case where this entity is a person but not a senior her age must be less than sixty years. The problem is that we do not know the age for this case. Moreover, simply saying that the probability that* e_1 *belongs to* 'Person' *is the same as the probability that* e_1 *belongs to* 'Senior' *is not a useful solution because we actually do not remove the incorrect ARM-dependency between* 'age' *and* 'Senior' *and the preparation activity becomes effectless.*

The most intuitive solution to this problem is to add an alternative to the x-tuple representing e_1 *in table* 'Person' *so that* e_1 *is represented by two x-tuples alternatives: one with the age* 65 *and a probability of* 0.7 *and one having a null value in the attribute* 'age' *and having a probability of* 0.3*. Nevertheless, this solution does not produce a consistent database either because the null value implies all ages to be possible, including the ages above* 60*.*

A correct solution to this problem can often only constructed by explicitly enumerating all the possible values that do not violate the membership to the subtype, e.g. by adding one alternative per possible age under 60 years to the x-tuple in table *'Person'*. Such an approach, however, can be complex in computation effort and, even worse, can lead to an explosion of the database's size. Nevertheless, as already mentioned before, the preparation activity is only completed for the purpose of duplicate detection and duplicate detection is heuristic by nature. For that reason, it is not critical if the prepared database contains small inconsistencies.

7.8. Incorporation of Process Uncertainty

Until now, we only consider uncertainty of the source data, but do not consider uncertainty of the duplicate detection process itself. Nevertheless, as already mentioned in this thesis duplicate detection is heuristic by nature and the detection quality that results from using a specific detection process essentially depends on the domain of the input data. Since there is no process configuration that universally works suitable for duplicate detection in every possible application scenario and it is even not clearly defined what duplicate detection quality exactly means (see several measures for quality in Section 4.3.8),

a large number of approaches has been developed for each of the individual detection phases. Moreover, the most of these techniques have several parameters that can be configured in multiple ways. As a consequence, it seems logical to incorporate process uncertainty into the detection process.

This section is structured as follows. First we point out some sources of uncertainty in a duplicate detection process and shortly describe in which certain duplicate detection approach handle these uncertainties. Then, we present our strategy to incorporate process uncertainty into the description-based detection approach. Finally, we consider the affect of this incorporation on the different strategies for matching cost optimization that we have introduced in Section 7.6.

7.8.1. Sources of Uncertainty in Duplicate Detection

We start with a short discussion on the different sources of uncertainty that can typically emerge in a duplicate detection process. The moments of the individual sources within the detection process[12] are graphically presented in Figure 7.79.

- **Preparation Uncertainty:** The first source of uncertainty in the detection process is the preparation phase. As described in Section 4.3.3, the purpose of preparation is to eliminate simple errors and to standardize the data. In both kind of activities it is often not clear what solution fits best.

  Example 163 *For illustration we consider a preparation activity that should resolve abbreviations in a universal way. Sometimes, however, it is not completely clear what an abbreviation means and more than one preparation are plausible. Let us consider a table about American Cities that has an attribute* 'state'. *The values in that attribute are not standardized so that sometimes the whole state name is written, e.g. 'California', and sometimes only an abbreviation is used, e.g. 'CAL' or 'CA'. Now let us assume that one database entity has the abbreviation 'TE' in that attribute. Due to 'TE' could refer to the state Texas, but could also refer to the state Tennessee, we are not sure how to prepare it. In certain data duplicate detection only one of the two preparations can be utilized (maybe this one that is most likely in combination with the city name). Nevertheless, the correct preparation can be still unknown and deciding for one of them automatically implies to accept the risk of introducing an error into the data.*

- **Feature Matching Uncertainty:** In the feature matching phase, the similarities between two attribute values and the similarities between two related entities sets are measured by any mean of resemblance. The problem here is that there are not only several meanings of similarity (syntactical, semantical, phonetic, etc.), but also a large number of different measures have been developed for each of them (see Section 4.3.5). Thus, the general meaning of similarity is not completely clear and can become even less clear when we consider a specific case of given source data for which the suitability of similarity measures has not been studied so far.

  To overcome this problem, in conventional detection approaches a variety of similarity measures (and hence multiple meanings of similarity) is used and the resultant similarity scores are finally

[12] Recall, that it make no sense to handle uncertainty in the phase of candidate pair space construction, because the pair space is only reduced for efficiency reasons and 'maybe' pairing two entities is not meaningful.

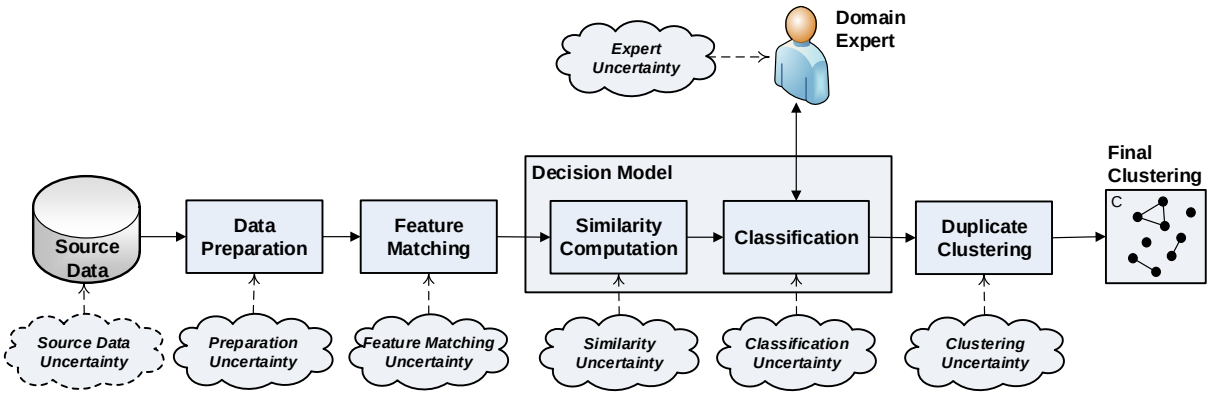


Figure 7.79.: The phases of a conventional duplicate detection process along with the identified sources of uncertainty

aggregated to a single score by computing the average, minimum, or maximum. Thus, uncertainty on the best feature matching configuration is indeed considered, but is resolved immediately at the end of the phase.

- **Similarity Uncertainty:** As we have presented in Section 4.3.6, a score of description similarity can be computed from the feature score in several ways. One way is to use a distance function that considers the comparison vectors of the feature score as points in a Cartesian System. Another way is to use a learned classifier as a Support Vector Machine or a Decision Tree. The suitability of the individual computation methods for the considered use case, however, is often not known. Moreover, learning-based approach require the availability of training data, but it is often not known how good the training data represents the complete database.

  As in the feature matching phase, conventional duplicate detection approaches for certain data resolve uncertainty on similarity computation instantly by aggregating the similarity scores that has been computed by different computation methods.

- **Classification Uncertainty:** After computing a single similarity score per entity pair, one or two thresholds are used to classify this pair as a MATCH, a POSSIBLE MATCH, or an UNMATCH. To restrict the number of POSSIBLE MATCHES (and hence to restrict the number of clerical reviews) to only a few pairs, both thresholds should be closed to each other. On the other hand, the lower the threshold $\theta_{P/M}$ that demarcates the sets of POSSIBLE MATCHES and MATCHES the higher is the risk of producing false positives and the greater the threshold $\theta_{U/P}$ that demarcates the sets of UNMATCHES and POSSIBLE MATCHES the higher is the risk of producing false negatives. Selecting some suitable thresholds is consequently not straightforward and a bad selection can lead to poor detection results. This problematic is graphically illustrated in Figure 7.80.

- **Clustering Uncertainty:** The purpose of the duplicate clustering phase is to build a clustering of all database entities based on their pairwise duplicate decisions. Nevertheless, the pairwise duplicate decision have been computed independently and can contradict each other. What clustering

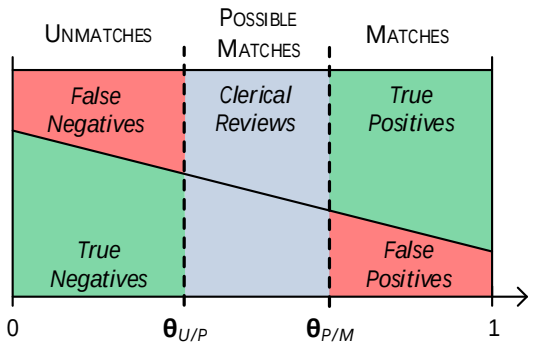


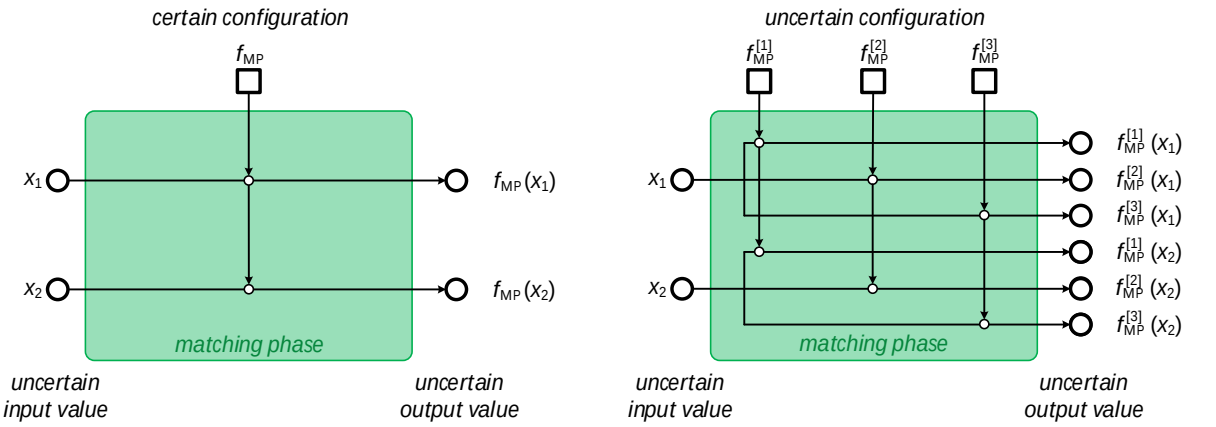
Figure 7.80.: Trade-Off between number of false positives, number of false negatives, and number of clerical reviews

approach solves such conflicts best, however, depends on the considered use case, because sometimes false positives are worse than false negatives or vice versa. Moreover, the influence of a clustering approach on the final detection quality is sometimes hard to predict. Finally, not only several approaches for duplicate clustering exist (see Section 4.3.7), but some of these approaches resolve ambiguous situations, e.g. the selection of center nodes, by taking one of the possible solutions randomly. Thus, these approaches already bear uncertainty by themselves.

In general, the impact of the used clustering approach on the quality of the finally produced duplicate detection result is often underrated. However, experiments show that the differences in the number of proposed MATCHES and the number of proposed UNMATCHES can considerably differ from one clustering approach to another. For instance, Partitioning based on Connected Components produces large clusters and thus typically produces much more false positives than false negatives. In contrast, Partitioning based on Centers produces small clusters and therefore usually produces much more false negatives than false positives. Other approaches show less significant differences in favoring false negatives to false positives or vice versa. In conclusion, it is important to select a clustering approach that is appropriate for the considered use case. For that purpose, however, the user need to know the ratio between the number of false negatives and the number of false positives that is typically produced by the individual clustering approaches.

Intuitively, this property does not depend on the domain of the input data and it should be easy to study it experimentally for all available clustering approaches once and then to use this knowledge in subsequent use cases. Nevertheless, this property strongly depends on the characteristics of the processed duplicate decisions, e.g. the structure of the duplicate-pair graph, and these characteristics often depend on the characteristics of the considered input data. Thus, the clustering property indirectly depends on the domain of the input data and an experimental study that produces universally valid results is actually not possible.

- **Expert Uncertainty:** After classification domains experts are usually responsible to review the POSSIBLE MATCHES by hand. A domain expert, however, is usually not an all-knowing being and sometimes it is hard to decide whether or not a POSSIBLE MATCH should be changed into a MATCH or should be changed into an UNMATCH even if the reviewer possesses a lot of domain knowledge.



(a) Certain matching phase configuration

(b) Uncertain matching phase configuration

Figure 7.81.: Differences in using certain matching phase configurations and using uncertain matching phase configurations

As presented above, a duplicate detection process contains different sources of uncertainty. In conventional duplicate detection approaches uncertainty is eagerly resolved in the corresponding phase. If the detection process, however, can handle probabilistic intermediate result, such an eager resolution is not necessary and uncertainty can be resolved during a later phase of the detection process. This in turn can improve detection quality in many scenarios.

7.8.2. Incorporation Approach

Uncertainty on a matching phase can be incorporated into the detection process by performing the phase repeatedly with some of its plausible configurations. The idea behind this way of incorporation is that several less probable configurations of a matching phase can produce values that lead to same consequences of the detection process and therefore maybe represents the matching result better than the value produced by the most probable configuration. Since different interim results can lead to same consequences, e.g. different feature scores can lead to the same similarity score, the most probable consequence of the whole detection process can be different from the consequence that result from selecting the most probable interim matching result at the end of each the detection phases as deterministic duplicate detection approaches typically do. Thus, the later uncertainty is resolved, the closer is the computed consequence to the most probable consequence.

Formally, we consider each matching phase as a function that maps a value of its input domain to a value of its output domain. As a consequence, incorporate process uncertainty into the detection process corresponds to interpreting each matching phase as an uncertain function according to Definition 53 (Incomplete Function) or according to Definition 54 (Probabilistic Function). The difference to the certain approach is presented in Figure 7.81. If a matching phase is executed with a single configuration, each alternative of the probabilistic input value is processed by the matching phase once and if no two input alternatives are mapped to the same output, the number of output alternatives is equivalent to the number of input alternatives. This procedure is presented in Figure 7.81(a) with two input alternatives and two output alternatives. In contrast, if a matching phase is associated with an uncertain configuration,

each input alternative is not processed once per matching phase, but is processed once per configuration alternative. Thus, as illustrated in Figure 7.81(b) by using three possible configurations of the feature matching phase from processing the two input alternatives six output alternatives result, because each input alternative is processed once by using each of the different configuration alternatives.

Example 164 *For illustration we reconsider the abbreviation 'TE' that should be prepared to the full written name of a state. Since the correct state is either 'Texas' or 'Tennessee', the preparation activity is considered as an uncertain mapping with the two alternatives 'TE'→'Texas' and 'TE'→'Tennessee'.*

Now let us assume that we assign a probability to each of these alternatives, e.g. the first is associated with the probability 0.6 *and the second is associated with the probability* 0.4*. Moreover, let us assume that the value 'TE' belongs to an alternative of a probabilistic attribute description that has the probability* 0.2*. Then the preparation activity applied to that alternative produces two alternatives where the first contains the value 'Texas' and has the probability* $0.2 \times 0.6 = 0.12$ *and the second alternative contains the value 'Tennessee' and has the probability* $0.2 \times 0.4 = 0.08$*.*

As mentioned above, each matching phase can be considered as a probabilistic function that has one alternative per used configuration. Therefore, let $f_{FMP}^{[\rho]} = (f_{FMP}^{[\iota]}, Pr_{FMP})$ be a probabilistic configuration of the feature matching phase and let $APS(\mathfrak{d}_r, \mathfrak{d}_s) = (W_{APS}, Pr_{APS})$ be an alternative pair space, according to Definition 54 the result of applying $f_{FMP}^{[\rho]}$ to $APS(\mathfrak{d}_r, \mathfrak{d}_s)$ is the probabilistic feature score *fs* that is defined as:

$$
\begin{aligned}
& fs = (W_{fs}, Pr_{fs}) \\
\text{where} \quad & W_{fs} = \{f_{FMP}(d_r, d_s) \mid (d_r, d_s) \in W_{APS}, f_{FMP} \in f_{FMP}^{[\iota]})\} \\
\text{and} \quad & \forall x \in W_{fs} \colon Pr_{fs}(x) = \sum\nolimits_{f_{FMP} \in f_{FMP}^{[\iota]}} \sum\nolimits_{(d_r, d_s) \in W_{APS}} Pr_{FMP}(f_{FMP}) \times Pr_{APS}(d_r, d_s)
\end{aligned}
$$

In the above described approach, the probability of each output alternative is computed by multiplying the probability of the input alternative and the probability of the configuration alternative that lead to that output alternative. Nevertheless, it is theoretically also possible to introduce correlations between the processed input alternatives and the used configuration alternatives. For instance, it could be useful to condition the feature score computation on some characteristics of the considered alternative description pair. Therefore, it can be useful to define a joint probability distribution on the alternatives of the input value and the matching phase configuration. Nevertheless, in this thesis, we restrict our examples to the case of independence.

7.8.3. Uncertainty Resolution / Uncertainty Reduction

The advantage of this incorporation approach is that each matching phase still produces a probabilistic value of its output domain. Thus, every reduction methods and every resolution methods that does not require information on matching result provenance, i.e. the method does not need to know which output alternatives result from which combination of alternative description pair and phase configurations, can be applied as proposed in the previous sections of this thesis. For instance, if we want to resolve the uncertainty of a probabilistic similarity score by computing the expected value, we do not need to know

alt. pair	feature matching	similarity computation	*probability*	*similarity score*
x	***α***	*r*	*0.6×0.5×0.8=0.24*	*sim1*
x	***α***	*s*	*0.6×0.5×0.2=0.06*	*sim2*
x	***β***	*r*	*0.6×0.4×0.8=0.192*	*sim2*
x	***β***	*s*	*0.6×0.4×0.2=0.048*	*sim3*
x	***φ***	*r*	*0.6×0.1×0.8=0.048*	*sim4*
x	***φ***	*s*	*0.6×0.1×0.2=0.012*	*sim1*
y	***α***	*r*	*0.4×0.5×0.8=0.16*	*sim5*
y	***α***	*s*	*0.4×0.5×0.2=0.04*	*sim6*
y	***β***	*r*	*0.4×0.4×0.8=0.128*	*sim1*
y	***β***	*s*	*0.4×0.4×0.2=0.032*	*sim2*
y	***φ***	*r*	*0.4×0.1×0.8=0.032*	*sim7*
y	***φ***	*s*	*0.4×0.1×0.2=0.008*	*sim5*

(a) Provenance table with three dimensions

alt.	(*sim*,*imp*)	*Pr*
sim1	*(0.3,1.0)*	0.048
sim2	*(0.4,1.0)*	0.38
sim3	*(0.5,1.0)*	0.284
sim4	*(0.6,1.0)*	0.048
sim5	*(0.7,1.0)*	0.032
sim6	*(0.8,1.0)*	0.168
sim7	*(0.9,1.0)*	0.04

(b) Probabilistic similarity score

Figure 7.82.: A sample provenance table with three dimensions and its corresponding probabilistic similarity score

the provenance of the individual similarity score alternatives, but can compute it directly on the score itself.

In contrast, if we want to use a resolution method or a reduction method that requires information on matching result provenance or if we do not want to resolve / reduce all dimensions at once, we need to maintain a provenance table as introduced in Section 7.5.4.4. The formerly presented provenance table has one key column (the alternative description pair) and two value columns (the pair's probability and the pair's matching result). If we additionally incorporate process uncertainty we need to extend this table by one key column per matching phase that additionally introduce uncertainty by the use of a probabilistic configuration, because the alternatives of the output value does not only depend on the alternative description pair, but also depends on the configuration alternatives that have been used to compute this output value from the considered alternative description pair. Note, since data preparation uncertainty is modeled in the descriptions themselves and preparation is not part of the pairwise matching process, we do not consider the used preparation configurations in the provenance table.

Example 165 *For illustration, Figure 7.82(a) presents a provenance table that stores the provenance of probabilistic similarity score alternatives where uncertainty has been introduced by the input data, the feature matching phase, and the similarity computation phase. The first key column represents the alternative description pairs x (probability 0.6) and y (probability 0.4), the second key column represents the three different configurations of the feature matching phase α (probability 0.5), β (probability 0.4), and φ (probability 0.1), and the third key column represents the two different configuration of the similarity computation phase r (probability 0.8) and s (probability 0.2). The first value column represents the joint probability of the combination in its corresponding key column values and the last column contains a reference to the corresponding alternative of the probabilistic similarity score that is presented in Figure 7.82(b).*

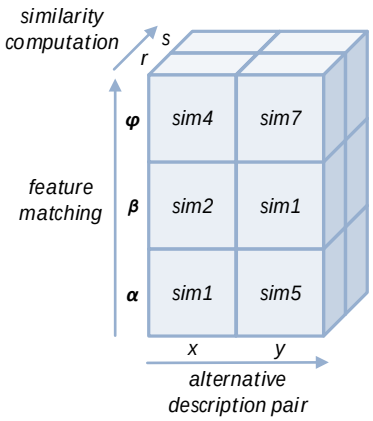


Figure 7.83.: Similarity cube that represents the provenance table from Figure 7.82(a)

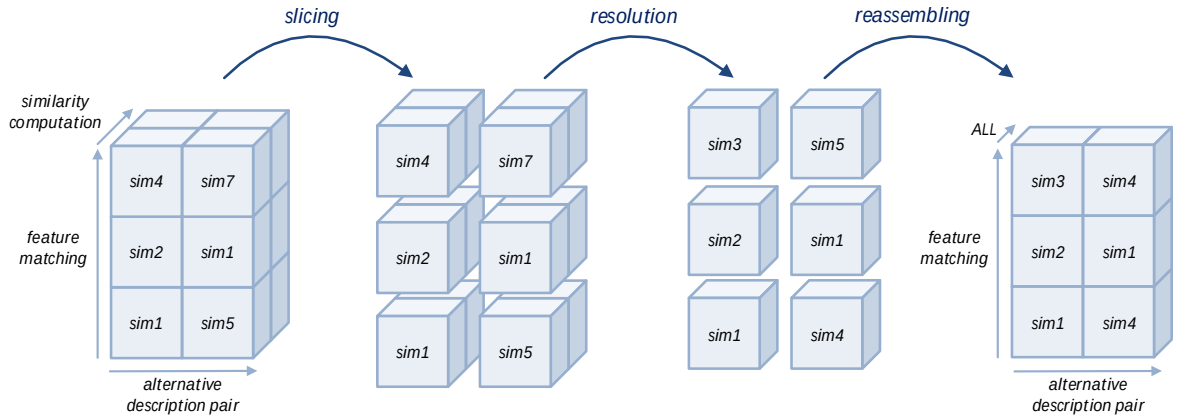
It is simple to see that the provenance table presented in Figure 7.82(a) is structurally equivalent to the fact table of a star schema that is typically used in data warehousing [Leh03] where the alternative description pairs and the phase configurations are the dimensions and their joint probabilities as well as their similarity scores are the facts. Consequently, we can interpret the provenance table as a data cube. Since each matching phase adds a new dimension to the provenance table, we can distinguish between three types of cubes:

- **Feature Cube:** The feature cube has only two dimensions. The first dimension contains the alternative description pairs and the second dimension contains the used configuration of the feature matching phase. The cube's cells contain the facts, i.e. the probabilities and the feature scores.
- **Similarity Cube:** Besides the two dimensions of the feature cube, the similarity cube has a third dimensions that contains the used configurations of the similarity computation phase. Each fact (and hence cell) of the similarity cube is a pair of probability and similarity score.
- **Decision Cube:** The decision cube extends the similarity cube by a fourth dimension that contains the used configurations of the classification phase. The facts are the pairs of probabilities and the duplicate decisions.

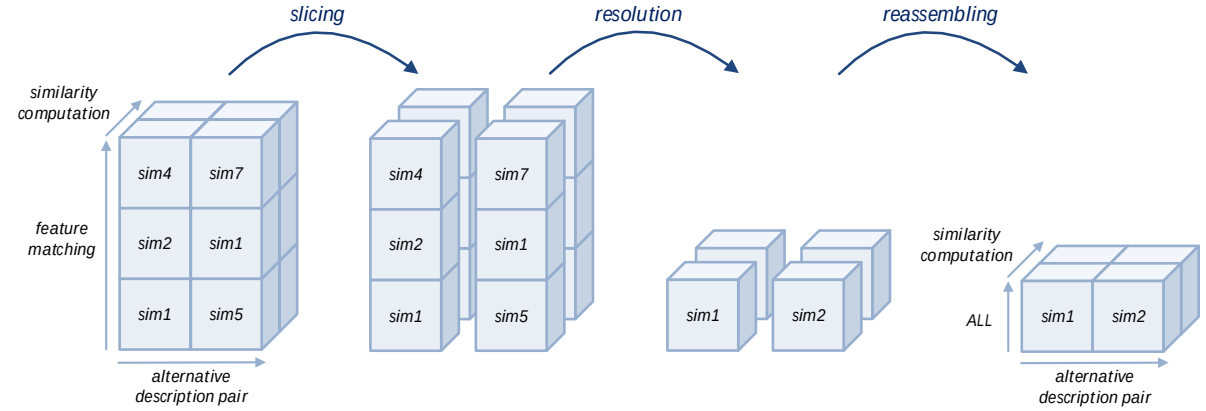
For illustration, Figure 7.83 presents the similarity cube that corresponds to the provenance table from Figure 7.82(a).

Under the cube-based interpretation, uncertainty reduction and uncertainty resolution can be considered as cube operators. Resolution can be applied to the whole cube at once or can be applied to single dimensions individually[13]. In the first case, the cube is compressed to a single cell and in the second case we divide the n-dimensional cube into slices (each a one dimensional cube), compress each slice to a single cell by the given resolution method and then reassemble all cells to a $(n-1)$-dimensional cube. Consequently, resolving a single dimensions leads to the loss of this dimension. This process of resolution is graphically illustrated for the similarity computation phase in Figure 7.84(a) and is graphically illustrated for the feature matching phase in Figure 7.84(b).

[13]Theoretically, resolution and reduction can be applied to every subset of dimensions. Nevertheless, in this thesis we restrict to resolution /reduction of either a single dimension or the whole cube.



(a) Resolution of the third dimension that represents the uncertain configuration of the similarity computation phase



(b) Resolution of the second dimension that represents the uncertain configuration of the feature matching phase

Figure 7.84.: Two examples for resolving the uncertainty of a single dimension

In the actual provenance table the changes caused by a resolution method can only be demonstrated less visualizable. If we resolve the whole cube, all the table's rows are replaced by a single row that contains the value "ALL" in each key column (dimension) and contains the accumulative probability (usually 1.0) as well as the resolved matching result in the value columns (facts). If we resolve only a single dimension, all rows that have the same values in the non-resolved dimensions are consolidated to a single row that contains the value *ALL* in the resolved dimension, and contains the accumulative probability of these rows as well as the resolved matching result in the fact columns. Since rows are consolidated, the number of rows shrinks. Note, an uncertainty resolution of a single dimension corresponds to an uncertainty reduction of the whole matching result.

Example 166 *Figure 7.85(a) presents the provenance table that result from resolving the third dimension of the previously considered provenance table from Figure 7.82(a) by computing the expected similarity score. Since we want to resolve only the uncertainty of the third dimension (similarity computation phase), we group all rows by the values of the remaining dimensions (in the cube based visualization, each group represents a slice). Each group is then resolved independently by computing its expected similarity score and by summing up the probabilities of its rows. Finally each group result in a single*

alt. pair	feature matching	similarity computation	*probability*	*similarity score*
x	***α***	*ALL*	*0.24+0.06=0.3*	*sim1*
x	***β***	*ALL*	*0.192+0.048=0.24*	*sim2*
x	***φ***	*ALL*	*0.048+0.012=0.06*	*sim3*
y	***α***	*ALL*	*0.16+0.04=0.2*	*sim4*
y	***β***	*ALL*	*0.128+0.032=0.16*	*sim1*
y	***φ***	*ALL*	*0.032+0.008=0.04*	*sim5*

(a) Provenance table after resolving the uncertainty of the similarity computation phase

alt.	(*sim*,*imp*)	*Pr*
sim1	*(0.32,1.0)*	0.06
sim2	*(0.42,1.0)*	0.46
sim3	*(0.52,1.0)*	0.24
sim4	*(0.72,1.0)*	0.04
sim5	*(0.82,1.0)*	0.2

(b) Probabilistic similarity score

Figure 7.85.: The provenance table and probabilistic similarity score that result from resolving the uncertainty of the similarity computation phase configuration

row that gets the value "ALL" in the resolved dimension. The reduced probabilistic similarity score is presented in Figure 7.85(b). Note, the described resolution process corresponds to the one that is graphically presented in Figure 7.84(a).

Theoretically, the values of one dimension can be hierarchically categorized. For instance, the set of all plausible feature matching configurations can be categorized into configurations that use token-based similarity measures and configurations that use edit-based similarity measures. In that case, resolution can be also performed to single categorizations. This kind of resolution is equivalent to the roll-up operator of data cubes.

Reduction can be performed similar to resolution, but does not change the structure of the cube. Theoretically, reduction can reduce the size of the cube, but because in any slice of the reduced dimension another value can be discarded, size reduction is not obligatory and we define reduction in a way that it always retains the number of cells. Nevertheless, whereas the size of the cube and thus the size of the provenance table do not decrease, the number of alternatives of the corresponding probabilistic matching result is reduced. As a consequence, more cells refer to the same matching result or are empty, i.e. contain the reference *nil*. Note that an empty cell represents a combination that is not possible anymore and therefore has a joint probability of zero.

Example 167 *Figure 7.86(a) presents a provenance table that result from reducing the second domain of the provenance table in Figure 7.82(a). For reduction we first group all rows that have same values in the first and the third dimension and then perform reduction in each of these groups separately. In this example, reduction is realized by discarding the matching result alternative that is least probable. In the groups where the rows have the value x in the first dimension (alternative description pair), the matching result of the rows with the value φ in the second dimension (feature matching phase) are discarded and therefore are set to zero probability. In contrast, in the groups where the rows have the value y in the first dimension, the rows with the value β in the second dimension are discarded. According to the definition of this reduction method (see Section 7.6.2), the probability of the discarded alternatives is uniformly redistribute to the non-discarded alternatives of the reduced probabilistic similarity score. Since reduction is performed in each group separately (otherwise the other dimensions would be affected by reduction), the probability of all discarded rows of one group is uniformly redistribute to the non-*

alt. pair	feature matching	similarity computation	*probability*	*similarity score*
x	***α***	*r*	*0.6×0.5/0.9×0.8=0.267*	*sim1*
x	***α***	*s*	*0.6×0.5/0.9×0.2=0.067*	*sim2*
x	***β***	*r*	*0.6×0.4/0.9×0.8=0.213*	*sim2*
x	***β***	*s*	*0.6×0.4/0.9×0.2=0.053*	*sim3*
x	***φ***	*r*	*0*	-
x	***φ***	*s*	*0*	-
y	***α***	*r*	*0.4×0.5/0.6×0.8=0.267*	*sim5*
y	***α***	*s*	*0.4×0.5/0.6×0.2=0.067*	*sim6*
y	***β***	*r*	*0*	-
y	***β***	*s*	*0*	-
y	***φ***	*r*	*0.4×0.1/0.6×0.8=0.053*	*sim7*
y	***φ***	*s*	*0.4×0.1/0.6×0.2=0.013*	*sim5*

(a) Provenance table after reducing the uncertainty of the feature matching phase

alt.	(*sim*,*imp*)	*Pr*
sim1	*(0.4,1.0)*	0.267
sim2	*(0.5,1.0)*	0.28
sim3	*(0.6,1.0)*	0.053
sim4	*(0.7,1.0)*	0.053
sim5	*(0.8,1.0)*	0.28
sim6	*(0.9,1.0)*	0.067

(b) Probabilistic similarity score

Figure 7.86.: The provenance table and probabilistic similarity score that result from reducing the uncertainty of the feature matching phase configuration

discarded rows of that group. The resultant provenance table has the same size as the unreduced table, but as presented in Figure 7.86(b) the reduced probabilistic similarity score has one alternative less.

Whereas some resolution methods / reduction methods can be used for every dimension, others are dimension-specific. For example, the expected similarity score can be applied to every dimension of the similarity cube. In contrast, the Probabilistic Monge-Elkan Similarity is only applicable for probabilistic entity descriptions and therefore can only applied to the first dimension of the similarity cube. For that reason, we either have to resolve all other dimensions before we compute the Probabilistic Monge-Elkan Similarity or we have to compute it once for several slices. Note, if we need to resolve several slices separately, the runtime advantage of the Probabilistic Monge-Elkan Similarity compared to the Earth Mover's Distance becomes even more significant.

The dimensions can be resolved in any order and after any of the pairwise matching phases, but at least need to be resolved until the duplicate clustering phase because existing clustering approaches are not defined for getting cubes as input.

A special resolution method for decision cubes is to compute the probability that the corresponding entity pair $\{e_r, e_s\}$ is a MATCH by simply summing up the probabilities of all facts that have a MATCH as matching class. By doing so we actually do not resolve uncertainty completely, but quantify it by a simple number (instead of a probabilistic value). The resultant score is called the *matching probability* ($Pr(\{e_r, e_s\} \in \text{MATCH})$) of this entity pair and can be used as input to a duplicate clustering approach that we present in Section 8.7.

7.8.4. Matching Cost Optimization

Of course, besides uncertainty reduction other strategies for matching cost optimization are affected by the incorporation of process uncertainty as well. One of them is the in-depth pruning we have presented

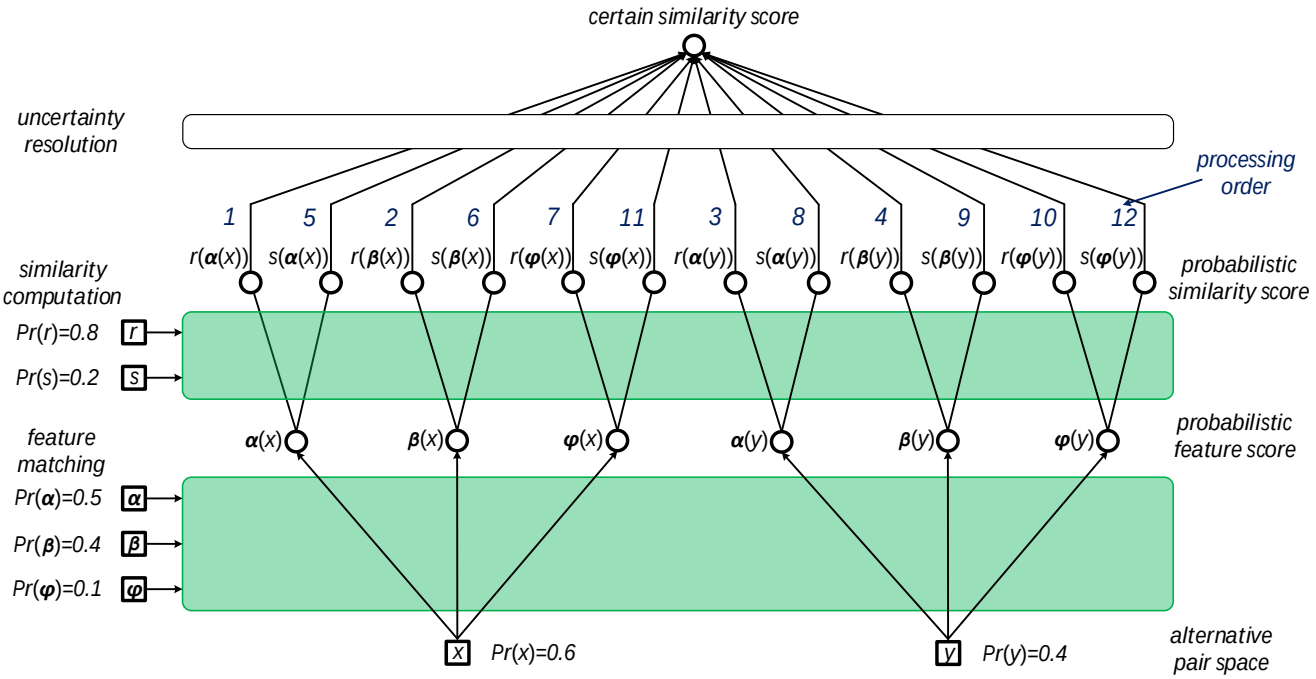


Figure 7.87.: In-depth pruning in the presence of process uncertainty

in Section 7.6.1.2. Recall, in this optimization strategy, the alternative description pairs are processed by the individual matching phases one after another until the used resolution method can compute a deterministic result. To minimize the number of matched alternative description pairs, the pairs are processed in order of their probabilities, i.e. the most probable pair is processed first. By doing so, the one dimensional provenance table (only one dimension column) that we have considered without process uncertainty is filled with facts step by step.

In the presence of process uncertainty, this strategy need to be adapted, because the uncertainty of the individual phases affects the final result as well. Naturally, the most probable phase configurations are processed first. Thus, instead of ordering alternative description pairs by their probability we order combinations of alternative description pairs and phase configurations by their joint probability and then perform these combinations one after another until the resolution method can compute a deterministic result. In conclusion, we fill the finally resolved cube cell by cell.

Example 168 *For illustration we reconsider the sample provenance table from Figure 7.82(a) and assume that at the beginning the table is empty, because we do not have matched any description alternatives so far. Moreover, we assume that we want to perform uncertainty resolution to all the dimensions at once after the phase of similarity computation. The in-depth pruning on this sample is presented in Figure 7.87. Recall that we assume independence and thus the joint probability of a combination of an alternative description pair and two configuration alternatives is equal to the product of the probabilities of these three components. Therefore, the most probable combination is to apply the feature matching configuration α and the similarity computation configuration r to the alternative description pair x. The result of this combination is the similarity score $r(\alpha(x))$. The second most probable combination is (x, β, r) and the third most probable combination is (y, α, r). In Figure 7.87, the ordering in which the*

combinations are processed is marked by the blue numbers above the resultant similarity scores. Usually we do not need to match all combinations to compute a deterministic resolution result and hence can save matching costs by using this iterative approach.

Obviously, if the feature score $\alpha(x)$ has been already computed for $r(\alpha(x))$, the similarity score $s(\alpha(x))$ can be computed with less costs than the similarity score $r(\beta(x))$ if we cache the interim results of the feature matching phase. By doing so, we only need to perform the similarity computation phase to compute $s(\alpha(x))$, but need to perform the feature matching phase and the similarity computation phase to compute $r(\beta(x))$. As a consequence, it can be profitable to determine the ordering of the input combinations not only based on their probabilities, but also on their required costs, because matching multiple less probable but 'cheap' combinations can be more useful than matching a single high probable but 'expensive' combination if the accumulative probability of the first is greater than the probability of the second. For that reason, we can draw a cost-benefit list that contains the costs (the accumulative costs of all phases that still need to be computed for this combination) and benefits (the probability of this combination) for each combination of dimension values and then order the combinations by their benefit-cost ratio.

As mentioned above, uncertainty resolution can be performed for each dimension individually. Thus, the moments of resolutions and the consequent ordering of resolution is another aspect that affects optimization, because some dimensions can be resolved earlier than others. In that case we need to recompute the former resolution on each slice for each of the newly matched combinations of dimension values in order to determine whether or not the later resolution can compute a deterministic result. Thus, the Best Match Scenario (or Worst Match Scenario respectively) of the latter resolution is computed based on the Best Match Scenarios (or Worst Match Scenarios respectively) of the slices of the former resolution. For that reason, it is essential that the former resolutions can be computed iteratively as well, because otherwise we have to completely recompute these values each time a new combination has been matched for the later resolution.

7.9. Approach Comparison

In this section, we provide a theoretical comparison of the world-based detection approach and the description-based detection approach. Both detection approaches reuse methods that have been designed for duplicate detection in certain databases. The trump card of the world-based detection approach, is its simplicity because it can be build upon a conventional duplicate detection system easily. The only methods that are additionally required are a method for generating possible worlds and a method for aggregating the detection results that have been computed by performing a conventional duplicate detection process on each of the possible worlds separately. In contrast, besides aggregation methods the description-based detection approach requires a method for extracting probabilistic entity descriptions from the probabilistic input database, requires a special implementation of the candidate pair space construction phase, and requires a phase model that can deal with probabilistic output values that are produced by the individual detection phases. Nevertheless, the description-based detection approach has several advantages compared to the world-based detection approach.

- In the description-based detection approach all alternative description pairs of two database entities are match collectively. Thus, each pair of description alternatives is matched only once. In contrast, entities can be represented by the same description in different possible worlds. As a consequence, it is not uncommon that by using the world-based detection approach same entity descriptions are matched for several times.

  Example 169 *This benefit can be illustrated by a simple example. Let us assume a set of ten mutual independent database entities where each of these entities has five possible instances. Moreover, let us assume that the used extraction process computes another entity description per possible instance and let us assume that we do not reduce the candidate pair space.*

  In the description-based approach we need to match $\frac{10 \times 9}{2} = 45$ *entity pairs. Without alternative pair space reduction, each entity pair comparison requires a matching of* $5 \times 5 = 25$ *alternative description pairs. In summary, the pairwise detection phases need to be performed for* $45 \times 25 = 1,125$ *times. Since we compare each possible instance of one entity with each possible instance of any other entity, this approach corresponds to perform duplicate detection in every possible world. Because the assumed independence between the ten entities, the number of possible worlds is* $5^{10} = 9,765,625$. *In each world we need to match* 45 *description pairs. Thus, altogether we need to perform the pairwise detection phases for* $439,453,125$ *times which is around* $400,000$ *as much as we need in the description-based approach.*

  Of course, this difference can be smaller if the processed database entities are correlated. Nevertheless, as we have described in Section 7.7.1 we recommend to remove entity dependencies from the input database and therefore can assume an entity-independent input database in most cases.

- In the world-based detection approach, a domain expert can only review a single pair of description alternatives at one time and cannot review all the alternative description pairs of one entity pair collectively because each world is considered to be processed independently. Therefore, interactions of domain experts with the detection process are much more complicated in the world-based detection approach than in the description-based detection approach.

- The most matching cost optimization techniques that we have presented in Section 7.6 are only applicable if all alternative description pairs of one entity pair are processed collectively. For instance, the In-Depth Pruning method saves computation time by only comparing as many alternative description pairs as we require for making a deterministic decision for the considered entity pair. As a consequence, we can use them in the description-based detection approach, but cannot use them in the world-based detection approach.

  Similar holds for soft pruning methods, reducing a probabilistic matching result is simple because we only need to discard some of its alternatives. In contrast, a pair-based reduction of uncertainty is difficult if matching is performed in a world-based fashion and discarding whole worlds is usually less appropriate.

- As we have described in Section 7.5.4.5, similarity measures for probability mass functions such as the Probabilistic Monge-Elkan Similarity are often only less useful for uncertainty resolution

in the world-based detection approach if the number of processed possible worlds is rather small. Thus, a detection of scattered duplicates is much more difficult by using the world-based detection approach than by using the description-based detection approach.

- It is more convenient to incorporate process uncertainty into the description-based detection approach than into the world-based detection approach because the first is already defined to deal with probabilistic results of the individual detection phases, but the second is not.

In summary, from a theoretical point of view the description-based detection approach enables a better trade-off between effectiveness and efficiency than the world-based detection approach.

Chapter 8

Indeterministic Duplicate Detection

In this chapter, we present the concept of indeterministic duplicate detection. We start with a formal introduction of indeterministic duplicate detection processes, indeterministic deduplication processes, and uncertain clusterings in Section 8.1. and adopt properties and methods from certain clusterings to uncertain clusterings in Section 8.2. Since a separate storage of all possible clusterings is usually impractical, we consider a factorization of uncertain clusterings in Section 8.3. Then, we present approaches for modeling a factorized indeterministic deduplication result within a probabilistic database in Section 8.4 and discuss processing indeterministic deduplication results in Section 8.5. In Section 8.6, we discuss some peculiarities for performing indeterministic deduplication on probabilistic databases. Then, we propose an approach for producing uncertain duplicate clusterings from pairwise duplicate decisions in Section 8.7. Finally, we investigate the meaning of detection quality for cases in which the detection result corresponds to an uncertain clustering in Section 8.8.

In the previous chapter, we discuss several approaches for incorporating data uncertainty and process uncertainty into a duplicate detection process, but in the end they all produce a deterministic result, i.e. they need to resolve uncertainty for constructing a single clustering of all the considered database entities. Thus, regardless of its underlying ambiguity, we finally make a deterministic duplicate decision for each entity pair with all consequences this may have. Uncertainty resolution, however, is not required if we model the deduplicated database within a probabilistic representation system that is powerful enough to represent exclusions between sets of entities. In contrast, by using such a representation system we are able to model uncertainty on duplicate decisions in the detection result and hence can avoid any cases of uncertainty resolution that are not based on solid information. By doing so, all significantly likely duplicate clusters find their way into the database and any query result or other derived data will reflect this inherent uncertainty.

Since we relax the mandatory requirement of making deterministic duplicates decisions and instead provide methods for handling decisions indeterministically, we denote this concept as indeterministic duplicate detection [PR12] (only duplicate detection) and as indeterministic deduplication [PvKR13] or indeterministic duplicate elimination (duplicate detection combined with duplicate merging) respectively. It is important to note that only the duplicate detection result is of an indeterministic nature, but

the detection process itself can be still deterministic[1], i.e. it always produces the same result if the same process configuration and the same input database are given.

The main object of indeterministic duplicate detection is to reduce the risk of producing false decisions, because they can cause massive costs (financial area), can effect physical harms (medical area), or can provoke damages of several other kinds. Nevertheless, modeling uncertainty in the detection result implicates some further advantages.

- Because no absolute decision needs to be taken, we can decrease (or maybe completely avoid) the use of clerical reviews which is always a time consuming and monetary expensive task. A complete eschewal of clerical reviews even enables a fully automatically detection process that is required in several application domains as for example in virtual data integration [LN06, HDI12].

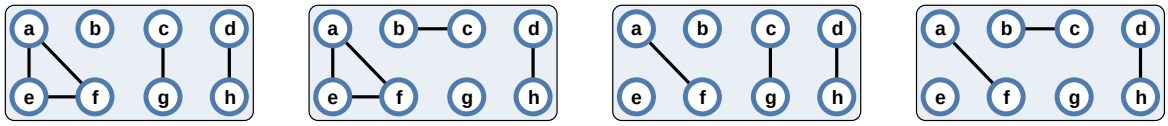
- Domain experts are usually not all-knowing. Thus, even in cases where clerical reviews are required, we can profit from modeling uncertainty in the detection result because domain experts are no longer forced to make concrete decisions in ambiguous situations, but instead can incorporate their incomplete knowledge in terms of degrees of belief.

- Modeling uncertainty on duplicate decisions unresolved in the database can save a lot of processing time, because as a rough guide around 10% of duplicate decisions usually causes 90% of processing time [vKdK09].

Of course representation complexity and query complexity increase with a growing amount of uncertainty in the database, but are still manageable in many cases. Moreover, modeling decision uncertainty in the deduplicated database must not be a fixed state. In contrast, it can be rather considered as a prompt answer whose uncertainty is later reduced from time to time, but is immediately queryable. Moreover, the subsequently performed resolution of the indeterministically modeled decisions does not need to be made by a handful of experts, but can be made interactively by a complete community. Moreover, the indeterministically modeled decisions do not need to be resolved by a handful of experts, but can be resolved interactively with the support and the knowledge of a complete community.

The remainder of this chapter is structured as follows: First we consider a formal definition of uncertain clusterings and indeterministic duplicate detection in Section 8.1. Then, we adopt properties and operations from certain clusterings to uncertain clusterings in Section 8.2. A factorization of uncertain clusterings is considered in Section 8.3. A representation of indeterministic duplicate detection result within a probabilistic database is considered in Section 8.4. Processing indeterministic deduplication results is discussed in Section 8.5. In Section 8.7, we propose a clustering approach that produces a factorized uncertain duplicate clustering as a result. Several peculiarities of utilizing an indeterministic duplicate detection to probabilistic databases is considered in Section 8.6. Finally, we discuss quality evaluation of indeterministic duplicate detection results in Section 8.8.

[1] Note, even in the context of deterministic duplicate detection some processes are non-deterministic. Thus, they in fact produce a certain clustering as a result (the result is deterministic), but can produce different clusterings in different runs even if the same input database and the same process configurations are given (the process is non-deterministic). For example, the Star Clustering algorithm [APR04] is an non-deterministic process because in this algorithm seeds are chosen randomly.



(a) Possible clustering $\mathcal{C}_1$ (b) Possible clustering $\mathcal{C}_2$ (c) Possible clustering $\mathcal{C}_3$ (d) Possible clustering $\mathcal{C}_4$

Figure 8.1.: An incomplete clustering $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$

8.1. Problem Definition

Since duplicate decisions on individual entity pairs can be correlated, we incorporate uncertainty into the detection result by modeling it as an uncertain clustering (incomplete or probabilistic) instead of a certain clustering.

Recall C-All(S) is the set of all clusterings of an element set S and C-All$_{\circledast}(S)$ is the set of all cluster-disjoint clusterings of an element set S.

Definition 59 (Incomplete Clustering): *An incomplete clustering Γ of the element set S is defined as an incomplete value of the domain C-All(S), i.e. $\Gamma \in^{[\iota]}$ C-All(S), and hence is a set of certain (crisp) clusterings of S.*

An alternative of an incomplete clustering is in the following also denoted as a *possible clustering*.

Example 170 *For demonstration, we consider the incomplete clustering Γ that is depicted in Figure 8.1. This clustering is defined on the element set $\{a, b, c, d, e, f, g, h\}$ and contains the four possible clusterings $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$:*

$$\mathcal{C}_1 = \{\langle a, e, f\rangle, \langle b\rangle, \langle c, g\rangle, \langle e, h\rangle\} \qquad \mathcal{C}_3 = \{\langle a, f\rangle, \langle e\rangle, \langle b\rangle, \langle c, g\rangle, \langle e, h\rangle\}$$
$$\mathcal{C}_2 = \{\langle a, e, f\rangle, \langle b, c\rangle, \langle g\rangle, \langle e, h\rangle\} \qquad \mathcal{C}_4 = \{\langle a, f\rangle, \langle e\rangle, \langle b, c\rangle, \langle g\rangle, \langle e, h\rangle\}$$

If we consider these elements as database entities and if we consider the clusterings as duplicate detection results, we say that the duplicate decision on entity pair $\{e, f\}$ is modeled indeterministically because both entities belong to the same cluster in the possible clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$, but belong to different clusters in the possible clusterings $\mathcal{C}_3$ and $\mathcal{C}_4$.

Definition 60 (Probabilistic Clustering): *A probabilistic clustering $\mathfrak{C}$ of the element set S is defined as a probabilistic value of the domain C-All(S), i.e. $\mathfrak{C} \in^{[\rho]}$ C-All(S), and hence is a pair $\mathfrak{C} = (\Gamma, Pr)$ where Γ is an incomplete clustering of S and Pr is a probability distribution on C-All(S).*

Definition 61 (Possible Cluster): *A set of elements is called to be a possible cluster of an uncertain clustering if it is a cluster in at least one of its alternatives. Thus, let Γ be an incomplete clustering, the set of all possible clusters of Γ is defined as: possCl$(\Gamma) = \bigcup \Gamma$. The set of all possible clusters of a probabilistic clustering is defined accordingly.*

The range function rng has only one parameter. Thus, its incomplete version is defined as $rng(\Gamma) = \{rng(\mathcal{C}) \mid \mathcal{C} \in \Gamma\}$. Since all alternatives are defined on the same set of elements, the range of an

incomplete clustering is always a crisp value. The same holds for a probabilistic clustering. Therefore, let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering of an element set S, its range is defined as $rng(\mathfrak{C}) = rng(\Gamma) = S$.

An uncertain clustering of the set S is considered to be cluster-disjoint, if all its alternatives are cluster-disjoint and hence if it is an uncertain value of the domain C-All$_{\otimes}(S)$. In the rest of the thesis we assume uncertain clusterings to be cluster-disjoint if not explicitly stated otherwise.

By modeling ambiguous duplicate decisions in the form of an uncertain clustering, the result of an indeterministic duplicate detection process is either an incomplete clustering or a probabilistic clustering.

Definition 62 (Indeterministic Duplicate Detection): *An indeterministic duplicate detection process is a function δ_{idet} that maps a set of database entities to a cluster-disjoint incomplete clustering Γ or a cluster-disjoint probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ respectively.*

Let Δ_{det} be the space of all possible deterministic duplicate detection processes. An indeterministic duplicate detection process can be considered as an uncertain function on the domain Δ_{det}, because for each alternative of the uncertain output clustering there must be a deterministic duplicate detection process that produces this clustering as a detection result.

Example 171 *For illustration, we reconsider the duplicate detection scenario from Example 60. In this example, we consider a duplicate clustering phase that gets the duplicate pair graph from Figure 4.18(a). as input and that has to produce a duplicate clustering as output. As presented in Example 60, different clustering approaches lead to different duplicate clusterings (see Figure 4.18(c) to Figure 4.18(e)) and it is often not clear which of these clusterings represents reality best. Thus, instead of selecting only one of these clusterings as output, it maybe make sense to use each of them as an alternative of an incomplete clustering. By doing so, we produce the incomplete clustering Γ that has the three alternatives*

$$\mathcal{C}_1 = \{\langle e_1, e_2, e_3, e_4, e_5, e_6, e_{10}\rangle, \langle e_7, e_8\rangle, \langle e_9\rangle\}$$
$$\mathcal{C}_2 = \{\langle e_1, e_2, e_4\rangle, \langle e_3\rangle, \langle e_5, e_6\rangle, \langle e_7, e_8\rangle, \langle e_9\rangle, \langle e_{10}\rangle\}$$
$$\mathcal{C}_3 = \{\langle e_1, e_2, e_4\rangle, \langle e_3, e_5, e_6, e_{10}\rangle, \langle e_7, e_8\rangle, \langle e_9\rangle\}$$

where each of these alternatives is computed by another deterministic duplicate detection process.

Using Definition 62 and Definition 23 an indeterministic duplicate elimination process is defined as:

Definition 63 (Indeterministic Duplicate Elimination): *Formally, an indeterministic duplicate elimination process is a function ϵ_{idet} that maps a set of database entities to an incomplete set of database entities or to a probabilistic set of database entities respectively. Let δ_{idet} be an indeterministic duplicate detection process and let μ be a duplicate merging process, the indeterministic duplicate elimination process $\epsilon_{idet} = \mu \circ \delta_{idet}$ that is performed on a set of database entities $\mathfrak{E}$ is defined as:*

$$\epsilon_{idet}(\mathfrak{E}) = \mu \circ \delta_{idet}(\mathfrak{E}) = \{\mu(\mathcal{C}) \mid \mathcal{C} \in \delta_{idet}(\mathfrak{E})\}$$

if δ_{idet} produces an incomplete clustering, and is defined as:

$$\begin{aligned} \epsilon_{idet}(\mathfrak{E}) &= \mu \circ \delta_{idet}(\mathfrak{E}) = (\mathcal{E}, Pr_{\mathcal{E}}) \\ \textit{where} \quad & \mathcal{E} = \{\mu(\mathcal{C}) \mid \mathcal{C} \in \Gamma\} \quad \textit{and} \quad \forall \mathfrak{E} \in \mathcal{E}\colon Pr_{\mathcal{E}}(\mathfrak{E}) = \sum\nolimits_{\mathcal{C} \in \Gamma} Pr(\mathcal{C}) \end{aligned}$$

if δ_{idet} produces the probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$,

Thus, if an indeterministic duplicate elimination process gets a certain database as input and if it uses a conflict resolution approach for duplicate merging, it produces an uncertain database as output where each possible world of the output database corresponds to a possible deduplication result.

Modeling duplicate detection results by using uncertain clusterings implicates a set of challenges:

- The number of clustering alternatives exponentially increases with the number of uncertain pairwise decisions. As a consequence, modeling all possible clusterings separately is typically impractical. For that reason, a more compact representation is required instead. Usually the most uncertain decisions are independent to each other. Thus, we can factorize the uncertain clustering into as set of smaller uncertain clusterings and hence can dramatically reduce the total number of clustering alternatives. For that reason, we consider factorization of uncertain clusterings in Section 8.3.

- In order to model an indeterministic duplicate detection result as an uncertain clustering, we need to compute this clustering from the given uncertainty on the pairwise duplicate decisions. A simple solution is to run the several duplicate detection processes (each with another configuration) on the input database and to consider each result as another clustering alternative. By doing so, however, uncertainty need to be completely resolved in every run and we need to perform one run per resultant alternative. It seems less meaningful to use configurations that are completely different, because uncertainty usually remains in the vernier adjustment of the configuration parameters. Thus, for producing reliable results, the configurations should be highly similar and should only vary in the settings of some few parameters. Running detection processes with similar configurations in turn implies a redundant computation of same intermediate results. In addition, an effective incorporation of detection uncertainty into the detection result will require a large number of runs. As a consequence, this approach can lack in efficiency and effectiveness.

  Moreover, if the number of clustering alternatives is large, a subsequent factorization is not possible and we require a detection approach that directly computes a clustering in its factorized representation.

  For these reasons, we strive for an approach that directly models the underlying uncertainty in a factorized detection result by a single run and hence strive for duplicate detection approaches that do not completely resolve uncertainty.

- Although the uncertain clustering serves as duplicate detection result, it is not the final result of a duplicate elimination process. Instead the duplicate entities need to be merged before they can be stored into the output database. By merging all entities of one cluster into a new entity, the uncertain clustering changed into an uncertain set of database entities (see Definition 63). This begs the question how to store an indeterministic duplicate elimination result within a probabilistic database.

- The purpose of modeling uncertainty in the duplicate detection result is to improve the correctness of the deduplicated database by reducing the number of false decisions. In which way a data

application can benefit from the stored uncertainty, however, essentially depends on the way the application processes the database's inherent uncertainty. For that reason, we need to study what types of applications can process an indeterministic deduplication result.

- The last challenge concerns the quality of indeterministic duplicate detection results because rating detection quality is an essential part in tuning the configuration of a detection process with respect to a specific domain. As we have presented in Section 4.3.8, conventional quality measures rate the quality of certain clusterings by computing their similarity to a given gold standard. However, the meaning of similarity between an uncertain clustering and a gold standard is not clearly defined. Moreover, the incorporation of uncertainty into the detection result requires for quality measures that do not compute the clustering's similarity to the gold standard, but capture its degree of uncertainty. For that reason, we need to elaborate the meaning of duplicate detection quality in the presence of uncertainty and we need to design measures that capture this meaning by numbers. Moreover, because an uncertain clustering is usually provided in its factorized representation, we need to develop methods that enable an efficient computation of the newly designed quality measures based on the clustering's factorization.

8.2. Properties of Uncertain Clusterings

Before we consider a factorization of uncertain clusterings, we need to adopt the properties and operations that we have introduced in Section 2.2 from certain clusterings to uncertain clusterings.

The projection of an uncertain clustering on an element set is defined by the incomplete version (or the probabilistic version respectively) of the projection operator that we introduced for certain clusterings in Definition 8. Consequently, a projection of an incomplete clustering Γ on a set S results in the incomplete clustering:

$$\pi_S(\Gamma) = \{\pi_S(\mathcal{C}) \mid \mathcal{C} \in \Gamma\} \tag{8.1}$$

and a projection of a probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ on a set S results in the probabilistic clustering:

$$\begin{aligned} & \pi_S(\mathfrak{C}) = (\Gamma_S, Pr_S) \\ \text{where} \quad & \Gamma_S = \pi_S(\Gamma) \\ \text{and} \quad & \forall \mathcal{C} \in \text{C-All}(S)\colon Pr_S(\mathcal{C}) = \sum\nolimits_{\mathcal{C}' \in \Gamma, \mathcal{C} = \pi_S(\mathcal{C}')} Pr(\mathcal{C}') \end{aligned} \tag{8.2}$$

Accordingly, the integration of two incomplete clusterings or the integration of two probabilistic clusterings is defined by the incomplete version and the probabilistic version of the integration operator that we introduced in Definition 11. The integration of a set of uncertain clusterings can be considered as a kind of composition (see Section 6.6), because the integration result is an uncertain clustering that models the possible combinations of the alternatives of all integrated uncertain clusterings. For that reason, the integration operator always assumes that its input uncertain clusterings are mutual independent[2].

Consequently, the integration of two incomplete clusterings Γ_i and Γ_j results in the incomplete clustering:

$$\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j = \{\mathcal{C}_i \boxtimes \mathcal{C}_j \mid \mathcal{C}_i \in \Gamma_i, \mathcal{C}_j \in \Gamma_j\} \tag{8.3}$$

[2]Otherwise we would end in the chicken-and-egg problem that we described in Section 6.6 in the context of uncertain value composition.

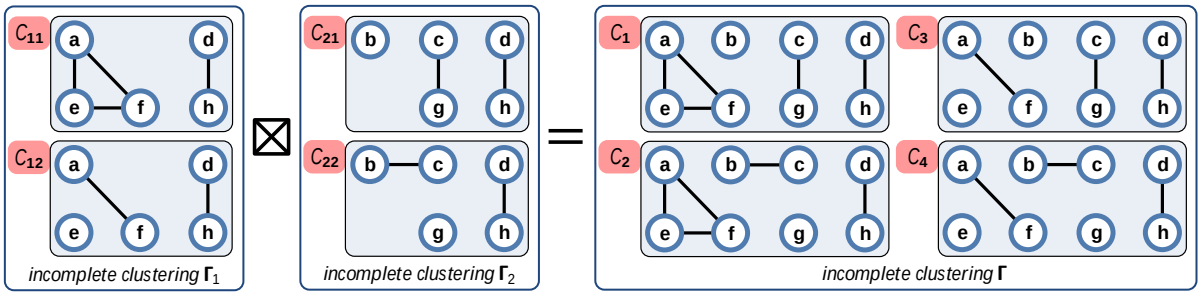


Figure 8.2.: Sample integration of two incomplete clusterings

and the integration of two probabilistic clusterings $\mathfrak{C}_i = (\Gamma_i, Pr_i)$ and $\mathfrak{C}_j = (\Gamma_j, Pr_j)$ results in the probabilistic clustering:

$$\mathfrak{C}_{ij} = \mathfrak{C}_i \boxtimes \mathfrak{C}_j = (\Gamma_{ij}, Pr_{ij}) \tag{8.4}$$

where $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$

and $\forall \mathcal{C} \in \text{C-All}(rng(\mathfrak{C}_i) \cup rng(\mathfrak{C}_j)) : Pr_{ij}(\mathcal{C}) = \sum_{\mathcal{C}_i \in \Gamma_i, \mathcal{C}_j \in \Gamma_j, \mathcal{C} = \mathcal{C}_i \boxtimes \mathcal{C}_j} Pr_i(\mathcal{C}_i) \times Pr_j(\mathcal{C}_j)$

Example 172 *As an example, we consider the two incomplete clusterings* $\Gamma_1 = \{\mathcal{C}_{11}, \mathcal{C}_{12}\}$ *and* $\Gamma_2 = \{\mathcal{C}_{21}, \mathcal{C}_{22}\}$ *with:*

$$\mathcal{C}_{11} = \{\langle a, e, f\rangle, \langle d, h\rangle\} \qquad \mathcal{C}_{21} = \{\langle b\rangle, \langle c, g\rangle, \langle d, h\rangle\}$$
$$\mathcal{C}_{12} = \{\langle a, f\rangle, \langle e\rangle, \langle d, h\rangle\} \qquad \mathcal{C}_{22} = \{\langle b, c\rangle, \langle g\rangle, \langle d, h\rangle\}$$

Integrating Γ_1 *and* Γ_2 *exactly results in the incomplete clustering* $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ *that we considered in Example 170. The integration is graphically presented in Figure 8.2.*

Domination of uncertain clusterings is defined accordingly to the certain case. Thus, an incomplete clustering Γ_i is dominated by an incomplete clustering Γ_j (in symbols $\Gamma_i \prec \Gamma_j$), iff $rng(\Gamma_i) \subseteq rng(\Gamma_j) \wedge \pi_{rng(\Gamma_i)}(\Gamma_j) = \Gamma_i$ and a probabilistic clustering $\mathfrak{C}_i = (\Gamma_i, Pr_i)$ is dominated by a probabilistic clustering $\mathfrak{C}_j = (\Gamma_j, Pr_j)$ (in symbols $\mathfrak{C}_i \prec \mathfrak{C}_j$), iff $rng(\mathfrak{C}_i) \subseteq rng(\mathfrak{C}_j) \wedge \pi_{rng(\mathfrak{C}_i)}(\Gamma_j) = \mathfrak{C}_i$.

Theorem 17 *An incomplete clustering* Γ_i *is dominated by an incomplete clustering* Γ_j*, iff each possible clustering* $\mathcal{C}_i \in \Gamma_i$ *is dominated by a possible clustering* $\mathcal{C}_j \in \Gamma_j$ *and iff each possible clustering* $\mathcal{C}_j \in \Gamma_j$ *dominates a possible clustering* $\mathcal{C}_i \in \Gamma_i$*.*

Theorem 17 is proved by Proof 16 in Section A.2.

Example 173 *For illustration, let* $\mathcal{C}_{11}$*,* $\mathcal{C}_{12}$*,* $\mathcal{C}_{21}$*, and* $\mathcal{C}_2$ *be the following four clusterings:*

$$\mathcal{C}_{11} = \{\langle a, b\rangle, \langle c\rangle\} \qquad \mathcal{C}_{21} = \{\langle a, b, d\rangle, \langle c, e\rangle\}$$
$$\mathcal{C}_{12} = \{\langle a\rangle, \langle b, c\rangle\} \qquad \mathcal{C}_{22} = \{\langle a, d\rangle, \langle b, c\rangle, \langle e\rangle\}$$

The incomplete clustering $\Gamma_1 = \{\mathcal{C}_{11}, \mathcal{C}_{12}\}$ *is dominated by the incomplete clustering* $\Gamma_2 = \{\mathcal{C}_{21}, \mathcal{C}_{22}\}$ *because* $\mathcal{C}_{11} \prec \mathcal{C}_{21}$ *and* $\mathcal{C}_{12} \prec \mathcal{C}_{22}$*.*

Theorem 18 *A probabilistic clustering* $\mathfrak{C}_i = (\Gamma_i, Pr_i)$ *is dominated by a probabilistic clustering* $\mathfrak{C}_j = (\Gamma_j, Pr_j)$*, iff* Γ_i *is dominated by* Γ_j *and it holds that* $\forall \mathcal{C} \in$ *C-All*$(rng(\mathfrak{C}_i))\colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$.

Theorem 18 is proved by Proof 17 in Section A.2.

Recall, Theorem 1 describes that an integration of two clusterings is cluster-disjoint, if the integrated clusterings are cluster-disjoint and either have disjoint ranges or the shared elements form the same clusters in both clusterings.

This theorem can be adopted to incomplete clusterings as follows:

Theorem 19 *The incomplete clustering* $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$ *is a cluster-disjoint clustering, iff* Γ_i *and* Γ_j *are cluster-disjoint clusterings and either their ranges are disjoint or the elements of the ranges' overlap only form clusters that belongs to all possible clusterings of* Γ_i *and belongs to all possible clusterings of* Γ_j*, i.e.* $\forall \mathcal{C}_i \in \Gamma_i\colon \forall \mathcal{C}_j \in \Gamma_j\colon \forall C_1 \in \mathcal{C}_i\colon \forall C_2 \in \mathcal{C}_j\colon C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$.

Theorem 19 is proved by Proof 18 in Section A.2.

Since the probabilities of the possible clusterings do not affect cluster-disjointness, a probabilistic clustering $\mathfrak{C}_{ij} = \mathfrak{C}_i \boxtimes \mathfrak{C}_j$ with $\mathfrak{C}_{ij} = (\Gamma_{ij}, Pr_{ij})$, $\mathfrak{C}_i = (\Gamma_i, Pr_i)$, and $\mathfrak{C}_j = (\Gamma_j, Pr_j)$ is cluster-disjoint, iff $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$ is cluster-disjoint.

Recall, Theorem 2 describes that integrating two projections of an uncertain clustering on disjoint sets is equivalent to a projection on the union of these sets, if none of the clustering's alternatives contains a cluster that overlap with both of these sets. Since the integration operator assumes a mutual independence between the alternatives of the input clusterings, Theorem 2 cannot be directly adopted to incomplete clusterings, but need to be reformulated as follows:

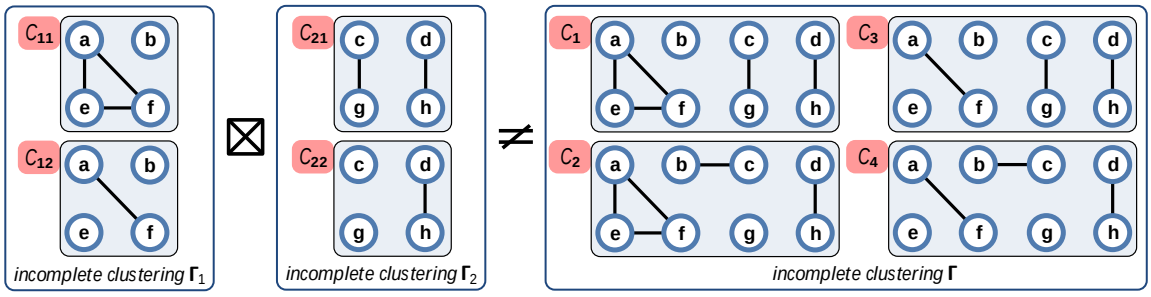
Theorem 20 *Let* Γ *be an incomplete clustering and let* $S_i, S_j \subseteq rng(\Gamma)$ *be two disjoint subsets of the clustering's range. It holds that* $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) = \pi_{S_i \cup S_j}(\Gamma)$ *iff the two clusterings* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are mutually independent with respect to* Γ*, i.e. iff* $\forall \mathcal{C}_i \in \pi_{S_i}(\Gamma)\colon \forall \mathcal{C}_j \in \pi_{S_j}(\Gamma)\colon \exists \mathcal{C} \in \Gamma\colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ *and* $\forall \mathcal{C} \in \Gamma\colon \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma)\colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma)\colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$.

Theorem 20 is proved by Proof 19 in Section A.2.

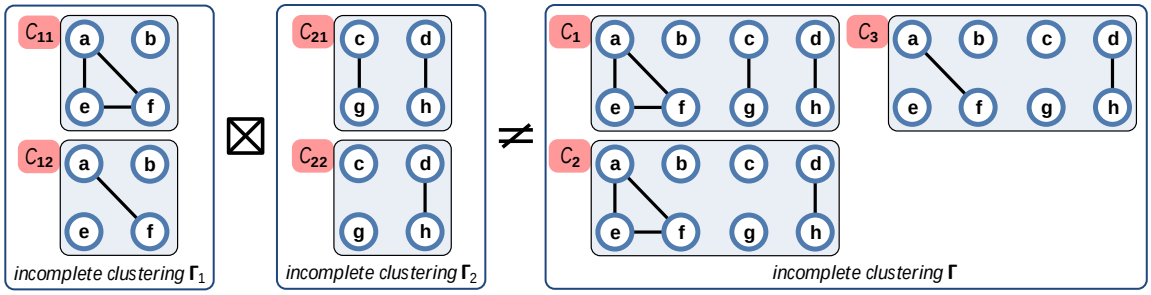
Note, the first independence condition $\forall \mathcal{C}_i \in \pi_{S_i}(\Gamma)\colon \forall \mathcal{C}_j \in \pi_{S_j}(\Gamma)\colon \exists \mathcal{C} \in \Gamma\colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ ensures that $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) \subseteq \pi_{S_i \cup S_j}(\Gamma)$ and the second independence condition $\forall \mathcal{C} \in \Gamma\colon \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma)\colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma)\colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ ensures that $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) \supseteq \pi_{S_i \cup S_j}(\Gamma)$.

Example 174 *According to Theorem 20, a non-equivalence between the projection of an incomplete clustering* Γ *on* $S_i \cup S_j$ *and the integration of the two individually projected clusterings* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *exists if the two clusterings* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are not independent with respect to* Γ*. A dependency can be caused by two reasons. First some alternative of* Γ *has a cluster that overlaps with* S_i *and* S_j*. Second for some alternative of* $\pi_{S_i}(\Gamma)$ *and some alternative of* $\pi_{S_j}(\Gamma)$ *there is no alternative of* $\pi_{S_i \cup S_j}(\Gamma)$ *that corresponds to their integration, i.e. a correlation between elements of* S_i *and elements of* S_j *has been lost by the projection.*

To demonstrate the first reason, we consider the integration scenario presented in Figure 8.3(a). In this sample, the incomplete clustering Γ *is projected on the sets* $S_1 = \{a, b, e, f\}$ *and* $S_2 = \{c, d, g, h\}$.



(a) Integration scenario 1



(b) Integration scenario 2

Figure 8.3.: Two sample scenarios in which integrating the results two individually computed projections of the incomplete clustering Γ on the sets $S_1 = \{a, b, e, f\}$ and $S_2 = \{c, d, g, h\}$ is in not equivalent to compute the projection of Γ on $S_1 \cup S_2$

Since some alternatives of Γ *contain the cluster* $\langle b, c\rangle$ *and these cluster's elements belong to* S_1 *and* S_2*, the result of integrating* $\Gamma_1 = \pi_{S_1}(\Gamma)$ *and* $\Gamma_2 = \pi_{S_2}(\Gamma)$ *is not equivalent to* Γ*.*

To demonstrate the second reason, we consider the integration scenario presented in Figure 8.3(b). In this sample, the incomplete clustering Γ *(note this clustering is different to the one that is used in the first scenario) is projected on the sets* $S_1 = \{a, b, e, f\}$ *and* $S_2 = \{c, d, g, h\}$*. In this scenario, no alternative of* Γ *contains a cluster whose elements belong to* S_1 *and* S_2*. However,* Γ *models correlations between some elements in* S_1 *and some elements in* S_2 *and these correlations are lost by projecting* Γ *on these sets individually. For instance, the existence of cluster* $\langle c, g\rangle$ *implicates the existence of cluster* $\langle a, e, f\rangle$*, i.e.* $\forall \mathcal{C} \in \Gamma\colon \langle c, g\rangle \in \mathcal{C} \Rightarrow \langle a, e, f\rangle \in \mathcal{C}$*. Because the lost correlation is not reconstructed by the integration operator, the result of integrating* $\Gamma_1 = \pi_{S_1}(\Gamma)$ *and* $\Gamma_2 = \pi_{S_2}(\Gamma)$ *is not equivalent to* Γ*.*

Theorem 3 can be adapted to incomplete clusterings in the same way as Theorem 2. Since both theorems are proved by the property of domination and because this property considers the probabilities of probabilistic clusterings in its definition, both theorems can be adopted to probabilistic clusterings by simply replacing any appearance of the incomplete clustering Γ with the appearance of the probabilistic clustering $\mathfrak{C}$.

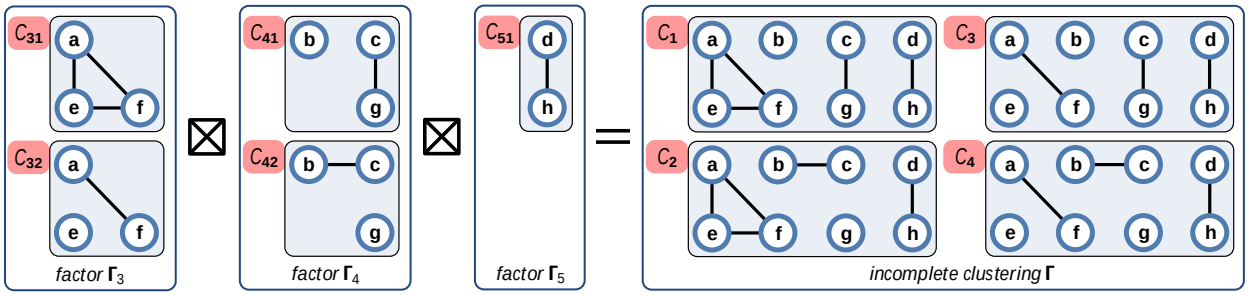


Figure 8.4.: Sample of a range-disjoint factorization

8.3. Factorization of Uncertain Clusterings

Because the alternatives of an uncertain clustering often considerably overlaps and it is usually even impossible to store them separately (in our experiments we worked on probabilistic clusterings with around $7.6 \cdot 10^{44}$ alternatives), we require for a more succinct representation of uncertain clusterings. One way to improve the succinctness of representation is to factorize an uncertain clustering into its independent components.

8.3.1. Factorization based on Absolute Independence

In this section, we define factorization based on the integration operator. Since the integration operator considers its input clusterings to be mutual independent, we automatically restrict our consideration to factorizations into the absolutely independent components of an uncertain clustering. We will shortly present a factorization based on conditional independence in Section 8.3.2.

Definition 64 (Factorization of an Incomplete Clustering): *A set of incomplete clusterings* $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *is called a factorization of the incomplete clustering* Γ*, iff* Γ *can be reconstructed from* $\mathcal{F}$ *by using the integration operator, i.e. iff* $\Gamma = \boxtimes\mathcal{F}$*. If* $\mathcal{F}$ *is a factorization of* Γ*, the incomplete clusterings in* $\mathcal{F}$ *are called factors of* Γ *and the sets in* $\mathcal{F}^{set} = \{rng(\Gamma_{F_i}) \mid \Gamma_{F_i} \in \mathcal{F}\}$ *are called factor sets of* Γ*. A factorization is called* range-disjoint*, if all its factor sets are pairwise disjoint.*

Example 175 *For illustration, the integration scenario that is described in Example 172 presents a factorization of* Γ *into* Γ_1 *and* Γ_2 *because the can be reconstructed from the latter. Since both factor sets contain the elements 'd' and 'h' the factorization is not range-disjoint.*

Note, an incomplete clustering can have several factorizations.

Theorem 21 *Let* Γ *be an incomplete clustering. Every non range-disjoint factorization* $\mathcal{F}_i$ *of* Γ *can be further factorized into a range-disjoint factorization* $\mathcal{F}_j$ *of* Γ*.*

Theorem 21 is proved by Proof 20 in Section A.3.

Example 176 *Theorem 21 can be simply illustrated by the non range-disjoint factorization from Example 172. Both factor sets contain the elements 'd' and 'h'. Moreover these two elements form the same*

cluster in all two possible clusterings of Γ_1 *and form the same cluster in all two possible clusterings of* Γ_2*. As a consequence, this cluster can be extracted from both factors into an own factor. The resultant factors* $\Gamma_3 = \{\mathcal{C}_{31}, \mathcal{C}_{32}\}$*,* $\Gamma_4 = \{\mathcal{C}_{41}, \mathcal{C}_{42}\}$*, and* $\Gamma_5 = \{\mathcal{C}_{51}\}$ *as well as their integration result are graphically presented in Figure 8.4. Since the factor sets are now disjoint, the resultant factorization is range-disjoint. As you can see, the integration result remains unchanged, although we transform the non range-disjoint factorization* $\mathcal{F}_1 = \{\Gamma_1, \Gamma_2\}$ *into the range-disjoint factorization* $\mathcal{F}_2 = \{\Gamma_3, \Gamma_4, \Gamma_5\}$*.*

A factorization is called *complete* if none of its factors cannot be factorized further on.

Theorem 22 *A complete factorization is range-disjoint.*

Theorem 22 is proved by Proof 21 in Section A.3.

Finding a complete factorization is sometimes not straightforward, but transforming a non range-disjoint factorization into a range-disjoint factorization is trivial (see Example 176). Therefore, in the remainder of this thesis we restrict to range-disjoint factorizations (complete or not complete).

From the given definition of a factorization we can derive some interesting observations:

Theorem 23 *Let* $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *be a range-disjoint factorization of the incomplete clustering* Γ*, then the following statements are true:*

- *Each factor* $\Gamma_F \in \mathcal{F}$ *can be computed as:* $\Gamma_F = \pi_F(\Gamma)$*.*
- *The factor sets form a factorization of every alternative of* Γ*, i.e. for each* $\mathcal{C} \in \Gamma$ *the set* $\{\pi_F(\mathcal{C}) \mid F \in \mathcal{F}^{set}\}$ *is a factorization of* $\mathcal{C}$*.*
- *Each factor set is a subset of the range of* Γ*, i.e.* $\forall F \in \mathcal{F}^{set} \colon F \subseteq rng(\Gamma)$
- *Each element* $e \in rng(\Gamma)$ *is covered by exact one factor set, i.e.* $\bigcup \mathcal{F}^{set} = rng(\Gamma)$
- *The factor sets* $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ *form a partition of* $rng(\Gamma)$ *and hence form a cluster-disjoint clustering of* $rng(\Gamma)$*.*
- *Each two factors* Γ_{F_i} *and* Γ_{F_j} *are mutually independent with respect to* Γ*, i.e.* $\forall \mathcal{C}_i \in \Gamma_{F_i} \colon \forall \mathcal{C}_j \in \Gamma_{F_j} \colon \exists \mathcal{C} \in \Gamma \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ *and* $\forall \mathcal{C} \in \Gamma \colon \exists \mathcal{C}_i \in \Gamma_{F_i} \colon \exists \mathcal{C}_j \in \Gamma_{F_j} \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$*.*
- *Each factor* $\Gamma_F \in \mathcal{F}$ *is dominated by* Γ*, i.e.* $\forall \mathcal{C}_F \in \Gamma_F \colon \exists \mathcal{C} \in \Gamma \colon \mathcal{C}_F \prec \mathcal{C}$ *and* $\forall \mathcal{C} \in \Gamma \colon \exists \mathcal{C}_F \in \Gamma_F \colon \mathcal{C}_F \prec \mathcal{C}$*.*
- *No possible cluster of* Γ *contains elements from different factor sets, i.e.* $\forall C \in possCl(\Gamma) \colon \forall F_i, F_j \in \mathcal{F}^{set} \colon \neg(C \otimes F_i) \wedge \neg(C \otimes F_j) \Rightarrow F_i = F_j$*.*

Theorem 23 is proved by Proof 22 in Section A.3.

Since the factor sets can be derived from the factors (each factor set is the range of a factor) and the factors can be derived from the factorized clustering using the factor sets, a factorization can be clearly described by the set of factors or can be clearly described by the factorized clustering in combination with the factor sets.

A factorization of a probabilistic clustering is defined accordingly to the incomplete case.

Definition 65 (Factorization of a Probabilistic Clustering): *A set of probabilistic clusterings* $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ *is called a factorization of the probabilistic clustering* $\mathfrak{C}$*, iff* $\mathfrak{C}$ *can be reconstructed from* $\mathcal{F}$ *by using the integration operator, i.e.* $\mathfrak{C} = \boxtimes\mathcal{F}$*. If* $\mathcal{F}$ *is a factorization of* $\mathfrak{C}$*, the probabilistic clusterings in* $\mathcal{F}$ *are called factors of* $\mathfrak{C}$ *and the sets in* $\mathcal{F}^{set} = \{rng(\mathfrak{C}_{F_i}) \mid \mathfrak{C}_{F_i} \in \mathcal{F}\}$ *are called factor sets of* $\mathfrak{C}$*.*

The definition of range-disjointness and completeness can be adopted from incomplete clustering to probabilistic clusterings without adaptations.

Besides the observations that we have made for a factorization of an incomplete clustering, we can make some additional observations for a factorization of a probabilistic clustering.

Theorem 24 *Let* $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ *be a range-disjoint factorization of the probabilistic clustering* $\mathfrak{C} = (\Gamma, Pr)$ *where* $\forall i \in \{1, \ldots, k\}\colon \mathfrak{C}_{F_i} = (\Gamma_{F_i}, Pr_i)$*, then the following statements are true:*

- *Each factor* $\mathfrak{C}_{F_i} \in \mathcal{F}$ *can be computed as:* $\mathfrak{C}_{F_i} = \pi_{F_i}(\mathfrak{C})$*.*
- $\{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *is a range-disjoint factorization of* Γ*.*
- *The probabilities of the original clustering alternatives are preserved by the factorization, i.e.* $\forall \mathfrak{C}_{F_i} \in \mathcal{F}\colon \forall \mathcal{C}_i \in$ *C-All*$(rng(\mathfrak{C}_{F_i}))\colon Pr_i(\mathcal{C}_i) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \prec \mathcal{C}} Pr(\mathcal{C}) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \subseteq \mathcal{C}} Pr(\mathcal{C})$ *(the last equality holds because of range-disjointness).*
- *The factors are mutually independent with respect to* $\mathfrak{C}$*, i.e.* $\forall \mathfrak{C}_{F_i}, \mathfrak{C}_{F_j} \in \mathcal{F}\colon \forall \mathcal{C}_i \in$ *C-All*$(rng(\mathfrak{C}_{F_i}))\colon \forall \mathcal{C}_j \in$ *C-All*$(rng(\mathfrak{C}_{F_j}))\colon Pr_i(\mathcal{C}_i) \times Pr_j(\mathcal{C}_j) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}} Pr(\mathcal{C}) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \boxtimes \mathcal{C}_j \subseteq \mathcal{C}} Pr(\mathcal{C})$*.*

Theorem 24 is proved by Proof 23 in Section A.3.

In a range-disjoint factorization, the factor sets corresponds to the partition classes of a partition of the considered range. Thus, we can adopt properties from partitions to the set of all factor sets and hence to factorizations in general. One interesting property are the two unidirectional relationships *is finer than* and *is coarser than*. Following [Maz10], a partition is finer than another partition, if each partition class of the first is a subset of a partition class of the second. On the contrary, a partition is coarser than another partition if the second is finer than the first. We adopt that notion from partitions to factor sets. Let $\mathcal{F}_1^{set}$ and $\mathcal{F}_2^{set}$ be the factor sets of two range-disjoint factorizations. If every factor set of $\mathcal{F}_1^{set}$ is the subset of some factor set of $\mathcal{F}_2^{set}$, $\mathcal{F}_1^{set}$ is called to be finer than $\mathcal{F}_2^{set}$ and $\mathcal{F}_2^{set}$ is called to be coarser than $\mathcal{F}_1^{set}$.

An adoption to the factorization itself, however, is only meaningful for factorizations of the same clustering.

Definition 66 (Finer/Coarser Relationships of Factorizations): *Let* Γ *be an incomplete clustering and let* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *be two range-disjoint factorizations of* Γ *that have the factor sets* $\mathcal{F}_1^{set}$ *and* $\mathcal{F}_2^{set}$ *respectively. If* $\mathcal{F}_1^{set}$ *is finer than* $\mathcal{F}_2^{set}$*,* $\mathcal{F}_1$ *is called to be a finer factorization of* Γ *than* $\mathcal{F}_2$ *and* $\mathcal{F}_2$ *is called to be coarser factorization of* Γ *than* $\mathcal{F}_1$*.*

These relationships are defined for certain clusterings and probabilistic clusterings accordingly.

The finer/coarser relationship of all possible range-disjoint factorizations of a given clustering (whether certain or uncertain) form a partial order where the complete factorization of that clustering is the bottom of this order and the clustering itself is the top of this order.

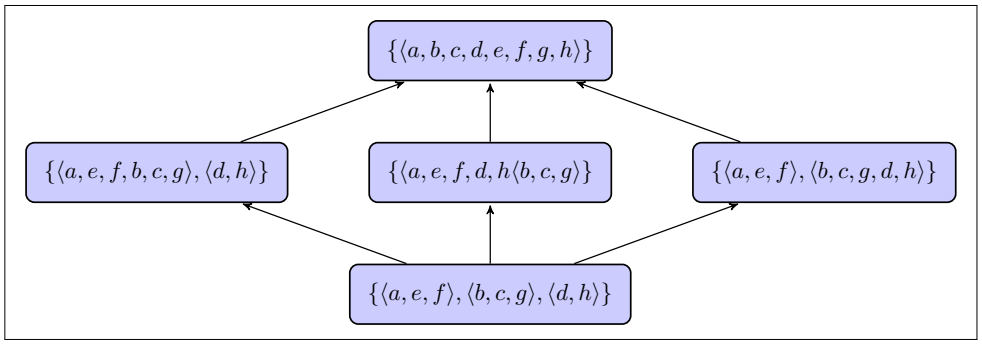


Figure 8.5.: Hasse-diagram presenting the partial order of the *is finer than* relationship between the factorizations of the incomplete clustering Γ

Example 177 *For illustration, we consider the incomplete clustering Γ from Figure 8.4. Besides the trivial factorization $\mathcal{F}_1$ of Γ into Γ itself ($\mathcal{F}_1^{set} = \{\{a,b,c,d,e,f,g,h\}\}$) and besides the complete factorization $\mathcal{F}_2$ of Γ into to three incomplete clusterings Γ_3, Γ_4, and Γ_5 ($\mathcal{F}_2^{set} = \{\{a,e,f\},\{b,c,g\},\{d,h\}\}$), Γ can be factorized in three other ways. Each one by using the factor set $\mathcal{F}_3^{set} = \{\{a,e,f,b,c,g\},\{d,h\}\}$ (factorization $\mathcal{F}_3$), the factor set $\mathcal{F}_4^{set} = \{\{a,e,f,d,h\},\{b,c,g\}\}$ (factorization $\mathcal{F}_4$), and the factor set $\mathcal{F}_5^{set} = \{\{a,e,f\},\{b,c,g,d,h\}\}$ (factorization $\mathcal{F}_5$). The first factorization is trivially coarser than the latter four and the second factorization is finer than all other. Between the factorizations $\mathcal{F}_3$, $\mathcal{F}_4$, and $\mathcal{F}_5$, however, no one is finer (and hence no one is coarser) than some of the others. Thus, these five factorizations form the partial order that is depicted by the Hasse-diagram in Figure 8.5.*

If one factorization is finer than another, the first can be computed by further factorizing the second and the second can be computed by integrating some factors of the first.

Theorem 25 *Let Γ be an incomplete clustering and let $\mathcal{F}_m$ and $\mathcal{F}_n$ be two range-disjoint factorizations of Γ. If $\mathcal{F}_m$ is a finer factorization of Γ than $\mathcal{F}_n$, the factors $\{\Gamma_{F_m} \mid \Gamma_{F_m} \in \mathcal{F}_m, rng(\Gamma_{F_m}) \subseteq rng(\Gamma_F)\}$ form a range-disjoint factorization of the factor $\Gamma_F \in \mathcal{F}_n$.*

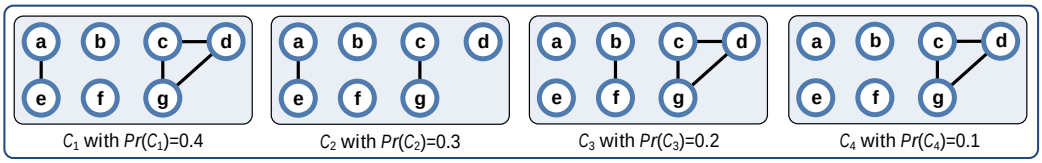
Theorem 25 is proved by Proof 24 in Section A.3.

Similar theorems can be proved for certain clusterings and probabilistic clusterings accordingly.

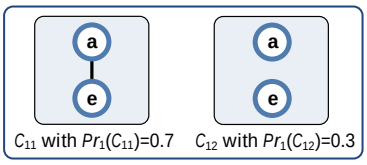
8.3.2. Factorization based on Conditional Independence

In theory, a probabilistic clustering $\mathfrak{C}$ can be factorized into a set of probabilistic clusterings $\mathcal{F}$ if there exist any function that can be used to reconstruct $\mathfrak{C}$ from $\mathcal{F}$. As mentioned above, by using the integration operator for reconstruction, we automatically restrict to factorizations that are based on absolutely independence. Nevertheless, a factorization can be also defined on the basis of conditional independence. In this case, reconstruction is done by the use of conditional probability distributions and factorization is based on the theorem of Bayes, i.e. $Pr_{ij}(\mathcal{C}_i \boxtimes \mathcal{C}_j) = Pr_{i|j}(\mathcal{C}_i, \mathcal{C}_j) \times Pr_j(\mathcal{C}_j)$. A common way to compute such distributions is the *chain rule factorization* described by Castillo et al. [CGH97].

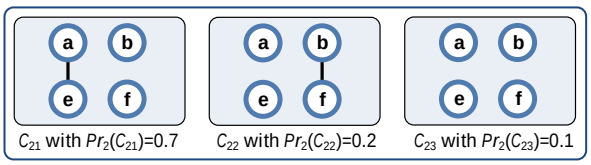
A general definition of factorizations that is also applicable to factorizations that are based on conditional independence is given by Borgelt et al. [BSK09]. According to them, a probability distribution Pr on a domain U is factorizable with respect to a set $\mathcal{M} = \{M_1, \ldots, M_k\} \subseteq 2^U$, if it can be computed



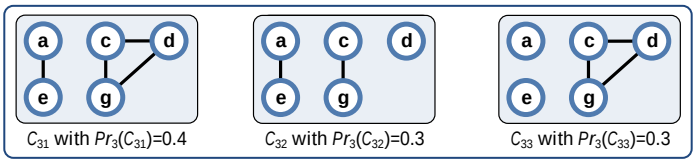
(a) The probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ with $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$



(b) $\mathfrak{C}_1 = (\Gamma_1, Pr_1)$ with $\Gamma_1 = \{\mathcal{C}_{11}, \mathcal{C}_{12}\}$



(c) $\mathfrak{C}_2 = (\Gamma_2, Pr_2)$ with $\Gamma_2 = \{\mathcal{C}_{21}, \mathcal{C}_{22}, \mathcal{C}_{23}\}$



(d) $\mathfrak{C}_3 = (\Gamma_3, Pr_3)$ with $\Gamma_3 = \{\mathcal{C}_{31}, \mathcal{C}_{32}, \mathcal{C}_{33}\}$

Figure 8.6.: Sample for factorization based on conditional independence

by the product of k non-negative functions $\phi_M, M \in \mathcal{M}$, i.e. iff

$$\forall u \in U\colon Pr(u) = \prod\nolimits_{M \in \mathcal{M}} \phi_M(u) \tag{8.5}$$

Example 178 *For illustration, we consider the probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ from Figure 8.6(a). It is obvious that no part of the clustering is absolutely independent from the rest. Thus, according to Definition 65 this clustering cannot be factorized further on. In contrast, if we also take conditional independence into account, this clustering can be factorized into the three probabilistic clusterings $\mathfrak{C}_1$, $\mathfrak{C}_2$, and $\mathfrak{C}_3$ that are presented in Figure 8.6(b), Figure 8.6(c), and Figure 8.6(d) respectively. This factorization can be performed because the true clusterings of the element sets $\{b, f\}$ and $\{c, d, g\}$ are mutual independent if the true clustering of the elements $\{a, e\}$ is given.*

As presented in Table 8.1, if the true clustering of the factor set $\{a, e\}$ is given as $\{\langle a, e\rangle\}$, the true clustering of the factor set $\{b, f\}$ is always $\{\langle b\rangle, \langle f\rangle\}$ and the true clustering of the factor set $\{c, d, g\}$ is either $\{\langle c, d, g\rangle\}$ (with probability $4/7$) or $\{\langle c, g\rangle, \langle d\rangle\}$ (with probability $3/7$). In contrast, if the true clustering of $\{a, e\}$ is given as $\{\langle a\rangle, \langle e\rangle\}$, the true clustering of $\{b, f\}$ is either $\{\langle b, f\rangle\}$ (with probability $2/3$) or $\{\langle b\rangle, \langle f\rangle\}$ (with probability $1/3$) and the true clustering of $\{c, d, g\}$ is always $\{\langle c, d, g\rangle\}$. Thus, in both cases, the true clustering of $\{b, f\}$ is independent of the true clustering of $\{c, d, g\}$.

Therefore, the original clustering $\mathfrak{C}$ can be reconstructed by using the functions:

$$\phi_1(\mathcal{C}) = 1/Pr_1(\pi_{rng(\mathfrak{C}_1)}(\mathcal{C})) \qquad \phi_2(\mathcal{C}) = Pr_2(\pi_{rng(\mathfrak{C}_2)}(\mathcal{C})) \qquad \phi_3(\mathcal{C}) = Pr_3(\pi_{rng(\mathfrak{C}_3)}(\mathcal{C}))$$

The probability of a possible clustering $\mathcal{C} \in \Gamma$ is consequently computed as:

$$Pr(\mathcal{C}) = \prod\nolimits_{i \in \{1,\ldots,3\}} \phi_i(\mathcal{C}) = \frac{Pr_2(\pi_{rng(\mathfrak{C}_2)}(\mathcal{C})) \times Pr_3(\pi_{rng(\mathfrak{C}_3)}(\mathcal{C}))}{Pr_1(\pi_{rng(\mathfrak{C}_1)}(\mathcal{C}))}$$

	Case 1: $\{\langle a, e\rangle\}$	**Case 2: $\{\langle a\rangle, \langle e\rangle\}$**
$\{b, f\}$	$\{\langle b\rangle, \langle f\rangle\}$: 1.0	$\{\langle b, f\rangle\}$: 2/3 $\{\langle b\rangle, \langle f\rangle\}$: 1/3
$\{c, d, g\}$	$\{\langle c, d, g\rangle\}$: 4/7 $\{\langle c, g\rangle, \langle d\rangle\}$: 3/7	$\{\langle c, d, g\rangle\}$: 1.0

Table 8.1.: The probabilities of the true clusterings of the factor sets $\{b, f\}$ and $\{c, d, g\}$ conditioned by the event that the true clustering of the factor set $\{a, e\}$ is known

A factorization of $\mathfrak{C}$ that is based on the considered case of conditional independence can be stored even more efficiently by discarding the first factor and by defining the functions $\phi_2(\mathcal{C})$ and $\phi_3(\mathcal{C})$ as:

$$\phi_2(\mathcal{C}) = Pr_2(\pi_{rng(\mathfrak{C}_2)}(\mathcal{C})) \qquad \phi_3(\mathcal{C}) = \begin{cases} Pr_3(\pi_{rng(\mathfrak{C}_3)}(\mathcal{C}))/0.3, & \textit{iff } \pi_{rng(\mathfrak{C}_3)}(\mathcal{C})) = \mathcal{C}_{33}, \\ Pr_3(\pi_{rng(\mathfrak{C}_3)}(\mathcal{C}))/0.7, & \textit{else.} \end{cases}$$

Note that factorizing a probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ based on the integration operator as we have defined it in Definition 65 is a special case of the definition made by Borgelt et al. [BSK09] because it corresponds to Equation 8.5 if we consider the domain U to be Γ, consider $\mathcal{M}$ as the set of all factor sets, i.e. $\mathcal{M} = \mathcal{F}^{set}$, and use the functions $\phi_M(\mathcal{C}) = Pr_M(\pi_M(\mathcal{C}))$ for all $M \in \mathcal{M}$ for reconstruction. In other words, the probability of a possible clustering $\mathcal{C}$ is computed by the product of the probabilities of its projection on the individual factor sets, because $\mathcal{C}$ is only the true clustering of $\mathfrak{C}$ if its projection $\pi_{F_i}(\mathcal{C})$ is the true clustering for each factor $\mathfrak{C}_{F_i} \in \mathcal{F}$ and all factors are assumed to be mutually independent.

Because factorizations that are based on conditional independence are much more complex to represent and process than factorizations that are based on absolute independence, we restrict to the latter in this thesis. Moreover, the most pairwise duplicate decisions are actually mutual independent. Therefore, factorizing a probabilistic clustering in its absolutely independent components is already sufficient to reduce the representation complexity and querying complexity of indeterministic deduplication results to a large extent which makes a consideration of stronger factorization concepts superfluous.

8.4. Modeling Indeterministic Deduplication Results

The result of an indeterministic duplicate detection process is an uncertain clustering. Thus, before we can store the deduplication result into a probabilistic database we need to merge the detected duplicates. Theoretically, for an uncertain clustering a duplicate merging process is defined by the uncertain version of a conventional merging process. As a consequence, it changes the uncertain clustering into an uncertain set of database entities by merging each cluster of every possible clusterings separately. Each two possible clusterings differ in at least two clusters. Thus, provided that the used merge function is injective, i.e. no two different clusters are merged to the same entity, from each possible clustering another entity set results. Typically, the merge of one cluster does not depend on the merge of the other clusters in the same clustering. In this case, we need to merge each possible cluster only once and do not need to merge it per possible clustering it belongs to.

possible clusterings of $\mathfrak{C}$	Pr_1
$\mathcal{C}_1$=$\{\langle e_1,e_2\rangle,\langle e_3\rangle,\langle e_4\rangle,\langle e_5\rangle,\langle e_6\rangle,\langle e_7\rangle\}$	0.12
$\mathcal{C}_2$=$\{\langle e_1,e_2\rangle,\langle e_3,e_4\rangle,\langle e_5\rangle,\langle e_6\rangle,\langle e_7\rangle\}$	0.18
$\mathcal{C}_3$=$\{\langle e_1,e_2\rangle,\langle e_3,e_4,e_5\rangle,\langle e_6\rangle,\langle e_7\rangle\}$	0.3
$\mathcal{C}_4$=$\{\langle e_1,e_2\rangle,\langle e_3\rangle,\langle e_4\rangle,\langle e_5\rangle,\langle e_6,e_7\rangle\}$	0.08
$\mathcal{C}_5$=$\{\langle e_1,e_2\rangle,\langle e_3,e_4\rangle,\langle e_5\rangle,\langle e_6,e_7\rangle\}$	0.12
$\mathcal{C}_6$=$\{\langle e_1,e_2\rangle,\langle e_3,e_4,e_5\rangle,\langle e_6,e_7\rangle\}$	0.2

(a) Probabilistic clustering $\mathfrak{C} = (\Gamma, Pr_1)$

possible entity sets of E	Pr_2
$\mu(\mathcal{C}_1)$=$\mathfrak{E}_1$=$\{f_\mu(\{e_1,e_2\}),e_3,e_4,e_5,e_6,e_7\}$	0.12
$\mu(\mathcal{C}_2)$=$\mathfrak{E}_2$=$\{f_\mu(\{e_1,e_2\}),f_\mu(\{e_3,e_4\}),e_5,e_6,e_7\}$	0.18
$\mu(\mathcal{C}_3)$=$\mathfrak{E}_3$=$\{f_\mu(\{e_1,e_2\}),f_\mu(\{e_3,e_4,e_5\}),e_6,e_7\}$	0.3
$\mu(\mathcal{C}_4)$=$\mathfrak{E}_4$=$\{f_\mu(\{e_1,e_2\}),e_3,e_4,e_5,f_\mu(\{e_6,e_7\})\}$	0.08
$\mu(\mathcal{C}_5)$=$\mathfrak{E}_5$=$\{f_\mu(\{e_1,e_2\}),f_\mu(\{e_3,e_4\}),e_5,f_\mu(\{e_6,e_7\})\}$	0.12
$\mu(\mathcal{C}_6)$=$\mathfrak{E}_6$=$\{f_\mu(\{e_1,e_2\}),f_\mu(\{e_3,e_4,e_5\}),f_\mu(\{e_6,e_7\})\}$	0.2

(b) Probabilistic entity set $\mathbf{E} = (\mathcal{E}, Pr_2)$

Figure 8.7.: A probabilistic clustering $\mathfrak{C}$ and its corresponding probabilistic entity set **E** that results from applying the probabilistic version of a duplicate merging process μ that uses the merge function f_μ to $\mathfrak{C}$

Example 179 *For illustration we consider the probabilistic clustering $\mathfrak{C}$ from Figure 8.7(a) and perform the duplicate merging phase by applying the probabilistic version of a merging process μ that uses the merge function f_μ to $\mathfrak{C}$. Applying μ to a certain clustering implicates to apply f_μ to each of its clusters. As a consequence, applying μ to an uncertain clustering means to apply f_μ to each of its possible clusters. By doing so from each possible clustering another entity set is computed. For instance, from merging the cluster of the possible clustering $\mathcal{C}_3 = \{\langle e_1,e_2\rangle,\langle e_3,e_4,e_5\rangle,\langle e_6\rangle,\langle e_7\rangle\}$ the entity set $\mathfrak{E}_3 = \{f_\mu(\{e_1,e_2\}),f_\mu(\{e_3,e_4,e_5\}),e_6,e_7\}$ result. The probabilistic entity set **E** that results from applying μ to $\mathfrak{C}$, i.e. the indeterministic deduplication results, is presented in Figure 8.7(b).*

Obviously, the alternatives of a probabilistic entity set are mutually exclusive, i.e. for each possible world of the database at most one alternative of the uncertain entity set is allowed to be part of this world. Moreover, it is assumed that the 'true' deduplication result is one of the possible entity sets, i.e. for each possible world of the database at least one alternative of the uncertain entity set must be part of this world. For that purpose, we introduce the concept of *entity set complementation*.

Definition 67 (Entity Set Complementation): *The entity sets of $\mathcal{S} = \{\mathfrak{E}_1,\ldots,\mathfrak{E}_k\}$ are complementing (short $\mathfrak{cpl}(\mathcal{S})$) in a probabilistic database PDB $= (\boldsymbol{W}, Pr)$, iff their existences are jointly exhaustive and mutually exclusive events in $\boldsymbol{W}$. Naturally speaking either all entities of $\mathfrak{E}_1$ exist and none of $\mathfrak{E}_2,\ldots,\mathfrak{E}_k$ that do not belong to $\mathfrak{E}_1$, or all entities of $\mathfrak{E}_2$ exist and none of $\mathfrak{E}_1,\mathfrak{E}_3,\ldots,\mathfrak{E}_k$ that do not belong to $\mathfrak{E}_2$, and so on:* $\mathfrak{cpl}(\mathcal{S}) \Leftrightarrow \forall W \in \boldsymbol{W}\colon \exists \mathfrak{E} \in \mathcal{S}\colon \mathfrak{E} \subseteq W \wedge \forall e \in \bigcup \mathcal{S} - \{\mathfrak{E}\}\colon e \notin W$.

According to this definition, an entity set complementation can only be correctly modeled within a probabilistic database if the used representation system is able to model a mutual exclusion between the existence of several sets of database entities. As a consequence, many probabilistic representation systems such as TI-databases, AOR-databases, AOR?-databases, or BID-databases are not able to model entity set complementations if they are used stand-alone (without views). Recall, this was even one of the reasons that we have developed approaches for deterministic duplicate detection in probabilistic databases in the previous chapter. Nevertheless, more powerful representation systems such as pc-databases can be used to model entity set complementations within a probabilistic database. Furthermore, the views that are required to model an entity set complementation within a BID-database are of a simple nature. For that reason, we consider a modeling within pc-databases in Section 8.4.2 and

then discuss a modeling within BID-databases in Section 8.4.3. For presentation reasons we restrict to certain databases as deduplication input in these sections and consider the peculiarities of indeterministic deduplication on probabilistic databases separately in Section 8.6. Moreover, before describing these two modeling approaches in detail, we shortly introduce a concept that stores useful information on the performed deduplication process.

8.4.1. The Merge-Base

As discussed in Section 7.7.1 it can be useful to retrace the origin of tuple dependencies. Moreover, it can be useful to know from which entities a particular entity has been merged from, for example in order to revert false positives. For this purpose, we additionally store a so-called *merge base* for each entity. The merge base is an entity set that contains all entities the considered entity is merged from. This information helps subsequent data applications to identify the reasons of the introduced dependencies because two entities with an overlapping merge base cannot belong to the same possible world.

Since each of the merged entities can be the result of a former deduplication process, the merge base of an entity e_r (denoted as $mbase(e_r)$) is composed by the set union of the merge bases of the entities it is merged from, i.e. $e_r = f_\mu(S) \Rightarrow mbase(e_r) = \bigcup_{e_s \in S} mbase(e_s)$. If for some entity no merge base is available, it is treated as an empty set. Actually, the merge base of a database entity is a kind of lineage [CWW00] (also known as provenance [BT07, CCT09]), but there is an essential difference to the meaning of lineage as it used by the ULDB model [BSHW06]. In ULDB an entity only exists in a possible world if its lineage condition is evaluated to true for this world. Thus, an entity e_r with the lineage $\lambda(e_r) = e_s$ only belongs to a world, if the entity e_s belongs to this world as well. In the merge base, however, the correlation between entities is modeled in the opposite way because an entity can only belong to a possible world if none of the entities that is referenced in its merge base belong to this world. For that reason, the merge base cannot be modeled by the using the standard lineage of ULDB.

All possible clusterings of an indeterministic duplicate detection result are defined on the same range. As a consequence, every entity of the input database is considered in each of the possible deduplication results: either as it is or as a part of the merge base of a newly created entity. Thus, the accumulative merge base of all entities is equivalent for each possible deduplication results, i.e. let $\mathfrak{E}_p$ and $\mathfrak{E}_q$ be two complementing sets that are alternatives of the same uncertain deduplication result, it holds that $\bigcup_{e \in \mathfrak{E}_p} mbase(e) = \bigcup_{e \in \mathfrak{E}_q} mbase(e)$.

If the merge base should be stored within a relational database, we use an extra table that we call *MBase-table*. This table has the two attributes *'DEI'* and *'base'*. The attribute *'DEI'* describes the considered database entity and the attribute *'base'* describes an entity of its merge base. As consequence, we need $|S|$ tuples of the *MBase-table* (one per entity $e \in S$) to store the merge base $mbase(e) = S$.

8.4.2. Modeling Indeterministic Deduplication Results within PC-Databases

In a pc-database, an entity set complementation $\mathfrak{cpl}(\mathcal{S})$ can be modeled by the use of some extra variables. For this purpose, we introduce one variable per entity set complementation. This variable has as many possible values as the set $\mathcal{S}$ has elements because we need another variable assignment per complementing set. Of course, since one of these sets must belong to the 'true' world, the probabilities of all

Product

DEI	name	class	sales
e1	product A	R	7
e2	product A	R	9
e3	product B	R	4
e4	product BB	R	6
e5	product BBB	R	11
e6	product C	S	2
e7	product CC	S	4

(a) Initial database

Product

RK	DEI	name	class	sales	p	condition
1	e8	product A	R	8	1.0	true
2	e3	product B	R	4	0.2	X=1 v X=4
3	e4	product BB	R	6	0.2	X=1 v X=4
4	e5	product BBB	R	11	0.5	X=1 v X=2 v X=4 v X=5
5	e6	product C	S	2	0.6	X=1 v X=2 v X=3
6	e7	product CC	S	4	0.6	X=1 v X=2 v X=3
7	e9	product BB	R	5	0.3	X=2 v X=5
6	e10	product BBB	R	7	0.5	X=3 v X=6
7	e11	product CC	S	3	0.4	X=4 v X=5 v X=6

World-Table

var	value	Prob
X	1	0.12
X	2	0.18
X	3	0.3
X	4	0.08
X	5	0.12
X	6	0.2

MBase-Table

DEI	base
e8	e1
e8	e2
e9	e3
e9	e4
e10	e3
e10	e4
e10	e5
e11	e6
e11	e7

(b) Deduplicated database

Figure 8.8.: Modeling a non-factorized indeterministic deduplication result by using a pc-database

the variable's possible values sum up to 1. Whereas the database entities of the first set are conditioned by an assignment of the variable to the first possible value, the entities of the second set are conditioned by an assignment of the variable to the second possible value, and so forth. Recall that entities can belong to several sets and therefore can be conditioned by several assignments, i.e. the conditions of these entities are disjunctions of atomic conditions that each proves the assignment of the considered variable to a particular possible value. Note that in this case, the condition holds for the whole database entity and hence for all of its possible instances. Therefore, we consider that kind of condition on entity granularity and thus add the same condition to all tuples that represent the same database entity. Due to the possible values of the newly introduced variable are mutually exclusive, this dependency holds for the conditioned entity sets as well. Since the variable is not maybe, one of the entity sets exists for sure, i.e. they are jointly exhaustive. In other words, we use the system-specific 'complementation' of a variable's possible values in order to model complementing sets of database entities.

Example 180 *For illustration we consider the conventional database table* 'Product' *that is presented in Figure 8.8(a). This table has the seven entities* e_1 *to* e_7*. We assume that an indeterministic duplicate detection process has been performed on these entities and exactly returns the probabilistic clustering that is presented in Figure 8.7(a). After transforming the possible clustering to entity sets by merging the entities of each cluster with the merge function* f_μ*, i.e.* $e_8 = \mu(\{e_1, e_2\})$*,*

$e_9 = \mu(\{e_3, e_4\})$, $e_{10} = \mu(\{e_3, e_4, e_5\})$, *and* $e_{11} = \mu(\{e_6, e_7\})$, *we need to model the entity set complementation* $\mathfrak{cpl}(\{\mathfrak{E}_1, \mathfrak{E}_2, \mathfrak{E}_3, \mathfrak{E}_4, \mathfrak{E}_5, \mathfrak{E}_6\})$ *with* $\mathfrak{E}_1 = \{e_8, e_3, e_4, e_5, e_6, e_7\}$, $\mathfrak{E}_2 = \{e_8, e_9, e_5, e_6, e_7\}$, $\mathfrak{E}_3 = \{e_8, e_{10}, e_6, e_7\}$, $\mathfrak{E}_4 = \{e_8, e_3, e_4, e_5, e_{11}\}$, $\mathfrak{E}_5 = \{e_8, e_9, e_5, e_{11}\}$, $\mathfrak{E}_6 = \{e_8, e_{10}, e_{11}\}$ *within a pc-database.*

For merging we assume a simple merge function that takes the longest value for the attributes 'name' *and* 'class'*, and that computes the average value for the attribute* 'sale'*. To model the given entity set complementation, we create the variable* X *that has six possible values (each one per entity set) and condition the database entities by some assignments of* X *to the corresponding possible values. The resultant pc-database is presented in Figure 8.8(b). Since we consider a certain database as input and because we consider a conflict resolution approach for merging duplicate database entities, each database entity of the produced pc-database as a certain instance. However, each entity that does not belong to all six possible deduplication results* $\mathfrak{E}_1$ *to* $\mathfrak{E}_6$ *has an uncertain existence. For that reason, entity* e_8 *is the only one that is not maybe. Due to the entities of the input database were certain, the entity conditions of the deduplicated database only consist of the disjunctions that have been computed for modeling the given entity set complementation. For instance, entity* e_5 *belongs to the four possible entity sets* $\mathfrak{E}_1$*,* $\mathfrak{E}_2$*,* $\mathfrak{E}_4$*, and* $\mathfrak{E}_5$ *and therefore is associated with a condition that evaluates to true if the variable* X *takes one of the values* 1*,* 2*,* 4*, or* 5*.*

Note that these modeling steps become a bit more complicated if we use a ULDB (Trio) or if we use a U-database (MayBMS) as representation system because in the first we need to model correlations by the use of lineage and the second uses ordinary attributes to model tuple conditions. We propose algorithms for modeling an entity set complementation within each of these systems in [PvKR13].

In pc-databases, the presence of variables in tuple conditions can be negated (e.g. $\neg(X = 1)$). As a consequence, for modeling a complementation between k entity sets we actually require only a variable with $k-1$ possible values instead of a variable with k possible values. In this case, the variable becomes maybe and the fact that one of the entity sets must exist is modeled by the used negation. Nevertheless, for query processing reasons it is desirable to minimize the complexity of condition formulas. For that reason, we use negation only for modeling complementations between two entity sets ($k = 2$). Otherwise we always use a variable with k possible values.

For modeling the deduplication result within a pc-database, we need one variable with one possible value per possible clustering. Thus, if we work on a non-factorized representation, the number of required possible values can be infeasible large in size which makes modeling an indeterministic deduplication result within a pc-database impractical (recall, in the relational database approach we store each possible value a variable can take by an own tuple in the world-table). However, this high degree of representation complexity can be dramatically reduced if the uncertain clustering is factorized into many small factors. In this case, instead of modeling a single complementation of whole worlds, we only need to model several complementations of small parts of these worlds. Since we have a set of entity set complementations instead of one, we can model the indeterministic deduplication result by creating several independent variables (one variable per factor). This in turn can considerably reduce the total number of required variable-value pairs and thus can decrease the number of tuples that we need to store in the world-table to a large extent.

possible clusterings of $\mathfrak{C}_{F_1}$	Pr_{F_1}
$\mathcal{C}_{11} = \{\langle e_1, e_2\rangle\}$	1.0

factor $\mathfrak{C}_{F_1} = (\Gamma_{F_1}, Pr_{F_1})$

possible clusterings of $\mathfrak{C}_{F_2}$	Pr_{F_2}
$\mathcal{C}_{21} = \{\langle e_3\rangle, \langle e_4\rangle, \langle e_5\rangle\}$	0.2
$\mathcal{C}_{22} = \{\langle e_3, e_4\rangle, \langle e_5\rangle\}$	0.3
$\mathcal{C}_{23} = \{\langle e_3, e_4, e_5\rangle\}$	0.5

factor $\mathfrak{C}_{F_2} = (\Gamma_{F_2}, Pr_{F_2})$

possible clusterings of $\mathfrak{C}_{F_3}$	Pr_{F_3}
$\mathcal{C}_{31} = \{\langle e_6\rangle, \langle e_7\rangle\}$	0.6
$\mathcal{C}_{32} = \{\langle e_6, e_7\rangle\}$	0.4

factor $\mathfrak{C}_{F_3} = (\Gamma_{F_3}, Pr_{F_3})$

(a) The three factors $\mathfrak{C}_{F_1}$, $\mathfrak{C}_{F_2}$, and $\mathfrak{C}_{F_3}$ of the sample probabilistic clustering $\mathfrak{C}$

Product

RK	DEI	name	class	sales	p	condition
1	e8	product A	R	8	1.0	true
2	e3	product B	R	4	0.2	X=1
3	e4	product BB	R	6	0.2	X=1
4	e5	product BBB	R	11	0.5	X=1 ∨ X=2
5	e6	product C	S	2	0.6	Y=1
6	e7	product CC	S	4	0.6	Y=1
7	e9	product BB	R	5	0.3	X=2
6	e10	product BBB	R	7	0.5	X=3
7	e11	product CC	S	3	0.4	¬ (Y=1)

World-Table

var	value	Prob
X	1	0.2
X	2	0.3
X	3	0.5
Y	1	0.6

MBase-Table

DEI	base
e8	e1
e8	e2
e9	e3
e9	e4
e10	e3
e10	e4
e10	e5
e11	e6
e11	e7

(b) Deduplicated database (factorized)

Figure 8.9.: Modeling a factorized indeterministic deduplication result by using a pc-database

Example 181 *For demonstration, we reconsider the conventional database table* 'Product' *from Figure 8.8(a), assume the probabilistic clustering* $\mathfrak{C}$ *that is presented in Figure 8.7(a), and assume the same merge function* f_μ *as it is described in the previous example. If we take a look at the probabilistic clustering* $\mathfrak{C}$*, we see that the decisions on the entity sets* $F_1 = \{e_1, e_2\}$*,* $F_2 = \{e_3, e_4, e_5\}$*, and* $F_3 = \{e_6, e_7\}$ *are mutually independent to each other. Therefore, we can factorize* $\mathfrak{C}$ *into the three factors* $\mathfrak{C}_{F_1}$*,* $\mathfrak{C}_{F_2}$*, and* $\mathfrak{C}_{F_3}$ *by using the disjoint factor sets* F_1*,* F_2*, and* F_3*. The resultant factors are presented in Figure 8.9(a). Each of these factors is transformed into a probabilistic entity set. As a consequence, instead of one we need to model three entity set complementations within the probabilistic output database. The resultant pc-database is presented in Figure 8.9(b). Since the complementation* $\mathfrak{cpl}(\{\{e_6, e_7\}, \{e_{11}\}\})$ *has only two elements, we can use a maybe variable* Y *that has only one possible value, i.e. the value* 1*. Thus, whereas the existence of the first set* $\{e_6, e_7\}$ *is conditioned by the assignment of* Y *to* 1*, the existence of the second set* $\{e_{11}\}$ *is conditioned by the negated assignment of* Y *to* 1*. If we compare the resultant pc-database with the pc-database that is presented in Figure 8.8(b), we see that the total number of variable-value pairs has been decreased from six to four. Moreover, the condition formulas are less complex.*

Note that although we have several variables in the factorized case, the atomic conditions of one tuple are always defined on the same variable. That is because we use one variable per factor and the factor sets are disjoint, i.e. each database entity only belongs to one factor and therefore is only conditioned by assignments of the factor's corresponding variable.

ProductBasic

RK	DEI	name	class	sales	p
1	e8	product A	R	8	1.0
2	e3	product B	R	4	1.0
3	e4	product BB	R	6	1.0
4	e5	product BBB	R	11	1.0
5	e6	product C	S	2	1.0
6	e7	product CC	S	4	1.0
7	e9	product BB	R	5	1.0
6	e10	product BBB	R	7	1.0
7	e11	product CC	S	3	1.0

Correlation-Table

DEI	var	value
e8	Z	1
e3	X	1
e4	X	1
e5	X	1
e5	X	2
e6	Y	1
e7	Y	1
e9	X	2
e10	X	3
e11	Y	2

World-Table

var	value	p
Z	1	1.0
X	1	0.2
X	2	0.3
X	3	0.5
Y	1	0.6
Y	2	0.4

MBase-Table

DEI	base
e8	e1
e8	e2
e9	e3
e9	e4
e10	e3
e10	e4
e10	e5
e11	e6
e11	e7

(a) Deduplicated BID-database with the two BID-tables *'ProductBasic'* and *'World-Table'* as well as the certain *'Correlation-Table'*

CREATE VIEW	Product
AS SELECT	p.DEI, p.name, p.class, p.sales
FROM	ProductBasic p, Correlation-Table c, World-Table w
WHERE	p.DEI = c.DEI
AND	c.var = w.var
AND	c.value = w.value

(b) View $Q_{Product}$ that reconstructs the pc-table *'Product'* from Figure 8.9(b) by querying the BID-database from Figure 8.10(a)

Figure 8.10.: Modeling a factorized indeterministic deduplication result by using a BID-database coupled with views

In the example described above, the number of required variable-value pairs is only decreased from $2 \times 3 = 6$ to $2 + 3 = 5$ if we do not use negation and is decreased from six to four if we use negation. This reduction is hardly worth mentioning. Nevertheless, we only consider a small example and whereas in the factorized approach we only need as many variable-value pairs as the accumulative sum of all numbers of factor alternatives (linear complexity), in the non-factorized approach we need as many variable-value pairs as the multiplication of all numbers of factor alternatives (exponential complexity). For instance, if we can factorize a probabilistic clustering into $n = 100$ factors each with $m = 2$ alternatives, we only need $n \times m = 200$ variable-value pairs by using the factorized representation, but need $m^n = 2^{100} = 1.27 \times 10^{30}$ variable-value pairs by using the non-factorized representation.

8.4.3. Modeling Indeterministic Deduplication Results within BID-Databases

Recall from Section 3.3.7 that although BID-databases are no complete representation systems by themselves, they are complete in combination with CQ queries. Moreover, recall that all tuple conditions that we compute in the approach for modeling entity set complementations within a pc-database are always disjunctive formulas where all of the formula's atomic conditions are defined on the same variable. This circumstance enables us to reconstruct these conditions by a CQ query and hence enable us to represent entity set complementations by BID-databases. For that purpose, we expand the set of database tables

that we need to represent the entities of the deduplication result by two additional tables. The first of these tables corresponds to the world-table of a pc-database where each variable represents another entity set complementation[3]. Note that the variables of the world-table are mutually independent and thus this table is a BID-table by definition. The second table connects the entities of the actual BID-database with the tuples, i.e. the variable-value pairs, of the world-table and therefore replaces the tuple conditions that are used in a pc-database. We call this table *correlation-table* in this thesis. Note that the correlation-table is certain because each element of an entity set complementation is a deterministic set of entities. Finally, we need to define one view per actual database table. This view is an SQL-query that connects the considered database table with the world-table by joining them both with the correlation-table and by finally projecting on the attributes of the considered database table.

Note, according to the conventions that we have introduced in Chapter 3, the correlation-table and the world-table would be modeled differently than in the approach described above. For instance, the world-table would contain a surrogate attribute that serves as representation key (and hence can be used to identify a variable-value pair of a corresponding pc-database) and the correlation-table would only have two attributes: one for storing the *DEI*s of the actual database entities and one for storing the representation key values of the world-table. However, we think that the modeling approach described above is more intuitive for the reader and the connections between this modeling approach and the modeling approach that uses pc-databases are simpler to see.

Example 182 *For illustration, we consider the same deduplication scenario that we have used in the previous examples, but restrict to the case where the deduplication result is factorized. The resultant BID-database is depicted in Figure 8.10(a). Since we only have one actual database table, namely the table* 'ProductBasic'*, the database consists of three tables plus the MBase-table. Note that in this example the actual database table is only certain because we consider a certain database as deduplication input and because we use a conflict resolution approach for merging duplicate database entities. In other use cases, however, the blocks (entities) of this table may have several tuples. Since the table* 'ProductBasic' *is the only actual database table in this example, we require only a single view that joins this table with the correlation-table and the world-table in order to reconstruct[4] the pc-table* 'Product' *from Figure 8.9(b). This view is presented in Figure 8.10(b).*

Note that we cannot use foreign keys in order to reference from the correlation-table to the actual database tables because one tuple of the correlation-table can reference to multiple tuples from several tables, i.e. it references to all tuples that represent the described database entity.

8.5. Processing Indeterministic Deduplication Results

It is intuitive that modeling the uncertainty on ambiguous duplicate decision within a probabilistic database can increase the quality of several database applications because a reduction of false dupli-

[3] It is important to note that the world-table is a system table in pc-databases, but is a conventional table in this modeling approach.

[4] Actually, the resultant pc-table is not exactly the one from Figure 8.9(b) because it uses a certain variable Z and the variable Y has two possible values instead of one, i.e. we do not use negations in tuple conditions. However, it models the same set of possible worlds.

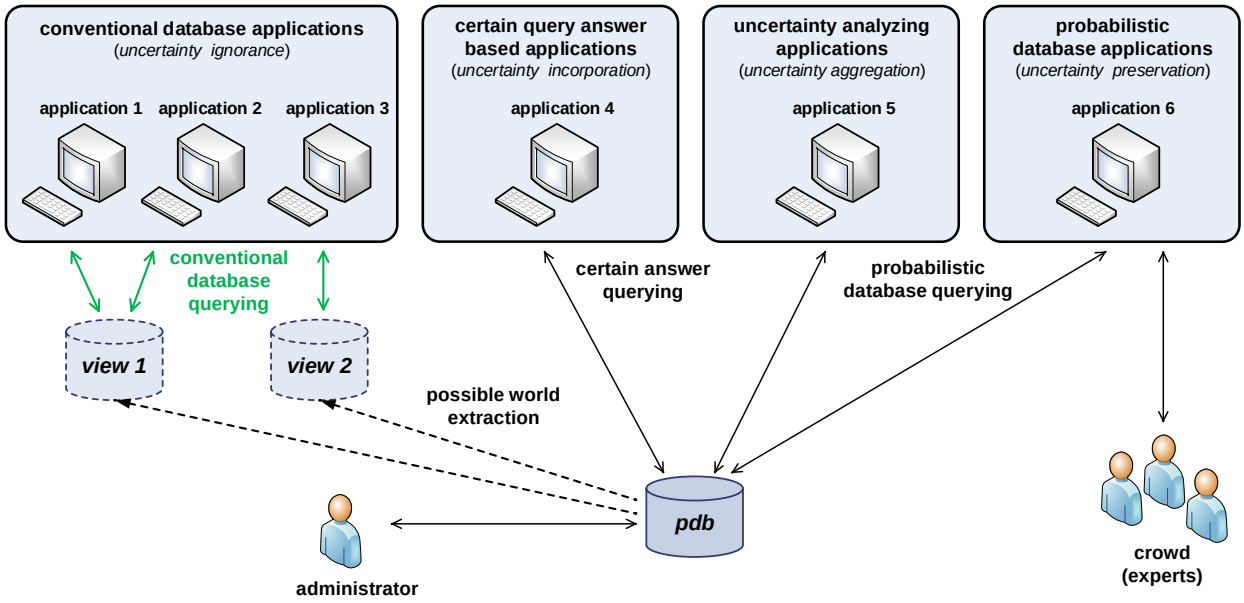


Figure 8.11.: The four classes of database applications that can work on a probabilistic database

cate decisions can decrease the risk of producing erroneous output data. Nevertheless, it is not clear in which way a database application can benefit from an indeterministic deduplication result. Furthermore, an indeterministic deduplication result is a complex probabilistic database in which the existence of database entities can be correlated. An evaluation of aggregate queries on such databases, however, is not straightforward. For that reason, we study several ways of processing indeterministic deduplication results in Section 8.5.1 and then consider an evaluation of aggregate queries on indeterministic deduplication results in Section 8.5.2.

8.5.1. Classes of Database Applications

In this section we study the different classes of database applications that can process indeterministic deduplication results. As graphically illustrated in Figure 8.11 and as we discuss in the following, we identified four of these classes.

1. **Probabilistic Database Applications:** The most native way to process a probabilistic database is to consider any kind of uncertainty during query evaluation and hence to produce a probabilistic query result. Recall that we extensively present probabilistic database querying in Section 3.5 where we describe that the query result is either a probabilistic database (possible worlds semantics) or is a list of tuple-probability pairs (possible answers semantics).

2. **Uncertainty Analyzing Applications:** The second class includes applications that are designed to analyze the uncertainty of a complete probabilistic database or of a particular part of this database. For instance, an application can compute the minimal/maximal/expected number of database entities that satisfy a specific selection criterion as for example by posing a query that asks for the

expected number of persons living in Germany. In such cases, aggregate functions can be used to extract several characteristics of the database's uncertainty.

Actually, analyses on uncertainty can be build upon probabilistic query results so that the actual database access can be considered to be equal to the access of a probabilistic database application. However, as we described in Section 3.5.4 many probabilistic query languages support the use of specific aggregate functions, e.g. the expected sum, that makes it possible to perform the uncertainty analyses directly on the database. For that reason, we consider it as an own class of applications.

3. **Conventional Database Applications:** A deduplicated database is usually processed by conventional database applications that need concrete query results. Thus, these applications are not designed to process an uncertain database, but need a certain database as input. In this case, the application has to ignore uncertainty by evaluating queries only on one of the database's possible worlds instead on all of them. Naturally it is most meaningful to take the most probable world (or one of them if multiple most probable worlds exist) as application input. Nevertheless, in some use cases it makes sense to take another possible world as input if it satisfies some specific criterion. For example, if an application considers false positives to be worse than false negatives it can be advantageous to select the possible world with the most number of database entities instead of the most probable world.

   If the application accesses the database only for reading purposes, processing a single possible world is equivalent to processing a deterministic deduplication result. Thus, the uncertainty of the database can be hidden from this application and the only overhead compared to conventional database processing is the computation of the desired possible world. The situation becomes different if applications need to modify the database (update, delete or insert operations) because modifying one possible world usually implicates modifications on some other possible worlds if the probabilistic database is modeled within a compact representation system. Therefore, database modifications can cause a variety of side effects the modifying database application is even not aware of because the other possible worlds are hidden from this application, i.e. the application does not know which possible worlds are affected and does not know in which way these worlds are affected. Moreover, the caused side effects can unwantedly affect the processed world itself because these effects can change the world that is selected by the application's criterion.

   At a first glance, we have the same situation as we have in the use of database views that are not changeable because key attributes are projected out or aggregate functions are used to transform the viewed data. And indeed by taking a closer look we see that providing a particular possible world to a database application is nothing else than providing a tailor-made view. In both cases we provide the data that the considered application needs for its purpose and in both cases the actual instance of the database is unknown to the application and write access cannot be given.

   The effort of computing the desired world depends on the selection criterion and the considered representation system. Recall, in Section 3.3 we described for each representation system in which way the most probable worlds can be computed. For computing the world with the most (or least)

entities, the merge bases can be used because the greater the merge bases of the selected entities, the more entities are excluded from the considered world.

Note, due to each application can select a world by its own criterion, an indeterministically modeling of ambiguous duplicate decisions can be even useful if the deduplicated database is solely processed by conventional database applications.

4. **Certain Query Answer based Applications:** A lot of applications require query answers that are dead certain or at least are certain with a given level of tolerance τ (recall from Section 3.5.1 that certain query answers are those answers that result from every possible world [IL84, Gra91]). In this case, a probabilistic database need to be processed by using mechanisms for *certain query answering* [Wij10, Wij12, DPW12, GPW14] that is also known as *consistent query answering* [ABC99, FFM05, Wij09, Ber11, Wij14] or *sure information answering* [Kle01, Kle94] (a similar concept is known as *clean query answering* [AFM06, BKL11, BBKL12]). By doing so, uncertainty is resolved by evaluating a query only on the certain facts of the probabilistic database. It is important to note that the certain query answers that result from querying an indeterministic deduplication results are typically more correct than the certain query answers that result from querying a deterministic duplicate detection result because the query answering algorithm can distinguish between uncertain duplicate decisions and certain duplicate decisions.

   As conventional database applications, certain query answer based applications do not produce an uncertain query result. However, in contrast to conventional database applications that only query a single possible world, certain query answer based applications consider every kind of data uncertainty for producing their certain query result.

   Of course, for certain query answering the used database query language need to contain a function that enables the user to check whether or not an answer is certain. For example, the query language of the Trio system (called *TriQL*) provides the function *Conf()* that can be used to quantify the confidence of a subquery result and therefore can be utilized to distinguish between certain answers (*Conf()* $\geq 1 - \tau$) and uncertain answers (*Conf()* $< 1 - \tau$).

As mentioned above, using a probabilistic database can become critical if multiple applications write on this database separately because in most representation systems manipulating one possible world can implicate a manipulation of some other possible worlds. Nevertheless, there are a lot of scenarios in which the processing applications only need to read the database and the only writing access is given to the application that maintains the database, i.e. the administrator.

If we take a closer look at these four classes of database applications, we can identify three different ways of handling a database's inherent uncertainty. Conventional database applications need to ignore uncertainty. In contrast, certain query answer based applications resolve uncertainty by distinguishing between certain facts and non-certain facts. Finally, probabilistic database applications as well as uncertainty analyzing applications process the database's uncertainty and push it directly to the query result. As a consequence, although all these applications work on the same database, each of them deal with the database's uncertainty in a complete different way and therefore are differently affected by the number of possible clusterings and the correctness of these clusterings. As we will discuss in Section 8.8,

these differences have an essential influence on the meaning of the quality of indeterministic duplicate detection results.

As mentioned in Section 1.3.1, an indeterministic deduplication result can be considered as a prompt answer and feedback from the crowd (or domain experts) can help to resolve detection uncertainty time after time. Of course a user is only enabled to give feedback on an indeterministically modeled duplicate decision if she is able to got information on this ambiguity. Since only probabilistic database applications present uncertainty to the user in a non-aggregated form, such a feedback process can only be realized by using such an application.

Example 183 *For instance, an indeterministic duplicate detection process classifies two entities of a movie database as a* POSSIBLE MATCH*. This ambiguity is presented to the crowd by a movie webpage where each user can vote whether these movies are duplicates or not. In a simple approach, the probability that both movies are duplicates increases for each positive vote and decreases for each negative vote. If the probability changes to one or if the probability changes to zero the uncertainty on this duplicate decision is considered to be resolved, the indeterministically modeled duplicate decision is changed into a deterministic one by either removing the merged entity (*UNMATCH*) or by removing the original entities (*MATCH*), and the voting process stops.*

De Keijzer and van Keulen [dKvK07b, vKdK09, dK10b] propose different techniques for tracing back user feedback (positive or negative) on query results to possible worlds in order to reduce the uncertainty of the queried probabilistic database step by step. Of course, these techniques can be used to resolve uncertainty on duplicate decisions.

Zhang et al. [ZCJC13] consider uncertainty reduction via crowdsourcing in the context of schema matching. Because of the large semantical resemblance between schema element correspondences and duplicate decisions, it seems natural that the most of their insights can be adopted to the context of indeterministic deduplication with only less adaptation.

Example 184 *To illustrate the different classes of database applications that we have present in the previous part of this section we consider the indeterministic deduplication result from Figure 8.8(b) and request for all product classes that contain less than four products. Note that the result of this request only depends on the detected duplicates and does not depend on the used merge function. Thus, this request is perfectly suitable to demonstrate in which way indeterministically modeled duplicate decisions affect the query result.*

In conventional database querying, this request can be expressed by using the SQL query Q_{CD} that is presented in Figure 8.12(a). As mentioned above, a conventional database application only process a single possible world of the considered probabilistic database. In this example, we assume that the most probable world is processed. This world results by using the variable assignment $\theta(X \mapsto 3, Y \mapsto 1)$ and contains the four entities e_6, e_7, e_8 and e_{10}. The result of applying query Q_{CD} to the most probable world is shown in Figure 8.12(e) and contains two tuples: each one for the product classes R and S.

In the case of a probabilistic database application, the SQL query need to be rewritten into a query of the language that is used by the considered probabilistic database system. In this example, we use the Trio Query Language. For representation purposes we evaluate the query by the possible answers

```
SELECT  P2.class
FROM    (SELECT  P.class
                 COUNT(*) AS Qty
         FROM    product P
         GROUP BY(P.class)) AS P2
WHERE   P2.Qty < 4;
```

(a) Conventional database query Q_{CD}

```
SELECT  P2.class
FROM    (SELECT  P.class
                 COUNT(*) AS Qty
         FROM    product P
         GROUP BY(P.class)) AS P2
WHERE   P2.Qty < 4
ORDER BY Confidences  ASC;
```

(b) Probabilistic database query Q_{PD}

```
SELECT  P2.class
FROM    (SELECT  P.class
                 Hcount(*) AS Qty
         FROM    product P
         GROUP BY(P.class)) AS P2
WHERE   P2.Qty < 4;
```

(c) Certain answer query Q_{CA}

```
SELECT  P.class, Lsum(P.sales) AS minQty,
                 Hsum(P.sales) AS maxQty,
                 Esum(P.sales) AS expQty
FROM    product P
GROUP BY P.class;
```

(d) Uncertainty analyzing query Q_{UA}

class
R
S

(e) Result of Q_{CD}

class	p
S	1.0
R	0.8

(f) Result of Q_{PD}

class
S

(g) Result of Q_{CA}

class	minQty	maxQty	expQty
R	2	4	2.7
S	1	2	1.6

(h) Result of Q_{UA}

Figure 8.12.: Sample queries and their results for the different classes of database applications

semantics and additionally sort the resultant tuples by their probabilities. The corresponding TriQL query Q_{PD} is presented in Figure 8.12(b). The result of this query is shown in Figure 8.12(f) and contains two tuples. Whereas the tuple that represents the product class S is certain, the tuple that represents the product class R is only maybe (probability 0.8) because its number of products is four in the possible worlds that result from the variable assignments $\theta(X \mapsto 1, Y \mapsto 1)$ (probability 0.12) and $\theta(X \mapsto 1, Y \mapsto 2)$ (probability 0.08).

In the context of certain query answering, we are only interested in the answers that result from querying every possible world. For that reason, the original SQL query from Figure 8.12(a) need to be rewritten. The rewriting process depends on the used probabilistic database system. In this example we use TriQL again. The rewritten query Q_{CA} is shown in Figure 8.12(c) and filters out all product classes that does not satisfy the given condition with absolute confidence by using the aggregate function 'Hcount' in the subquery . This function returns the highest possible values that result from evaluated the aggregate function 'Count' under the distribution semantics. Since a product class only certainly belongs to the query result if all its alternatives have less than four products a class can only be certain query answer if its highest possible number of products is below this threshold. The result of Q_{CA} is shown in Figure 8.12(g) and contains only the product class S as a query answer. Note that the product

class R is missing because it has four products in the case of the aforementioned variable assignments $\theta(X \mapsto 1, Y \mapsto 1)$ (probability 0.12) and $\theta(X \mapsto 1, Y \mapsto 2)$ (probability 0.08).

The last class of applications is a bit different to the three classes that we have considered before because requesting all product classes that satisfy the desired criterion is not an uncertainty analyzing query. For that reason we slightly modify this request and instead ask for the lowest possible quantity, the highest possible quantity, and the expected quantity of each product class. By using TriQL this request can be expressed by the query Q_{UA} that is presented in Figure 8.12(d). The result of this query is shown in Figure 8.12(h). The product class S has at least one product, has at most two products, and has 1.6 products in expectation. In contrast, the product class R has at least two products, has at most four products, and has 2.7 products in expectation.

8.5.2. Aggregate Query Answering on Indeterministic Deduplication Results

In the previous example, we use an aggregate query in order to illustrate the different classes of database applications because this query is independent from the used merge function and therefore is particularly suitable to demonstrate in which way indeterministically modeled duplicate decisions affect the query result. In real-world applications, however, indeterministic deduplication results are also useful for queries that does not include aggregate functions. In the context of aggregate queries, however, it is important to note that the currently available Trio version[5] does not support ordinary aggregate functions in vertical queries although it is part of the TriQL manual[6]. Therefore, even query Q_{PD} is defined accordingly to the TriQL manual it cannot be executed by this Trio version. The available version of MayBMS does not support ordinary aggregate functions as well. For this reason, the TriQL query Q_{PD} is currently only of a theoretical nature and exact evaluation of this query is not possible. The reason for this lack in implementation is simply the fact that an exact computation of aggregate queries on pc-databases under the distribution semantics has an exponential time complexity. Recall from Section 3.5.4 that the number of possible aggregation results can be equal to the number of possible worlds and therefore even enumerating all these possible results is impractical in many cases. Nevertheless, although an exact computation is often infeasible the aggregation result can be approximated (e.g. by using a Monte-Carlo Simulation). Moreover, Trio only supports an evaluation of aggregate queries under the range semantics and aggregate queries under the expected value semantics for ULDBs without lineage and therefore only for BID-tables. MayBMS support these semantics. However, as described in Section 3.5.4, it is not clear whether MayBMS computes them exactly or approximately. As a consequence, the queries Q_{CA} and Q_{UA} cannot be exactly computed in many cases.

However, if the input to the indeterministic deduplication process is a certain database or is an entity-independent probabilistic database, the deduplicated database only contains entity correlations that have been introduced by the deduplication process. In this case, many aggregate queries can be exactly computed in an efficient way and thus without enumerating all the possible worlds because the entities from different factors are independent to each other. This fact can be best illustrated by considering the approach of modeling an indeterministic deduplication result within a BID-database (recall that in this case the input database is required to be entity-independent).

[5] http://infolab.stanford.edu/trio/

[6] http://infolab.stanford.edu/~widom/triql.html

The advantage of modeling an indeterministic deduplication result within a BID-database is that we can exactly compute some aggregate queries on the deduplicated database, i.e. the pc-table that is defined by the queried view, even under the distribution semantics based on the insights that have been result on the research on evaluating aggregate queries on BID-databases (see Section 3.5.4). For demonstration, we consider an evaluation of the aggregate query Q_{COUNT} that is presented in Figure 8.13 under the distribution semantics. Note that this query is posed to the table *'Product'* that is defined by view $Q_{Product}$ from Figure 8.10(b) and therefore is actually posed to a pc-table instead of a BID-table.

```
SELECT  COUNT(*) AS Qty
FROM    Product p
WHERE   p.class = R
```

Figure 8.13.: Aggregate query Q_{COUNT} that counts all products per class 'R' in the pc-table *'Product'*

Since *'Product'* is a pc-table that is constructed by the view that we have presented in Figure 8.10(b), we have the same shortcoming then discussed above, i.e. an algorithm for exactly computing this query under the distribution semantics is not available. However, as we will show below an exact computation of this query can be realized if we rewrite this query into a sequence of two aggregate queries that each work on a BID-database instead of a pc-table. In the first step, we extend the world-table by a new attribute that stores the number of products per variable-value pair (and thus per factor alternative). This can be accomplished by the database view that is depicted in Figure 8.14.

```
CREATE VIEW  WorldTableWithQuantity
AS SELECT    w.var, w.value, t.QtyWorld
FROM         World-Table w,
             (SELECT   COUNT(*) as QtyWorld
             FROM      Correlation-Table c, ProductBasic p
             WHERE     w.var = c.var
             AND       w.value = c.value
             AND       p.DEI = c.DEI
             AND       p.class = R) AS t
```

Figure 8.14.: View Q_{Qty} that extends the world-table by an additional attribute that stores the number of products per factor alternative

Note that although table *'ProductBasic'* is a certain table in Example 182, it can be a BID-table without maybe blocks if we use a conflict repairing approach for merging duplicate entities and can be a BID-table with maybe blocks if we consider a probabilistic database as deduplication input. Nonetheless, the correlation-table is always certain and joining a BID-table with a certain table always results in another BID-table. As a consequence, the subquery of Q_{Qty} always corresponds to an aggregate query on a BID-table (if table *'ProductBasic'* is certain, this subquery even corresponds to an aggregate query that is posed to a conventional database table). According to Gal et al. [GMSS09], a COUNT query on a BID-table under the distribution semantics (and thus view Q_{Qty}) can be computed in polynomial time.

In the second step, we modify the original considered aggregate query Q_{COUNT} into a query that sums up the quantity values of the database entities (i.e. blocks) in the newly created table *'WorldTableWithQuantity'*. This modification is correct, i.e. it does not change the query result, because the total number of products corresponds to the sum of products per factor. As a consequence, we replace query Q_{COUNT} by the aggregate-query Q_{SUM} that is presented in Figure 8.15.

```
SELECT  SUM(QtyWorld) AS Qty
FROM    WorldTableWithQuantity
```

Figure 8.15.: Aggregate query Q_{SUM} that sums up all products numbers per factor

According to Gal et al. [GMSS09], a SUM query cannot be evaluated efficiently on a BID-table under the distribution semantics in general because the number of possible aggregation results can be infeasible in size (theoretically from each possible world another output value can result). In the case of query Q_{SUM}, however, the aggregation result represents the number of products and therefore the output range is bounded by the number of tuples in table *'ProductBasic'*. For that reason, the complexity of query Q_{SUM} is exactly the same as the complexity of query Q_{Qty} and hence can be computed in polynomial time. This can be explained as follows: If we consider the size of table *'ProductBasic'* as a constant, i.e. $n = |$*'ProductBasic'*$|$, the computation time of query Q_{SUM} is linear to the number of blocks in the world-table because in the worst case we need to scan n values for each of these blocks.

Example 185 *An evaluation of query Q_{SUM} is illustrated in Figure 8.16 for the sample database instance from Figure 8.10(a). The only difference in this illustration is that we count all products and do not restrict to products of class R because otherwise the example would be restricted to a single uncertain factor and therefore would be less illustrative.*

In the first step, the world-table is extended by the attribute 'QtyWorld' *and this attribute is filled up with values by using the view Q_{Qty}. The resultant BID-table is presented in Figure 8.16(a). Since block Z has only one tuple and this tuple is only correlated with a single entity in table* 'ProductBasic'*, for Z the attribute* 'QtyWorld' *is filled up with the value* 1*. The tuples of block X are correlated with three, two, or one product respectively and the two tuples of block Y are correlated with two or one product respectively.*

Recall, each of the blocks represents a factor of the indeterministic deduplication result and whereas the first factor always consists of one product, the second factor consists of one, two, or three products and the third factor consists of one or two products. Since the factors are mutually independent and even so the blocks, the total number of products results from summing up the products per factor (i.e. blocks). Figure 8.16(b) shows the successive process of summing up all possible combinations of the blocks' values in the attribute 'QtyWorld' *in order to compute the total count of all products. Note that each row is a probability distribution among the values considered thus far, i.e. each cell contains the probability that its corresponding number of products is true. For example, block Z has exact one product. Thus, after the first computation step the number of products is certainly one. In contrast, block X has one product (probability* 0.5*), two products (probability* 0.3*), or three products (probability* 0.2*) respectively. As a consequence, the total number of products within the two factors that are represented by the blocks Z and X is either two, three, or four where the probabilities of these alternative values correspond to*

WorldTableWithQuantity

RK	var	value	QtyWorld	p
1	Z	1	1	1.0
2	X	1	3	0.2
3	X	2	2	0.3
4	X	3	1	0.5
5	Y	1	2	0.6
6	Y	2	1	0.4

(a) The constructed BID table with quantity values

		number of products						
step	**var**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
1	**Z**	0	1.00	0	0	0	0	0
2	**X**	0	0	0.50	0.30	0.20	0	0
3	**Y**	0	0	0	0.20	0.42	0.26	0.12

(b) The three steps of summing up the number of products

Aggregation Semantics	**Qty**
range	$[3, 6]$
distribution	$\{(3, 0.2), (4, 0.42), (5, 0.26), (6, 0.12)\}$
expected value	4.3

(c) The aggregation result presented in all three aggregation semantics

Figure 8.16.: Sample for computing the COUNT aggregate of the table *'Product'*

these probabilities that are introduced by block X. Finally, we add the number of products that are contained in the factor that is represented by block Y. For instance, since the probability that Y has one product is 0.4 and the probability that Z and X together have two products is 0.5, the probability that all three blocks together have three products is $0.5 \times 0.4 = 0.2$.

The results for all three aggregation semantics are presented in Figure 8.16(c). The lowest number of possible products is three. This number results if the last tuple of each block is chosen to be true. In contrast, the highest number of possible products is six. This number is given if the first tuple of each block in the BID-table 'WorldTableWithQuantity' *is chosen to be true. As a consequence, under the range semantics the queries Q_{COUNT} and Q_{SUM} both results in the pair $[3, 6]$. Under the distribution semantics both queries results in the number-probability pair list $\{(3, 0.2), (4, 0.42), (5, 0.26), (6, 0.12)\}$. Finally, the expected number of products[7] results in $(1.0 \times 1) + (0.2 \times 3 + 0.3 \times 2 + 0.5 \times 1) + (0.6 \times 3 + 0.4 \times 2) = 4.3$.*

Following the above described approach we can compute a COUNT query on an indeterministic deduplication result under the distribution semantics in decent time although a modeling of such a deduplication result requires the representation power of pc-databases. Of course, since the result of an aggregate query that is evaluated under the range semantics or that is evaluated under the expected value semantics can be easily derived from the result of this query evaluated under the distribution semantics, this insights can be used for the range semantics and the expected value semantics, too.

In query Q_{CA} from Figure 8.12(c) we group all products by their class. This grouping, however, cannot directly incorporated into the subquery of the view Q_{Qty} because otherwise the resultant table is not a BID-table anymore. In contrast, we need to ensure that each tuple of the world-table is represented by a set of mutually exclusive tuples in the BID-table *'WorldTableWithQuantity'* (i.e. tuples of the same

[7] Note, the expected value of a sum of random variables is equivalent to the sum of the expected values of these variables [GT96, CA03].

block) and therefore we either need to extend the schema of this table by one attribute per group or need to define one view per group.

Similar rewriting approaches can be used for aggregate queries that apply one of the aggregate functions SUM, MIN, MAX, or AVG on the attribute *'sales'* of pc-table *'Product'*. In contrast to the COUNT function, however, the results of the SUM function and the AVG function are not bounded and the number of possible output values can be considerably large. For that reason, these functions cannot be exactly computed in polynomial time under the distribution semantics in some cases [MW07, MIW11] although we use the rewriting approach presented above. Nevertheless, in such cases a Monte-Carlo Simulation (or any other approximation technique) can be used for approximating the aggregation result. Moreover, this shortcoming does not necessarily hold for the range semantics and the expected value semantics.

Of course, if we model an indeterministic deduplication result within a pc-database and if the entities of the input database are independent to each other, i.e. the only entity correlations in the database are the correlations that represent the entity set complementations of the indeterministic deduplication result, this computation process can be adopted from BID-databases to pc-databases. Nevertheless, pc-databases cannot make this independence assumption for all its database entities in general. For that reason, we cannot simply rewrite queries, but need to directly change the querying mechanisms instead.

8.6. Indeterministic Deduplication in Probabilistic Databases

In indeterministic duplicate detection in certain databases we map a single database instance to a set of possible clusterings and in deterministic duplicate detection in uncertain databases we map a set of possible database instances to a single clusterings. Consequently, in indeterministic duplicate detection in uncertain databases we map a set of possible database instances to a set of possible clusterings. This mapping can be done in two ways: conditioned or unconditioned.

- The idea of the conditioned approach is to connect alternatives of the output, i.e. possible clusterings, to alternatives of the input, i.e. possible worlds, and therefore to condition duplicate decisions by the given possible entity instances. For example, two database entities are only considered to be duplicates in these possible worlds in which they share the same last name, but are considered to be non-duplicates in the remaining worlds. Of course, as in indeterministic duplicate detection in certain databases (which is nothing else than a single possible world), several possible clusterings can be connected to the same possible world. In this case, the probability of this possible world is redistributed to the possible clusterings that are conditioned by this world. Moreover, as for deterministic duplication detection in uncertain databases, one possible clustering can be connected to several possible worlds, i.e. this duplicate clustering is the 'true' clustering if some of its connected worlds is the 'true' world. Altogether, the probabilities of the possible worlds can be redistributed to the possible clusterings in a variety of ways.

- In contrast, the idea of the unconditioned approach is to decouple the clustering alternatives from the possible worlds they have been derived from. Thus, even if a clustering has been computed on a particular possible world we discard this connection by simply considering this clustering as an alternative of an uncertain output clustering.

Product

	RK	DEI	name	class	sales	p	condition
t_1	1	e1	product A	R	7	1.0	true
t_2	2	e2	product A	R	9	0.8	V=1
t_3	3	e2	product AA	R	10	0.2	V=2
t_4	4	e3	product B	R	4	0.8	V=1
t_5	5	e4	product BB	R	6	1.0	true
t_6	6	e5	product BBB	R	11	0.6	W=1
t_7	7	e6	product C	S	2	0.6	W=1
t_8	8	e6	product C	R	5	0.4	W=2
t_9	9	e7	product CC	S	4	1.0	true

World-Table

var	value	Prob
V	1	0.8
V	2	0.2
W	1	0.6
W	2	0.4

(a) Initial database

Product

	RK	DEI	name	class	sales	p	condition
t_{10}	1	e8	product AA	R	8	1.0	true
t_{11}	2	e3	product B	R	4	0.16	V=1 ∧ X=1
t_{12}	3	e4	product BB	R	6	0.2	X=1
t_{13}	4	e5	product BBB	R	11	0.3	W=1 ∧ (X=1 ∨ X=2)
t_{14}	5	e6	product C	S	2	0.3	W=1 ∧ Y=1
t_{15}	6	e6	product C	R	5	0.2	W=2 ∧ Y=1
t_{16}	7	e7	product CC	S	4	0.5	Y=1
t_{17}	8	e9	product BB	R	5	0.4	X=2
t_{18}	9	e10	product BBB	R	7	0.4	X=3
t_{19}	10	e11	product CC	S	4	0.5	¬ (Y=1)

World-Table

var	value	Prob
V	1	0.8
V	2	0.2
W	1	0.6
W	2	0.4
X	1	0.2
X	2	0.3
X	3	0.5
Y	1	0.6

MBase-Table

DEI	base
e8	e1
e8	e2
e9	e3
e9	e4
e10	e3
e10	e4
e10	e5
e11	e6
e11	e7

(b) Deduplicated database

Figure 8.17.: Modeling an indeterministically deduplicated probabilistic database by using a pc-database

It is quite evident that an introduction of world conditioning makes the detection process more powerful. Moreover, this approach is even very intuitive in situations where the possible clusterings are computed by processing some of the database's possible world separately. However, it is also obvious that a conditioned indeterministic duplicate detection result cannot be simply modeled by an uncertain clustering, but requires some additional concepts for modeling the connections of the possible clusterings to the input worlds. For this reason we restrict in this thesis to the unconditioned approach.

In the unconditioned approach we can choose between the same two modeling approaches that we have presented for certain input databases in Section 8.4, i.e. we can model the indeterministic deduplication result by using a pc-database or we can model it by using a BID-database. If we use a pc-database we have to adapt the modeling approach from Section 8.4.2 to an appropriate handling of the tuples' conditions. Obviously, the conditions of the output tuples depend on the used merge function because a merging of database entities can include a merging of correlations to other database entities. Note that the dependency on the merge function does not only hold for the conditions of the tuples of the newly created entities. In contrast, merging the conditions of some tuples can also affect the conditions of tuples that

ProductBasic

	RK	DEI	name	class	sales	p
t_{10}	1	e8	product AA	R	8	1.0
t_{11}	2	e3	product B	R	4	0.8
t_{12}	3	e4	product BB	R	6	1.0
t_{13}	4	e5	product BBB	R	11	0.6
t_{14}	5	e6	product C	S	2	0.6
t_{15}	6	e6	product C	R	5	0.4
t_{16}	7	e7	product CC	S	4	1.0
t_{17}	8	e9	product BB	R	5	1.0
t_{18}	9	e10	product BBB	R	7	1.0
t_{19}	10	e11	product CC	S	4	1.0

Correlation-Table

	DEI	var	value
t_{20}	e8	Z	1
t_{21}	e3	X	1
t_{22}	e4	X	1
t_{23}	e5	X	1
t_{24}	e5	X	2
t_{25}	e6	Y	1
t_{26}	e7	Y	1
t_{27}	e9	X	2
t_{28}	e10	X	3
t_{29}	e11	Y	2

World-Table

	var	value	p
t_{30}	Z	1	1.0
t_{31}	X	1	0.2
t_{32}	X	2	0.3
t_{33}	X	3	0.5
t_{34}	Y	1	0.6
t_{35}	Y	2	0.4

MBase-Table

DEI	base
e8	e1
e8	e2
e9	e3
e9	e4
e10	e3
e10	e4
e10	e5
e11	e6
e11	e7

Figure 8.18.: Modeling a indeterministically deduplicated probabilistic database by using a BID-database

are correlated with the merged tuples. For modeling the deduplicated database, we have to combine the actual tuple conditions, i.e. the conditions of the input database or the conditions that are computed by the merge function respectively, with the tuple conditions that represent the entity set complementations of the indeterministic deduplication result. For each tuple both conditions are mandatory requirements for its existence. Therefore, these conditions need to be combined by computing their conjunction.

Example 186 *For illustration we consider the pc-database that is presented in Figure 8.17(a) and assume an unconditioned indeterministic duplicate detection process that produces the probabilistic clustering from Figure 8.7(a). For merging duplicate entities we use a conflict resolution function that selects the longest value in the attribute* 'name'*, that selects the most probable value in the attribute* 'class'*, and that aggregates the values in the attribute* 'sales' *by computing the rounded expected value. Moreover, the function merges tuple conditions by computing their disjunction. Note, by using this merge function each of the newly created entities consists of a single tuple and merging the conditions of some tuples does not affect the conditions of other tuples. For instance, the value of the tuple* t_{10} *in the attribute* 'sales' *is* 8 *because the expected sales of the tuples* t_1*,* t_2*, and* t_3 *is* $(7\times1.0+9\times0.8+10\times0.2)/2 = 8.1$*. Moreover, its condition results in* $\Phi(t_{10}) = true$ *because* $\Phi(t_1)\vee\Phi(t_2)\vee\Phi(t_3) = true\vee(V=1)\vee(V=2) = true$*. As described above, the final condition of an output tuple is computed by the conjunction of its actual condition and the condition that models the indeterministic deduplication result. As an example, the condition of tuple* t_{13} *is* $\Phi(t_{13}) = (W=1)\wedge(X=1\vee X=2)$*. Whereas the first operand of the conjunction corresponds to the condition of its corresponding input tuple* t_5*, the second operand models the membership of entity* e_5 *to the first two alternatives of the third factor. The deduplicated pc-database is presented in Figure 8.17(b).*

We cannot model entity correlations within a BID-database and the extended model that we have presented in Section 8.4.3 is only designed to incorporate the entity correlations that represent the entity set complementations of the indeterministic deduplication result. As a consequence, modeling an indeterministic deduplication result within a BID-database as presented in Section 8.4.3 is only possible if the input database is a BID-database or if it is a probabilistic database that is modeled within a less powerful representation system. If this condition is satisfied, the required adaptation of this modeling

approach to probabilistic input databases is only marginally because the original database entities as well as the newly created database entities are stored in a BID-database as we know it from a deterministic deduplication approach and the correlation-table as well as the world-table are constructed as described in Section 8.4.3.

Example 187 *For illustration we use the pc-database that is presented in Figure 8.17(a) and consider it as a BID-database by ignoring the tuple conditions. Moreover, we assume an unconditioned indeterministic duplicate detection process that produces the probabilistic clustering from Figure 8.7(a) and use the same merge function as in the previous example. The deduplicated BID-database is presented in Figure 8.18. The view is the same as in Example 182 (see Figure 8.10(b)). Note, whereas we consider a certain database as deduplication input in Example 182, the database table* 'ProductBasic' *is a BID-table in this example because we adopt the uncertainty from the input database to the output database. This example also illustrates that a tuple of the correlation-table can reference to several tuples of the table* 'ProductBasic' *because the correlation-table contains a tuple that references to entity* e_6 *and therefore references to the tuples* t_{14} *and* t_{15}.

8.7. Duplicate Clustering with Uncertainty

Separately performing a conventional duplicate clustering process for each combination of possible pairwise decisions is usually not performant. Therefore, we need a duplicate clustering approach that gets a set of uncertain duplicate decisions as input and produces an uncertain clustering as output. For that purpose we propose a graph-based clustering approach in [PvKR13]. The input to this approach is a matching probability per candidate pair and its output is a (factorized) probabilistic clustering. The approach can be divided into five steps. In the first step, a so-called *matching-graph* is constructed from the pairwise matching probabilities. In the second step, this *matching-graph* is decomposed into its independent components. Then, from each *matching-graph* component a set of so-called *world-graphs* is derived. In the fourth step, each *world-graphs* is checked for consistency. Finally, from each consistent *world-graph* a cluster-disjoint clustering is constructed. Each of these clusterings serves then as an alternative of the probabilistic clustering, i.e. a factor, that results from the corresponding *matching-graph* component.

Incorporating any kind of decision uncertainty into the uncertain output clustering (full-indeterministic approach) is usually not manageable in computation time and space because the number of independent factors remains low. Therefore, the computation of the algorithm as well as storing the resultant clustering becomes infeasible in most cases. For that reason, we introduce some strategies that carefully reduce the number of indeterministically handled decisions in Section 8.7.5. Nevertheless, due such strategies can be seen as restrictions on the full-indeterministic approach, we present the latter first. Moreover, because decomposition does not change the basic algorithm and the presentation of the underlying idea becomes more clear without considering it, we initially ignore decomposition and introduce it afterwards in Section 8.7.3.

8.7.1. The Full-Indeterministic Approach

The full-indeterministic approach computes a probabilistic clustering from a set of uncertain duplicate decisions. For reasons of presentation, we consider the computation of the probabilistic clustering as a graph-based process. For this purpose, we define two kinds of graphs: a *matching-graph* representing the independently computed uncertain duplicate decisions on the individual candidate pairs and a *world-graph* representing a conceivable clustering alternative (note, each clustering alternative represents the idea of a possible world).

8.7.1.1. Construction of the Initial Matching-graph

We start with constructing a *matching-graph* from the uncertain duplicate decisions. A *matching-graph* is a weighted undirected and fully connected graph, where each node represents a database entity. The weight of an edge denotes the probability that the connected entities are classified as a MATCH. If two entities have not been matched, i.e. this pair does not belong to the candidate pair space, their weight is initiliazed as zero, but can change if the pair is later post-fetched for some reason.

Definition 68 (Matching-Graph): *A* matching-graph *(short* M-graph*) is a triple* $G = (N, E, \gamma)$ *where* N *is a set of nodes,* $E = \{\{n_r, n_s\} \mid n_r, n_s \in N\}$ *is a set of undirected edges, and* γ *is a weight function* $\gamma\colon E \to [0, 1]$ *denoting matching probabilities.*

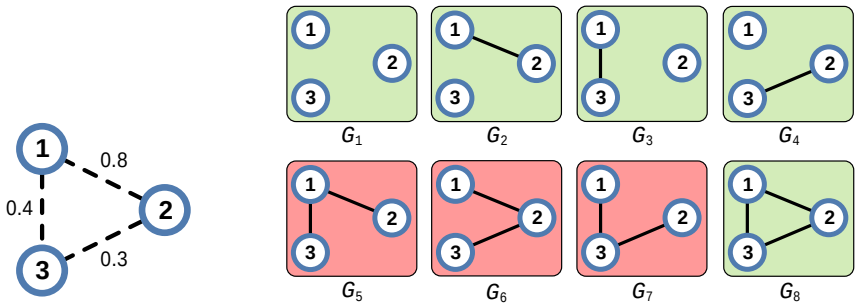
We call an edge to be *uncertain* if its weight is between 0 and 1 ($\gamma \in]0, 1[$) and *certain* otherwise. As a consequence, the set of all edges can be partitioned into two classes, the class of *certain edges* $E^!$ and the class of *uncertain edges* $E^?$. Certain edges are either positive (hard MATCH) or negative (hard UNMATCH). Therefore the class of certain edges can be further partitioned into the class of *positive edges* E^+ and the class of *negative egdes* E^-. These four classes are defined as:

$$E^+ = \{e \mid e \in E, \gamma(e) = 1\} \qquad E^! = E^+ \cup E^-$$
$$E^- = \{e \mid e \in E, \gamma(e) = 0\} \qquad E^? = E - E^!$$

Example 188 *An exemplary* matching-graph *is shown in Figure 8.19(a). For ease of presentation uncertain edges are generally shown dashed. The* M-graph $M = (N, E, \gamma)$ *has the three nodes* $N = \{n_1, n_2, n_3\}$ *and has the three edges* $E = \{\{n_1, n_2\}, \{n_1, n_3\}, \{n_2, n_3\}\}$ *where all these edges are uncertain, i.e.* $\gamma(\{n_1, n_2\}) = 0.3$*,* $\gamma(\{n_2, n_3\}) = 0.8$*, and* $\gamma(\{n_3, n_4\}) = 0.4$*. Each node* n_i *represents a database entity* e_i *and the weight of each edge represents the probability that the connected entities are a* MATCH*. Thus,* M *models the duplicate detection scenario in which the database entities* e_1*,* e_2 *and* e_3 *have been pairwise compared with each other and from these comparisons the matching probabilities* $Pr(\{e_1, e_2\} \in M) = 0.8$*,* $Pr(\{e_1, e_3\} \in M) = 0.4$*, and* $Pr(\{e_2, e_3\} \in M) = 0.3$ *have been resulted.*

8.7.1.2. Computation of World-graphs

A *world-graph* is an unweighted undirected graph representing one conceivable world of the clustering result, i.e. representing a possible duplicate clustering, where an edge $\{n_r, n_s\}$ means that the associated database entities e_r and e_s are considered to represent the same real-world entity, i.e. $\omega(e_r) = \omega(e_s)$.



(a) The sample *M-graph* M (b) The *W-graphs* $f^{\text{Naive}}_{MG \mapsto WG}(M) = \{G_1, \ldots, G_8\}$

W-graph	**probability**	**world (duplicate clustering)**
G_1	$0.2 \times 0.6 \times 0.7 = 0.084$	$W_1 : \{\langle e_1 \rangle, \langle e_2 \rangle, \langle e_3 \rangle\}$
G_2	$0.8 \times 0.6 \times 0.7 = 0.336$	$W_2 : \{\langle e_1, e_2 \rangle, \langle e_3 \rangle\}$
G_3	$0.2 \times 0.4 \times 0.7 = 0.056$	$W_3 : \{\langle e_1, e_3 \rangle, \langle e_2 \rangle\}$
G_4	$0.2 \times 0.6 \times 0.3 = 0.036$	$W_4 : \{\langle e_1 \rangle, \langle e_2, e_3 \rangle\}$
G_5	$0.8 \times 0.4 \times 0.7 = 0.224$	$W_5 : \{\langle e_1, e_2 \rangle, \langle e_1, e_3 \rangle\}$
G_6	$0.8 \times 0.6 \times 0.3 = 0.144$	$W_6 : \{\langle e_1, e_2 \rangle, \langle e_2, e_3 \rangle\}$
G_7	$0.2 \times 0.4 \times 0.3 = 0.024$	$W_7 : \{\langle e_1, e_3 \rangle, \langle e_2, e_3 \rangle\}$
G_8	$0.8 \times 0.4 \times 0.3 = 0.096$	$W_8 : \{\langle e_1, e_2, e_3 \rangle\}$

(c) The probabilities and the corresponding worlds (clusterings) of the *W-graphs* G_1 to G_8

Figure 8.19.: A sample *M-graph* and the eight *W-graph* that result from using the naive *W-graph* computation approach

Definition 69 (World-Graph): *A* world-graph *(short* W-graph*) is a triple* $G = (N, E, Pr)$ *where* N *is a set of nodes,* $E \subseteq \{\{n_r, n_s\} \mid n_r, n_s \in N\}$ *is a set of undirected edges, and* Pr *is the probability of the corresponding world.*

Obviously, each two entities of an hard MATCH have to belong to the same cluster in every possible clustering and each two entities of an hard UNMATCH have to belong to different clusters in every possible clustering. Therefore the most intuitive way to compute a set of *W-graphs* from the initial *M-graph* is to remove all *negative edges* and to eliminate each *uncertain edge* by either removing it or changing it to a *positive edge*. This naive process of *W-graph* computation is formalized by the mapping $f^{\text{Naive}}_{MG \mapsto WG} \colon \mathcal{G}_{MG} \to 2^{\mathcal{G}_{WG}}$ where $\mathcal{G}_{MG}$ is the set of all possible *matching-graphs* and $2^{\mathcal{G}_{WG}}$ is the power set of all possible *world-graphs*. Let $M = (N, E, \gamma)$ be the initial *M-graph*, the naive *W-graph* computation mapping $f^{\text{Naive}}_{MG \mapsto WG}$ is defined as:

$$f^{\text{Naive}}_{MG \mapsto WG}(M) = \{(N, E^+ \cup K, Pr(K)) \mid K \subseteq E^?\} \tag{8.6}$$

where $Pr(K) = \prod_{e \in K} \gamma(e) \prod_{e \in (E^? - K)} (1 - \gamma(e))$.

This means that we exactly create one *W-graph* for each possible set $K \subseteq E^?$ of uncertain edges. All *W-graphs* have the same nodes (the set N) as the initial *M-graph* and contain each edge $e \in E^+$. The probability of each *W-graph* results from the weights of the edges belonging to this *W-graph* and the inverse weights of the edges not belonging to this *W-graph*.

Example 189 *For illustration, we reconsider the* M-graph M *from Figure 8.19(a). Based on the given matching probabilities eight worlds along with their corresponding* W-graphs *can be derived (see Figure 8.19(b) and Figure 8.19(c)).*

8.7.1.3. Removing Inconsistent World-graphs

Since identity is a transitive relation, worlds that does not satisfy this transitivity are considered to be impossible, i.e. only a cluster-disjoint clustering can be the 'true' clustering. Thus, a world W with the entity set $\mathfrak{E}$ is possible, if and only if $\forall e_r, e_s, e_t \in \mathfrak{E}\colon \omega(e_r) = \omega(e_s) \wedge \omega(e_r) = \omega(e_t) \Rightarrow \omega(e_s) = \omega(e_t)$ and hence if grouping its entities by their identities results in a cluster-disjoint clustering. A *W-graph* is called consistent, if it represents a possible world.

Theorem 26 *A* W-graph $G = (N, E, P)$ *is consistent, if and only if* G *is equivalent to its transitive closure, i.e.* $G = G^*$.

Theorem 26 is proved by Proof 25 in Appendix A.4.

In the previous matching phases, the individual candidate pairs have been matched independently. Thus, worlds are created from independent considerations and hence can be impossible. Each inconsistent *W-graph* represents an impossible world and hence is removed from the set of considered graphs. In other words, dependencies between individually taken duplicate decisions are introduced by only considering consistent *W-graphs*.

Example 190 *For illustration, we reconsider the example from Figure 8.19(b). Since the transitivity of identity is violated, the three worlds* W_5*,* W_6*, and* W_7 *are definitely not the 'true' world and hence the* W-graphs G_5*,* G_6*, and* G_7 *are removed from further considerations.*

After removing inconsistent *W-graphs* (impossible worlds), the probabilities of the remaining *W-graphs* (worlds) do not sum up to 1 anymore. For that reason, the probabilities of the remaining *W-graphs* are conditioned with the event $\mathcal{B}$ that the true world must be a possible world. The probability of event $\mathcal{B}$ is the accumulative probability of all consistent *W-graphs*.

Example 191 *For instance, in our example, the conditioned probability of* G_1 *(and hence* W_1*) is computed as:*

$$Pr(G_1 \mid \mathcal{B}) = \frac{Pr(G_1)}{Pr(\mathcal{B})} = \frac{0.084}{0.608} = 0.138$$

8.7.1.4. Constructing the Probabilistic Clustering

In the final step from each consistent *W-graph* a clustering is constructed by simply inserting all nodes that are connected by an edge into the same duplicate cluster. Recall that the considered *W-graph* is consistent and hence is equivalent to its transitive closure. Therefore, the resultant clustering is always cluster-disjoint and the set of clusters corresponds to its connected components. Finally, we create a probabilistic clustering that has each of the previously constructed clusterings as an alternative. The probability of each alternative is set to the probability of its corresponding *W-graph*.

W-graph	possible clusterings of Γ	probability
G_1	$\mathcal{C}_1 = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3\rangle\}$	$Pr_{G_1} = Pr(\mathcal{C}_1) = 0.138$
G_2	$\mathcal{C}_2 = \{\langle e_1, e_2\rangle, \langle e_3\rangle\}$	$Pr_{G_2} = Pr(\mathcal{C}_2) = 0.553$
G_3	$\mathcal{C}_3 = \{\langle e_1, e_3\rangle, \langle e_2\rangle\}$	$Pr_{G_3} = Pr(\mathcal{C}_3) = 0.092$
G_4	$\mathcal{C}_4 = \{\langle e_1\rangle, \langle e_2, e_3\rangle\}$	$Pr_{G_4} = Pr(\mathcal{C}_4) = 0.059$
G_8	$\mathcal{C}_5 = \{\langle e_1, e_2, e_3\rangle\}$	$Pr_{G_8} = Pr(\mathcal{C}_5) = 0.158$

Figure 8.20.: Constructing the probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ from the set of consistent *W-graphs*

Example 192 *For illustration, we reconsider the five consistent* W-graphs G_1, G_2, G_3, G_4, *and* G_8 *from Figure 8.19(b). From each of these* W-graphs *a clustering is constructed. These clusterings* $\mathcal{C}_1$ *to* $\mathcal{C}_5$ *serves then as the alternatives (possible clusterings) of the finally produced probabilistic clustering. In summary, from processing the initial* M-graph M *from Figure 8.19(a) the probabilistic clustering* $\mathfrak{C} = (\Gamma, Pr)$ *that is presented in Figure 8.20 results.*

8.7.2. Consistency

In the third step of the clustering approach, all inconsistent *W-graphs* are removed from further considerations. Consequently, we run into a problem if all *W-graphs* are inconsistent and hence none of the generated worlds is possible.

Definition 70 (*M-graph* Consistency): *An* M-graph M *is called to be consistent with respect to the* W-graph *computation mapping* $f_{MG \mapsto WG}$ *if from* $f_{MG \mapsto WG}(M)$ *at least one consistent* W-graph *result.*

As we will prove below, the problem of validating *M-graph* consistency can be reduced to the problem of the proving the existence of so-called $\triangle$*-conflicts* (pronounced as *triangle-conflict*).

Definition 71 ($\triangle$-*conflict*): *Let* $M = (N, E, \gamma)$ *be an* M-graph*, a* $\triangle$*-conflict in* M *denotes a situation where two adjacent edges* $\{n_r, n_s\}$ *and* $\{n_r, n_t\}$ *are in* E^+*, but the edge* $\{n_s, n_t\}$ *is in* E^-.

As a consequence, an *M-graph* has a $\triangle$*-conflict* if two entity pairs $\{e_r, e_s\}$ and $\{e_r, e_t\}$ have been classified as an hard MATCH, but the entity pair $\{e_s, e_t\}$ has been classified as an hard UNMATCH. Obviously, these three duplicate decisions are contradictory and make all considered worlds per se impossible.

Theorem 27 *An* M-graph $M = (N, E, \gamma)$ *is inconsistent with respect to the* W-graph *computation mapping* $f^{Naive}_{MG \mapsto WG}$ *if and only if it contains at least one* $\triangle$*-conflict.*

Theorem 27 is proved by Proof 29 in Appendix A.4.

8.7.2.1. Validation of *M-graph* Consistency

Following Theorem 27, the consistency of an *M-graph* can be validated by checking for the existence of $\triangle$*-conflicts*. Theoretically, the complexity of checking the existence of a $\triangle$*-conflict* is enormous, because we need to verify all combinations of three different edges. Nevertheless, each $\triangle$*-conflict* includes two

positive edges and the number of positive edges is rather low in typical duplicate detection scenarios. Moreover, the two positive edges need to be adjacent. Thus, in practice we can check the existence of a $\triangle$-*conflict* in an acceptable time.

First we create two indexes. The first index I_{PE} maps each node to its positive edges (complexity $\mathcal{O}(|E^+|)$) and the second index I_{NNE} maps each node to its non-negative edges (complexity $\mathcal{O}(|E^?| + |E^+|)$). Alternatively, the negative edges can be stored instead of the non-negative edges. However the class E^- is usually many times larger than the union of the classes $E^?$ and E^+. After creating the indexes we scan over all nodes and check for each node n if its edges form a $\triangle$-*conflict*. A $\triangle$-*conflict* exists if there are two positive edges $\{n, n_r\}, \{n, n_s\} \in I_{PE}(n)$, and the nodes n_r and n_s are not connected by a positive edge or an uncertain edge, i.e. $\{n_r, n_s\} \notin I_{NNE}(n_r)$. Since only the fewest nodes have two positive adjacent edges, the checking process must only be performed for a few times.

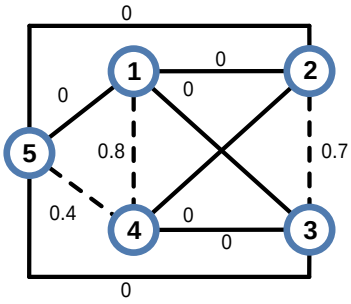
8.7.2.2. Repairing Inconsistent *M-graphs*

If an *M-graph* is inconsistent, we need to repair it because otherwise we cannot compute a detection result. An inconsistent *M-graph* can be repaired by resolving all its $\triangle$-*conflicts*. A $\triangle$-*conflict* can in turn be resolved by changing one of the involved edges into an uncertain edge. To change the matching probabilities as less as possible, we change the weight of a positive edge $e \in E^+$ from $\gamma(e) = 1$ to $\gamma(e) = 0.99$ and change the weight of a negative edge $e' \in E^-$ from $\gamma(e') = 0$ to $\gamma(e') = 0.01$. Of course, it is desirable to change as less edges as possible. Computing the repair that has the minimal amount of changes is a complex optimization problem and usually does not scale well for large graphs. For that reason, we use a simply heuristic. First we discover all $\triangle$-*conflicts*. This can be already incorporated into the process of validating consistency. Then we sort all edges in descending order by the number of $\triangle$-*conflicts* they are involved in. Finally, we change the edge from the top of the sorted list from certain to uncertain until all $\triangle$-*conflicts* are resolved. Since changing a certain edge into an uncertain edge cannot cause a new $\triangle$-*conflict*, the *M-graph* becomes consistent if all initially discovered $\triangle$-*conflicts* have been resolved. If some of the involved edges was negative because the corresponding entity pair does not belong to the candidate pair space (and hence was actually not matched), a post-fetching can be used to compute the actual matching probability of this pair (see Section 5.2.3.3).

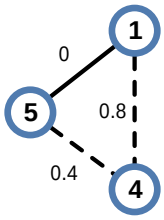
8.7.3. Decomposition of Matching-graphs

If the nodes of two different and node-disjoint subgraphs of the considered *M-graph* are only connected with *negative edges*, a node of the first subgraph cannot be connected with a node of the second subgraph in any *W-graph* and hence their corresponding entities can never belong to the same duplicate cluster. Thus if we use the mapping $f^{\text{Naive}}_{MG \mapsto WG}$ to compute a set of *W-graphs* from a given *M-graph*, such two subgraphs can be processed separately without changing the final clustering result.

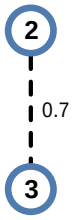
Definition 72 (Matching-Graph Decomposition): *Let $M = (N, E, \gamma)$ be an* M-graph*, the decomposition of M into its set of* partial M-graphs *is defined as dividing the graph $M = (N, E^+ \cup E^?, \gamma)$ in its connected components. The function $f_{dec} \colon \mathcal{G}_{MG} \to 2^{\mathcal{G}_{MG}}$ maps an* M-graph *M to its* partial M-graphs*.*



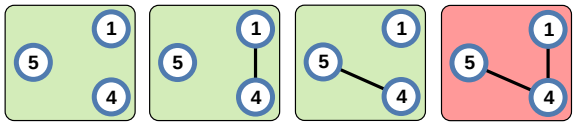
(a) Initial *M-graph* M_2



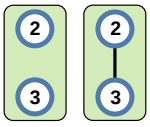
(b) *partial M-graph* M_{21}



(c) *partial M-graph* M_{22}



(d) The *W-graphs* of M_{21}



(e) The *W-graphs* of M_{22}

Figure 8.21.: Decomposition of an *M-graph* M_2 in its independent *partial M-graphs* M_{21} and M_{22}

Note that the set of all *W-graphs* that are computed from the *partial M-graphs* in $f_{dec}(M)$ is a compact representation of the *W-graphs* that are computed from the *M-graph* M directly because we can reconstruct the latter from the first.

Theorem 28 *Let* $M{=}(N, E.\gamma)$ *be an* M-graph *and let* $f_{dec}(M){=}\{M_1, \ldots, M_k\}$ *be its set of* partial M-graphs *where* $M_i{=}(N_i, E_i, \gamma)$ *for each* $i \in \{1, \ldots, k\}$. *The set* $f^{Naive}_{MG\mapsto WG}(M)$ *can be reconstructed from the sets* $\{f^{Naive}_{MG\mapsto WG}(M_1), \ldots, f^{Naive}_{MG\mapsto WG}(M_k)\}$ *by using the* W-graph *integration operator* ${}^{G}\boxtimes$ *that is defined as:*

$$ {}^{G}\boxtimes_{i=1}^{k} f^{Naive}_{MG\mapsto WG}(M_i) = \{(\bigcup_{i=1}^{k} N_i, \bigcup_{i=1}^{k} E_{G_i}, \prod_{i=1}^{k} Pr_{G_i}) \mid \forall i{\in}\{1, \ldots, k\}\colon G_i \in f^{Naive}_{MG\mapsto WG}(M_i)\} $$

where $G_i = (N_i, E_{G_i}, Pr_{G_i})$ *for each* $i \in \{1, \ldots, k\}$.

Theorem 28 is proved by Proof 30 in Appendix A.4.

Thus, each *W-graph* of $f^{\text{Naive}}_{MG\mapsto WG}(M)$ is reconstructed by combining exact one *W-graph* per *partial M-graph* of $f_{dec}(M)$. It is simple to see that the *W-graph* integration operator is the graph equivalent of the clustering integration operator that we have introduced in Definition 11 and have described by Equation 8.4 if we restrict the former to the integration of consistent *W-graphs* and restrict the latter to the integration of range-disjoint clusterings.

Example 193 *For illustration, we consider the* M-graph $M_2 = (N, E, \gamma)$ *with* $N = \{n_1, \ldots, n_5\}$ *as presented in Figure 8.21(a). Since there is no uncertain edge or positive edge that connects the disjoint subgraphs* $M_{21} = (\{n_1, n_4, n_5\}, \{\{n_1, n_3\}, \{n_1, n_5\}, \{n_3, n_5\}\}, \gamma))$ *(see Figure 8.21(b)) and* $M_{22} = (\{n_2, n_3\}, \{\{n_2, n_3\}\}, \gamma)$ *(see Figure 8.21(c)), the initial* M-graph *can be decomposed into these subgraphs. Both subgraphs are then considered as stand-alone* M-graphs *and hence are separately*

processed further on. From M_{21} *four* W-graphs *can be computed (see Figure 8.21(d)). Three of these* W-graphs *are consistent. Therefore, this* partial M-graph *produces the probabilistic clustering* $\mathfrak{C}_{21}$ *that has three alternatives. In contrast, from the* partial M-graph M_{22} *only two, each consistent,* W-graphs *can be computed (see Figure 8.21(e)). As a consequence, from processing* M_{22} *the probabilistic clustering* $\mathfrak{C}_{22}$ *that has two alternatives results. It can be easily seen that from processing the initial* M-graph M_2 *eight* W-graphs *result where six of them are consistent. Each of these six* W-graphs *is exactly the integration of a consistent* W-graph *from* $f^{Naive}_{MG\mapsto WG}(M_{21})$ *and a consistent* W-graph *from* $f^{Naive}_{MG\mapsto WG}(M22)$.

Since we can reconstruct the *W-graphs* that are computed for a non-decomposed *M-graph* from the *W-graphs* that are computed for its *partial M-graphs*, the probabilistic clusterings that result from processing all *partial M-graphs* independently form a factorization of the probabilistic clustering that results from processing the corresponding *M-graph* without decomposition.

Theorem 29 *Let* $f^{Naive}_{MG\mapsto Cl}$ *be the mapping that maps an* M-graph *to its probabilistic clustering by using the* W-graph *computation mapping* $f^{Naive}_{MG\mapsto WG}$. *Moreover, let* M *be an* M-graph *and let* $f_{dec}(M) = \{M_1 \ldots, M_k\}$ *be its set of* partial M-graphs. *The set of probabilistic clusterings that results from applying* $f_{MG\mapsto Cl}$ *to each* partial M-graph *of* M *separately is a factorization of the probabilistic clustering that results from applying* $f^{Naive}_{MG\mapsto Cl}$ *to* M *itself, i.e.*

$$f^{Naive}_{MG\mapsto Cl}(M) = \boxtimes_{i=1}^{k} f^{Naive}_{MG\mapsto Cl}(M_i)$$

Theorem 29 is proved by Proof 31 in Appendix A.4.

In summary, we profit from *M-graph* decomposition in two ways.

- First, it simplifies the clustering algorithm because it reduces the number of computed *W-graphs* from $\prod_{M^* \in f_{dec}(M)} |f^{\text{Naive}}_{MG\mapsto WG}(M^*)|$ to $\sum_{M^* \in f_{dec}(M)} |f^{\text{Naive}}_{MG\mapsto WG}(M^*)|$. Moreover, each computed *W-graph* covers only a subset of all considered nodes, i.e. the *W-graphs* become smaller and therefore become less expensive to process. In particular the step of validating the consistency of the computed *W-graphs* becomes much more efficient. Finally, the *partial M-graphs* are smaller than the initial *M-graph*. Therefore, *M-graph* consistency validation and inconsistent *M-graph* reparation become cheaper as well.

- Second, *M-graph* decomposition directly produces the probabilistic clustering in a factorized way because from each *partial M-graph* another factor results. Thus, *M-graph* decomposition reduces the complexity of modeling the indeterministic deduplication result within the probabilistic output database (see Section 8.4).

Obviously, the more edges are *negative*, the larger is the number of *partial M-graphs* the initial *M-graph* can be decomposed in. For that reason, *M-graph* decomposition is often only useful in combination with a semi-indeterministic approach that initially reduces the number of uncertain edges.

8.7.4. Complexity of the Full Indeterministic Approach

Before introducing some semi-indeterministic approaches in Section 8.7.5, we formally discuss the complexity of the full-indeterministic approach. As any other technique that is based on the possible worlds semantics, a full-indeterministic approach has, in theory, a very high complexity. The number of *W-graphs* that are generated from an *M-graph* with q uncertain edges by using the *W-graph* computation mapping $f^{\text{Naive}}_{MG \mapsto WG}$ is:

$$N^{\text{Naive}}_{WG}(q) = 2^q \tag{8.7}$$

A fully connected *M-graph* with n nodes has $|E| = \frac{n \times (n-1)}{2}$ edges.

$$N^{\text{Naive}}_{WG\text{-max}}(n) = N^{\text{Naive}}_{WG}(|E|) = 2^{\frac{n \times (n-1)}{2}} \tag{8.8}$$

In the case where each duplicate decision is uncertain (and hence no *M-graph* decomposition is possible), the number of possible worlds (consistent *W-graphs*) that result from processing n database entities is equal to the number of possible partitions of these entities. Thus, the maximal possible number of resultant worlds can be reduced to the complexity of set partitioning and is:

$$N^{\text{Naive}}_{PW\text{-max}}(n) = N^{\text{max}}_{PW} = B_n = \frac{1}{e} \sum_{i=1}^{\infty} \frac{i^n}{i!} \tag{8.9}$$

where B_n is the n-th Bell number [Rot64].

Let db_i be the input database and let db_o be the output database. If each edge of the initial *M-graph* is uncertain, the entities of db_o can be mapped to the power set of the input database's entities without the empty set ($2^{\mathfrak{E}(db_i)} - \emptyset$). As a consequence, in the worst case the number of resultant database entities is:

$$N^{\text{Naive}}_{\mathfrak{E}\text{-max}}(db_o) = |2^{\mathfrak{E}(db_i)}| - 1 = 2^{|\mathfrak{E}(db_i)|} - 1 \tag{8.10}$$

Example 194 *To get an idea of the dramatic complexity scale, we assume an input database db_i with* 10 *entities. The number of* W-graphs *which can be generated from the initial* M-graph *is maximal:*

$$N^{Naive}_{WG\text{-}max}(10) = 2^{45} \simeq 3.5184 \times 10^{13}$$

The number of resultant possible worlds and hence the number of variable-value pairs that is required to model the complementation between all these worlds into the output database is at most:

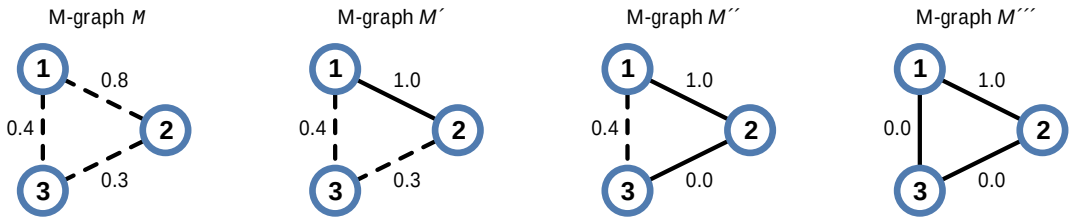
$$N^{Naive}_{PW\text{-}max}(10) = B_{10} = 115,975$$

Finally, the maximal number of entities of the output database db_o is:

$$N^{Naive}_{\mathfrak{E}\text{-}max}(db_o) = 2^{10} - 1 = 1,023$$

which is nearly as hundred times larger than the number of entities of the input database.

In conclusion, the complexity of the full-indeterministic approach as well as the size of the resultant database dramatically increase with the number of uncertain edges. For that reason, a full-indeterministic approach is in general only of theoretical value and is typically not usable in practice.



(a) Setting: $\alpha < 0.2$ (b) Setting: $\alpha \in [0.2, 0.3[$ (c) Setting: $\alpha \in [0.3, 0.4[$ (d) Setting: $\alpha \geq 0.4$

Figure 8.22.: The modified *M-graphs* M', M'', and M''' that result from applying different settings of the (α, β)-restriction to the *M-graph* M

8.7.5. Semi-Indeterministic Approaches

To make the clustering approach feasible in practice, the number of the resultant worlds need to be reduced (preferably to the most probable ones). This can be done in two ways: First by reducing the number of uncertain edges of the initial *M-graph* and second by modifying the function that computes the *W-graphs* from a given *M-graph*. Of course, both concepts can be used in combination to restrict the number of resultant worlds further on.

In the end, the probabilities of all worlds must sum up to 1. Thus, the probabilities of the resultant worlds are conditioned and hence may be distorted. However, the result is usually more accurate than the one world that results from performing a deterministic duplicate detection process.

In our work, we developed two restriction strategies (each one for both restriction concepts). Although we consider them to be most intuitive it cannot be ruled out that other restriction strategies are more useful. The two developed restriction strategies are called (α, β)-restriction and HC-restriction respectively.

8.7.5.1. (α, β)-restriction

To restrict the detection result to the most probable worlds, it make sense to consider only the most uncertain duplicate decisions in an indeterministic way. A decision is most uncertain if its matching probability is 0.5. As a consequence, we utilize the two thresholds $\alpha \leq 0.5$ and $\beta \geq 0.5$ and initially change all matching probabilities that are lower than α to 0 and change all matching probabilities that are equal to or greater than β to 1. To make sure that the most probable worlds result, we always use $\beta = 1 - \alpha$.

For well-chosen values of α and β, the duplicate decisions that have a matching probability outside the range $]\alpha, \beta]$ can be considered to be quite evident. Therefore, they can be deterministically handled without running a high risk of failure and the number of uncertain decisions (and hence the number of uncertain edges) can be reduced to a large extent without considerably decreasing the correctness of the detection result.

Nonetheless, the smaller the range $]\alpha, \beta]$ the more ambigious decisions are considered to be confident and hence the risk of making false decisions increases. In contrast, the larger the range between α and β, the less false decision are included into the detection result, but the greater is its uncertainty. As a consequence, selecting values for α and β is often a trade-off between correctness and certainty.

Example 195 *As an example, we reconsider the* M-graph M *from Example 8.19(a). If* α *is set to a value lower than* 0.2*, none of the three edges is changed and the modified* M-graph *is still the initial* M-graph M *(see Figure 8.22(a)). In contrast, if* α *is between* 0.2 *and* 0.3*, the edge* $\{n_1, n_2\}$ *becomes positive (see* M-graph M' *in Figure 8.22(b)). Consequently, the set of possible* W-graphs *is reduced to* $\{G_2, G_5, G_6, G_8\}$*. If* α *is set to a value larger than* 0.3 *but smaller than* 0.4*, the edge* $\{n_2, n_3\}$ *is additionally changed into a negative one and the* M *is changed into the* M-graph M'' *that is depicted in Figure 8.22(c). From this* M-graph *only the two* W-graphs G_2 *and* G_5 *can be computed. Finally, if we set* α *to a value larger than* 0.4 *all three edges become certain and a single* W-graph*, i.e.* G_2*, can be derived from the modified* M-graph M''' *(see Figure 8.22(d)).*

8.7.5.2. HC-restriction

In the second restriction strategy we utilize the principle of hierarchical clustering as it has been used by Beskales et al. [BSIBD09]. A hierarchical restriction can be incorporated into the clustering approach by replacing $f^{\text{Naive}}_{MG \mapsto WG}$ with another *W-graph* computation mapping.

An intuitive hierarchical restriction is not to create a *W-graph* per every combination of uncertain edges of the *M-graph*, but to process the uncertain edges in the order of their weights. An uncertain edge e of the *M-graph* is then only considered as an edge of the resultant *W-graph* if all other edges of the *M-graph* that have a weight greater than or equal to the weight of e are considered as an edge of this *W-graph*, too. More precisely: Let $M = (N, E, \gamma)$ be an *M-graph*, for each $\tau \in \{\gamma(e) | e \in E\}$ we generate a *W-graph* by replacing each edge $e \in E$ that has a weight greater than or equal to τ with a *positive edge* and by removing all other edges.

The corresponding *W-graph* computation mapping $f^{\text{HC}}_{MG \mapsto WG}$ is defined as:

$$f^{\text{HC}}_{MG \mapsto WG}(M) = \{(N, K, Pr(K)) \mid K \in \bigcup\nolimits_{\tau \in \{\gamma(e) | e \in E\}} \{e \mid e \in E, \gamma(e) \geq \tau\}\} \tag{8.11}$$

where $Pr(K) = \prod_{e \in K} \gamma(e) \prod_{e \in (E^? - K)} (1 - \gamma(e))$.

Note, by doing so a generated *W-graph* is only consistent if either all nodes of two subgraphs are connected or none of them. Thus, this HC-restriction corresponds to the Complete Link clustering method that we have described in Section 4.3.6.5. Of course, other hierarchical clustering methods such as the Single Link method can be used as well.

The probabilities of the resultant *W-graphs* are computed in the same way than in the naive *W-graph* computation mapping. However, the number of resultant *W-graphs* is usually smaller than the number of *W-graphs* that are generated by using the naive mapping (recall, this is even the purpose of the HC-restriction). Thus, the accumulative probability of all *W-graphs* that are computed by using $f^{\text{HC}}_{MG \mapsto WG}$ can be lower than one. In such cases, a final conditioning is actually required. Nevertheless, since we perform a conditioning after removing the inconsistent *W-graphs* anyway, we do not need to perform an extra conditioning.

Example 196 *For illustration, we reconsider the sample* M-graph M *from Figure 8.19(a). By applying the mapping* $f^{HC}_{MG \mapsto WG}$ *to* M *only the four* W-graphs $\{G_1, G_2, G_5, G_8\}$ *result (see Figure 8.19(b)). Since* G_5 *is inconsistent, the clustering process performs the hierarchical clustering that is presented in Figure 8.23.*

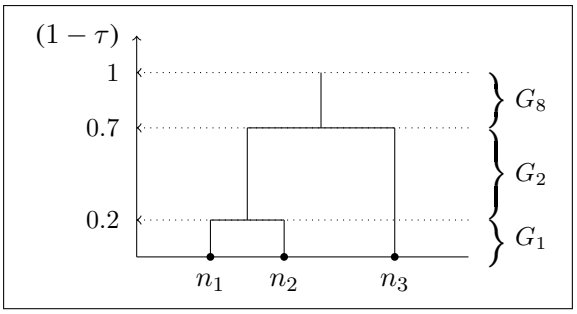


Figure 8.23.: Hierachical clustering on the *M-graph* M from Figure 8.19(a)

In the *W-graph* computation mapping described above, we compute the probability of a *W-graph* by the product of the edges' (inverse) weights. By using the HC-restriction, however, the probability of a particular *W-graph* can also be considered as the range of the matching probability that leads to this graph. Obviously, the resultant probabilities are different to these that result from multiplying the edges' weights. However, this computation method is much more efficient and the remaining probabilities sum up to one. For clarification we will use the mapping ${}^{*}f^{\text{HC}}_{MG\mapsto WG}$ in cases where the original mapping $f^{\text{HC}}_{MG\mapsto WG}$ is modified by the use of an alternative probability computation method.

In the naive approach we compute one *W-graph* from each possible set of uncertain edges. In contrast, by using the HC-restriction we consider only some of these sets for *W-graph* computation. As a consequence, a *W-graph* can only result from the HC-restriction if it results from using the naive *W-graph* computation mapping as well.

Theorem 30 *Let M be an arbitrary* M-graph*, it holds that:* $f^{HC}_{MG\mapsto WG}(M) \subseteq f^{Naive}_{MG\mapsto WG}(M)$

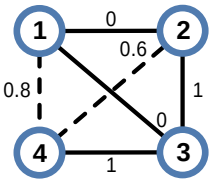
Theorem 30 is proved by Proof 32 in Appendix A.4.

Note that the removal of inconsistent *W-graphs* and thus the conditioning of the remaining *W-graphs* is not included into the *W-graph* computation mapping. Therefore, both *W-graph* computation mappings always compute the same probability for a given *W-graph*.

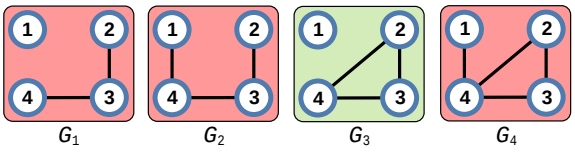
If we use an alternative method for computing the probabilities of the resultant *W-graphs*, Theorem 30 becomes invalid. Nevertheless, the used probability computation method only affects the *W-graphs*' probabilities, but does not affect their nodes and edges. As a consequence, let S be a set of *W-graphs* and let f_{Graph} be a function that maps S to a set of undirected graphs that represent the *W-graphs*' actual graph structures, i.e. $f_{Graph}(S) = \{(N,E) \mid (N,E,Pr) \in S\}$, the set inclusion $f_{Graph}({}^{*}f^{\text{HC}}_{MG\mapsto WG}(M)) \subseteq f_{Graph}(f^{\text{Naive}}_{MG\mapsto WG}(M))$ holds for every possible probability computation method.

8.7.5.3. Consistency

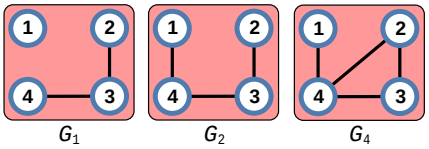
The problem of an inconsistent *M-graph* becomes much more acute if an (α,β)-restriction is used, because the number of certain edges increases enormously. In general, the closer α and β, the higher is the likelihood that the modified *M-graph* becomes inconsistent. Repairing an inconsistent *M-graph* works in the same way as presented in Section 8.7.2, i.e. by changing certain edges to uncertain edges until all $\triangle$-*conflicts* are resolved. Nevertheless, in contrast to the standard repairing approach, we do not need to modify the weight of a changed edge from 1 to 0.99 or from 0 to 0.01 respectively, but instead



(a) The sample *M-graph* M'



(b) The *W-graphs* that result from $f^{\text{Naive}}_{MG\mapsto WG}(M')$



(c) The *W-graphs* that result from $f^{\text{HC}}_{MG\mapsto WG}(M')$

Figure 8.24.: Demonstration that an *M-graph* can be inconsistent w.r.t. $f^{\text{HC}}_{MG\mapsto WG}$ without having a $\triangle$-*conflict*

can use the original weight of the considered edge if this edge was uncertain in the initial *M-graph* and only was changed from uncertain to certain by the used (α, β)-restriction.

According to Theorem 30, an HC-restriction does not produce a *W-graph* that would not be produced by the unrestricted approach, i.e. for every possible *M-graph* M holds that $f^{\text{HC}}_{MG\mapsto WG}(M) \subseteq f^{\text{Naive}}_{MG\mapsto WG}(M)$. As a consequence, if an *M-graph* M has a $\triangle$-*conflict* it is not consistent with respect to $f^{\text{Naive}}_{MG\mapsto WG}$ and hence is not consistent with respect to $f^{\text{HC}}_{MG\mapsto WG}$ either, because $f^{\text{HC}}_{MG\mapsto WG}(M)$ can only contain a consistent *W-graph* if $f^{\text{Naive}}_{MG\mapsto WG}(M)$ contains a consistent *W-graph* as well. Nevertheless, with respect to $f^{\text{HC}}_{MG\mapsto WG}$ the absence of $\triangle$-*conflicts* is only a necessary condition for consistency, but is not a sufficient condition, i.e. an *M-graph* can be inconsistent with respect to $f^{\text{HC}}_{MG\mapsto WG}$ even it does not contain a $\triangle$-*conflict*. As a consequence, Theorem 27 is not valid if an HC-restriction is used.

Example 197 *We will illustrate this fact by the simple* M-graph M' *that is presented in Figure 8.24(a).* M' *does not contain a* $\triangle$*-conflict and from applying* $f^{Naive}_{MG\mapsto WG}$ *to* M' *the four* W-graphs G_1, G_2, G_3, *and* G_4 *that are presented in Figure 8.24(b) result. Since* G_3 *is consistent,* M' *is consistent with respect to* $f^{Naive}_{MG\mapsto WG}$.

In contrast, from applying $f^{HC}_{MG\mapsto WG}$ *to* M' *only the three inconsistent* W-graphs G_1, G_2, *and* G_4 *result (see Figure 8.24(c)). The consistent* W-graph G_3 *does not result because the edge* $\{n_2, n_4\}$ *has a lower weight than the edge* $\{n_1, n_4\}$ *and hence is only contained in* W-graphs *that contain* $\{n_1, n_4\}$ *as well. Since none of the resultant* W-graphs *is consistent,* M' *is not consistent with respect to* $f^{HC}_{MG\mapsto WG}$.

For that reason, repairing an inconsistent *M-graph* cannot be reduced to resolve all its $\triangle$-*conflicts* if we use the HC-restriction. Currently, we do not have developed an efficient strategy that initially validates the consistency or initially repairs the inconsistency of an *M-graph* for the mapping $f^{\text{HC}}_{MG\mapsto WG}$. Notwithstanding, the number of *W-graphs* generated from an *M-graph* by using the mapping $f^{\text{HC}}_{MG\mapsto WG}$ is linear in the number of uncertain edges in the considered *M-graph*. Therefore, consistency can be efficiently checked by performing $f^{\text{HC}}_{MG\mapsto WG}$ itself. If no consistent *W-graph* result, the *M-graph* is inconsistent with respect to $f^{\text{HC}}_{MG\mapsto WG}$ and we use the mapping $f^{\text{Naive}}_{MG\mapsto WG}$ instead.

In summary, given an *M-graph* M, we use the mapping $f^{\text{HC/Naive}}_{MG\mapsto WG}$ that corresponds to the mapping $f^{\text{HC}}_{MG\mapsto WG}$ if it produces a consistent *W-graph* and corresponds to the mapping $f^{\text{Naive}}_{MG\mapsto WG}$ otherwise:

$$f^{\text{HC/Naive}}_{MG\mapsto WG}(M) = \begin{cases} f^{\text{HC}}_{MG\mapsto WG}(M), & \text{if } \exists G \in f^{\text{HC}}_{MG\mapsto WG}(M) : G \text{ is consistent,} \\ f^{\text{Naive}}_{MG\mapsto WG}(M), & \text{else.} \end{cases}$$

Of course, by doing so the actual advantage of the HC-restriction can become effectless for some detection scenarios and we fall back to the high complexity of the naive approach. However, if an HC-restriction is combined with an initial *M-graph* decomposition we only need to use the naive mapping $f^{\text{Naive}}_{MG\mapsto WG}$ for all inconsistent *partial M-graphs* and hence can still profit from the HC-restriction if at least one of the *partial M-graphs* is consistent with respect to $f^{\text{HC}}_{MG\mapsto WG}$.

Another option is to transform the inconsistent *M-graph* into a Duplicate-Pair Graph and then to use a conventional clustering approach as described in Section 4.3.7. By doing so, we do not model any of the corresponding duplicate decisions indeterministically. However, if we combine HC-restriction with *M-graph* decomposition this circumstance only concerns a small part of the whole set of database entities.

8.7.5.4. Decomposition

The more edges are *negative*, the larger is the number of *partial M-graphs* the initial *M-graph* can be decomposed in. For that reason, an (α, β)-restriction enables a decomposition into more and smaller *partial M-graphs*. This in turn improves the efficiency of the clustering approach enormously.

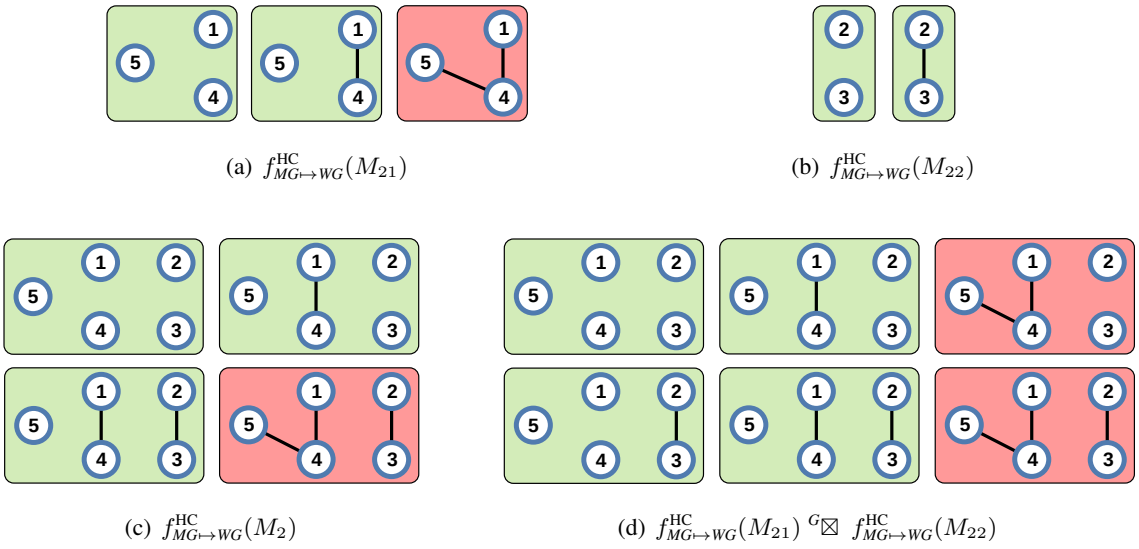
Unfortunately and in contrast to the unrestricted approach, by using $f^{\text{HC}}_{MG\mapsto WG}$ the *W-graphs* that are computed from the non-decomposed *M-graph* cannot be reconstructed from the *W-graphs* that are computed from its *partial M-graphs*.

Theorem 31 *Let* $M = (N, E, \gamma)$ *be an* M-graph *and let* $f_{dec}(M)=\{M_1, \ldots, M_k\}$ *be its set of* partial M-graphs. *The set* $f^{HC}_{MG\mapsto WG}(M)$ *cannot be reconstructed from the sets* $\{f^{HC}_{MG\mapsto WG}(M_1), \ldots, f^{HC}_{MG\mapsto WG}(M_k)\}$ *by using the* W-graph *integration operator* ${}^{G}\boxtimes$, *i.e. it holds that:*

$$f^{HC}_{MG\mapsto WG}(M) \neq {}^{G}\boxtimes_{i=1}^{k}\ f^{HC}_{MG\mapsto WG}(M_i)$$

Theorem 31 can be proved by a simple example.

Example 198 *Let us reconsider the* M-graph M_2 *and its two* partial M-graphs M_{21} *and* M_{22} *that are presented in Figure 8.21. By using the mapping* $f^{HC}_{MG\mapsto WG}$ *from the non-decomposed* M-graph M_2 *four* W-graphs *result from which three are consistent (see Figure 8.25(c)). In contrast, if we compute* W-graphs *from* M_{21} *and* M_{22} *by using the mapping* $f^{HC}_{MG\mapsto WG}$ *three* W-graphs *and two* W-graphs *result respectively. Whereas from the three* W-graphs *in* $f^{HC}_{MG\mapsto WG}(M_{21})$ *two are consistent and one is inconsistent (see Figure 8.25(a)), both* W-graphs *in* $f^{HC}_{MG\mapsto WG}(M_{22})$ *are consistent (see Figure 8.25(b)). From integrating* $f^{HC}_{MG\mapsto WG}(M_{21})$ *with* $f^{HC}_{MG\mapsto WG}(M_{22})$ *the six* W-graphs *that are presented in Figure 8.25(d) result. Since the two sets* $f^{HC}_{MG\mapsto WG}(M_2)$ *and* $f^{HC}_{MG\mapsto WG}(M_{21})\ {}^{G}\boxtimes\ f^{HC}_{MG\mapsto WG}(M_{22})$ *are unequal, Theorem 31 is proved.*



(a) $f^{\text{HC}}_{MG\mapsto WG}(M_{21})$ (b) $f^{\text{HC}}_{MG\mapsto WG}(M_{22})$

(c) $f^{\text{HC}}_{MG\mapsto WG}(M_2)$ (d) $f^{\text{HC}}_{MG\mapsto WG}(M_{21})\ {}^{G}\boxtimes\ f^{\text{HC}}_{MG\mapsto WG}(M_{22})$

Figure 8.25.: Sample for demonstrating that the *W-graphs* of an *M-graph* cannot be reconstructed from the *W-graphs* of its *partial M-graphs* in the case the hierarchical *W-graph* computation mapping $f^{\text{HC}}_{MG\mapsto WG}$ is used.

Whereas in the naive approach *M-graph*-decomposition reduces the total number of computed *W-graphs* (see Section 8.7.3), in the HC-restriction *M-graph*-decomposition increases the total number of computed *W-graphs*. Let $q > 0$ be the number of uncertain edges of an *M-graph* M, the set $f^{\text{HC}}_{MG\mapsto WG}(M)$ has at least two elements (all uncertain edges in M have the same weight) and has at most $q+1$ elements (all uncertain edges in M have different weights). In contrast, let q_i be the number of uncertain edges in the *partial M-graph* $M_i, i \in \{1,\ldots,k\}$ and let $l < k$ be the number of *partial M-graphs* with at least one certain edge. Due to $\sum_{i=1}^{k} q_i = q$ (each uncertain edge of M belongs to exact one *partial M-graph* of M), the total number of *W-graphs* that are computed from all *partial M-graphs* using the mapping $f^{\text{HC}}_{MG\mapsto WG}$ is at least:

$$N^{\text{HC}}_{WG\text{-min}}(q) = \sum_{i=1}^{k} N^{\text{HC}}_{WG\text{-min}}(q_i) = (k-l) + \sum_{i=1}^{l} 2 = (k-l) + 2 \times l = k + l$$

and is at most:

$$N^{\text{HC}}_{WG\text{-max}}(q) = \sum_{i=1}^{k} N^{\text{HC}}_{WG\text{-max}}(q_i) = (q-l) + \sum_{i=1}^{l} (q_i + 1) = (k-l) + (\sum_{i=1}^{l} q_i) + l = k + q$$

Consequently, the total number of *W-graphs* in $\bigcup_{i=1}^{q} f^{\text{HC}}_{MG\mapsto WG}(M_i))$ is greater than the number of *W-graphs* in $f^{\text{HC}}_{MG\mapsto WG}(M)$ and hence using *M-graph* decomposition implicates an overhead in *W-graph* processing.

Actually, if we compute the probabilities of the *W-graphs* based on the (inverse) weights of the *M-graph*'s edges (see Equation 8.11), we can conclude some interesting set inclusions.

Theorem 32 *Let* $M = (N, E, \gamma)$ *be an* M-graph *and let* $f_{dec}(M) = \{M_1, \ldots, M_k\}$ *be its set of* partial M-graphs*, it holds:* $f^{HC}_{MG\mapsto WG}(M) \subseteq {}^{G}\boxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i) \subseteq f^{Naive}_{MG\mapsto WG}(M)$.

Theorem 32 is proved by Proof 33 in Appendix A.4.

As for Theorem 30, the set inclusions become invalid if an alternative probability computation method is used. Nevertheless, as already mentioned for Theorem 30, the set inclusions still hold if we disregard probabilities and only consider the actual graph structure consisting of nodes and edges. As a consequence, Theorem 32 can be generalized to:

$$f_{Graph}({}^{*}f^{\text{HC}}_{MG\mapsto WG}(M)) \subseteq f_{Graph}({}^{G}\boxtimes_{i=1}^{k} {}^{*}f^{\text{HC}}_{MG\mapsto WG}(M_i)) \subseteq f_{Graph}(f^{\text{Naive}}_{MG\mapsto WG}(M))\}$$

From the above discussion we can conclude that in combination with an HC-restriction *M-graph* decomposition increases the total number of *W-graphs*. Moreover, the resultant clustering is much more uncertain than in the non-decomposed case because ${}^{G}\boxtimes_{i=1}^{k} f^{\text{HC}}_{MG\mapsto WG}(M_i) \supseteq f^{\text{HC}}_{MG\mapsto WG}(M)$. Nevertheless, due to the following reasons the non-equivalence between both sets of *W-graphs* is rather an advantage than a disadvantage:

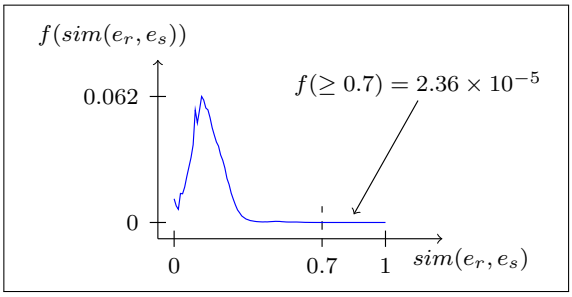
- As presented above, the total number of *W-graphs* increases with *M-graph* decomposition. Nevertheless, the resultant *W-graphs* are smaller than the *W-graphs* that result from using the clustering approach without *M-graph* decomposition and therefore are cheaper to process. Moreover, even if the total number of *W-graphs* increases the computation complexity is still linear and hence combining *M-graph* decompositon with hierarchical clustering is much more efficient than using the naive clustering approach.

- An HC-restriction reduces the number of resultant *W-graphs* by introducing dependencies between the individual duplicate decisions that are actually not given. These introductions, however, are the reason that the resultant probabilistic clustering cannot be factorized into many and small factors because the most of the clustering's uncertain decisions are correlated and therefore cannot be assigned to different absolute independent factors. This in turn is a serious problem because storing the non-factorized representation of a probabilistic clustering is usually not practical for large sets of database entities (see Section 8.4).

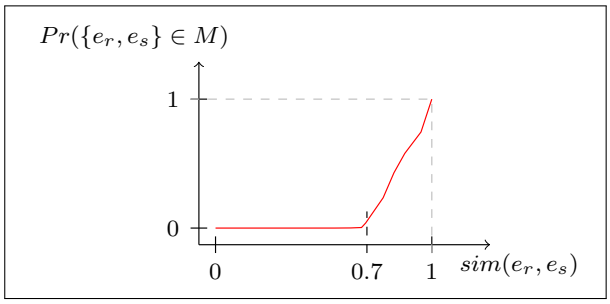
  In contrast, by using *M-graph* decomposition we preserve the independence between the duplicate decisions from different *partial M-graphs* that becomes ignored by performing an HC-restriction on the non-decomposed *M-graph*. As a consequence, the resultant probabilistic clustering can be stronger factorized than the clustering that results from the non-decomposed *M-graph*. Moreover, *M-graph* decomposition directly produces a factorized representation of the final probabilistic clustering. As a consequence, factorization does not need to be computed afterwards.

- The probabilistic clustering that results from processing the decomposed *M-graph* is clearly more uncertain than the probabilistic clustering that results from processing the non-decomposed *M-graph*. However, this circumstance is not necessarily a disadvantage because according to Theorem 32 the resultant probabilistic clustering is closer to the probabilistic clustering that results from using the naive approach than the probabilistic clustering that results from using HC-restriction without *M-graph* decomposition.

In summary, combining an HC-restriction with *M-graph* decomposition increases the number of resultant possible clusterings, but its result comes closer to the result of the naive approach. Moreover, the



(a) Relative frequency of similarity scores



(b) Matching probability per similarity score

Figure 8.26.: Statistics on a CD-data set

processed graphs are smaller and the resultant probabilistic clustering is factorized. As a consequence, combining an HC-restriction with *M-graph* decomposition is a win-win situation because it increases the correctness of the produced probabilistic clustering without losing the great benefits of linear complexity.

8.7.5.5. Complexity

In the worst case, the (α, β)-restriction does not eliminate even one uncertain edge and hence the full-indeterministic approach as well as the semi-indeterministic approaches have theoretically the same complexity. Notwithstanding, usually the most matching probabilities are close to zero. Thus, even marginal restrictions come nowhere near the worst case because they eliminate the majority of the uncertain edges rather than none of them. Since the number of uncertain edges is the dominant factor in the complexity formulas of the full-indeterministic approach (see Section 8.7.4), it means that in practice an (α, β)-restriction moves us into an entirely different area of computation complexity, an area that we have shown in [PvKR13] is well manageable in practice.

Example 199 *For illustration, in [PvKR13] we perform a statistical analysis on the similarities and matching probabilities of database entities from a CD-data set. As presented in Figure 8.26(a), the most similarity scores were close to zero and only around* 0.00236% *of the scores were greater than* 0.7*. In the experiments that we present in [PvKR13] we derive the matching probability of two entities from their similarity. For that purpose, we computed a mapping from similarity to matching probability based on an empirical analysis that was performed on a labeled training data set with* 5000 *entities. As depicted in Figure 8.26(b) only for high similarity scores a noteworthy score of matching probability result. Thus, in this sample scenario even for low values of* α *the number of uncertain edges was reduced to a large extent.*

The complexity reduction that is caused by an HC-restriction can be even theoretically noticed. In the naive approach, the number of *W-graphs* that result from an *M-graph* with q uncertain edges is fixed to $N^{\text{Naive}}_{WG}(q) = 2^q$. In the case of using the *W-graph* computation mapping $f^{\text{HC}}_{MG \mapsto WG}$, this number depends on the diversity of the uncertain edges' weights, because all edges with the same weight are considered collectively. For that reason, the number of *W-graphs* that result from an *M-graph* with q uncertain edges is at least $N^{\text{HC}}_{WG\text{-min}}(q) = 1 + \mathbf{min}(1, q)$ (all uncertain edges have the same weight) and

is at most $N^{\text{HC}}_{WG\text{-max}}(q) = q + 1$ (all uncertain edges have different weights). Nevertheless, even in the worst case the number of resultant *W-graphs* increases only linear with the number of uncertain edges. As a consequence, if we use an HC-restriction the computation complexity is reduced from exponential complexity to linear complexity and the clustering approach performs well even for *M-graphs* with high numbers of uncertain edges.

8.7.6. Sources of Matching Probabilities

The quality of the clustering approach essentially depends on the quality of the used matching probabilities. The matching probabilities can originate from different sources. Of course matching probabilities can result from the two detection approaches (world-based and description-based) that we have presented in Chapter 7 because computing the matching probability can be used as a method for uncertainty resolution (see Section 7.2.9). In this case, the matching probabilities model data uncertainty as well as process uncertainty.

Nevertheless, this clustering approach can be also useful, if no uncertainty has been incorporated into the pairwise matching results and the input of the duplicate clustering phase is a set of certain duplicate decisions that contain certain similarity scores instead of matching probabilities. In this case, the matching probabilities can be derived from the similarity scores in several ways:

1. **Specifications based on Empirical Analyses:** The first way is to analyse some labeled training data as we did in [PvKR13]. By doing so, we first construct a histogram that represents the frequency of the true MATCHES per similarity score. For that purpose we discretize the range of possible similarity scores. Then, we derive a function (we call it a *sim2p-mapping*) that maps from similarity scores to matching probabilities. A higher similarity score is generally an higher indication for a duplicate than a lower similarity score. Therefore, the resultant function is only meaningful if it is non-decreasing. This requirement can be satisfied by applying some curve-fitting modification steps.

   The disadvantage of this way for computing matching probability is that we need labeled training data and that the analyses as well as the curve fitting modifications are time consuming. Furthermore, the resultant mapping is tailor-made for the domain of the training data and maybe cannot be reused for other domains without expensive adaptations.

2. **Specifications based on Threshold Distances:** Another way to compute matching probabilities is to restrict the indeterministic handling of duplicate decisions to the set of POSSIBLE MATCHES. In that case, we can use the distance between the similarity scores to the two thresholds $\theta_{M/P}$ and $\theta_{P/U}$ in order to define the matching probability. The closer the similarity score is to $\theta_{M/P}$ the greater is the likelihood that the corresponding entities are duplicates. In contrast, the closer the similarity score is to $\theta_{P/U}$ the less likely these entities are duplicates. Thus, we can compute the matching probability of two entities e_r and e_s as:

$$Pr(\{e_r, e_s\} \in M) = 1 - \frac{\theta_{P/M} - sim(e_r, e_s)}{\theta_{P/M} - \theta_{U/P}} \tag{8.12}$$

   Since we only compute matching probabilities for all POSSIBLE MATCHES, the clustering approach is already restricted by setting the two thresholds $\theta_{M/P}$ and $\theta_{P/U}$. It is obvious that the

effectiveness and correctness of such a restriction is lower than the effectiveness and correctness of an approach that resolves all POSSIBLE MATCHES by clerical reviews. However, this restriction is a full-automatic approach and hence does not require any effort of domain experts and does not need any labeled training data.

3. **Manual Specifications:** One of the sources of process uncertainty that we have pointed out in Section 7.8.1 is the imperfect knowledge of domain experts. Usually the expert is enforced to classify an entity pair as a MATCH or an UNMATCH. Nevertheless, by using a clustering approach that gets matching probabilities as input, the experts are unburden from making decisions they are not confident about and can specify some matching probabilities instead.

   If manual specifications are the only source of matching probabilities, the indeterministic handling is only applied to a small number of individual entity pairs and therefore the number of resultant worlds remains low.

8.8. Quality of Indeterministic Duplicate Detection Results

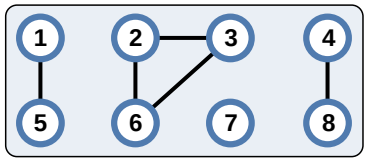
Duplicate detection is heuristic by nature and appropriate configuration settings are usually detected by performing test runs on labeled sample data. For that reason, it is important that we are able to rate the quality of a duplicate detection result with respect to a given gold standard. The quality measures that we have presented in Section 4.3.8, however, are only designed to deal with certain clusterings and hence cannot be used to rate the quality of indeterministic duplicate detection results.

In general, the purpose of quality evaluation is not only to rate the absolute quality of a single detection result, but also to compare the qualities of several detection results. Thus, evaluating the quality of an uncertain duplicate clustering can be useful to

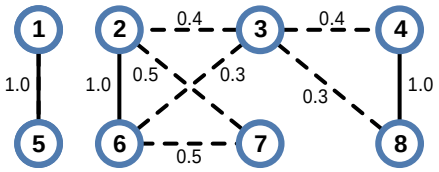
- compare a deterministic duplicate detection result with an indeterministic duplicate detection result and hence to enable a quantification of the benefit that results from modeling ambiguous duplicate decisions in an indeterministic way.
- compare a full-indeterministic duplicate detection result with a semi-indeterministic duplicate detection result and hence to enable a cost-benefit analysis of the individual semi-indeterministic detection approaches that we have proposed in Section 8.7.5.
- find the configuration setting of a specific (full- or semi-)indeterministic duplicate detection approach that fits best with a given application scenario.

Moreover, a quality evaluation of uncertain clusterings can be also useful for tuning the configuration of deterministic duplicate detection processes that intermediately produce a set of POSSIBLE MATCHES because clerical reviews are expensive and rating the correctness of the certain duplicate decisions can help to select the set of POSSIBLE MATCHES carefully.

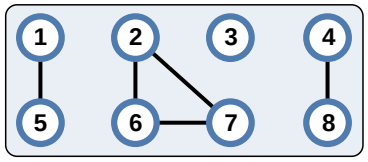
The broad meaning of data quality is typically broken down into a set of quality dimensions [WZL01, BS06]. The underlying idea of indeterministic duplicate detection is to improve the correctness of the detected duplicates at the expense of certainty. For that reason, we have to consider two quality dimensions: *decision correctness* and *decision certainty*. The absolute quality of a detection result is



(a) Gold standard $\mathcal{C}_{\text{gold}}$



(b) Matching graph M



(c) Certain clustering $\mathcal{C}_1$

possible clusterings of Γ_2	Pr_2
$\mathcal{C}_{21} = \{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle, \langle e_7\rangle, \langle e_4, e_8\rangle\}$	0.4
$\mathcal{C}_{22} = \{\langle e_1, e_5\rangle, \langle e_2, e_6, e_7\rangle, \langle e_3, e_4, e_8\rangle\}$	0.4
$\mathcal{C}_{23} = \{\langle e_1, e_5\rangle, \langle e_2, e_6, e_7\rangle, \langle e_3\rangle, \langle e_4, e_8\rangle\}$	0.2

(d) Probabilistic clustering $\mathfrak{C}_2 = (\Gamma_2, Pr_2)$

possible clusterings of Γ_3	Pr_3
$\mathcal{C}_{31} = \{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle, \langle e_7\rangle, \langle e_4, e_8\rangle\}$	0.22
$\mathcal{C}_{32} = \{\langle e_1, e_5\rangle, \langle e_2, e_6, e_7\rangle, \langle e_3\rangle, \langle e_4, e_8\rangle\}$	0.22
$\mathcal{C}_{33} = \{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3, e_4, e_8\rangle, \langle e_7\rangle\}$	0.22
$\mathcal{C}_{34} = \{\langle e_1, e_5\rangle, \langle e_2, e_6, e_7\rangle, \langle e_3, e_4, e_8\rangle\}$	0.2
$\mathcal{C}_{35} = \{\langle e_1, e_5\rangle, \langle e_2, e_3, e_6\rangle, \langle e_4, e_8\rangle, \langle e_7\rangle\}$	0.14

(e) Probabilistic clustering $\mathfrak{C}_3 = (\Gamma_3, Pr_3)$

possible clusterings of Γ_4	Pr_4
$\mathcal{C}_{41} = \{\langle e_1, e_5\rangle, \langle e_2, e_3, e_6\rangle, \langle e_4, e_8\rangle, \langle e_7\rangle\}$	0.35
$\mathcal{C}_{42} = \{\langle e_1, e_5\rangle, \langle e_2\rangle, \langle e_3, e_4\rangle, \langle e_6\rangle, \langle e_7, e_8\rangle\}$	0.22
$\mathcal{C}_{43} = \{\langle e_1, e_2, e_5, e_6\rangle, \langle e_3\rangle, \langle e_4\rangle, \langle e_7\rangle, \langle e_8\rangle\}$	0.2
$\mathcal{C}_{44} = \{\langle e_1, e_5\rangle, \langle e_2, e_3, e_6, e_7\rangle, \langle e_4\rangle, \langle e_8\rangle\}$	0.13
$\mathcal{C}_{45} = \{\langle e_1, e_5\rangle, \langle e_2, e_7\rangle, \langle e_3, e_6\rangle, \langle e_4\rangle, \langle e_8\rangle\}$	0.1

(f) Probabilistic clustering $\mathfrak{C}_4 = (\Gamma_4, Pr_4)$

Figure 8.27.: The gold standard, the sampe *M-graph*, and the four detection results of our motivating example

always a trade-off between both dimensions. Nevertheless, we can generally say that a detection result is of better quality than another detection result, if it is more correct than it and at least as certain as it or if it is more certain than it and at least as correct as it. In this case, we say that the first detection result is superior to the second. Obviously, the meaning of superiority depends on the meaning of decision correctness and decision certainty.

Broadly, the quality of a database can be defined as its *fitness for use* [TB98] (also denoted as *fitness for purpose* [HG93]). As a consequence, the meaning of decision correctness and the meaning of decision certainty essentially depend on the use of the considered indeterministic duplicate detection result. For that reason, we introduce several quality semantics in Section 8.8.1 and then present measures of decision correctness and decision certainty for each of these semantics in Section 8.8.2 or Section 8.8.3 respectively. An uncertain clustering that represents an indeterministic duplicate detection result is usually given in a factorized way and reconstructing the non-factorized representation is computational expensive (or sometimes even impossible). Therefore, we discuss methods to efficiently compute the decision correctness and the decision certainty of an uncertain clustering based on the clustering's factors for each of the presented semantics in Section 8.8.5.

Before starting the quality discussion in detail, we want to motivate this discussion by a simple example.

Example 200 *For motivation we cop a look at the example that is presented in Figure 8.27. This example considers duplicate detection in a set of eight database entities. The gold standard is depicted in Figure 8.27(a) and contains four duplicate clusters. For comparison we consider one deterministic duplicate detection result (the certain clustering $\mathcal{C}_1$) and three indeterministic duplicate detection results (the probabilistic clusterings $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$). These clusterings are depicted in Figure 8.27(c) to Figure 8.27(f). The certain clustering $\mathcal{C}_1$ as well as the first two probabilistic clusterings $\mathfrak{C}_2$ and $\mathfrak{C}_3$ are computed based on the* matching-graph M *that is presented in Figure 8.27(b). The clustering $\mathcal{C}_1$ is computed by constructing a consistent* W-graph *from M that consists every edge of M whose weight is greater than or equal to 0.5. As a consequence, this clustering results from utilizing the clustering approach that we have presented in Section 8.7 by combining it with a $(0.5, 0.5)$-restriction. Note, since all entity pairs that have a matching probability of 0.5 are considered as* MATCHES*, producing $\mathcal{C}_1$ as the detection result means to marginally prefer false positives to false negatives. In contrast, the two probabilistic clusterings $\mathfrak{C}_2$ and $\mathfrak{C}_3$ are computed from M without an (α, β)-restriction. However, whereas $\mathfrak{C}_3$ is computed from M by using the naive* W-graph *computation mapping $f^{Naive}_{MG \mapsto WG}$ (full-indeterministic approach), $\mathfrak{C}_2$ is computed from M by using the hierarchical* W-graph *computation mapping $f^{HC}_{MG \mapsto WG}$ (semi-indeterministic approach). The probabilistic clustering $\mathfrak{C}_4$ is not computed based on M, but results from another detection process that is assumed to be unknown to us, i.e. we only know the result.*

We will reuse these four sample clusterings in the rest of this section in order to illustrate the individual quality semantics and quality measures. Some illustrations require several equally probable clustering alternatives. For that reason, we do not use the matching probabilities of M to compute the probabilities of the alternatives as described in Section 8.7, but set the individual probabilities as presented in Figure 8.27(c) to Figure 8.27(f).

At a first glance the relative quality of the individual clusterings seems completely unclear. Even with the gold standard at hand, we cannot prefer one probabilistic clustering to another. Obviously, the most certain result is this presented by $\mathcal{C}_1$. Nevertheless, it contains two false negatives as well as two false positives and hence is only less decision correct, e.g. its Pair F_1-score is only $pF_1(\mathcal{C}_1, \mathcal{C}_{gold}) = 0.6$. The most probable alternative of $\mathfrak{C}_4$ is absolutely decision correct, but its other alternatives are only less decision correct. Clustering $\mathfrak{C}_3$ contains an absolute decision correct alternative as well, but this alternative is the least probable alternative of $\mathfrak{C}_3$. In general, the most probable alternatives of $\mathfrak{C}_2$ and $\mathfrak{C}_3$ are less decision correct than the most probable alternative of $\mathfrak{C}_4$, but their remaining alternatives are much more decision correct than the remaining alternatives of $\mathfrak{C}_4$. Moreover, the alternatives of $\mathfrak{C}_2$ have many duplicate decisions in common. Thus, much of its clustering information is certain. The same holds for clustering $\mathfrak{C}_3$.

In summary, we can see that each of the presented detection results scores in another quality aspect, but we cannot simply see whether or not one of these detection results is more suitable than the others and hence generally fits best. This demonstrates that we need to consider the meaning of quality in more detail as we will do in the remainder of this section.

At the beginning of this section we introduced the concept of superiority for comparing uncertain duplicate clusterings with respect to the two quality dimensions decision correctness and decision certainty. Each of these dimensions, however, can be represented by several quality measures. Moreover, we introduce a set of quality semantics in Section 8.8.1 and in each of these semantics we propose different ways

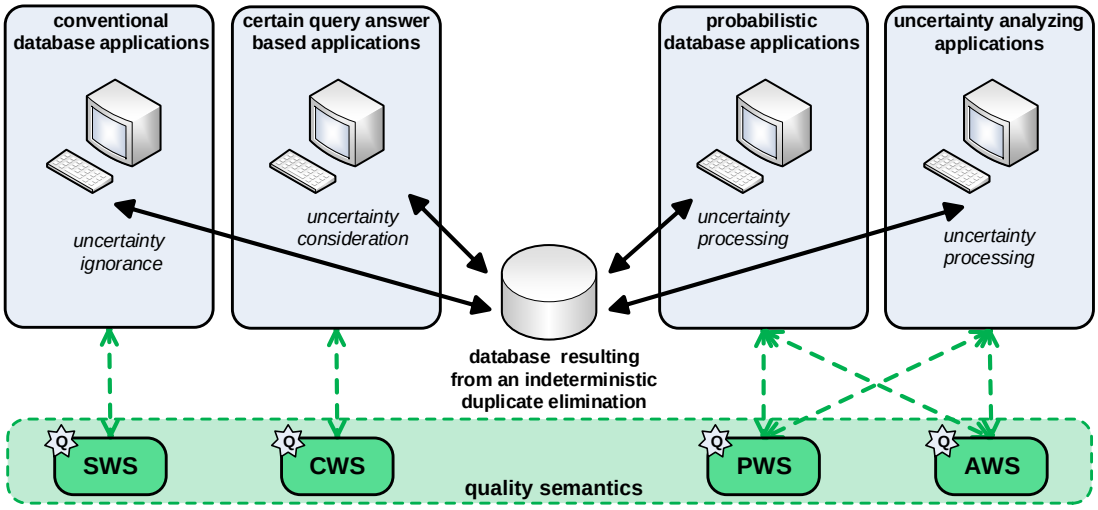


Figure 8.28.: The different handlings of data uncertainty and their corresponding quality semantics

for adapting a quality measure to the clusterings' uncertainty. Therefore it is not uncommon that we need to compare duplicate clusterings with respect to several categories of detection quality. For that reason, we generalize the concept of superiority and say that a first duplicate clustering is superior to a second duplicate clustering with respect to a set of categories if its quality is better in some of these categories and is not worse in the remaining categories.

8.8.1. Quality Semantics

As mentioned before the meaning of data quality primarily depends on the data's fitness for use and hence depends on the considered use case, i.e., the way the database is further processed. As a consequence, the way decision correctness is affected by the uncertainty of a duplicate clustering and the way we have to define the meaning of decision certainty depends on the way the clustering uncertainty is processed by a database application. In Section 8.5.1, we pointed out that all applications that potentially process indeterministic deduplication results can be distinct into four classes: conventional database applications, certain query answer based applications, probabilistic database applications, and uncertainty analyzing applications. Recall from Section 8.5.1 that we can derive three different kinds of handling data uncertainty from these four classes. Whereas conventional database applications only use a single possible world (and hence only a single duplicate clustering alternative) as input and therefore ignore any kind of uncertainty, certain query answer based applications resolve duplicate uncertainty by distinguishing between certain duplicate decisions and uncertain duplicate decisions. In contrast, probabilistic database applications and uncertainty analyzing applications directly process data uncertainty and pass it to the application result. Consequently, the meaning of quality is different for some of these application classes and we can derive three *quality semantics*. Since the quality semantics that we propose for the uncertainty processing applications cannot be efficiently computed in many cases, we also introduce a fourth semantics that can be considered as a simpler version of the third. The connections between the different handlings of data uncertainty and the four quality semantics are graphically presented in Figure 8.28.

1. **Possible Worlds Semantics (short PWS):** If the database is processed accordingly to the *possible worlds semantics* (see Section 3.2), it seems most meaningful to rate the quality of an indeterministic duplicate detection result by an uncertain quality score. Moreover, this approach allows a subtle analysis of the uncertain clustering's quality by further processing the quality score's uncertainty. For example, it allows to determine to what probability a quality score is greater than a user-specific threshold. Thus, this semantics is also appropriate for applications that analyze the data's inherent uncertainty. Nevertheless, this semantics is computationally intensive because in the worst case for each of the uncertain clustering's alternatives another quality score results and hence the uncertain quality score can have as many alternatives as the uncertain clustering.

2. **Aggregated World Semantics (short AWS):** Since the possible worlds semantics is computationally intensive, we additionally design the *aggregated world semantics* that can be used for probabilistic database applications and uncertainty analyzing applications instead of the possible worlds semantics. In contrast to the possible worlds semantics it does not compute an uncertain quality score, but aggregate the quality scores over all possible clusterings. Since it is based on the use of aggregation functions like **min**, **max** or **exp**, this semantics allows a rough analysis of the clustering's quality. For example, it allows to query the worst case scenario (minimal quality score), the expected scenario (expected quality score), and the best case scenario (maximal quality score) of the evaluated indeterministic duplicate detection result.

3. **Single World Semantics (short SWS):** The *single world semantics* is designed to rate the quality of an indeterministic duplicate detection result with conventional database applications in mind. Since these applications pick one of the possible clusterings (worlds) as input, it is most meaningful to rate the quality of an indeterministic duplicate detection result as the quality of the selected clustering alternative.

4. **Certain World Semantics (short CWS):** The *certain world semantics* is tailor-made for applications that are based on certain query answers. Since the correctness of a certain query result can be only affected (positive as well as negative) by the correctness of the certain duplicate decisions, it is most meaningful to rate the decision correctness of an indeterministic duplicate detection result as the quality of its certain clustering information.

It is important to note that these semantics are no competitors in general, but each of them has been developed for a specific application scenario. Moreover, all the presented semantics are designed to take uncertainty of duplicate clusterings into account and hence are equal in the case of absolute certainty. As a consequence, the quality of the sample clustering $\mathcal{C}_1$ is the same in all these semantics, i.e. its decision certainty is always one and its decision correctness only depends on the quality measure QM that is used for comparing clustering alternatives with the gold standard.

8.8.2. Decision Correctness

In deterministic duplicate detection, decision correctness is defined as the similarity of the duplicate clustering to the gold standard. Thus, every measure that we have presented in Section 4.3.8 is a measure of decision correctness. Whereas in the certain case the meaning of decision correctness is clearly defined,

it is not clearly defined for the uncertain case. As we have shown in the motivating example, decision correctness actually does not describe certainty, but is affected by it because an uncertain clustering has multiple alternatives and each alternative can have another similarity to the gold standard. As discussed above, the concrete meaning of decision correctness depends on the way clustering uncertainty is processed by the considered database application. Therefore, the meaning of decision correctness differs from one quality semantics to another.

8.8.2.1. Possible Worlds Semantics

Probabilistic database applications process the probabilistic database based on the principles of the possible worlds semantics (see Section 3.2 and Section 6.1). Consequently, it seems logical to interpret quality in the same way.

A quality measure for decision correctness can be considered as a function that gets two certain clusterings (the detection result and the gold standard) as input and produces a certain quality score as output. Consequently, in the possible world semantics, the result of applying such a measure to an uncertain clustering (detection result) and a certain clustering (gold standard) is defined by its incomplete version or by its probabilistic version respectively. Therefore, let *QM* be a measure for the decision correctness of certain clusterings that has the score domain dom_Q, the decision correctness of an incomplete clustering Γ with respect to a gold standard $\mathcal{C}_{\text{gold}}$ and the measure *QM* under the Possible Worlds Semantics is defined as the incomplete quality score:

$$y_{QM} = QM^{PWS}(\Gamma, \mathcal{C}_{\text{gold}}) = \{QM(\mathcal{C}, \mathcal{C}_{\text{gold}}) \mid \mathcal{C} \in \Gamma\}$$

Accordingly, the decision correctness of a probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ with respect to a gold standard $\mathcal{C}_{\text{gold}}$ and the measure *QM* under the Possible Worlds Semantics is defined as the probabilistic quality score:

$$\begin{aligned} z_{QM} = QM^{PWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}}) &= (y_{QM}, Pr_{QM}) \\ \text{where} \quad & y_{QM} = \{QM(\mathcal{C}, \mathcal{C}_{\text{gold}}) \mid \mathcal{C} \in \Gamma\} \\ \text{and} \quad & \forall x \in dom_Q \colon Pr_{QM}(x) = \sum\nolimits_{\mathcal{C} \in \Gamma, QM(\mathcal{C}, \mathcal{C}_{\text{gold}}) = x} Pr(\mathcal{C}) \end{aligned}$$

Example 201 *For illustration, we consider the four clusterings $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ from the motivating example. The pairwise computed decision correctness of the alternatives of each of these clusterings are presented in Table 8.2. In the Possible Worlds Semantics, the decision correctness of a probabilistic clustering is considered as a probabilistic quality score that has each quality score of the possible clusterings as an alternative. Since different possible clusterings can have the same similarity to the gold standard, the quality score of an uncertain clustering has always equal or less alternatives than the corresponding clustering. For demonstration, the probabilistic Pair Recall scores of the four sample clusterings are listed in Table 8.3. Since the alternatives of $\mathfrak{C}_2$ have all the same Pair Recall, the Pair Recall of $\mathfrak{C}_2$ is a certain score although $\mathfrak{C}_2$ is uncertain. Considering only Pair Recall, $\mathfrak{C}_3$ is superior to $\mathcal{C}_1$ and is superior to $\mathfrak{C}_2$ because its Pair Recall is always greater than or equal to the Pair Recall of $\mathcal{C}_1$ or $\mathfrak{C}_2$ respectively.*

Clustering	Alternative	Pr	$\lvert p\text{TP}\rvert$	$\lvert p\text{FP}\rvert$	$\lvert p\text{FN}\rvert$	pRec	pPrec	pF_1
$\mathcal{C}_1$	-	1.0	3	2	2	0.6	0.6	0.6
$\mathfrak{C}_2$	$\mathcal{C}_{21}$	0.4	3	0	2	0.6	1.0	0.75
	$\mathcal{C}_{22}$	0.4	3	4	2	0.6	0.429	0.5
	$\mathcal{C}_{23}$	0.2	3	2	2	0.6	0.6	0.6
$\mathfrak{C}_3$	$\mathcal{C}_{31}$	0.22	3	0	2	0.6	1.0	0.75
	$\mathcal{C}_{32}$	0.22	3	2	2	0.6	0.6	0.6
	$\mathcal{C}_{33}$	0.22	3	2	2	0.6	0.6	0.6
	$\mathcal{C}_{34}$	0.2	3	4	2	0.6	0.429	0.5
	$\mathcal{C}_{35}$	0.14	5	0	0	1.0	1.0	1.0
$\mathfrak{C}_4$	$\mathcal{C}_{41}$	0.35	5	0	0	1.0	1.0	1.0
	$\mathcal{C}_{42}$	0.22	1	2	4	0.2	0.333	0.25
	$\mathcal{C}_{43}$	0.2	2	4	3	0.4	0.333	0.364
	$\mathcal{C}_{44}$	0.13	4	3	1	0.8	0.571	0.67
	$\mathcal{C}_{45}$	0.1	2	1	3	0.4	0.667	0.5

Table 8.2.: The decision correctness of the alternatives of the four sample clusterings $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ computed by several pairwise measures

Since an uncertain correctness score contains all the information on the uncertain clustering's decision correctness, it allows a subtle analysis of different quality aspects. For example, we can compute the probability that the decision correctness of a considered uncertain clustering is below a given threshold and hence can compute a kind of risk.

Example 202 *Let us consider the probabilistic Pair Recall $z = pRec(\mathfrak{C}_3)$ with $z = (y, Pr)$ from Table 8.3 and let us assume that we need to guarantee that the Pair Recall of the duplicate detection result is greater than 0.7. The risk of violating this requirement can now be computed as the probability that the probabilistic Pair Recall z is lower than or equal to 0.7, i.e.:*

$$risk = \sum\nolimits_{x \in y, x \leq 0.7} Pr(x) = 0.52$$

8.8.2.2. Aggregated World Semantics

Although an uncertain quality score is very useful for analyzing the uncertainty of the clustering's decision correctness, it is expensive to compute. For that reason, we additionally introduce the Aggregated World Semantics that resolves the uncertainty of an uncertain quality score by utilizing an aggregate function and hence derives a certain quality score from an uncertain clustering.

Let *QM* be a measure for decision correctness, let **agg** be an aggregate function that is defined on the score domain of *QM*, the decision correctness of an incomplete clustering Γ with respect to a gold standard $\mathcal{C}_{\text{gold}}$, the measure *QM*, and the function **agg** under the aggregated world semantics is defined as the certain quality score:

$$QM^{AWS}(\Gamma, \mathcal{C}_{\text{gold}}) = \mathbf{agg}(QM^{PWS}(\Gamma, \mathcal{C}_{\text{gold}})) = \mathbf{agg}(\{QM(\mathcal{C}, \mathcal{C}_{\text{gold}}) \mid \mathcal{C} \in \Gamma\})$$

Quality Score	Alternative	Pr
$p\text{Rec}(\mathcal{C}_1)$	0.6	1.0
$p\text{Rec}(\mathfrak{C}_2)$	0.6	1.0
$p\text{Rec}(\mathfrak{C}_3)$	0.6	0.86
	1.0	0.14
$p\text{Rec}(\mathfrak{C}_4)$	1.0	0.35
	0.4	0.3
	0.2	0.22
	0.8	0.13

Table 8.3.: The probabilistic Pair Recall of the four sample clusterings

	pRec			pPrec			pF_1		
Clustering	**min**	**max**	**exp**	**min**	**max**	**exp**	**min**	**max**	**exp**
$\mathcal{C}_1$	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
$\mathfrak{C}_2$	0.6	0.6	0.6	0.429	1.0	0.692	0.5	0.75	0.62
$\mathfrak{C}_3$	0.6	1.0	0.656	0.429	1.0	0.71	0.5	1.0	0.669
$\mathfrak{C}_4$	0.2	1.0	0.618	0.333	1.0	0.631	0.25	1.0	0.615

Table 8.4.: The aggregated scores of Pair Recall, Pair Precision, and Pair F_1-score of the sample clusterings

Accordingly, for a probabilistic clustering $\mathfrak{C}$ holds:

$$QM^{AWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}}) = \mathbf{agg}(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}}))$$

Theoretically, an aggregated quality score results from aggregating the alternatives of an uncertain quality score and hence its computation theoretically implicates the same complexity as a computation of quality under the Possible Worlds Semantics. However, for many aggregate functions, the aggregated value can be computed directly on the factorized uncertain clustering without computing the uncertain quality score first (see Section 8.8.5.4).

Each aggregate function extracts some specific characteristics from the uncertain quality score. Consequently, by using multiple aggregate functions such as the minimal value, the maximal value, and the expected value, we can derive a useful description on the uncertainty of the clustering's quality without producing an uncertain quality score first.

Moreover, although a set aggregated quality scores is usually less informative than an uncertain quality score, aggregation can improve the interpretability of the quality evaluation result because it can be easier to compare two uncertain clusterings by comparing two small sets of quality scores than by comparing two uncertain quality scores that each have hundred of alternatives.

Example 203 *For illustration, we consider the aggregated Pair Recall, the aggregated Pair Precision, and the aggregated Pair F_1-score of the four sample clusterings. All these quality scores are presented in Table 8.4. Again, if we restrict our consideration to Pair Recall, $\mathfrak{C}_3$ is superior to $\mathcal{C}_1$ and $\mathfrak{C}_2$ because its*

minimal Pair Recall is equal to the maximal Pair Recall of the other two. If we additionally consider Pair Precision and Pair F_1-score we cannot generally prefer one clustering to another because each of them has their benefits. The clustering $\mathfrak{C}_3$ has the highest expect Pair F_1-score. Nevertheless, the minimal Pair F_1-score of $\mathcal{C}_1$ is greater than these of the three probabilistic clusterings. Thus, if we need to guarantee that the Pair F_1-score is above 0.5 this clustering fits best. In contrast, the clusterings $\mathfrak{C}_2$ and $\mathfrak{C}_4$ impress with their maximal Pair Precision.

8.8.2.3. Single World Semantics:

Conventional database applications ignore data uncertainty by only processing a single possible world. Thus, the quality of an application result only depends on the quality of the possible world that has been selected by this application. Each of the uncertain database's worlds relies on a possible clustering of the performed indeterministic duplicate detection result (note, because duplicate merging can produce uncertainty as well, several worlds can rely on the same possible clustering, but each world relies on exact one possible clustering). As a consequence, it is most meaningful to define the decision correctness of an uncertain clustering as the quality of its alternative the selected world relies on. Therefore, let *QM* be a measure for the decision correctness of certain clusterings, the decision correctness of an incomplete clustering Γ with respect to a gold standard $\mathcal{C}_{\text{gold}}$ and the measure *QM* under the Single World Semantics is defined as the incomplete quality score:

$$QM^{SWS}(\Gamma, \mathcal{C}_{\text{gold}}) = QM(\mathcal{C}, \mathcal{C}_{\text{gold}}) \tag{8.13}$$

where $\mathcal{C} \in \Gamma$ is the duplicate clustering the selected world relies on.

Duplicate detection quality is usually evaluated for training purposes and hence before the actual deduplication process has been performed or is evaluated to a small sample of the detection result instantly after the detection process has been performed in order to verify the process's suitability. At this time, however, we sometimes do not know the exact possible worlds the database applications will later work on, but only know some criteria that these applications will use for world selection instead. For that reason, the decision correctness of an uncertain clustering need to be rated in another way.

In the case of a probabilistic database it make sense to take one of the most probable worlds as application input. Considering indeterministic deduplication results, high probable worlds relies on high probable clusterings. Therefore, it seems natural to select the most probable clusterings for quality evalution.

Definition 73 (Top-Level of a Probabilistic Clustering): *Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering, and let $Pr_{max}(\mathfrak{C}) = \mathbf{max}_{\mathcal{C}\in\Gamma} Pr(\mathcal{C})$ be the maximal probability of all alternatives in $\mathfrak{C}$. The incomplete clustering $\widehat{\mathfrak{C}} = \{\mathcal{C} \in \Gamma \mid Pr(\mathcal{C}) = Pr_{max}(\mathfrak{C})\}$ contains the most probable alternatives of $\mathfrak{C}$ and is called the top-level of $\mathfrak{C}$.*

The differences between the probabilities of the individual worlds can be extremely small. Moreover, because of duplicate merging the most probable world must not necessarily rely on the most probable clustering. For instance, assume that from the most probable clustering (0.5 probability) two equally probable worlds have been derived, but from the second most probable clustering (0.4 probability) only

one world has been derived. Since the probability of a possible clustering is equal to the accumulative probability of all the worlds that rely on this clustering, the most probable world would rely on the second most probable clustering instead of the most probable clustering. For that reason, it makes sometimes sense to introduce a probability tolerance τ and then to define the top-level of a probabilistic clustering as a set that contains all its alternatives that have a probability that is greater than $Pr_{max}(\mathfrak{C}) - \tau$. Nevertheless, because such an approach complicates the factor-based computation that is presented in Section 8.8.5.5, we abstain from considering such a tolerance value in this thesis.

Besides probabilities each application may have some further property-based criteria to prefer some clustering alternatives to the others. For that reason, we can additionally reduce the set of candidate clusterings by performing a clustering selection[8] σ_ϕ that is based on an application-specific selection predicate ϕ. Two selection predicates that are meaningful in the context of duplicate detection are *maxCl* and *minCl*:

- The predicate *maxCl* selects the alternatives that have the largest number of clusters. This selection is especially appropriate for applications that consider false positives to be worse than false negatives. Thus, let Γ be an incomplete clustering, selecting alternatives of Γ by using *maxCl* is defined as:

$$\sigma_{maxCl}(\Gamma) = \{\mathcal{C} \mid \mathcal{C} \in \Gamma, |\mathcal{C}| = \mathbf{max}_{\mathcal{C}' \in \Gamma}|\mathcal{C}'|\} \tag{8.14}$$

- In contrast, the predicate *minCl* selects the alternatives that have the smallest number of clusters. This selection is especially appropriate for applications that consider false negatives to be worse than false positives. Thus, let Γ be an incomplete clustering, selecting alternatives of Γ by using *minCl* is defined as:

$$\sigma_{minCl}(\Gamma) = \{\mathcal{C} \mid \mathcal{C} \in \Gamma, |\mathcal{C}| = \mathbf{min}_{\mathcal{C}' \in \Gamma}|\mathcal{C}'|\} \tag{8.15}$$

By using the application-specific selection predicate we often can restrict the space of possibly selected clusterings to a large extent. Of course, besides *maxCl* and *minCl* other selection predicates are conceivable. If no property-based selection is specified (e.g. the application does not consider false negatives to be worse than false positives and vice versa), a tautology is automatically used as selection predicate, i.e. every alternative is selected as a candidate. In summary, let $\mathfrak{C}$ be a probabilistic clustering and let ϕ be an application-specific selection predicate, the set that contains all candidate clusterings of $\mathfrak{C}$ is defined as:

$$S_{Cand.}(\mathfrak{C}, \phi) = \sigma_\phi(\widehat{\mathfrak{C}}) \tag{8.16}$$

Since we do not know which of the candidate clusterings is actually used as application input, we rate the decision correctness of an uncertain clustering with respect to a particular application as the aggregated decision correctness of all its candidate clusterings. Therefore, let *QM* be a measure for the decision correctness of certain clusterings, let **agg** be an aggregate function that is defined on the score domain of *QM*, and let ϕ be an application-specific selection predicate, the decision correctness of a

[8] Note, the clustering selection operator is actually defined on a set of clusterings instead of an incomplete clustering. However, by ignoring the mutual exclusion between the clustering alternatives each incomplete clustering can be considered as an ordinary clustering set and we therefore use the crisp version of the selection operator for processing an incomplete input.

probabilistic clustering $\mathfrak{C}$ with respect to a gold standard $\mathcal{C}_{\text{gold}}$, the measure QM, the function **agg**, and the selection predicate ϕ under the Single World Semantics is defined as the certain quality score:

$$QM^{SWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}}) = \mathbf{agg}_{\mathcal{C} \in S_{Cand.}(\mathfrak{C},\phi)} QM(\mathcal{C}, \mathcal{C}_{\text{gold}}) \tag{8.17}$$

Of course, if we have to rate the quality with respect to multiple applications, we can compute an individual quality score per application and then can finally combine these scores in an appropriate way, e.g. by calculating the weighted average.

It is conceivable that many applications also have a quality requirement, i.e. for an application the quality score should be greater than the threshold θ_Q. In this case, decision correctness of the probabilistic clustering $\mathfrak{C}$ can be also considered as the share of candidate clusterings that fulfill the application's requirement, i.e.:

$$QM^{SWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}}) = \frac{|\{\mathcal{C} \mid \mathcal{C} \in S_{Cand.}(\mathfrak{C}, \phi), QM(\mathcal{C}, \mathcal{C}_{\text{gold}}) \geq \theta_Q\}|}{|S_{Cand.}(\mathfrak{C}, \phi)|} \tag{8.18}$$

Note, that computing the amount of candidate clusterings whose correctness is greater than θ_Q can be considered as an aggregate function and therefore is included in Equation 8.17. However, for identification reasons we call this approach as threshold-based aggregation in the rest of this thesis.

The decision correctness of an incomplete clustering under the Single World Semantics is defined accordingly. However, in contrast to a probabilistic clustering, an incomplete clustering does not model probabilities. For that reason, we cannot directly prefer one alternative to another and define the top-level of an incomplete clustering as the incomplete clustering itself.

The resultant quality scores can then be used for either compare two uncertain clusterings or can be used to verify the risk that the finally selected world of a given deduplication result does not satisfy the application's requirements.

Example 204 *For illustration, we reconsider the four clusterings $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ from the motivating example. The top-level as well as the final sets of candidate clusterings of $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ are presented in Table 8.5(a) for each of the three above discussed selection predicates (the predicate 1=1 represents the aforementioned tautology that is satisfied by every clustering). Whereas the top-level of $\mathfrak{C}_4$ contains only a single alternative, the top-levels of $\mathfrak{C}_2$ and $\mathfrak{C}_3$ each contains multiple alternatives. Moreover, Table 8.5(b) presents the decision correctness of each of the three probabilistic clusterings that results from aggregating the Pair F_1-score of its candidate clusterings. Here we distinct between several aggregation methods and the three selection predicates.*

By using the selection σ_{maxCl} for each three probabilistic clusterings only one candidate clustering results. Consequently, the minimum decision correctness, the maximum decision correctness, and the average decision correctness are the same for each of these clusterings. Since the Pair F_1-score of all three candidates is greater than $\theta_Q = 0.65$ the threshold-based aggregation results in 1.0 for each of the three probabilistic clusterings.

By using the selection σ_{minCl}, $\mathfrak{C}_3$ has two candidate clusterings. Nevertheless, since both have the same Pair F_1-score, the minimum decision correctness, the maximum decision correctness, and the average decision correctness are still equal. In this case, none of the candidate clusterings of $\mathfrak{C}_2$ are greater than θ_Q. Consequently, their threshold-based aggregation results in 0.0. The same holds for the probabilistic clustering $\mathfrak{C}_3$.

(a) The top-level and candidate clustering sets of $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$

$\mathfrak{C}_i$	$\widehat{\mathfrak{C}_i}$	$\sigma_{maxCl}(\widehat{\mathfrak{C}_i})$	$\sigma_{minCl}(\widehat{\mathfrak{C}_i})$	$\sigma_{1=1}(\widehat{\mathfrak{C}_i})$
$\mathfrak{C}_2$	$\{\mathcal{C}_{21}, \mathcal{C}_{22}\}$	$\{\mathcal{C}_{21}\}$	$\{\mathcal{C}_{22}\}$	$\{\mathcal{C}_{21}, \mathcal{C}_{22}\}$
$\mathfrak{C}_3$	$\{\mathcal{C}_{31}, \mathcal{C}_{32}, \mathcal{C}_{33}\}$	$\{\mathcal{C}_{31}\}$	$\{\mathcal{C}_{32}, \mathcal{C}_{33}\}$	$\{\mathcal{C}_{31}, \mathcal{C}_{32}, \mathcal{C}_{33}\}$
$\mathfrak{C}_4$	$\{\mathcal{C}_{41}\}$	$\{\mathcal{C}_{41}\}$	$\{\mathcal{C}_{41}\}$	$\{\mathcal{C}_{41}\}$

(b) Pair F_1-score of $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ for several selection predicates and aggregation methods

	σ_{maxCl}				σ_{minCl}				$\sigma_{1=1}$			
aggregation	$\mathcal{C}_1$	$\mathfrak{C}_2$	$\mathfrak{C}_3$	$\mathfrak{C}_4$	$\mathcal{C}_1$	$\mathfrak{C}_2$	$\mathfrak{C}_3$	$\mathfrak{C}_4$	$\mathcal{C}_1$	$\mathfrak{C}_2$	$\mathfrak{C}_3$	$\mathfrak{C}_4$
min	0.6	0.75	0.75	1.0	0.6	0.5	0.6	1.0	0.6	0.5	0.6	1.0
max	0.6	0.75	0.75	1.0	0.6	0.5	0.6	1.0	0.6	0.75	0.75	1.0
avg	0.6	0.75	0.75	1.0	0.6	0.5	0.6	1.0	0.6	0.625	0.65	1.0
$\theta_Q = 0.65$	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.5	0.333	1.0

Table 8.5.: Sample for quality evaluation by using the Single World Semantics

Without using any specific selection predicate (the selection predicate 1=1), all alternatives of the respective top-levels are selected as candidate clusterings. Now, the minimum decision correctness, the maximum decision correctness, and the average decision correctness of these candidates differ for clustering $\mathfrak{C}_2$ as well as differ for clustering $\mathfrak{C}_3$. Moreover, because for both probabilistic clusterings only one candidate is greater than θ_Q, the threshold-based aggregation results in 0.5 or 0.33 respectively.

The decision correctness of $\mathcal{C}_1$ is independent of the used selection predicate. Its minimum Pair F_1-score, its maximum Pair F_1-score, and its average Pair F_1-score and is always 0.6. In contrast, because 0.6 is lower than θ_Q, the result of the threshold-based aggregation is always zero.

In summary, it is obvious that the probabilistic clustering $\mathfrak{C}_4$ has the best decision correctness in the selected world semantics because its single top-level clustering is equivalent to the gold standard. Moreover, the decision correctness of $\mathfrak{C}_3$ is always equal to or greater than the decision correctness of the certain clustering $\mathcal{C}_1$. As a consequence, we can say that in the Single World Semantics $\mathfrak{C}_4$ is superior to all other sample clusterings and $\mathfrak{C}_3$ is superior to $\mathcal{C}_1$. The decision correctness of $\mathfrak{C}_2$ is better than the decision correctness of $\mathfrak{C}_3$ and $\mathcal{C}_1$ for some selection predicates and is worse for some other selection predicates. Therefore, we cannot say whether or not the former generally has a better decision correctness than the latter two, but this ordering depends on the used selection predicate and therefore depends on the considered database application.

8.8.2.4. Certain World Semantics

The Certain World Semantics is defined for evaluating the decision correctness of uncertain clusterings with respect to certain query answer based applications. Thus, we evaluate decision correctness only on the clustering information that is considered to be certain. The definition of clustering information depends on the used quality measure. For pair-based quality measures the clustering information is a set of pairs (MATCH or UNMATCH), but for cluster-based quality measures the clustering information

is always a clustering. As a consequence, we can distinguish between a pair-based interpretation and a cluster-based interpretation. In the cluster-based interpretation, certain clustering information can be considered as the clustering's facts[9] that belong to all its alternatives. Nevertheless, it is usually not possible to describe all certain clustering facts by a single clustering without implicating information that is not certain. Cluster-based measures for decision correctness, however, require a single clustering as input. As a consequence, we either have to ignore certain information or we have to consider information to be certain that actually is not certain. Whatever we decide to do, the reliability of the resultant quality scores suffers from this decision. For that reason, adapting the Certain World Semantics to cluster-based quality measures is very complicated at one hand and is very unsatisfactory on the other hand. This is why we restrict to the pair-based interpretation in this section. Nevertheless, an extensive consideration of the cluster-based interpretation is included in Appendix B.

We start with the pair-based interpretation by considering absolute certainty. Afterwards we discuss the consequences of relaxing the meaning of certainty and consider the situation where an uncertain gold standard is given. Since we consider the case of absolute certainty, the certain information of a probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ is equivalent to the certain information of its incomplete clustering Γ. For that reason, we restrict our consideration to incomplete clusterings in the following definitions.

In the pair-based interpretation, we consider the certain clustering information as the set of certain pairwise decisions where the set of entity pairs of an incomplete clustering Γ is defined as for a certain clustering, i.e. by every pairwise combination of entities from its range: $\mathit{Pairs}(\Gamma) = \{\{e_r, e_s\} \mid e_r, e_s \in rng(\Gamma), e_r \neq e_s\}$.

Definition 74 (Certain Duplicate Decisions): *Let Γ be an incomplete clustering. A certain duplicate decision of Γ is either a certain* MATCH *of Γ or a certain* UNMATCH *of Γ. The set of certain* MATCHES *(denoted as M_c) and the set of certain* UNMATCHES *(denoted as U_c) are defined as:*

$$M_c(\Gamma) = \bigcap\nolimits_{\mathcal{C}\in\Gamma} M(\mathcal{C}) = \{\{e_r, e_s\} \in \mathit{Pairs}(\Gamma) \mid \forall \mathcal{C} \in \Gamma\colon \{e_r, e_s\} \in M(\mathcal{C})\} \quad (8.19)$$

$$U_c(\Gamma) = \bigcap\nolimits_{\mathcal{C}\in\Gamma} U(\mathcal{C}) = \{\{e_r, e_s\} \in \mathit{Pairs}(\Gamma) \mid \forall \mathcal{C} \in \Gamma\colon \{e_r, e_s\} \in U(\mathcal{C})\} \quad (8.20)$$

$$= \{\{e_r, e_s\} \in \mathit{Pairs}(\Gamma) \mid \nexists \mathcal{C} \in \Gamma\colon \{e_r, e_s\} \in M(\mathcal{C})\} \quad (8.21)$$

Recall that throughout this thesis we usually have used the term 'hard MATCH' to refer to certain MATCHES and usually have used the term 'hard UNMATCH' to refer to certain UNMATCHES.

It is important to note that we have the equation $\mathit{Pairs}(\mathcal{C}) = M(\mathcal{C}) \cup U(\mathcal{C})$ for each certain clustering $\mathcal{C}$, but we usually have the inequation $\mathit{Pairs}(\Gamma) \neq M_c(\Gamma) \cup U_c(\Gamma)$ for an incomplete clustering Γ. As a consequence, we cannot imply that an entity pair is a certain UNMATCH if it is not a certain MATCH. Nevertheless, the most UNMATCHES are usually certain UNMATCHES and hence the number of certain UNMATCHES can be extremely large. For that reason we additionally consider the set of POSSIBLE MATCHES. The set of POSSIBLE MATCHES of an incomplete clustering Γ contains each entity pair whose decision is not certain, i.e.:

$$P(\Gamma) = \mathit{Pairs}(\Gamma) - \big(M_c(\Gamma) \cup U_c(\Gamma)\big) \quad (8.22)$$

$$= \{\{e_r, e_s\} \in \mathit{Pairs}(\Gamma) \mid \exists \mathcal{C}_p, \mathcal{C}_q \in \Gamma\colon \{e_r, e_s\} \in M(\mathcal{C}_p) \wedge \{e_r, e_s\} \in U(\mathcal{C}_q)\} \quad (8.23)$$

[9] As the facts of a clustering $\mathcal{C}$ we consider all clusterings that model some information of $\mathcal{C}$ and model no information that is not contained in $\mathcal{C}$, i.e. all clusterings that are dominated by $\mathcal{C}$. We consider clustering facts in more detail in Appendix B.1.

By using the set of POSSIBLE MATCHES, we have the equation $\mathit{Pairs}(\Gamma) = M_c(\Gamma) \cup U_c(\Gamma) \cup P(\Gamma)$ and hence can imply that an entity pair is a certain UNMATCH if it is neither a certain MATCH nor a POSSIBLE MATCH.

Using the sets of certain MATCHES, certain UNMATCHES, and POSSIBLE MATCHES, the pair-based measures of decision correctness that we have presented in Section 4.3.8 can be adapted to the Certain World Semantics as follows:

- **Certain Pair True Positives:** The set of certain true positives contains all certain MATCHES that are true duplicate pairs:

$$p\text{TP}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) \quad = \quad M_c(\Gamma) \cap M(\mathcal{C}_{\text{gold}}) \tag{8.24}$$

- **Certain Pair False Positives:** The set of Certain Pair False Positives contains all certain MATCHES that are not true duplicate pairs:

$$p\text{FP}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) \quad = \quad M_c(\Gamma) \cap U(\mathcal{C}_{\text{gold}}) = M_c(\Gamma) - M(\mathcal{C}_{\text{gold}}) \tag{8.25}$$

- **Certain Pair False Negatives:** The set of Certain Pair False Negatives contains all certain UNMATCHES that are true duplicate pairs:

$$p\text{FN}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) \quad = \quad U_c(\Gamma) \cap M(\mathcal{C}_{\text{gold}}) = M(\mathcal{C}_{\text{gold}}) - (M_c(\Gamma) \cup P(\Gamma)) \tag{8.26}$$

  Note that because of the inequation $\mathit{Pairs}(\Gamma) \neq M_c(\Gamma) \cup U_c(\Gamma)$ the last equivalence in Equation 8.26 differs from the equivalence that is given in the original definition of the set of Pair False Negatives in Equation 4.33.

- **Certain Pair False Decisions:** The set of Certain Pair False Decisions contains all Certain Pair False Positives and contains all Certain Pair False Negatives. As a consequence, it contains all certain MATCHES that are not true duplicate pairs and contains all certain UNMATCHES that are true duplicate pairs:

$$\begin{aligned} p\text{FDec}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) &= p\text{FP}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) \cup p\text{FN}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) && (8.27)\\ &= \big(M_c(\Gamma) - M(\mathcal{C}_{\text{gold}})\big) \cup \big(M(\mathcal{C}_{\text{gold}}) - (M_c(\Gamma) \cup P(\Gamma))\big) && (8.28) \end{aligned}$$

  An important consequence of the inequation $\mathit{Pairs}(\Gamma) \neq M_c(\Gamma) \cup U_c(\Gamma)$ are the inequations:

$$\begin{aligned} M(\Gamma) &\neq p\text{TP}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) \cup p\text{FP}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) && (8.29)\\ M(\mathcal{C}_{\text{gold}}) &\neq p\text{TP}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) \cup p\text{FN}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) && (8.30)\\ \text{and} \quad U(\mathcal{C}_{\text{gold}}) &\neq p\text{TN}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) \cup p\text{FP}^{\mathit{CWS}}(\Gamma, \mathcal{C}_{\text{gold}}) && (8.31) \end{aligned}$$

  For that reason, the number of Certain Pair False Decisions can capture quality aspects that cannot be captured by the Certain Pair Precision, the Certain Pair Recall, or the Certain Pair F_1-score.

(a) The certain MATCHES and the POSSIBLE MATCHES of the sample clusterings $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$

Clustering	**Set of Certain Matches: M_c**	**Set of Possible Matches: P**
$\mathcal{C}_1$	$\{\{e_1,e_5\},\{e_2,e_3\},\{e_2,e_6\},\{e_2,e_7\},\{e_4,e_8\}\}$	$\emptyset$
$\mathfrak{C}_2$	$\{\{e_1,e_5\},\{e_2,e_6\},\{e_4,e_8\}\}$	$\{\{e_2,e_7\},\{e_3,e_4\},\{e_3,e_8\},\{e_6,e_7\}\}$
$\mathfrak{C}_3$	$\{\{e_1,e_5\},\{e_2,e_6\},\{e_4,e_8\}\}$	$\{\{e_2,e_3\},\{e_2,e_7\},\{e_3,e_4\},\{e_3,e_6\},\{e_3,e_8\},\{e_6,e_7\}\}$
$\mathfrak{C}_4$	$\{\{e_1,e_5\}\}$	$\{\{e_1,e_2\},\{e_1,e_6\},\{e_2,e_3\},\{e_2,e_5\},\{e_2,e_6\},\{e_2,e_7\},\{e_3,e_4\},$ $\{e_3,e_6\},\{e_3,e_7\},\{e_4,e_8\},\{e_5,e_6\},\{e_6,e_7\},\{e_7,e_8\}\}$

(b) The pair-based decision correctness of the certain information of the sample clusterings $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$

Q-Measure	$\mathcal{C}_1$	$\mathfrak{C}_2$	$\mathfrak{C}_3$	$\mathfrak{C}_4$
$p\text{TP}^{CWS}$	$\{\{e_1,e_5\},\{e_2,e_6\},\{e_4,e_8\}\}$	$\{\{e_1,e_5\},\{e_2,e_6\},\{e_4,e_8\}\}$	$\{\{e_1,e_5\},\{e_2,e_6\},\{e_4,e_8\}\}$	$\{\{e_1,e_5\}\}$
$p\text{FP}^{CWS}$	$\{\{e_2,e_6\},\{e_2,e_7\}\}$	$\emptyset$	$\emptyset$	$\emptyset$
$p\text{FN}^{CWS}$	$\{\{e_2,e_3\},\{e_3,e_6\}\}$	$\{\{e_2,e_3\}\}$	$\emptyset$	$\emptyset$
$p\text{FDec}^{CWS}$	$\{\{e_2,e_3\},\{e_2,e_6\},\{e_2,e_7\},\{e_3,e_6\}\}$	$\{\{e_2,e_3\}\}$	$\emptyset$	$\emptyset$
$p\text{Rec}_2^{CWS}$	0.6	0.75	1.0	1.0
$p\text{Prec}^{CWS}$	0.6	1.0	1.0	1.0
pF_1^{CWS}	0.6	0.857	1.0	1.0

Table 8.6.: Pair-based evaluation of the decision correctness of the sample clusterings under the Certain World Semantics

- **Certain Pair Recall:** Due to the inequation $M(\mathcal{C}_{\text{gold}}) \neq p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) \cup p\text{FN}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})$, we can adapt the Pair Recall to the Certain World Semantics in two ways. The first adaptation is based on Equation 4.36 and results in:

$$p\text{Rec}_1^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) = \frac{|M_c(\Gamma) \cap M(\mathcal{C}_{\text{gold}})|}{|M(\mathcal{C}_{\text{gold}})|} \tag{8.32}$$

This adaptation considers the Certain Pair Recall as the amount of true duplicate pairs that are classified as certain MATCHES. In contrast, the second adaptation is based on Equation 4.37 and results in:

$$p\text{Rec}_2^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) = \frac{|p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})|}{|p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})| + |p\text{FN}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})|} \tag{8.33}$$

$$= \frac{|M_c(\Gamma) \cap M(\mathcal{C}_{\text{gold}})|}{|M(\mathcal{C}_{\text{gold}}) \cap \big(M_c(\Gamma) \cup U_c(\Gamma)\big)|} = \frac{|M_c(\Gamma) \cap M(\mathcal{C}_{\text{gold}})|}{|M(\mathcal{C}_{\text{gold}}) - P(\Gamma)|} \tag{8.34}$$

This adaptation considers Certain Pair Recall as the amount of certainly classified true duplicate pairs that are classified as a certain MATCH. Whereas, the second adaptation only rates the Pair Recall of the certain decisions, the first adaptation also takes all the true duplicates into account that have not been certainly classified at all. Therefore, the latter is actually not only a measure for decision correctness, but is more a mixture between a measure for decision correctness and a measure for decision certainty (recall this circumstance in the discussion on measuring decision

certainty in Section 8.8.3). For that reason, we restrict our following considerations to the second adaptation.

- **Certain Pair Precision:** The Certain Pair Precision is the amount of certain MATCHES that are true duplicate pairs:

$$p\text{Prec}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) = \frac{|M_c(\Gamma) \cap M(\mathcal{C}_{\text{gold}})|}{|M_c(\Gamma)|} = \frac{|p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})|}{|p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})| + |p\text{FP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})|} \quad (8.35)$$

- **Certain Pair F_1-score:** The Certain Pair F_1-score is the harmonic mean of the Certain Pair Recall and the Certain Pair Precision. Due to Certain Pair Recall can be adapted in two ways, we theoretically can adapt the Certain Pair F_1-score in two ways, too. Nevertheless, since we restrict our consideration to the second adaptation of the Certain Pair Recall, we only introduce the following adaptation of the Certain Pair F_1-score:

$$pF_1^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) = \frac{2 \times p\text{Rec}_2^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) \times p\text{Prec}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})}{p\text{Rec}_2^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) + p\text{Prec}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})} \quad (8.36)$$

$$= \frac{2 \times |p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})|}{2 \times |p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})| + |p\text{FP}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})| + |p\text{FN}^{CWS}(\Gamma, \mathcal{C}_{\text{gold}})|} \quad (8.37)$$

Example 205 *For illustration, we reconsider the four clusterings $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ from the motivating example. The clusterings' sets of certain* MATCHES *and* POSSIBLE MATCHES *are listed in Table 8.6(a). Obviously the certain clustering $\mathcal{C}_1$ has no* POSSIBLE MATCH *but has five certain* MATCHES. *In contrast, the clusterings $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ have four, six, and thirteen* POSSIBLE MATCHES *respectively. The clusterings $\mathfrak{C}_2$ and $\mathfrak{C}_3$ each have three certain* MATCHES *and the clustering $\mathfrak{C}_4$ has one certain* MATCH.

The pair-based quality of these four clusterings under the Certain World Semantics is then presented in Table 8.6(b). for each of the quality measures presented above. As in the previous quality semantics, the certain clustering $\mathcal{C}_1$ has three Pair True Positives, two Pair False Positives and two Pair False Negatives. The probabilistic clustering $\mathfrak{C}_2$ has three Certain Pair True Positives, but also has one Certain Pair False Negatives. Consequently, its Certain Pair Precision is one, but its Certain Pair Recall is only 0.75 and hence its Certain Pair F_1-score is only 0.857. In contrast, neither $\mathfrak{C}_3$ nor $\mathfrak{C}_4$ have any Certain Pair False Decision and hence each have a perfect Certain pair Recall, a perfect Certain Pair Precision, and a perfect Certain Pair F_1-score.

Considering only the certain information all three probabilistic clusterings are superior to the certain clustering $\mathcal{C}_1$. Moreover, the clusterings $\mathfrak{C}_3$ and $\mathfrak{C}_4$ are superior to the clustering $\mathfrak{C}_2$. This shows that in this example the computational benefits of using the hierarchical W-graph *computation mapping $f_{MG \mapsto WG}^{HC}$ (clustering $\mathfrak{C}_2$) instead of using the naive* W-graph *computation mapping $f_{MG \mapsto WG}^{Naive}$ (clustering $\mathfrak{C}_3$) comes to the price of decreasing the correctness of the certain decisions because it introduces a Certain Pair False Negative.*

Relaxing the Meaning of Certainty Sometimes it can be useful to relax the requirement of absolute certainty by introducing a tolerance value τ and to consider a duplicate decision to be certain if its certainty is within the given tolerance. In the case of a probabilistic clustering, an entity pair can be considered as a certain MATCH if the accumulative probability of all clusterings in which this pair belong

to the set of MATCHES is above the threshold $1 - \tau$. In the case of an incomplete clustering, we can use the percentage of possible clusterings instead of the probability[10].

In the pair-based interpretation such an extension does not pose a problem and we only need to adapt the definitions of the set of certain MATCHES, the set of certain UNMATCHES, and the set of POSSIBLE MATCHES by introducing the threshold $1 - \tau$ into the formulas. Thus, let $\mathfrak{C}$ be a probabilistic clustering, the set of certain MATCHES, the set of certain UNMATCHES, and the set of POSSIBLE MATCHES of $\mathfrak{C}$ by using the tolerance τ are defined as:

$$\begin{aligned} M_c(\mathfrak{C}) &= \{\{e_r, e_s\} \in \textit{Pairs}(\mathfrak{C}) \mid \sum\nolimits_{\mathcal{C}\in\Gamma, \{e_r,e_s\}\in M(\mathcal{C})} Pr(\mathcal{C}) \geq 1 - \tau\} \\ U_c(\mathfrak{C}) &= \{\{e_r, e_s\} \in \textit{Pairs}(\mathfrak{C}) \mid \sum\nolimits_{\mathcal{C}\in\Gamma, \{e_r,e_s\}\in U(\mathcal{C})} Pr(\mathcal{C}) \geq 1 - \tau\} \\ &= \{\{e_r, e_s\} \in \textit{Pairs}(\mathfrak{C}) \mid \sum\nolimits_{\mathcal{C}\in\Gamma, \{e_r,e_s\}\in M(\mathcal{C})} Pr(\mathcal{C}) < \tau\} \\ P(\mathfrak{C}) &= \{\{e_r, e_s\} \in \textit{Pairs}(\mathfrak{C}) \mid \sum\nolimits_{\mathcal{C}\in\Gamma, \{e_r,e_s\}\in M(\mathcal{C})} \tau \geq Pr(\mathcal{C}) < 1 - \tau\} \end{aligned}$$

The single requirement is that the threshold must be greater than 0.5 because otherwise the certain decisions can be contradictory, e.g. $\{e_1, e_2\}$ and $\{e_1, e_3\}$ are certain MATCHES but $\{e_2, e_3\}$ is a certain UNMATCH, and/or decisions can be considered to be certain and uncertain at the same time.

Uncertain Gold Standards In this section we compare an indeterministic detection result with a certain gold standard. Nevertheless, similar approaches can be used to compare a deterministic detection result with an uncertain gold standard. The truth is often not exactly known. Thus, it can be useful to select a deterministic detection result that is most likely similar to the truth. In that context, an entity pair is considered to be a certain true positive if it is classified as a MATCH by the detection process and if it is certainly classified as a MATCH by the uncertain gold standard.

A logical extension is then to compare an indeterministic detection result with an uncertain gold standard and hence to compare two uncertain clusterings directly. In this case, an entity pair is considered to be a certain true positive if it is certainly classified as a MATCH by both uncertain clusterings.

8.8.2.5. Motivating Example

As we have seen in the previous sections the meaning of quality essentially differs from one quality semantics to another. For illustration, we consider the four sample clusterings from Figure 8.27. The Pair F_1-scores of these clusterings under the different quality semantics are summarized in Table 8.7 (in the Single World Semantics we use the average Pair F_1-score of the most probable clustering alternatives). With respect to these scores, the clustering $\mathfrak{C}_3$ is superior to the clustering $\mathfrak{C}_2$, but is not superior to the clusterings $\mathcal{C}_1$ and $\mathfrak{C}_4$. Moreover, neither $\mathcal{C}_1$ nor $\mathfrak{C}_4$ is superior to $\mathfrak{C}_2$ and $\mathcal{C}_1$ is not superior to $\mathfrak{C}_4$ nor vice versa. The superiority of $\mathfrak{C}_3$ to $\mathfrak{C}_2$ illustrates that decreasing computation complexity and increasing decision certainty by using an HC-restriction often comes to the price of a decreasing decision correctness.

Under the Single World Semantics the clustering $\mathfrak{C}_4$ has by far the best decision correctness. In contrast, if we only consider the Pair F_1-score of the certain duplicate decisions (Certain World Semantics),

[10]Note, this approach corresponds to transforming an incomplete clustering into a probabilistic clustering by assuming a uniform probability distribution on all its alternatives.

Clustering	SWS	CWS	AWS-min	AWS-max	AWS-exp
$\mathcal{C}_1$	0.6	0.6	0.6	0.6	0.6
$\mathfrak{C}_2$	0.625	0.857	0.5	0.75	0.62
$\mathfrak{C}_3$	0.65	1.0	0.5	1.0	0.669
$\mathfrak{C}_4$	1.0	1.0	0.25	1.0	0.615

Table 8.7.: Pair F_1-scores of the four sample clusterings $\mathcal{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$

besides $\mathfrak{C}_4$ the clustering $\mathfrak{C}_3$ has a perfect decision correctness as well. The expected Pair F_1-scores is the best for clustering $\mathfrak{C}_3$. However, if we consider an application that requires that the Pair F_1-score is at least 0.6, the certain clustering $\mathcal{C}_1$ is most suitable. Altogether, each of these clusterings impresses by another quality aspect and a universal order between the decision correctness of these clusterings cannot be made. Finally, take into mind that we only consider decision correctness in this discussion and as we will see in the subsequent sections the two clusterings $\mathcal{C}_1$ and $\mathfrak{C}_2$ have a lower correctness than the clusterings $\mathfrak{C}_3$ and $\mathfrak{C}_4$ with respect to the most quality semantics but are also more certain.

This example illustrates that the meaning of data quality essentially depends on the way the database is intended to be used and therefore emphasizes the need for introducing the different quality semantics. Of course, if the deduplicated database is processed by applications from several classes, all these semantics need to be integrated into tuning the configuration of the considered duplicate detection process.

8.8.3. Decision Certainty

It contrast to its decision correctness, the decision certainty of an uncertain clustering is independent from the considered gold standard. The Single World Semantics has been designed for conventional database applications that each always processes a single possible world. Consequently, with respect to these applications the indeterministic duplicate detection result is always absolutely certain and hence we define decision certainty always to be 1. In the Certain World Semantics, we simply distinguish between certain clustering information and uncertain clustering information. Thus, we can rate decision certainty by the amount of all clustering information that is certain and hence can consider decision certainty as a kind of completeness. In contrast, in the Possible Worlds Semantics and in the Aggregated World Semantics we differentiate between the individual degrees of uncertainty. For that reason, we need measures for decision certainty that takes the exact degrees of uncertainties into account. Here we make use of the certainty measures that we have already presented in Section 6.10.

8.8.3.1. Certain World Semantics

In the Certain World Semantics we divide the set of clustering information (pairwise decisions or clustering facts) into two disjoint subsets, i.e. the set of certain clustering information and the set of uncertain clustering information, and then define the decision correctness of an uncertain clustering as the decision correctness of its certain clustering information. Thus, the exact degree of clustering uncertainty does not influence the application result, i.e. it does not make a difference if the information is absolutely uncertain or if it is only a little bit uncertain as long as it is finally treated as an uncertain information. In the

presence of tolerance, the demarcation between certain clustering information and uncertain clustering information shifts a little bit away from the situation of absolute certainty, but the binary classification into certain information and uncertain information remains. Of course, under this semantics a clustering is absolutely certain if all its clustering information is considered to be certain. Thus, by using a tolerance value a clustering can become absolutely certain under this semantics even if it has several alternatives. Because of the binary classification, we do not need to distinguish between an incomplete clustering and a probabilistic clustering and hence restrict our consideration to the first.

- **Decision Completeness:** The most intuitive approach to rate decision certainty is to compute the amount of clustering information that is certain. In the pair-based interpretation, clustering information is described by pairs. As a consequence, decision certainty can be rated by the amount of certain pairwise decisions that each is a certain MATCH or a certain UNMATCH and hence as can be measured by the *Decision Completeness* that is defined as:

$$\mathit{DecComp}(\Gamma) = \frac{|\mathit{Pairs}(\Gamma)| - |P(\Gamma)|}{|\mathit{Pairs}(\Gamma)|} = \frac{|M_C(\Gamma)| + |U_C(\Gamma)|}{|M_C(\Gamma)| + |U_C(\Gamma)| + |P(\Gamma)|} \tag{8.38}$$

  A shortcoming of this approach is that the number of UNMATCHES dominates the number of MATCHES by far and the majority of UNMATCHES are certain. Consequently, the amount of decisions that are certain will be close to 1 for (almost) all realistic indeterministic duplicate detection results, even if their numbers of indeterministically handled decisions are different to a large extent.

  One possible solution to this problem is to restrict decision completeness to the amount of all MATCHES that are certain. Nevertheless, because the certain UNMATCHES can be of interest as well we generalize the Decision Completeness by introducing the three weights $\alpha, \beta, \gamma \in \mathbb{R}$:

$$\mathit{DecComp}_{\alpha,\beta,\gamma}(\Gamma) = \frac{\alpha|M_C(\Gamma)| + \beta|U_C(\Gamma)|}{\alpha|M_C(\Gamma)| + \beta|U_C(\Gamma)| + \gamma|P(\Gamma)|} \tag{8.39}$$

  This measure can be specialized to the above given definition of *DecComp* by setting $\alpha = 1$, $\beta = 1$ and $\gamma = 1$, and can be specialized to the ratio between all certain MATCHES and all pairs that are not certain UNMATCHES by setting $\alpha = 1$, $\beta = 0$ and $\gamma = 1$.

  POSSIBLE MATCHES are potential MATCHES and are potential UNMATCHES at the same time. Therefore, we think it is most meaningful to set $\gamma = \mathbf{max}(\alpha, \beta)$ because they should have as much influence on the resultant quality score as the most important certain decision.

- **Possible Match-Ratio:** Another intuitive measure is the *Possible Match-Ratio* that rates decision certainty by the inverse number of POSSIBLE MATCHES. The idea behind this measure is that the less POSSIBLE MATCHES an uncertain clustering has, the greater is its the certainty.

$$\mathit{P\text{-}Ratio}(\Gamma) = \frac{1}{1 + |P(\Gamma)|} \tag{8.40}$$

  This measure, however, does not scale cardinally which is an important criterion for the interpretability of quality measures [HKK07]. Moreover, we do not make any relation between the

Clustering	$DecComp_{\mathbf{1,1}}$	$DecComp_{\mathbf{2,1}}$	$DecComp_{\mathbf{10,1}}$	*P-Ratio*
$\mathcal{C}_1$	1.0	1.0	1.0	1.0
$\mathfrak{C}_2$	0.857	0.771	0.560	0.2
$\mathfrak{C}_3$	0.786	0.676	0.450	0.143
$\mathfrak{C}_4$	0.536	0.381	0.156	0.071

Table 8.8.: The decision certainty of the four sample clusterings measured under the Certain World Semantics

number of POSSIBLE MATCHES and the number of certain decisions, but it makes a difference if we have 10 POSSIBLE MATCHES in a clustering with 10 elements or if we have 10 POSSIBLE MATCHES in a clustering with 1000 elements. Moreover, it is a special version of the Decision Completeness that results by setting $\alpha = 1/(2 \times |M_C(\Gamma)|)$, $\beta = 1/(2 \times |U_C(\Gamma)|)$, and $\gamma = 1$.

Example 206 *For illustration we consider the four clusterings from the motivating example. The certainty scores of these clusterings that result from rating them with the above presented measures are presented in Table 8.8. As we can see, all measures agree in the ordering of their scores, i.e. $\mathcal{C}_1$ is always absolute certain, $\mathfrak{C}_2$ is always more certain than $\mathfrak{C}_3$ and $\mathfrak{C}_4$, and $\mathfrak{C}_3$ is always more certain than $\mathfrak{C}_4$*

In this sample scenario we only consider clusterings with eight entities. Thus, using decision completeness with $\alpha = 1$ and $\beta = 1$ already produces significant results. Nevertheless, as we can see, with an increasing difference between α and β the scores become smaller and the differences between the scores become larger. Bear in mind that in this sample scenario we have five MATCHES *versus twenty-three* UNMATCHES *which is around twenty percent, but in real use cases this size is often only* 0.02 *percent or smaller. Moreover ninety-nine percent of* UNMATCHES *are usually certain ones. Therefore, if we consider the same number of certain* MATCHES *and* POSSIBLE MATCHES*, but assume two thousand certain* UNMATCHES *instead of twenty-three, the setting $\alpha = \beta = 1$ would produce in the scores $DecComp_{1,1}(\mathfrak{C}_2) = 0.998$, $DecComp_{1,1}(\mathfrak{C}_3) = 0.997$, and $DecComp_{1,1}(\mathfrak{C}_4) = 0.994$. In contrast, the setting $\alpha = 10$ and $\beta = 1$ would produce in the scores $DecComp_{10,1}(\mathfrak{C}_2) = 0.98$, $DecComp_{10,1}(\mathfrak{C}_3) = 0.971$, and $DecComp_{10,1}(\mathfrak{C}_4) = 0.939$. The latter is much more discriminating than the first and hence is more suitable for our purpose. In general, selecting the best setting of α, β, and γ depends on the considered use case. Nevertheless, since different settings only change the discrepancies between the scores but do not change their order, the actually used setting is often less crucial for relatively comparing a set of uncertain clusterings.*

8.8.3.2. Possible Worlds Semantics / Aggregated World Semantics

In contrast to the Certain World Semantics, in the Possible Worlds Semantics and the Aggregated World Semantics we have to take the exact degree of uncertainty of every clustering information into account because it can have an influence on the final application result. In this case, we have to differentiate between incomplete clusterings and probabilistic clusterings. The meaning of the degree of uncertainty is not clearly defined for an incomplete clustering and it is up to the user to define it by her own need. Nevertheless, the most meaningful definition for the certainty of a specific part of clustering information

is the amount of alternatives that contains this information. This definition, however, corresponds to transforming an incomplete clustering into a probabilistic clustering by assuming a uniform distribution on all alternatives. For that reason, we primarily restrict the following consideration to probabilistic clusterings.

Since uncertain clusterings are uncertain values, we can make use of the certainty measures that we have presented in Section 6.10. However, not all of these measures are equally suitable for our purpose. For that reason, in the remainder of this section we adapt them to our needs and discuss their advantages as well as disadvantages in detail.

- **Number of Alternatives:** The number of alternatives can be used for an uncertain clustering without any adaptation. Thus, let $\mathfrak{C} = (\Gamma, Pr)$ be a cluster-disjoint probabilistic clustering on $n = |rng(\mathfrak{C})|$ entities, the Ratio of Excluded Alternatives and the Inverse Number of Alternatives of $\mathfrak{C}$ are defined as:

$$\mathit{RExA}(\mathfrak{C}) = \frac{B_n - |\Gamma|}{B_n - 1} \qquad (8.41) \qquad\qquad \mathit{invNA}(\mathfrak{C}) = \frac{1}{|\Gamma|} \qquad (8.42)$$

  where B_n is the n-th Bell number [Rot64]. The size of B_n is the primary shortcoming of the Ratio of Excluded Alternatives. For example, the total number of cluster-disjoint clusterings with eight entities is $B_8 = 4140$ and the number of possible cluster-disjoint clusterings with twenty five entities is already $B_{25} = 4.63 \times 10^{18}$. Duplicate detection, however, is usually performed on a database with thousand of entities and hence the Bell number becomes incredible large which makes computation expensive and let the certainty scores become nearly identical even in cases where the certainty of the corresponding clusterings clearly distinct. A potential solution to this problem is to use replace B_n with a number that is only marginally larger than the maximal number of alternatives of all considered uncertain clusterings. Nevertheless, as described in Section 8.7 even this number can become extremely large.

  A disadvantage of both measures is that the number of clustering alternatives grows exponentially with the number of mutually independent uncertain decisions. For example introducing a single uncertain pairwise decision that is independent to all other uncertain decisions doubles the number of the clustering's alternatives. However, saying that a clustering with thousand mutually independent pairwise uncertain decisions is twice as certain as a clustering with thousand and one mutually independent pairwise uncertain decisions seems very counterintuitive.

- **Shannon Entropy:** Since probabilistic clusterings are probabilistic values, the Shannon Entropy can be used for probabilistic clusterings without any adaptation. Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering, the entropy of $\mathfrak{C}$ is defined as:

$$H(\mathfrak{C}) = -\sum\nolimits_{\mathcal{C} \in \Gamma} Pr(\mathcal{C}) \times \log_2(Pr(\mathcal{C}))$$

  The same holds for the Inverse Normalized Entropy. Therefore, let $\mathfrak{C} = (\Gamma, Pr)$ be a cluster-disjoint probabilistic clustering on $n = |rng(\mathfrak{C})|$ entities, the Inverse Normalized Entropy

of $\mathfrak{C}$ is defined as:

$$\mathit{invH}_{\mathit{norm}}(\mathfrak{C}) = 1 - H_{\mathit{norm}}(\mathfrak{C}) = 1 + \sum\nolimits_{\mathcal{C}\in\Gamma} Pr(\mathcal{C}) \times \log_{B_n}(Pr(\mathcal{C}))$$

where B_n is again the n-th Bell number.

Due to the size of the n-th bell number is incredibly large for commonly processed number of entities, this measure has a similar shortcoming as the Ratio of Excluded Alternatives. Nevertheless, in computing entropy this number is only used as the basis of the logarithm and hence has a lower influence on the size of the final scores. For instance, $\frac{\log_{10^{50}}(0.6)}{\log_{10^5}(0.6)} = \frac{-0.0044}{-0.044} = 0.1$ whereas $\frac{10^{50}}{10^5} = 10^{45}$.

Using the unnormalized entropy in lieu of the Inverse Normalized Entropy resolves the problem of using the bell number, but involves the disadvantages that (i) the meaning of the resultant scores is hard to interpret and hence these scores are most often only suitable to compare the relative quality of multiple uncertain clusterings instead of computing an absolute score of quality, and (ii) we cannot compute the inverse score anymore and hence cannot directly define a measure of certainty. The latter is especially a problem if we are interested to finally combine the computed scores of decision correctness and decision certainty to a single quality score as for example by computing their harmonic mean (see Section 8.8.4).

- **(Un)Certain Density / Answer Decisiveness:** The Certain Density and the Answer Decisiveness can be adapted to uncertain clusterings in two ways. First we can consider each uncertain clustering as a single choice point. By doing so, the Certain Density is equivalent to the Inverse Number of Alternatives and hence suffers from the same problems (similar problems result for the Answer Decisiveness). Second, we can consider the clustering's factors as choice points because all factors are mutually independent and each of them has a set of alternatives. Consequently, we have to distinct between the certainty of the non-factorized representation of an uncertain clustering and the certainty of its factorized representation. Moreover, an uncertain clustering can be factorized in different ways. As a consequence, the computed certainty of an uncertain clustering always depends on the clustering's factorization the certainty is computed from. Thus, let Γ be an incomplete clustering with the factorization $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$, we define the Certain Density of Γ that is computed based on $\mathcal{F}$ as:

$$\mathit{CDens}_{\mathcal{F}}(\Gamma) = \mathit{CDens}(\mathcal{F}) = \frac{1}{k}\sum\nolimits_{i=1}^{k} \frac{1}{|\Gamma_{F_i}|} \tag{8.43}$$

Moreover, let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering with the factorization $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ where $\forall i \in \{1, \ldots, k\}\colon \mathfrak{C}_{F_i} = (\Gamma_{F_i}, Pr_{F_i})$, we define the Answer Decisiveness of Γ that is computed based on $\mathcal{F}$ as:

$$\mathit{ADec}_{\mathcal{F}}(\Gamma) = \mathit{ADec}(\mathcal{F}) = \frac{1}{k}\sum\nolimits_{i=1}^{k} \frac{\mathbf{max}_{\mathcal{C}\in\Gamma_{F_i}} Pr_{F_i}(\mathcal{C})}{(2 - \mathbf{max}_{\mathcal{C}\in\Gamma_{F_i}} Pr_{F_i}(\mathcal{C})) \times \log_2\left(\mathbf{max}(2, |\Gamma_{F_i}|)\right)} \tag{8.44}$$

Note that an uncertain clustering can be considered as a factorization of itself and hence the certainty score that is computed based on its non-factorized representation is included in these definitions, e.g. $\mathit{CDens}(\{\Gamma\}) = \mathit{CDens}_{\{\Gamma\}}(\Gamma)$ is the Certain Density of Γ computed on its non-factorized representation.

possible clusterings in $\boldsymbol{\Gamma_{F_1}}$	$\boldsymbol{Pr_{F_1}}$
$\mathcal{C}_{11} = \{\langle e_1, e_5\rangle\}$	1.0

(a) Factor $\mathfrak{C}_{F_1}$ of the probabilistic clustering $\mathfrak{C}_3$

possible clusterings in $\boldsymbol{\Gamma_{F_2}}$	$\boldsymbol{Pr_{F_2}}$
$\mathcal{C}_{21} = \{\langle e_2, e_6\rangle, \langle e_3\rangle, \langle e_7\rangle, \langle e_4, e_8\rangle\}$	0.22
$\mathcal{C}_{22} = \{\langle e_2, e_6, e_7\rangle, \langle e_3\rangle, \langle e_4, e_8\rangle\}$	0.22
$\mathcal{C}_{23} = \{\langle e_2, e_6\rangle, \langle e_3, e_4, e_8\rangle, \langle e_7\rangle\}$	0.22
$\mathcal{C}_{24} = \{\langle e_2, e_6, e_7\rangle, \langle e_3, e_4, e_8\rangle\}$	0.2
$\mathcal{C}_{25} = \{\langle e_2, e_3, e_6\rangle, \langle e_4, e_8\rangle, \langle e_7\rangle\}$	0.14

(b) Factor $\mathfrak{C}_{F_2}$ of the probabilistic clustering $\mathfrak{C}_3$

Figure 8.29.: The complete factorization of sample clustering $\mathfrak{C}_3$

possible clusterings in $\boldsymbol{\Gamma'}$	$\boldsymbol{Pr'}$
$\mathcal{C}'_1 = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3\rangle, \langle e_4\rangle\}$	0.25
$\mathcal{C}'_2 = \{\langle e_1, e_2\rangle, \langle e_3\rangle, \langle e_4\rangle\}$	0.25
$\mathcal{C}'_3 = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3, e_4\rangle\}$	0.25
$\mathcal{C}'_4 = \{\langle e_1, e_2\rangle, \langle e_3, e_4\rangle\}$	0.25

(a) Probabilistic clustering $\mathfrak{C}' = (\Gamma', Pr')$

possible clusterings in $\boldsymbol{\Gamma^*}$	$\boldsymbol{Pr^*}$
$\mathcal{C}^*_1 = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3\rangle, \langle e_4\rangle\}$	$1/3$
$\mathcal{C}^*_2 = \{\langle e_1, e_2\rangle, \langle e_3\rangle, \langle e_4\rangle\}$	$1/3$
$\mathcal{C}^*_3 = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3, e_4\rangle\}$	$1/3$

(b) Probabilistic clustering $\mathfrak{C}^* = (\Gamma^*, Pr^*)$

possible clusterings in $\boldsymbol{\Gamma'_{F_1}}$	$\boldsymbol{Pr'_1}$
$\mathcal{C}'_{11} = \{\langle e_1\rangle, \langle e_2\rangle\}$	0.5
$\mathcal{C}'_{12} = \{\langle e_1, e_2\rangle\}$	0.5

possible clusterings in $\boldsymbol{\Gamma'_{F_2}}$	$\boldsymbol{Pr'_2}$
$\mathcal{C}'_{21} = \{\langle e_3\rangle, \langle e_4\rangle\}$	0.5
$\mathcal{C}'_{22} = \{\langle e_3, e_4\rangle\}$	0.5

(c) Factorization $\mathcal{F} = \{\mathfrak{C}'_{F_1}, \mathfrak{C}'_{F_2}\}$ of $\mathfrak{C}'$ with $\mathfrak{C}'_{F_1} = (\Gamma'_{F_1}, Pr'_1)$ and $\mathfrak{C}'_{F_2} = (\Gamma'_{F_2}, Pr'_2)$

Figure 8.30.: The two sample clusterings $\mathfrak{C}'$ and $\mathfrak{C}^*$ as well as the complete factorization of $\mathfrak{C}'$

In conclusion, the certainty of an uncertain clustering depends on the clustering's representation. This dependency, however, is not a desired property because in this case the problem of correlated choice points that we have mentioned in Section 6.10.3 comes into effect.

Example 207 *We will illustrate this circumstance by two simple examples. The first example should demonstrate the enormous amount a certainty score of an uncertain clustering can change if it is computed based on different factorizations. The second example should demonstrate that the relative order of certainty of several uncertain clusterings can change if some of these clusterings are finer factorized than others. Here it is important to note that not all uncertain clusterings can be factorized in the same way and usually clusterings that are more uncertain can be finer factorized than clusterings that are less uncertain because the former have less correlations between the uncertain duplicate decisions than the latter.*

For the first example, we consider the clustering $\mathfrak{C}_3$ from the motivating example. Due to the duplicate decision on the entities e_1 and e_5 is independent from the other duplicate decisions, we can factorized the clustering into the two factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$ that are depicted in Figure 8.29. The first factor is certain and the second factor has exact the same probability distribution as

the non-factorized representation because compared to $\mathfrak{C}_3$ *in each alternative of* $\mathfrak{C}_{F_2}$ *only the entities* e_1 *and* e_5 *are missing. The Answer Decisiveness of the non-factorized representation of* $\mathfrak{C}_3$ *is* $\frac{0.22}{1.78\times\log_2(5)} = 0.053$*, but the Answer Decisiveness of the factorized representation of* $\mathfrak{C}_3$ *is* $\frac{1}{2}\big(1 + \frac{0.22}{1.78\times\log_2(5)}\big) = 0.527$ *which is ten times larger than the certainty of the non-factorized representation.*

For the second example, we consider the two simple probabilistic clusterings $\mathfrak{C}'$ *and* $\mathfrak{C}^*$ *that are presented in Figure 8.30(a) and Figure 8.30(b). Both clusterings contain four entities. Since the alternatives of both clusterings are uniformly distributed and because in* $\mathfrak{C}^*$ *one of the alternatives of* $\mathfrak{C}'$ *is excluded for sure, it is obvious that* $\mathfrak{C}^*$ *is more certain than* $\mathfrak{C}'$*. The certainty scores of the non-factorized representation confirm this intuition because the Answer Decisiveness of* $\mathfrak{C}'$ *is* $\frac{0.25}{1.75\times\log_2(4)} = 0.0714$ *and the Answer Decisiveness of* $\mathfrak{C}^*$ *is* $\frac{1/3}{5/3\times\log_2(3)} = 0.1247$*. Nevertheless, because the duplicate decision between* e_1 *and* e_2 *as well as the duplicate decision between* e_3 *and* e_4 *are independent in* $\mathfrak{C}'$ *but are not independent in* $\mathfrak{C}^*$*, the first can be factorized and the latter not. The factorization of* $\mathfrak{C}'$ *is presented in Figure 8.30(c). The Answer Decisiveness of the factorized representation of* $\mathfrak{C}'$ *is* $\frac{1}{2}\big(2 \times \frac{0.5}{1.5\times\log_2(2)}\big) = 0.333$ *which is much greater than the Answer Decisiveness of* $\mathfrak{C}^*$ *that we have computed for the non-factorized representation. As a consequence, if we would always compute the Answer Decisiveness of an uncertain clustering based on its finest factorization,* $\mathfrak{C}'$ *would result to be more certain than* $\mathfrak{C}^*$ *which is clearly not the case.*

Intuitively, the certainty score of an uncertain clustering is the more meaningful the finer the clustering's representation, from which this score is computed, is factorized, because otherwise we end up in the disadvantage that the uncertainty grows exponentially with the number of mutually independent uncertain decisions (see description of Number of Alternatives). However, comparing certainty scores of uncertain clusterings is only meaningful if the compared scores are computed on comparable representations of these clusterings. Two representations in turn are only comparable if they are factorized by the same factor sets and hence are based on the same partition of the considered set of database entities. Since this comparability is often not given, we run into a problem if we want to use the Answer Decisiveness for computing meaningful certainty scores.

Nonetheless, even if we want to compare uncertain clusterings that are given in non-comparable representations, we can ensure comparability by a simple trick because we only need to make the clusterings' representation comparable. For that purpose we transform some of the given representations until comparability is given. To make the resultant certainty scores as maximal meaningful as possible, the performed transformation should be minimal. Such a minimal transformation can be realized by computing the finest entity set partition that forms a factorization of each of the considered clusterings. Thus, let $\{\mathcal{F}_1^{set}, \ldots, \mathcal{F}_k^{set}\}$ be the k factor sets of the given representations and let each of them be a partition of the entity set $\mathfrak{E}$, we compute the finest partition $\mathcal{F}_\otimes^{set}$ of $\mathfrak{E}$ that is coarser than each of the k input partitions, i.e., every partition class in $\mathcal{F}_i^{set}$ is a subset of some partition class in $\mathcal{F}_\otimes^{set}$ for every $i \in \{1, \ldots, k\}$ and there is no other partition that is finer than $\mathcal{F}_\otimes^{set}$ but coarser than all of the k input partitions. The partition $\mathcal{F}_\otimes^{set}$ can be computed by merging all factor sets in $\{\mathcal{F}_1^{set}, \ldots, \mathcal{F}_k^{set}\}$ that share an entity.

Example 208 *For illustration, let* $\mathcal{F}_1^{set} = \{\{e_1, e_2\}, \{e_3\}, \{e_4, e_5, e_6\}\}$, $\mathcal{F}_2^{set} = \{\{e_1\}, \{e_2, e_3\}, \{e_4\}, \{e_5, e_6\}\}$, *and* $\mathcal{F}_3^{set} = \{\{e_1\}, \{e_2\}, \{e_3\}, \{e_4, e_5\}, \{e_6\}\}$ *be the three partitions of the entity set* $\{e_1, e_2, e_3, e_4, e_5\}$. *The finest partition that is coarser than all these partitions is* $\mathcal{F}_{\otimes}^{set} = \{\{e_1, e_2, e_3\}, \{e_4, e_5, e_6\}\}$.

The special clue of this trick is that we even do not need to reconstruct the newly computed factorizations, but can compute the Answer Decisiveness that result from the reconstructed factorizations based on the originally given factorizations (see Section 8.8.5.7).

- **Precision:** Recall from Section6.10.4 that the measures presented above are based on the certainty of point queries, but the Precision is based on the certainty of range queries. In the context of clusterings, a point query corresponds to a predicate that checks whether or not the true clustering alternative is equal to a particular clustering and a range query corresponds to a predicate that checks whether or not the true clustering alternative is similar to a particular clustering. Therefore, we can simply adapt Expected Pair Precision to cluster-disjoint uncertain clusterings by using a similarity measure that is defined for such clusterings. Thus, let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering and let sim be a similarity measure for cluster-disjoint clusterings, the Expected Pair Precision of $\mathfrak{C}$ is defined as:

$$\mathit{EPPrec}(\mathfrak{C}) = \sum\nolimits_{\mathcal{C}_i \in \Gamma} \sum\nolimits_{\mathcal{C}_j \in \Gamma} Pr(\mathcal{C}_i) \times Pr(\mathcal{C}_j) \times sim(\mathcal{C}_i, \mathcal{C}_j) \tag{8.45}$$

A commonly used similarity measure for clusterings is the Rand-Index [Ran71]. Nevertheless, any of the measures for decision correctness that we have presented in Section 4.3.8 theoretically can be used as well. Note, in this case it does not play a role whether the used measure is symmetric or not because the similarity between two alternatives is always computed in both directions.

The similarity of two clusterings can be also considered as the amount of shared clustering information. Thus, we can say that two clusterings are more similar the more clustering facts they have in common. Consequently, a range query can be also considered as a predicate that checks whether or not the true clustering alternative contains some specific clustering information and hence checks whether or not it dominates a specific clustering fact. Thus, the Precision of an uncertain clustering can be also considered as the amount of clustering facts that are shared by its alternatives. Since clustering facts can overlap and we want to avoid that same clustering information is considered for multiple times, we restrict this consideration to the smallest clustering facts possible, i.e. pairwise duplicate decisions.

A pairwise duplicate decision is in turn the more certain, the more clustering alternatives include this decision. Thus, the precision of a probabilistic clustering $\mathfrak{C}$ with $n = |rng(\mathfrak{C})|$ can be rated by the *Average Decision Certainty* that we define as follows:

$$\mathit{AvgDecC}(\mathfrak{C}) = \frac{1}{|\mathit{Pairs}(\mathfrak{C})|} \times \sum\nolimits_{\{e_r, e_s\} \in \mathit{Pairs}(\mathfrak{C})} \mathit{DecC}(\mathfrak{C}, \{e_r, e_s\}) \tag{8.46}$$

$$= \binom{n}{2}^{-1} \times \sum\nolimits_{\{e_r, e_s\} \in \mathit{Pairs}(\mathfrak{C})} \mathit{DecC}(\mathfrak{C}, \{e_r, e_s\}) \tag{8.47}$$

where $\mathit{DecC}(\mathfrak{C}, \{e_r, e_s\})$ is the certainty of the duplicate decision on the entity pair $\{e_r, e_s\}$ in $\mathfrak{C}$.

Clustering	*RExA*	*invNA*	$\boldsymbol{H}$	$inv\boldsymbol{H}_{norm}$	*CDens′*	*ADec′*	*CDens**	*ADec**	*EPPrec*	*AvgDecC*
$\mathcal{C}_1$	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\mathfrak{C}_2$	0.9995	$1/3$	-1.52	0.873	$1/3$	0.158	$2/3$	0.579	0.931	0.886
$\mathfrak{C}_3$	0.9990	$1/5$	-2.30	0.808	$1/5$	0.053	$3/5$	0.527	0.854	0.86
$\mathfrak{C}_4$	0.9990	$1/5$	-2.19	0.818	$1/5$	0.091	$1/5$	0.091	0.839	0.764

Table 8.9.: The certainty scores of the four sample clusterings in the Possible World Semantics

A duplicate decision is most certain if it is a MATCH with probability 1.0 or if it is an UNMATCH with probability 1.0 (and consequently is a MATCH with probability 0.0) and is most uncertain if it is a MATCH and an UNMATCH each with probability 0.5. Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering, the certainty of the duplicate decision on the entity pair $\{e_r, e_s\}$ in $\mathfrak{C}$ can be therefore defined as:

$$DecC(\mathfrak{C}, \{e_r, e_s\}) = \frac{|1/2 - \sum_{\mathcal{C}\in\Gamma, \{e_r,e_s\}\in M(\mathcal{C})} Pr(\mathcal{C})|}{1/2} \tag{8.48}$$

Example 209 *For illustration we reconsider the clusterings from the motivating example. The certainty scores of these clusterings that result from rating them with the measures that we have presented above are listed in Table 8.9. Note, among these measures the conventional entropy H is the only one that is unnormalized. We compute the Certainty Density and the Answer Decisiveness two times, one for the non-factorized representation (CDens′ and ADec′) and one for the complete factorized representation (CDens* and ADec*). Since $\mathfrak{C}_4$ cannot be factorized, it has the same scores in both cases.*

Some of the above discussed problems are perfectly demonstrated by these sample scores. For instance, the eight Bell number $B_8 = 4140$ is already so large that the Ratio of Excluded Alternatives becomes close to one for all four clusterings and hence the resultant scores become only less representative. In contrast, using the Bell number is still suitable for the Inverse Normalized Entropy. The Inverse Number of Alternatives is representative, but ignores probabilities and therefore only enable a coarse distinction between the four sample clusterings. The Answer Decisiveness of the non-factorized representations of the uncertain clusterings is already close to zero, even they exclude at least 4175 of the 4180 cluster-disjoint clusterings that are possible for the considered set of entities. Consequently, whereas the Ratio of Excluded Alternatives is too close to one, Answer Decisiveness seems to be too close to zero. Moreover, the Certainty Density and the Answer Decisiveness of the uncertain clusterings extremely depend on the representation and are many times larger if these clusterings can be factorized. Note, since only $\mathfrak{C}_2$ and $\mathfrak{C}_3$ can be factorized by comparable factor sets that are not a trivial partition, the scores CDens and ADec* are only comparable between these two uncertain clusterings.*

The sample clusterings also demonstrate that the two Precision measures capture aspects of decision certainty that are not captured by the other measures because only for the Expected Pair Precision and for the Average Decision Certainty $\mathfrak{C}_3$ is more certain than $\mathfrak{C}_4$ (recall that the CDens and the ADec* of $\mathfrak{C}_3$ and $\mathfrak{C}_4$ are not comparable).*

Finally, we can observe a large discrepancy between the scores of the individual measures. Thus, a relative comparison of several scores seems much more appropriate than to consider these scores as

absolute values of quality. Furthermore, comparing scores that are rated by different measures are only less meaningful. Such situations, however, can occur if we want to select data sources in an integration scenario where for each candidate source an independently computed certainty score is provided.

As we have demonstrated by the above discussions and illustrated by the given example, there is no single measure for decision certainty that perfectly fits to our purpose. The measures of Precision rate another aspect of certainty (range queries) than the measures of the other approaches (point queries). Therefore it makes sense to consider the measures of Precision not as competitors of the other measures, but to incorporate each one measure of both groups into the final quality evaluation.

For rating point query certainty none of the above presented measures is without a shortcoming. The Ratio of Excluded Alternatives, however, dependents too strong on the size of the corresponding Bell number. For that reason, we discourage from using this measure. The Inverse Number of Alternatives and the Certain Density are only suitable for incomplete clusterings. In the case of rating probabilistic clusterings we therefore recommend to use the conventional Shannon Entropy, the Inverse Normalized Entropy, or the Answer Decisiveness where the Inverse Normalized Entropy becomes more suitable the less entities are considered and the Answer Decisiveness become more suitable the finer the rated clusterings can be factorized into comparable representations. The final selection, however, depends on the considered scenario and hence is up to the user.

8.8.4. Decision Quality

As described in the previous sections, we rate the quality of an uncertain clustering by its decision correctness and its decision certainty. Of course, selecting the 'best' detection result from a set of detection results is simple if one of them is superior to the others. In many realistic situations, however, it is typical that each of the compared duplicate clusterings has its benefits and a superior clustering is not given. For that purpose we need to combine the computed quality scores for decision correctness and decision certainty in order to determine a single quality score. Both quality dimensions are usually contrary because the more decisions are certain the less correct these decisions tend to be. This trade-off is similar to the trade-off between Pair Precision and Pair Recall that we know from conventional duplicate detection processes. As a consequence, it seems intuitive to rate the decision quality of an uncertain clustering by the harmonic mean of its decision correctness and its decision certainty and hence to define decision quality as:

$$\mathit{quality} = \frac{2 \times \mathit{correctness} \times \mathit{certainty}}{\mathit{correctness} + \mathit{certainty}} \tag{8.49}$$

The problem with the harmonic mean, however, is that each of both dimensions can be rated by several measures and the absolute quality scores that are computed by the individual measures can vary to a large extent although they agree in the relative differences and they agree in their orderings. This circumstance introduces a strong dependency between the result of the harmonic mean and the actually used measures. For instance, the decision certainty of the two sample clusterings $\mathfrak{C}_2$ and $\mathfrak{C}_4$ are 0.873 and 0.818 respectively if we use the Inverse Normalized Entropy, but are 0.158 and 0.091 respectively if we use the Answer Decisiveness of its non-factorized representation. In the first case, the difference between the score of $\mathfrak{C}_2$ and the score of $\mathfrak{C}_4$ is 0.055 and in the second case the difference is 0.055 which is quite similar. Nevertheless, whereas a score of 0.8 has a strong positive impact on the harmonic

mean, a score of 0.1 has a strong negative impact on this mean. Thus, if we use the harmonic mean for computing the final quality score it is very likely that using different quality measures lead to different best detection results even if same set of duplicate clusterings is considered and the relative ordering of these clusterings is always the same in each measure per dimension.

For that reason, we suggest to adopt the more generalized F_β-score. If we consider decision correctness as the counterpart of Pair Precision and if we consider decision certainty as the counterpart of Pair Recall, the F_β-score can be adopted to decision quality as:

$$\textit{quality} = (1 + \beta^2) \times \frac{\textit{correctness} \times \textit{certainty}}{(\beta^2 \times \textit{correctness}) + \textit{certainty}} \tag{8.50}$$

where β should be adjusted to a value that fits best to the actually used measures for decision correctness and decision certainty.

As for Pair Recall and Pair Precision combining both quality scores is not sometimes not appropriate, but a separate consideration of both scores is more valuable. Similar to the Recall/Precision Graph [Chr12] constructing a Correctness/Certainty Graph can help to find the optimal configuration of an indeterministic duplicate detection approach.

8.8.5. Factor-based Quality Computation

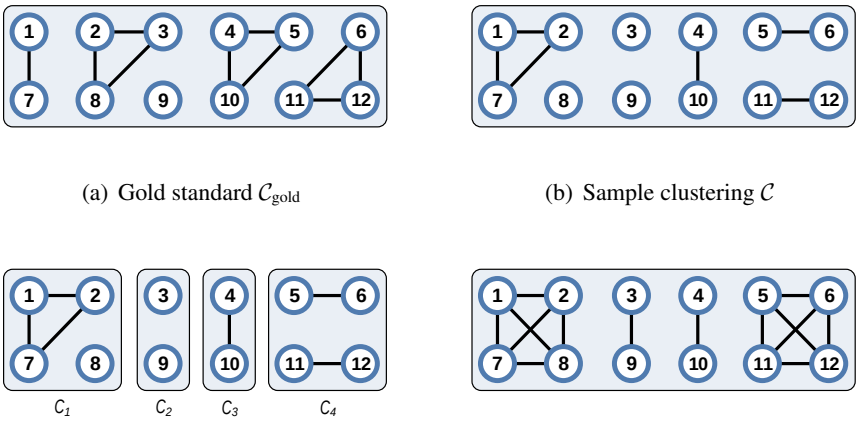
In the previous sections we present several measures for rating the decision correctness and the decision certainty of uncertain clusterings. Almost all of these measures are defined with a non-factorized clustering in mind. An uncertain clustering, however, is usually not given by its non-factorized representation, but is given by a factorization instead. Reconstructing the non-factorized representation of an uncertain clustering from its factorized representation for quality evaluation purposes is usually a pointless approach because it is too expensive in storage as well as computation time. As a consequence, we need methods for directly computing the quality of an uncertain clustering based on its factorized representation instead. Whether or not a quality score can be exactly computed in a factor-based fashion depends on the considered quality measure and the used quality semantics. For that reason, we initially consider some properties of quality measures in Section 8.8.5.1. Then we discuss a factor-based computation of decision correctness for each of the four quality semantics in Section 8.8.5.3 to Section 8.8.5.6. Finally, we consider a factor-based computation of decision certainty in Section 8.8.5.7.

8.8.5.1. Useful Properties of Quality Measures

Whether or not the quality of an uncertain clustering can be computed based on the clustering's factors depends on some properties of the considered quality measure. For that reason, we study such properties in this section.

Θ-decomposability An exact quality score of an uncertain clustering can be efficiently computed based on the quality scores of its factors, if we can simply combine the latter to get the first. Such a combination, however, is only possible if the used quality measure is Θ-decomposable.

Definition 75 (Θ-decomposability of Quality Measures): *Let S be an arbitrary set of elements and let $\mathcal{C} \in C\text{-}All_{\circledast}(S)$ be a cluster-disjoint clustering of S, and let $\mathcal{C}_{gold}$ be the corresponding gold standard.*



(a) Gold standard $\mathcal{C}_{\text{gold}}$

(b) Sample clustering $\mathcal{C}$

(c) Factorization of $\mathcal{C}$ into $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$

(d) Factor set clustering $\mathcal{C}_F = \{F_1, F_2, F_3, F_4\}$

Figure 8.31.: Sample for illustrating the properties Σ-decomposability and quasi-Σ-decomposability

A quality measure QM is called to be Θ***-decomposable****, iff for every range-disjoint factorization* $\mathcal{F}$ *of* $\mathcal{C}$ *with* $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ *there is an operator* Θ *and there is a function* f_A *so that:*

$$QM(\mathcal{C}, \mathcal{C}_{gold}) = \Theta_{i=1}^{k}\Big(f_A\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big)$$

Actually, we have to distinguish between measures for decision correctness that each get the rated clustering as well as the corresponding gold standard as input and quality measures that only get the rated clustering as input (e.g. measures for decision certainty). However, to make the notation convenient and to avoid further definitions we simply assume that for measures of the latter class the gold standard is ignored by the function f_A and hence the result is independent from this (actually not existing) input value. Thus, in the case of a quality measure that does not get the gold standard as input the equation in Definition 75 can be considered as:

$$QM(\mathcal{C}) = \Theta_{i=1}^{k}\Big(f_A\big(\pi_{F_i}(\mathcal{C})\big)\Big)$$

Since the individual results of the function f_A are combined by using the operator Θ, the former must produce a value as output that is a legal input to the latter. Usually, a quality measure *QM* is Θ-decomposable by summing up the factor-based computed values and by using itself as function f_A, i.e. by setting $\Theta = \sum$ and $f_A = QM$. Well known Σ-decomposable quality measures are for example the Number of Pair True Positives or the Number of Pair False Positives.

Example 210 *For illustration, we consider the clustering from Figure 8.31(b) and its corresponding gold standard from Figure 8.31(a). Then we assume that the clustering is factorized into the four factors* $\mathcal{C}_1$*,* $\mathcal{C}_2$*,* $\mathcal{C}_3$*, and* $\mathcal{C}_4$ *that are depicted in Figure 8.31(c) by partition the element set* $S = \{e_1, e_2, \ldots, e_{12}\}$ *into the four factor sets* $F_1 = \{e_1, e_2, e_7, e_8\}$*,* $F_2 = \{e_3, e_9\}$*,* $F_3 = \{e_4, e_{10}\}$*, and* $F_4 = \{e_5, e_6, e_{11}, e_{12}\}$*.*

To demonstrate the property of Σ-decomposability we first consider the Number of Pair True Positives. A Pair True Positive is always restricted to a single cluster and hence is restricted to a single factor. Therefore, the Number of all Pair True Positives results in summing up the Number of Pair True Positives per factor. The factors $\mathcal{C}_1$, $\mathcal{C}_3$ and $\mathcal{C}_4$ each contain one Pair True Positive, i.e. the pairs $\{e_1, e_7\}$ (factor $\mathcal{C}_1$), $\{e_4, e_{10}\}$ (factor $\mathcal{C}_3$), and $\{e_{11}, e_{12}\}$ (factor $\mathcal{C}_4$). If we consider the non-factorized representation of clustering $\mathcal{C}$, we will see that it has three Pair True Positives which are exactly the same as in its factors, namely $\{e_1, e_7\}$, $\{e_4, e_{10}\}$, and $\{e_{11}, e_{12}\}$. As a consequence, we can compute the Number of Pair True Positives of $\mathcal{C}$ without evaluating $\mathcal{C}$ itself, but only by summing up the Number of Pair True Positives of its factors $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$.

As a counterexample, we consider the Number of Pair False Negatives. This quality measure is not Σ-decomposable because the entities of a Pair False Negative can belong to clusters of different factors. For instance, in the given sample the Pair False Negative $\{e_2, e_3\}$ cannot be detected by considering the individual factors separately.

$\cup$-decomposable quality measures are the set of Pair True Positives and the set of Pair False Positives. For instance, in the example considered above the set of Pair True Positives of $\mathcal{C}$ results in $pTP(\mathcal{C}) = \{\{e_1, e_7\}, \{e_4, e_{10}\}, \{e_{11}, e_{12}\} = \{\{e_1, e_7\}\} \cup \emptyset \cup \{\{e_4, e_{10}\}\} \cup \{\{e_{11}, e_{12}\}\} = pTP(\mathcal{C}_1) \cup pTP(\mathcal{C}_2) \cup pTP(\mathcal{C}_3) \cup pTP(\mathcal{C}_4)$. A Π-decomposable measure is the Number of Alternatives (see Equation 8.64).

Quasi-Θ-decomposability Sometimes, the quality of a clustering cannot be simply computed by combining the scores that have been separately computed from its factors, but can be computed by finally aggregating the combined scores with an additional score that is computed based on the factor sets and the gold standard. A quality measure that can be computed in this way is called to be *quasi-Θ-decomposable*.

Definition 76 (Quasi-Θ-decomposability of Quality Measures): *Let S be an arbitrary set of elements and let $\mathcal{C} \in$ C-All$_{\otimes}(S)$ be a cluster-disjoint clustering of S, and let $\mathcal{C}_{gold}$ be the corresponding gold standard. A quality measure QM is called to be* quasi-Θ-decomposable*, iff for every range-disjoint factorization $\mathcal{F}$ of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ there is an operator Θ and there are some functions f_A, f_B, and f_C so that:*

$$QM(\mathcal{C}, \mathcal{C}_{gold}) = f_C\Big(\Theta_{i=1}^{k}\big(f_A\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big)$$

Since the result of the function f_C is the quality score, it must produce a value of the score domain of the considered quality measure. Note, similar to the property of w-decomposability e assume that the functions f_A and f_B ignore the gold standard if the quality measure only get the rated clustering as input. Thus, in the case of a quality measure that does not get the gold standard as input the equation in Definition 76 can be considered as:

$$QM(\mathcal{C}) = f_C\Big(\Theta_{i=1}^{k}\big(f_A\big(\pi_{F_i}(\mathcal{C})\big)\big), f_B(\mathcal{F}^{set})\Big)$$

The property of quasi-Θ-decomposability relaxes the requirements of Θ-decomposability because each Θ-decomposable quality measure can be formulated as a quasi-Θ-decomposable quality measure

Quality Measure	Θ-decomposability	quasi-Θ-decomposability	Requirements
Pair True Positives	$\Theta = \cup$	$\Theta = \cup$	-
Pair False Positives	$\Theta = \cup$	$\Theta = \cup$	-
Pair False Negatives	?	$\Theta = \cup$	-
Pair False Decisions	?	$\Theta = \cup$	-
Number of Pair True Positives	$\Theta = \Sigma$	$\Theta = \Sigma$	-
Number of Pair False Positives	$\Theta = \Sigma$	$\Theta = \Sigma$	-
Number of Pair False Negatives	?	$\Theta = \Sigma$	-
Number of Pair False Decisions	?	$\Theta = \Sigma$	-
Pair Recall	?	$\Theta = \Sigma$	-
Pair Precision	?	$\Theta = \Sigma$	-
Pair F_1-score	?	?	-
Closest Cluster Recall	?	?	-
Closest Cluster Precision	?	?	-
Closest Cluster F_1-score	?	?	-
Variation of Information	?	$\Theta = \Sigma$	-
Basic Merge Distance	?	$\Theta = \Sigma$	-
Generalized Merge Distance	?	$\Theta = \Sigma$	f_m is operation order independent
Generalized Merge Distance	?	?	-
Rand Index	?	$\Theta = \Sigma$	-

Table 8.10.: The properties of the individual measures for decision correctness that we have presented in Section 4.3.8

by setting $f_C(x, y) = x$. Thus, the class of all quasi-Θ-decomposable quality measures is a superclass of the class of all Θ-decomposable quality measures.

Example 211 *For illustration, we reconsider the clusterings from Figure 8.31 and reconsider the Number of Pair False Negatives. As mentioned above, this number cannot be computed by processing all factors separately, but can be computed in a factor-based fashion by using a simple trick. For that purpose, we partition the set of Pair False Negatives into two classes. The pairs of the first class are called as* intra-factor Pair False Negatives *and the pairs of the second class are called as* inter-factor Pair False Negatives. *The* intra-factor Pair False Negatives *are all Pair False Negatives its elements are within the same factor and the* inter-factor Pair False Negatives *are all Pair False Negatives its elements are within different factors. The Number of* intra-factor Pair False Negatives *is a Σ-decomposable quality measure because it can be computed by summing up the Number of the* intra-factor Pair False Negatives *per factor. In contrast, the Number of* inter-factor Pair False Negatives *is not a Σ-decomposable quality measure, but this number can be simply computed by counting the Pair False Negatives of the clustering $\mathcal{C}_F = \{F_1, F_2, F_3, F_4\}$ (see Figure 8.31(d)) that contains the factor sets as clusters. The total Number of Pair False Negatives results then in the sum of the Number of* intra-factor Pair False Nega-

tives *and the Number of* inter-factor Pair False Negatives. *As a consequence, the Number of Pair False Negatives is a quasi-Σ-decomposable quality measure with* $f_A(x,y) = f_B(x,y) = M(y) - M(x)$ *and* $f_C(x,y) = x + y$.

In this example, the factor $\mathcal{C}_1$ *has one Pair False Negative* ($\{e_2, e_8\}$)*, the factor* $\mathcal{C}_4$ *has two Pair False Negatives* ($\{e_6, e_{12}\}$ *and* $\{e_6, e_{12}\}$)*, and the factors* $\mathcal{C}_2$ *and* $\mathcal{C}_3$ *have no Pair False Negatives. Therefore, the Number of* intra-factor Pair False Negatives *is three. The pairs* $\{e_2, e_3\}$, $\{e_3, e_8\}$, $\{e_5, e_{10}\}$, *and* $\{e_4, e_5\}$ *are all* inter-factor Pair False Negatives *because its entities belong to different factors. This set of Pair False Negatives is identical to the set of Pair False Negatives of the clustering* $\mathcal{C}_F = \{F_1, F_2, F_3, F_4\}$. *In total we therefore have a set of seven Pair False Negatives which is exactly the same as the set of Pair False Negatives that results from directly evaluating the non-factorized representation of clustering* $\mathcal{C}$. *Consequently, we can compute the Number of Pair False Negatives without evaluating the non-factorized representation of* $\mathcal{C}$.

Further quasi-Σ-decomposable quality measures are the Number of Pair True Positives, the Number of Pairwise False Decisions, Pair Recall, the Closest Cluster Precision, and the (Generalized) Merge Distance. quasi-$\bigcup$-decomposable quality measures are the set of Pairwise True Positives and the set of Pairwise False Decisions.

Of course, with respect to a certain clustering a factor-based computation is worthless because evaluating the factors separately does not bring any advantage. However as mentioned before such a computation becomes crucial for evaluating uncertain clusterings because reconstructing the alternatives of the non-factorized representation of an uncertain clustering is usually not manageable. The trick with the quasi-Θ-decomposable quality measures is that they can be exactly computed for uncertain clusterings as efficient as the Θ-decomposable quality measures because the factor set clustering $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ and the gold standard $\mathcal{C}_{\text{gold}}$ are always certain. For illustration, the *inter-factor Pair False Negative* of an uncertain clustering are the same in any of its alternatives and thus the factor set clustering only need to be evaluated once.

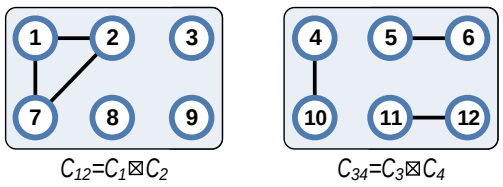
In Table 8.10, we list some measures for decision correctness that we have presented in Section 4.3.8 and mark whether or not they are Θ-decomposable or quasi-Θ-decomposable by naming the corresponding Θ operator. We can prove whether a given function is quasi-Θ-decomposable or not, but it is hard to exclude a measure from the class of all quasi-Θ-decomposable quality measures for sure because each measure can be computed in several ways and it could be the case that a measure can be computed by a quasi-Θ-decomposable function, but this function is currently unknown to us. For that reason, we indicate all quality measures for which we currently do not know a quasi-Θ-decomposable function that can be used to compute them by a question mark. The assertions that are summarized in this table are proved in Appendix A.6.

Approach of Minimal Defactorization As presented in Table 8.10, we currently do not know a function for computing the Closest Cluster Recall that is Θ-decomposable or is quasi-Θ-decomposable. Nevertheless, this measure (and other measures as for example the *Average Author Purity* [CGL07] as well) can be computed in a factor-based fashion by using a simple trick.

The function of the Closest Cluster Recall that we have presented in Equation 4.42 is not Θ-decomposable because a cluster $C \in \mathcal{C}_{\text{gold}}$ of the gold standard can intersect with multiple factor sets

$C_i \in \mathcal{C}_{\text{gold}}$	closest cluster $C_j \in \mathcal{C}$	$sim(C_i, C_j)$
$\{e_1, e_7\}$	$\{e_1, e_2, e_7\}$	$2/3$
$\{e_2, e_3, e_8\}$	$\{e_2\}$ or $\{e_3\}$ or $\{e_8\}$	$1/3$
$\{e_9\}$	$\{e_9\}$	1
$\{e_4, e_5, e_{10}\}$	$\{e_4, e_{10}\}$	$2/3$
$\{e_6, e_{11}, e_{12}\}$	$\{e_{11}, e_{12}\}$	$2/3$

(a) The closest clusters of $\mathcal{C}$ for each cluster of $\mathcal{C}_{\text{gold}}$



(b) Factorization of $\mathcal{C}$ into $\{\mathcal{C}_1 \boxtimes \mathcal{C}_2, \mathcal{C}_3 \boxtimes \mathcal{C}_4\}$

Figure 8.32.: Sample for minimal defactorization

of the factorized clustering $\mathcal{C}$ and hence we cannot compute the cluster of $\mathcal{C}$ that is most similar to C by considering each of the factors separately. However, the clusters of $\mathcal{C}_{\text{gold}}$ are usually not very large. For that reason, we do not need to reconstruct the non-factorized representation of $\mathcal{C}$, but only need to integrate the factors of $\mathcal{C}$ whose factor sets intersect with the same cluster of $\mathcal{C}_{\text{gold}}$.

Theorem 33 *Let $\mathcal{C}$ be a duplicate clustering and let $\mathcal{C}_{gold}$ be the corresponding gold standard, if $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ is a factor set that forms a range-disjoint factorization $\mathcal{F}_C$ of $\mathcal{C}$ and forms a range-disjoint factorization $\mathcal{F}_{\mathcal{C}_{gold}}$ of $\mathcal{C}_{gold}$, i.e. $\boxtimes_{i=1}^{k} \pi_{F_i}(\mathcal{C}) = \mathcal{C}$ and $\boxtimes_{i=1}^{k} \pi_{F_i}(\mathcal{C}_{gold}) = \mathcal{C}_{gold}$, the Closest Cluster Recall of $\mathcal{C}$ with respect to $\mathcal{C}_{gold}$ can be computed as:*

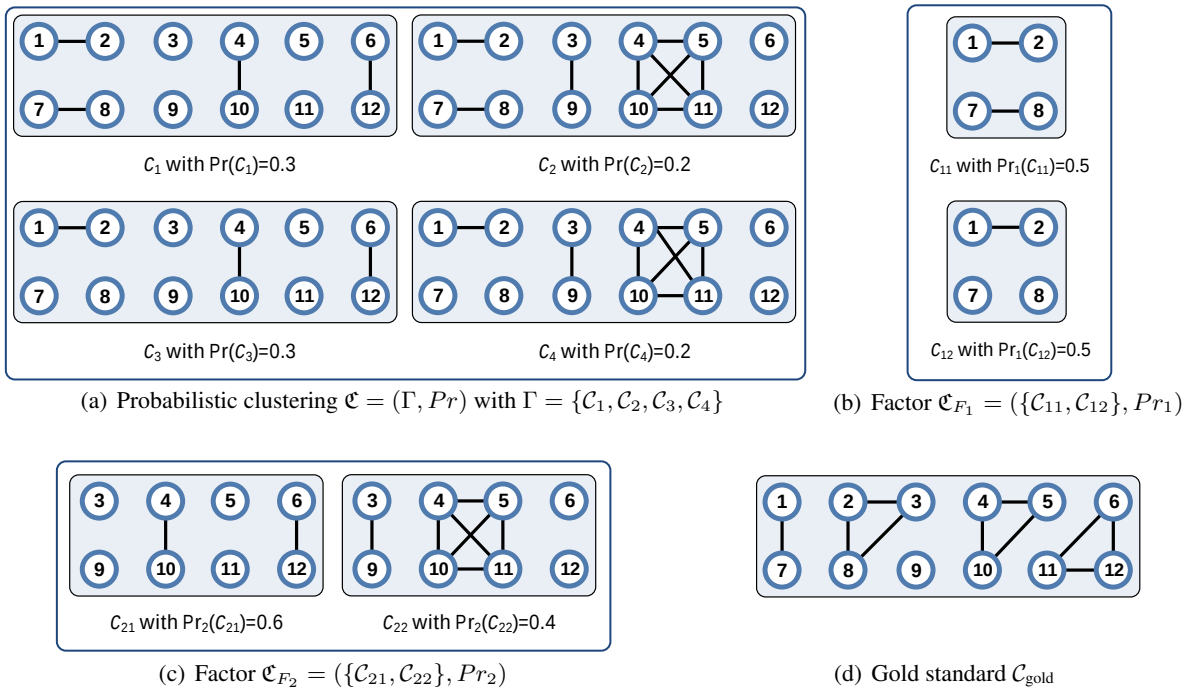
$$ccRec(\mathcal{C}, \mathcal{C}_{gold}) = \sum_{i=1}^{k} \frac{|\pi_{F_i}(\mathcal{C}_{gold})|}{|\mathcal{C}_{gold}|} \times ccRec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))$$

Theorem 33 is proved by Proof 34 in Section A.5.

As a consequence, to enable a factor-based computation of the Closest Cluster Recall of a clustering $\mathcal{C}$ with respect to the gold standard $\mathcal{C}_{\text{gold}}$, we only need to defactorize the given factorization $\mathcal{F}$ of $\mathcal{C}$ into a coarser factorization of $\mathcal{C}$ whose factor sets also form a factorization of $\mathcal{C}_{\text{gold}}$. This can be realized by integrating each two factors of $\mathcal{F}$ whose factor sets overlap with the same cluster of the gold standard.

Example 212 *For illustration we reconsider the clustering $\mathcal{C}$ and the gold standard $\mathcal{C}_{gold}$ from from Figure 8.31. The closest clusters of $\mathcal{C}$ for each cluster of $\mathcal{C}_{gold}$ are presented in Table 8.32(a) (for similarity computation we use the Jaccard Coefficient). Thus, the Closest Cluster Recall of $\mathcal{C}$ with respect to $\mathcal{C}_{gold}$ results in $ccRec(\mathcal{C}, \mathcal{C}_{gold}) = \frac{1}{5} \times (\frac{2}{3} + \frac{1}{3} + 1 + \frac{2}{3} + \frac{2}{3}) = \frac{2}{3}$.*

First, we consider the factorization of $\mathcal{C}$ that is presented in Figure 8.31(c). This factorization divides $\mathcal{C}$ into the four factors $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$. It is easy to see that we cannot compute the Closest Cluster Recall of $\mathcal{C}$ with respect to $\mathcal{C}_{gold}$ based on this factorization because some clusters of $\mathcal{C}_{gold}$ have entities that belong to different factors. For example, the entities of the cluster $\{e_2, e_3, e_8\}$ are distributed among the factors $\mathcal{C}_1$ and $\mathcal{C}_2$. Nevertheless, if we integrate $\mathcal{C}_1$ and $\mathcal{C}_2$ into $\mathcal{C}_{12}$ and if we integrate $\mathcal{C}_3$ and $\mathcal{C}_4$ into $\mathcal{C}_{34}$, we get a factorization of $\mathcal{C}$ whose factor sets $F_{12} = \{e_1, e_2, e_3, e_7, e_8, e_9\}$ and $F_{34} = \{e_4, e_5, e_6, e_{10}, e_{11}, e_{12}\}$ also form a factorization of $\mathcal{C}_{gold}$, i.e. these factor sets can be used to factorize $\mathcal{C}_{gold}$ into the two factors $\pi_{F_{12}}(\mathcal{C}_{gold}) = \{\langle e_1, e_7\rangle, \langle e_2, e_3, e_8\rangle, \langle e_9\rangle\}$ and $\pi_{F_{34}}(\mathcal{C}_{gold}) =$



(a) Probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ with $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$

(b) Factor $\mathfrak{C}_{F_1} = (\{\mathcal{C}_{11}, \mathcal{C}_{12}\}, Pr_1)$

(c) Factor $\mathfrak{C}_{F_2} = (\{\mathcal{C}_{21}, \mathcal{C}_{22}\}, Pr_2)$

(d) Gold standard $\mathcal{C}_{\text{gold}}$

Figure 8.33.: Demonstrating example for factor-based quality computation

$\{\langle e_4, e_5, e_{10}\rangle, \langle e_6, e_{11}, e_{12}\rangle\}$. *By considering the factorization into* $\mathcal{C}_{12}$ *and* $\mathcal{C}_{34}$ *no cluster of* $\mathcal{C}_{gold}$ *overlaps with different factor sets and the Closest Cluster Recall can be therefore computed as:*

$$\begin{aligned} ccRec(\mathcal{C}, \mathcal{C}_{gold}) &= \frac{|\pi_{F_{12}}(\mathcal{C}_{gold})|}{|\mathcal{C}_{gold}|} \times ccRec(\mathcal{C}_{12}, \pi_{F_{12}}(\mathcal{C}_{gold})) + \frac{|\pi_{F_{34}}(\mathcal{C}_{gold})|}{|\mathcal{C}_{gold}|} \times ccRec(\mathcal{C}_{34}, \pi_{F_{34}}(\mathcal{C}_{gold})) \\ &= \frac{3}{5} \times \left(\frac{2/3 + 1/3 + 1}{3}\right) + \frac{2}{5} \times \left(\frac{2/3 + 2/3}{2}\right) = \frac{2}{3} \end{aligned}$$

8.8.5.2. Demonstrating Example

As a demonstrating example throughout the remainder of this section we consider the probabilistic clustering $\mathfrak{C}$ from Figure 8.33(a) and its factorization $\mathcal{F}$ into the two factors $\mathfrak{C}_{F_1}$ (Figure 8.33(b)) and $\mathfrak{C}_{F_2}$ (Figure 8.33(c)) with the factor sets $\mathcal{F}^{set} = \{F_1 = \{e_1, e_2, e_7, e_8\}, F_2 = \{e_3, e_4, e_5, e_6, e_9, e_{10}, e_{11}, e_{12}\}\}$. Note that the factorization is not complete because it simplifies the following examples. For quality evaluation we use the same gold standard as in Section 8.8.5.1. For simplification it is depicted in Figure 8.33(d) once more. The quality scores of the individual alternatives of all three probabilistic clusterings as well as the clustering $\mathcal{C}_{\mathcal{F}^{set}}$ that results from considering the factor sets as clusters are presented in Table 8.11. Note that the Number of Pair False Negatives, the Pair Recall, and the Pair F_1-score of a clustering are always computed by using the entity equivalent part of the gold standard.

Clustering	$\lvert p\text{TP}\rvert$	$\lvert p\text{FP}\rvert$	$\lvert p\text{FN}\rvert^*$	$p\text{Rec}^*$	$p\text{Prec}$	pF_1^*
$\mathcal{C}_1$	2	2	8	$2/10 = 0.2$	$2/4 = 0.5$	$4/14 = 0.286$
$\mathcal{C}_2$	3	6	7	$3/10 = 0.3$	$3/9 = 0.333$	$6/19 = 0.316$
$\mathcal{C}_3$	2	1	8	$2/10 = 0.2$	$2/3 = 0.667$	$4/13 = 0.308$
$\mathcal{C}_4$	3	5	7	$3/10 = 0.3$	$3/8 = 0.375$	$6/18 = 0.333$
$\mathcal{C}_{11}$	0	2	2	$0/2 = 0.0$	$0/2 = 0.0$	$0/4 = 0$
$\mathcal{C}_{12}$	0	1	2	$0/2 = 0.0$	$0/1 = 0.0$	$0/3 = 0$
$\mathcal{C}_{21}$	2	0	4	$2/6 = 0.333$	$2/2 = 1.0$	$4/8 = 0.5$
$\mathcal{C}_{22}$	3	4	3	$3/6 = 0.5$	$3/7 = 0.429$	$6/13 = 0.46$
$\mathcal{C}_{\mathcal{F}^{set}}$	8	26	2	$8/10 = 0.8$	$8/34 = 0.235$	$16/44 = 0.364$

* evaluated on the entity equivalent part of $\mathcal{C}_{\text{gold}}$

Table 8.11.: The pair-based quality (with respect to $\mathcal{C}_{\text{gold}}$) of all certain clusterings of the demonstrating example

8.8.5.3. Possible Worlds Semantics

A factorization of an uncertain clustering always includes a factorization of its alternatives (see Theorem 23). Thus, the properties of Θ-decomposability and quasi-Θ-decomposability can be adopted from the crisp version of a quality measure to its incomplete version and its probabilistic version respectively.

Theorem 34 *Let QM be a Θ-decomposable quality measure and let $\mathcal{F}$ be a range-disjoint factorization of an incomplete clustering Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, the incomplete version of QM can be computed as:*

$$QM^{PWS}(\Gamma, \mathcal{C}_{gold}) = \Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\Gamma), \pi_{F_i}(\mathcal{C}_{gold})\big)$$

Theorem 34 is proved by Proof 35 in Section A.5.

Theorem 35 *Let QM be a quasi-Θ-decomposable quality measure and let $\mathcal{F}$ be a range-disjoint factorization of an incomplete clustering Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, the incomplete version of QM can be computed as:*

$$QM^{PWS}(\Gamma, \mathcal{C}_{gold}) = f_C\Big(\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\Gamma), \pi_{F_i}(\mathcal{C}_{gold})\big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big)$$

Theorem 35 is proved by Proof 36 in Section A.5.

A computation of the probabilistic version of a Θ-decomposable quality measure and a computation of the probabilistic version of a quasi-Θ-decomposable quality measure can be adapted to probabilistic clusterings accordingly.

The approach of minimal reconstruction uses the fact that the considered quality measure is quasi-Θ-decomposable in particular situations. Thus, for every situation this approach works in the certain case, it works in the uncertain case as well.

8.8.5.4. Aggregated World Semantics

In this section, we consider methods that can be used to compute the aggregated quality score of a uncertain clustering based on its factors. For some quality measures we can compute the aggregated quality score exactly, but for some not. For that reason, we initially present methods for an exact quality score computation and then propose strategies for factor-based computation of upper bounds and lower bounds that can be used in cases where an exact quality score cannot be computed in a factor-based fashion.

Factor-based Computation of Exact Quality Scores For a set of random variables (whether mutually independent or not), the expected value of the sum of these variables is equivalent to the sum of their expected values [GT96, CA03]. Moreover, for a set of mutually independent random variables (note in this case the covariance is zero), the expected value of the product of these variables is equivalent to the product of their expected values [GT96, CA03]. In the Aggregated World Semantics we restrict our consideration to quality measures that have a numerical score domain. Since numerical domains are measure spaces, each probabilistic value of a numerical domain in general and each probabilistic quality score in particular can be considered as a random variable. Thus, let $\{z_1, \ldots, z_k\}$ be a set of mutually independent probabilistic quality scores, it holds:

$$EXP\big(\sum_{i=1}^{k} z_i\big) = \sum_{i=1}^{k} EXP(z_i) \tag{8.51}$$

and

$$EXP\big(\prod_{i=1}^{k} z_i\big) = \prod_{i=1}^{k} EXP(z_i) \tag{8.52}$$

Moreover, let z be a probabilistic quality score and let $c, d \in \mathbb{R}$ be two constant real numbers, it holds:

$$EXP(c \times z + d) = c \times EXP(z) + d \tag{8.53}$$

This leads us to the following two theorems.

Theorem 36 *Let QM be a quality measure that is Θ-decomposable with the operator $\Theta \in \{\Sigma, \Pi\}$ and the function f_A. Moreover, let $\mathfrak{C}$ a probabilistic clustering that is represented by the range-disjoint factorization $\mathcal{F}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, it holds that:*

$$EXP\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{gold})\big) = \Theta_{i=1}^{k} EXP\Big(f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big)$$

Theorem 36 is proved by Proof 37 in Section A.5.

Theorem 37 *Let QM be a quality measure that is quasi-Θ-decomposable with the operator $\Theta \in \{\Sigma, \Pi\}$ and the functions f_A, f_B, and f_C. Moreover, let $\mathfrak{C}$ a probabilistic clustering that is represented by the range-disjoint factorization $\mathcal{F}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. If f_B produces a numerical value and if f_C is either the sum operator or the multiplication operator, i.e. $f_C(x, y) = x + y$ or $f_C(x, y) = x \times y$, it holds that:*

$$EXP\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{gold})\big) = f_C\Big(\Theta_{i=1}^{k} EXP\Big(f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big)$$

Theorem 37 is proved by Proof 38 in Section A.5.

By using these theorems we can propagate the expected value computation to the individual factors and hence can compute the expected quality score of a probabilistic clustering by combining the expected quality scores of its factors on condition that the used quality measure is Σ-decomposable, quasi-Σ-decomposable, Π-decomposable, or quasi-Π-decomposable.

A similar concept can be used for the minimum operator and the maximum operator. Trivially, the minimum of the sum of a set of mutually independent random variables is equivalent to the sum of their minimums and the maximum of the sum of a set of mutually independent random variables is equivalent to the sum of their maximums. The same holds for the product of a set of mutually independent random variables. Thus, let $\{z_1, \ldots, z_k\}$ be a set of mutually independent probabilistic (numerical) quality scores, it holds:

$$\mathbf{min}\big(\sum_{i=1}^{k} z_i\big) = \sum_{i=1}^{k} \mathbf{min}(z_i) \tag{8.54}$$

$$\mathbf{max}\big(\sum_{i=1}^{k} z_i\big) = \sum_{i=1}^{k} \mathbf{max}(z_i) \tag{8.55}$$

$$\mathbf{min}\big(\prod_{i=1}^{k} z_i\big) = \prod_{i=1}^{k} \mathbf{min}(z_i) \tag{8.56}$$

$$\mathbf{max}\big(\prod_{i=1}^{k} z_i\big) = \prod_{i=1}^{k} \mathbf{max}(z_i) \tag{8.57}$$

Moreover, let z be a probabilistic quality score and let $c, d \in \mathbb{R}$ be two constant real numbers, as for the expected value it holds:

$$\mathbf{min}(c \times z + d) = c \times \mathbf{min}(z) + d \tag{8.58}$$

$$\mathbf{max}(c \times z + d) = c \times \mathbf{max}(z) + d \tag{8.59}$$

As a consequence, we can derive equations for the minimum operator and the maximum operator that corresponds to the equations we have proved for the expected value in Theorem 36 and Theorem 37. Thus, let *QM* be a quality measure that is Θ-decomposable with the operator $\Theta \in \{\Sigma, \Pi\}$ and the function f_A. Moreover, let $\mathfrak{C}$ a probabilistic clustering that is represented by the range-disjoint factorization $\mathcal{F}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, it holds that:

$$\mathbf{min}\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}})\big) = \Theta_{i=1}^{k} \mathbf{min}\Big(f_A\big(\mathfrak{C}_F, \pi_F(\mathcal{C}_{\text{gold}})\big)\Big) \tag{8.60}$$

$$\mathbf{max}\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}})\big) = \Theta_{i=1}^{k} \mathbf{max}\Big(f_A\big(\mathfrak{C}_F, \pi_F(\mathcal{C}_{\text{gold}})\big)\Big) \tag{8.61}$$

Furthermore, let *QM* be a quality measure that is Θ-decomposable with the operator $\Theta \in \{\Sigma, \Pi\}$ and the functions f_A, f_B, and f_C, and let $\mathfrak{C}$ a probabilistic clustering that is represented by the range-disjoint factorization $\mathcal{F}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. If f_B produces a numerical value and if f_C is either the sum operator or the multiplication operator, i.e. $f_C(x, y) = x + y$ or $f_C(x, y) = x \times y$, it holds that:

$$\mathbf{min}\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}})\big) = f_C\Big(\Theta_{i=1}^{k} \mathbf{min}\Big(f_A\big(\mathfrak{C}_F, \pi_F(\mathcal{C}_{\text{gold}})\big)\Big), f_B(\mathcal{F}^{set}, \mathcal{C}_{\text{gold}})\Big) \tag{8.62}$$

$$\mathbf{max}\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{\text{gold}})\big) = f_C\Big(\Theta_{i=1}^{k} \mathbf{max}\Big(f_A\big(\mathfrak{C}_F, \pi_F(\mathcal{C}_{\text{gold}})\big)\Big), f_B(\mathcal{F}^{set}, \mathcal{C}_{\text{gold}})\Big) \tag{8.63}$$

In conclusion, for quality measures that are Σ-decomposable, quasi-Σ-decomposable, Π-decomposable, or quasi-Π-decomposable (with some meaningful restrictions to the functions f_B and

f_C), we can exactly compute the expected quality score, the minimal quality score, and the maximal quality score of an uncertain clustering based on the quality scores of its factors.

Example 213 *For illustration we consider the probabilistic clustering $\mathfrak{C}$ from Figure 8.33(a) that can be factorized into the two factors Γ_{F_1} and Γ_{F_2} as presented in Figure 8.33(b) and Figure 8.33(c). The maximal Number of Pair False Positives of $\mathfrak{C}$ is six (alternative $\mathcal{C}_2$) and the maximal Number of Pair False Negatives of $\mathfrak{C}$ is eight (alternatives $\mathcal{C}_1$ and $\mathcal{C}_3$). The Number of Pair False Positives is a Σ-decomposable quality measure. Thus, we can compute the maximal Number of Pair False Positives of $\mathfrak{C}$ by summing up the maximal Number of Pair False Positives of all its factors. Factor $\mathfrak{C}_{F_1}$ has either two Pair False Positives (alternative $\mathcal{C}_{11}$) or one Pair False Positive (alternative $\mathcal{C}_{12}$) and factor $\mathfrak{C}_{F_2}$ has either zero Pair False Positives (alternative $\mathcal{C}_{21}$) or has four Pair False Positives (alternative $\mathcal{C}_{22}$). Consequently, the maximal number of Pair False Positives of $\mathfrak{C}$ results in $2 + 4 = 6$ which is exactly the value that we have computed for the non-factorized representation above. The Number of Pair False Negatives is a quasi-Σ-decomposable quality measure. Therefore, we can compute the maximal Number of Pair False Negatives of $\mathfrak{C}$ by summing up the maximal Number of Pair False Negatives of all its factors (the total number of intra-factor Pair False Negatives) and by additionally adding the Number of Pair False Negatives of the factor set clustering $\mathcal{C}_F = \{\langle a, b, g, h\rangle, \langle c, d, e, f, i, j, k, l\rangle\}$ (the number of inter-factor Pair False Negatives). Recall that the latter is always a certain clustering. The alternatives of factor $\mathfrak{C}_{F_1}$ have each two Pair False Negatives and the alternatives of factor $\mathfrak{C}_{F_2}$ have either four Pair False Negatives (alternative $\mathcal{C}_{21}$) or has three Pair False Negatives (alternative $\mathcal{C}_{22}$). Thus, the maximal number of intra-factor Pair False Negatives of $\mathfrak{C}$ is $2 + 4 = 6$. The clustering $\mathcal{C}_F$ has two Pair False Negatives, i.e. $\{b, c\}$ and $\{h, c\}$. As a consequence, the maximal Number of Pair False Negatives of $\mathfrak{C}$ is $6 + 2 = 8$ which agrees with the Number of Pair False Negatives that we have manually evaluated on the non-factorized representation of $\mathfrak{C}$.*

Factor-based Computation of Quality Bounds If a quality measure is neither Σ-decomposable, quasi-Σ-decomposable, Π-decomposable nor quasi-Π-decomposable its exact result cannot be simply computed in a factor-based fashion, but need to be computed on the non-factorized representation instead. Such a computation, however, is impractical in many situations. Two solutions to this problem are to approximate the quality score or to restrict it by computing an upper bound or a lower bound respectively.

In the case of the expected quality score an approximation technique as the Monte-Carlo Simulation [KLM89] can be used, but for computing the minimal score and the maximal score such an approximation method is usually not very accurate. Nonetheless, in such cases we can use some measure-specific observations to compute upper bounds and/or lower bounds for these scores. Bounds in turn can be sufficient in many real-world scenarios to detect whether or not the solution satisfies the given quality requirements. For instance, let us assume an application that requires that the minimal possible quality score is above a specific threshold. If we can compute an upper bound of this score and this bound is already below this threshold, we do not need to compute the exact minimal quality score to come to the conclusion that the uncertain clustering does not satisfy the considered quality requirements.

The approach for bound computation that is presented below is based on the fact that all factors are mutually independent and hence every possible combination of one alternative per factor can be integrated to an alternative of the non-factorized representation. Moreover, the minimum operator and

the maximum operator are of a deciding nature, i.e. they always return a quality score that results from some alternative of the non-factorized representation. Thus, each quality score that we compute for a reconstructed alternative of the non-factorized representation is equal to or greater than the minimal quality score and is equal to or lower than the maximal quality score. Consequently, we can compute an upper bound of the minimal score and can compute a lower bound of the maximal score by reconstructing an alternative of the non-factorized representation. Obviously, it make sense to reconstruct an alternative whose quality score is expected to be close to the minimal quality score or the maximal quality score respectively. The most intuitive solution is to reconstruct an alternative for the lower bound computation of the maximal quality score by selecting from each factor the alternative that has the maximal quality score. Against intuition, the quality score of this alternative is not always equal to the exact score, but indeed is only a bound. We will illustrate this circumstance on the basis of the Pair F_1-score. Recall that the Pair F_1-score is the ratio between the double Number of Pair True Positives and the sum of the double Number of Pair True Positives, the Number of Pair False Negatives, and the Number of Pair False Positives.

Example 214 *For demonstration we reconsider the probabilistic clustering $\mathfrak{C}$ from Figure 8.33(a) that can be factorized into the two factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$ as presented in Figure 8.33(b) and Figure 8.33(c). Recall, the corresponding gold standard is depicted in Figure 8.33(d). The first factor $\mathfrak{C}_{F_1}$ has two alternatives. The first alternative $\mathcal{C}_{11}$ has two Pair False Negatives and two Pair False Positives and the second alternative $\mathcal{C}_{12}$ has two Pair False Negatives and one Pair False Positive, but none of them as any Pair True Positive. Thus, the Pair F_1-score of both alternatives is $pF_1(\mathcal{C}_{11}, \mathcal{C}_{gold}) = pF_1(\mathcal{C}_{12}, \mathcal{C}_{gold}) = 0.0$. The second factor $\mathfrak{C}_{F_2}$ has two alternatives as well. The first alternative has two Pair True Positives, four Pair False Negatives, and no Pair False Positives, and the second one has three Pair True Positives, three Pair False Negatives, and four Pair False Positives. Thus, the Pair F_1-score of these alternatives is $pF_1(\mathcal{C}_{21}, \mathcal{C}_{gold}) = 4/8 = 0.5$ and $pF_1(\mathcal{C}_{22}, \mathcal{C}_{gold}) = 6/13 = 0.46$ respectively. Although the second alternative of $\mathfrak{C}_{F_2}$ has a lower Pair F_1-score than the first alternative, the Pair F_1-score of $\mathfrak{C}$ is maximal by integrating $\mathcal{C}_{22}$ with $\mathcal{C}_{12}$. This is because the resultant clustering $\mathcal{C}_4 = \mathcal{C}_{12} \boxtimes \mathcal{C}_{22}$ has three Pair True Positives, five Pair False Negatives, and five Pair False Positives ($\Rightarrow pF_1(\mathcal{C}_4, \mathcal{C}_{gold}) = 6/16 = 0.375$), whereas the clustering $\mathcal{C}_1 = \mathcal{C}_{11} \boxtimes \mathcal{C}_{21}$ has two Pair True Positives, six Pair False Negatives, and two Pair False Positives ($\Rightarrow pF_1(\mathcal{C}_4, \mathcal{C}_{gold}) = 4/12 = 0.333$), and the the clustering $\mathcal{C}_3 = \mathcal{C}_{12} \boxtimes \mathcal{C}_{21}$ has two Pair True Positives, six Pair False Negatives, and one Pair False Positive ($\Rightarrow pF_1(\mathcal{C}_4, \mathcal{C}_{gold}) = 4/11 = 0.364$). It is also noticeable that integrating some alternative of $\mathfrak{C}_{F_2}$ with $\mathcal{C}_{12}$ leads to a higher Pair F_1-score than integrating this alternative with $\mathcal{C}_{11}$ although $\mathcal{C}_{11}$ and $\mathcal{C}_{12}$ have both the same Pair F_1-score. The reason is that $\mathcal{C}_{12}$ has less Pair False Positives than $\mathcal{C}_{11}$ and hence its quality has a weaker impact on the final Pair F_1-score than the quality of $\mathcal{C}_{11}$. In contrast, if $\mathcal{C}_{11}$ or $\mathcal{C}_{12}$ would have a Pair F_1-score of 1.0, the best choice from factor $\mathfrak{C}_{F_2}$ for computing the maximal Pair F_1-score of $\mathfrak{C}$ would be $\mathcal{C}_{21}$ instead of $\mathcal{C}_{22}$.*

This example illustrates that the Pair F_1-score of an uncertain clustering is not minimal (or maximal) if it is minimal (or maximal) in each of its factors. In other words, the alternative of the uncertain clustering that has minimal (maximal) Pair F_1-score is not equivalent to the clustering that result from integrating the alternative per factor that has the minimal (maximal) Pair F_1-score. In contrast, selecting

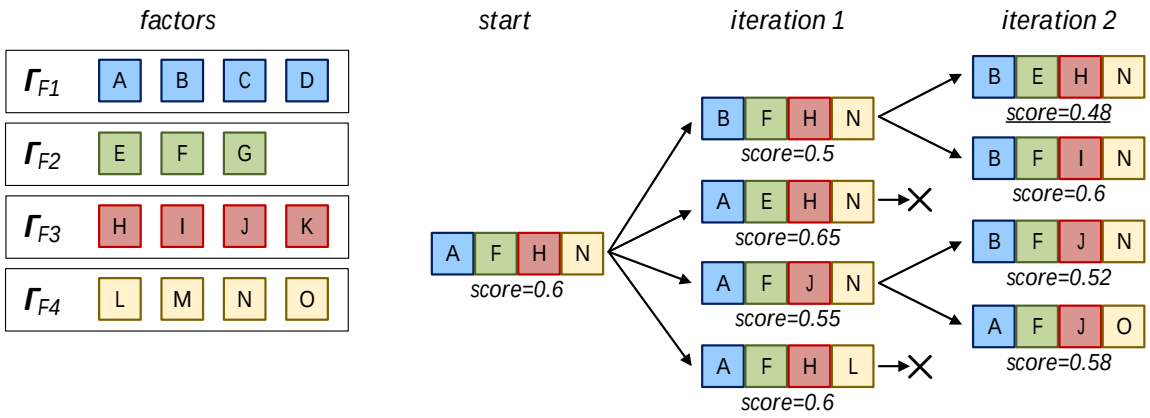


Figure 8.34.: Sample for bound computation by using a genetic programming approach

an alternative for one factor usually depends on the alternatives that have been selected for the other factors.

As we have illustrated in the previous example, propagating the optimization problem to the factors separately is often not the best approach for bound computation. Nevertheless, there is no reason why we should only reconstruct a single alternative of the non-factorized representation. In contrast, it could be useful to reconstruct a set of alternatives instead of a single one (of course, this set should be of a manageable size). An interesting approach for selecting a set of reconstructed alternatives in a meaningful way is to use a genetic programming algorithm. Such an algorithm starts from an initial solution and then applies genetic operations, i.e. crossovers (switch some solution characteristics with these of another solution) or mutations (replace some solution characteristics), to compute new solutions based on the given solutions. The quality scores of the newly computed solutions are then verified and only the best solutions are kept (survival of the fittest) and changed by genetic operations further on until a fixed number of solutions has been computed or the improvement of the best quality score becomes below a predefined threshold. To avoid local optima some of the non-best solutions are usually kept as well.

In the case of computing an upper bound for the minimal quality score, it seems useful to start with an alternative that is reconstructed from the worst alternative per factor. We think this is a good starting solution because the quality score of this alternatives is already close to the exact score in many cases. Because of the genetic programming approach, we consider each alternative of the non-factorized representation as a set of factor alternatives (each one per factor). Genetic operations can be realized by replacing a factor alternative by another alternative of the same factor. Of course, the selection of the replaced alternative and the selection of the replacing alternative can be computed randomly or can follow some specific criteria.

Example 215 *The underlying idea of the genetic programming approach is illustrated in Figure 8.34. The sample depicts the aforementioned scenario for computing an upper bound of the minimal quality score that stops after two iterations. From the four factors we start with a single solution that has the worst alternative per factor. Let us assume that the quality of the integration result is* 0.6*. From this solution, we derive four other solutions by mutating the start solution (an alternative of the start solution*

is replaced by another alternative of the same factor). The quality scores of their integration results are assumed to be 0.5*,* 0.65*,* 0.55*, and* 0.6*. Then, we select the two best solutions (the solutions that lead to the lowest quality scores) and start another iteration. In this iteration from each of the two selected solutions two other solutions are derived by the use of mutation. Let us assume that the quality scores of the mutations are* 0.48*,* 0.6*,* 0.52*, and* 0.58*. Then, we stop the process and use the quality score* 0.48 *as the final bound.*

Further Remarks Other concepts that are possibly useful to improve the efficiency of minimal score computation and maximal score computation are the properties of modularity, supermodularity, and submodularity of set functions [Sch03].

Let S be a set, a set function f (a function that gets a set as input) is called to be modular if for all sets $A, B \in 2^S$ it holds:

$$f(A) + f(B) = f(A \cap B) + f(A \cup B)$$

Interestingly, Σ-decomposability is a generalization of modularity because if we assume that f_A is the considered quality measure itself and if we define $QM(\emptyset, \emptyset) = 0$ for each quality measure QM, Σ-decomposability corresponds to modularity. Thus, every modular function is Σ-decomposable and modularity is already covered by Σ-decomposability.

Let S be a set, a set function f is supermodular if for all sets $A, B \in 2^S$ it holds:

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$$

and is submodular if the function $f' = -f$ is supermodular. From these definitions follows that if a quality measure is supermodular, the quality score of an incomplete clustering is always lower than or equal to the sum of the quality scores of its factors. Therefore, the properties of supermodularity and submodularity cannot be used for exact score computation, but they can be used to compute bounds. It seems intuitive that some quality measures that are not Σ-decomposable or quasi-Σ-decomposable are supermodular or submodular, but we have not proved it yet.

8.8.5.5. Single World Semantics

In the Single World Semantics, the quality of an uncertain clustering is computed based on the quality of a set of candidate clusterings that each is one of the uncertain clustering's alternatives. In most cases, only a few clustering alternatives are considered as candidates. Thus, it is already useful if we can compute the candidate clusterings of an uncertain clustering directly from its factors instead of reconstructing the non-factorized representation of this uncertain clustering first.

Theorem 38 *Let* $\mathfrak{C} = (\Gamma, Pr)$ *be a probabilistic clustering and let* $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ *be a range-disjoint factorization of* $\mathfrak{C}$*, the set* $\{\widehat{\mathfrak{C}_F} \mid \mathfrak{C}_F \in \mathcal{F}\}$ *is a range-disjoint factorization of the incomplete clustering* $\widehat{\mathfrak{C}}$*.*

Theorem 38 is proved by Proof 39 in Section A.5.

Following Theorem 38, the top-level of a probabilistic clustering $\mathfrak{C}$ with the range-disjoint factorization $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ can be computed by integrating the top-levels of its factors, i.e. $\widehat{\mathfrak{C}} = \boxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$.

A clustering alternative has the maximal number of clusters if it has the maximal number of clusters per factor set.

Theorem 39 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ, the set $\{\sigma_{maxCl}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ is a range-disjoint factorization of the incomplete clustering $\sigma_{maxCl}(\Gamma)$.*

Theorem 39 is proved by Proof 40 in Section A.5.

The same property holds for the selection predicate *minCl*.

Theorem 40 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ, the set $\{\sigma_{minCl}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ is a range-disjoint factorization of the incomplete clustering σ_{minCl}.*

Theorem 40 is proved by Proof 41 in Section A.5.

Following Theorem 39, a selection with the predicate *maxCl* can be pushed down to the factors. Thus, let Γ be an incomplete clustering with the factorization $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$, the incomplete clustering $\sigma_{maxCl}(\Gamma)$ can be computed by integrating the incomplete clusterings $\sigma_{maxCl}(\Gamma_{F_1})$ to $\sigma_{maxCl}(\Gamma_{F_k})$, i.e. $\sigma_{maxCl}(\Gamma) = \bigboxtimes_{i=1}^{k}\big(\sigma_{maxCl}(\Gamma_{F_i})\big)$. The same holds for a selection with the predicate *minCl* (Theorem 40).

By combining the insights that are provided by Theorem 38, Theorem 39, and Theorem 40, we can compute the candidate clusterings of an uncertain clustering based on its factors.

$$\sigma_{maxCl}(\widehat{\mathfrak{C}}) = \bigboxtimes_{i=1}^{k}\big(\sigma_{maxCl}(\widehat{\mathfrak{C}_{F_i}})\big) \quad \text{and} \quad \sigma_{minCl}(\widehat{\mathfrak{C}}) = \bigboxtimes_{i=1}^{k}\big(\sigma_{minCl}(\widehat{\mathfrak{C}_{F_i}})\big)$$
$$\text{or} \quad \sigma_{maxCl}(\Gamma) = \bigboxtimes_{i=1}^{k}\big(\sigma_{maxCl}(\Gamma_{F_i})\big) \quad \text{and} \quad \sigma_{minCl}(\Gamma) = \bigboxtimes_{i=1}^{k}\big(\sigma_{minCl}(\Gamma_{F_i})\big)$$

As a consequence, if the number of candidate clusterings is rather low (which should be the standard case), we can simply reconstruct the non-factorized representation of these candidates and then compute the decision correctness for each of them separately. Nevertheless, as we have shown in Section 8.8.5.4 we can directly compute the exact score or at least can directly compute a bound of the quality aggregation result based on the factors if aggregation is performed by a conventional aggregate function such as average (expected value), minimum, or maximum. Such a factor-based computation, however, is not possible if aggregation is performed by thresholding as we did it in Equation 8.18. In summary, even if the number of candidate clusterings is so large that a reconstruction of their non-factorized representation is infeasible, we can efficiently compute the decision correctness of an uncertain clustering under the Single World Semantics in the most cases.

Example 216 *For illustration we reconsider the sample clustering $\mathfrak{C}$ and its two factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$ from Figure 8.33. The top-levels as well as the incomplete clusterings that result from applying a selection with one of the two predicates maxCl or minCl to these clusterings (or its top-levels respectively) are presented in Table 8.12. Whereas, the top-level of $\mathfrak{C}_{F_1}$ contains the two alternatives $\mathcal{C}_{11}$ and $\mathcal{C}_{12}$, the top-level of $\mathfrak{C}_{F_2}$ has only one alternative, namely $\mathcal{C}_{21}$. The top-level of $\mathfrak{C}$ contains the two alternatives $\mathcal{C}_1$*

Clustering	$\widehat{\mathfrak{C}_i}$	$\sigma_{maxCl}(\mathfrak{C}_i)$	$\sigma_{minCl}(\mathfrak{C}_i)$	$\sigma_{maxCl}(\widehat{\mathfrak{C}_i})$	$\sigma_{minCl}(\widehat{\mathfrak{C}_i})$
$\mathfrak{C}$	$\{\mathcal{C}_1, \mathcal{C}_3\}$	$\{\mathcal{C}_3\}$	$\{\mathcal{C}_2\}$	$\{\mathcal{C}_3\}$	$\{\mathcal{C}_1\}$
$\mathfrak{C}_{F_1}$	$\{\mathcal{C}_{11}, \mathcal{C}_{12}\}$	$\{\mathcal{C}_{12}\}$	$\{\mathcal{C}_{11}\}$	$\{\mathcal{C}_{12}\}$	$\{\mathcal{C}_{11}\}$
$\mathfrak{C}_{F_2}$	$\{\mathcal{C}_{21}\}$	$\{\mathcal{C}_{21}\}$	$\{\mathcal{C}_{22}\}$	$\{\mathcal{C}_{21}\}$	$\{\mathcal{C}_{21}\}$

Table 8.12.: The topl-levels and some selection results for the sample clustering $\mathfrak{C}$ and its two factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$

and $\mathcal{C}_3$. Since $\mathcal{C}_1 = \mathcal{C}_{11} \boxtimes \mathcal{C}_{21}$ and $\mathcal{C}_3 = \mathcal{C}_{12} \boxtimes \mathcal{C}_{21}$, these two alternatives exactly result from integrating the top-level elements of the factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$. This demonstrates that the top-level of $\mathfrak{C}$ can be computed from the topl-levels of its factors.

The alternatives of $\mathfrak{C}$, $\mathfrak{C}_{F_1}$, and $\mathfrak{C}_{F_2}$ that have the most clusters are $\mathcal{C}_3$, $\mathcal{C}_{12}$, and $\mathcal{C}_{21}$ respectively. The fact that the alternative $\mathcal{C}_3$ results from integrating $\mathcal{C}_{12}$ and $\mathcal{C}_{21}$ illustrates that the selection operator can be pushed down to the factors.

8.8.5.6. Certain World Semantics

According to the definition of a factorization, the possible clusters of an uncertain clustering are always restricted to single factors, i.e. all entities that belong to different factor sets cannot belong to the same cluster in any possible clustering and hence are certain UNMATCHES. As a consequence, the set of certain MATCHES of an incomplete clustering Γ is equivalent to the union of the certain MATCHES of its factors. The same holds for the set of POSSIBLE MATCHES. The set of certain UNMATCHES results in the union of the certain UNMATCHES of the factors plus all entity pairs whose entities belong to different factor sets. Therefore M_c and P are $\cup$-decomposable functions and U_c is a quasi-$\cup$-decomposable function.

Theorem 41 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, it holds:*

$$\begin{aligned} M_c(\Gamma) &= \bigcup_{i=1}^{k} M_c(\Gamma_{F_i}) \\ P(\Gamma) &= \bigcup_{i=1}^{k} P(\Gamma_{F_i}) \\ U_c(\Gamma) &= \left(\bigcup_{i=1}^{k} U_c(\Gamma_{F_i})\right) \cup U(\mathcal{F}^{set}) \end{aligned}$$

Theorem 41 is proved by Proof 42 in Section A.5.

Example 217 *For illustration, we consider the sample factorization from Figure 8.33. The probabilistic clustering $\mathfrak{C}$ has two certain* MATCHES*, i.e. the entity pairs $\{e_1, e_2\}$ and $\{e_4, e_{10}\}$. Its set of* POSSIBLE MATCHES *contains the pairs $\{e_3, e_9\}$, $\{e_4, e_5\}$, $\{e_4, e_{11}\}$, $\{e_5, e_{10}\}$, $\{e_5, e_{11}\}$, $\{e_6, e_{12}\}$, $\{e_7, e_8\}$, and $\{e_{10}, e_{11}\}$. The factor $\mathfrak{C}_{F_1}$ has one certain Match, i.e. $\{e_1, e_2\}$, and has one* POSSIBLE MATCH*, i.e. $\{e_7, e_8\}$. In contrast, the factor $\mathfrak{C}_{F_2}$ has one certain Match, i.e. $\{e_4, e_{10}\}$, and has seven* POSSIBLE MATCHES*, i.e. $\{e_3, e_9\}$, $\{e_4, e_5\}$, $\{e_4, e_{11}\}$, $\{e_5, e_{10}\}$, $\{e_5, e_{11}\}$, $\{e_6, e_{12}\}$, and $\{e_{10}, e_{11}\}$. Thus, each certain* MATCH *of $\mathfrak{C}$ is either a certain* MATCH *of $\mathfrak{C}_{F_1}$ or is a certain* MATCH *of $\mathfrak{C}_{F_2}$. The same holds for the set of* POSSIBLE MATCHES.

Based on the given formulas, we can compute the quality of the certain decisions of an incomplete clustering Γ in two steps. First we compute all certain MATCHES and all POSSIBLE MATCHES in a factor-based fashion as described above. Second, we compute the final quality score based on these two sets as defined in Section 8.8.2.4. Of course, because the Number of Pair True Positives and the Number of Pair False Positives are Σ-decomposable quality measures, and because the Number of Pair False Negatives is a quasi-Σ-decomposable quality measure, we can improve quality computation further on by shifting the computation of these numbers into the factors.

Theorem 42 *Let Γ be an incomplete clustering, let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, and let $\mathcal{C}_{gold}$ be the corresponding gold standard, it holds:*

$$
\begin{aligned}
|p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{gold})| &= \sum_{i=1}^{k} |p\text{TP}^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))| \\
|p\text{FP}^{CWS}(\Gamma, \mathcal{C}_{gold})| &= \sum_{i=1}^{k} |p\text{FP}^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))| \\
|p\text{FN}^{CWS}(\Gamma, \mathcal{C}_{gold})| &= \left(\sum_{i=1}^{k} |p\text{FN}^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))|\right) + |p\text{FN}(\mathcal{F}^{set}, \mathcal{C}_{gold})|
\end{aligned}
$$

Theorem 42 is proved by Proof 43 in Section A.5.

Example 218 *As presented in Figure 8.33, the Number of Certain Pair True Positives of $\mathfrak{C}$ is exactly the sum of the Number of Certain Pair True Positives of $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$. The same holds for the Number of Certain Pair False Positives. In contrast, in the case of the Number of Certain Pair False Negatives, the Number of Pair False Negatives of the factor set clustering $\mathcal{F}^{set}$ additionally need to be added, i.e. $|p\text{FN}^{CWS}(\mathfrak{C}_{F1}, \pi_{F1}(\mathcal{C}_{gold}))| + |p\text{FN}^{CWS}(\mathfrak{C}_{F2}, \pi_{F2}(\mathcal{C}_{gold}))| + |p\text{FN}^{CWS}(\mathcal{F}^{set}, \mathcal{C}_{gold})| = 2 + 2 + 2 = 6 = |p\text{FN}^{CWS}(\mathfrak{C}, \mathcal{C}_{gold})|$.*

8.8.5.7. Factor-based Computation of Decision Certainty

As for decision correctness, it is desirable to compute the decision certainty of an uncertain clustering without reconstructing its non-factorized representation first. As we will show below, all the certainty measures that we have considered in Section 8.8.3 can be computed in a factor-based fashion.

Decision Completeness & Possible Match-Ratio The Decision Completeness and the Possible Match-Ratio are based on the sizes of the set of certain MATCHES, the set of certain UNMATCHES and the set of POSSIBLE MATCHES. As we have shown in Section 8.8.5.6, all three sets can be computed in a factor-based fashion. Moreover, since the number of certain UNMATCHES can be derived from the number of certain MATCHES and the number of POSSIBLE MATCHES, we only need to compute the latter two sets for computing the Decision Completeness and the Possible Match-Ratio based on the individual factors.

Number of Alternatives Let Γ be an incomplete clustering of a domain with n elements and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ. From each possible combination of one alternative per factor another alternative of Γ can be reconstructed. Therefore, the number of alternatives of Γ is a Π-decomposable measure and can be computed as:

$$|\Gamma| = \prod_{i=1}^{k} |\Gamma_{F_i}| \tag{8.64}$$

Consequently, the Ratio of Excluded Alternatives and the Inverse Number of Alternatives of Γ are quasi-Π-decomposable quality measures and can be computed as:

$$\mathit{RExA}(\Gamma) = \frac{B_n - \prod_{i=1}^{k} |\Gamma_{F_i}|}{B_n - 1} \tag{8.65}$$

$$\mathit{invNA}(\Gamma) = \frac{1}{\prod_{i=1}^{k} |\Gamma_{F_i}|} \tag{8.66}$$

Shannon Entropy Each factor of a factorization can be considered as a discrete random variable and the integration of all these factors, i.e. the non-factorized representation, can be considered as a discrete random variable that models all possible combinations of the factors' random variables and hence is a joint probability distribution on them. The joint entropy of a set of random variables $\{X_1, \ldots, X_k\}$ is defined as [Sin13]:

$$H(X_1, \ldots, X_k) = -\sum_{x_1 \in X_1} \ldots \sum_{x_k \in X_k} Pr(x_1, \ldots, x_k) \times \log_2(Pr(x_1, \ldots, x_k)) \tag{8.67}$$

Recall we consider only factorizations into absolutely independent factors. The joint entropy of k mutually independent random variables $\{X_1, \ldots, X_k\}$ is equal to the sum of their entropies [Sin13], i.e. $H(X_1, \ldots, X_k) = \sum_{i=1}^{k} H(X_i)$. Consequently, entropy is a Σ-decomposable quality measure and we can compute the entropy of a probabilistic clustering based on the entropies of its factors. Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering and let $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ be a factorization of $\mathfrak{C}$, the entropy of $\mathfrak{C}$ can computed as:

$$H(\mathfrak{C}) = \sum_{i=1}^{k} H(\mathfrak{C}_{F_i}) \tag{8.68}$$

Similar holds for the Inverse Normalized Entropy.

Theorem 43 *Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering, let N be the normalization factor of $\mathfrak{C}$, and let $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ be a factorization of $\mathfrak{C}$ where $\mathfrak{C}_{F_i} = (\Gamma_{F_i}, Pr_{F_i})$ and N_i is the normalization factor of $\mathfrak{C}_{F_i}$, the Inverse Normalized Entropy of $\mathfrak{C}$ can be computed as:*

$$\mathit{invH}_{\mathit{norm}}(\mathfrak{C}) = 1 - \sum_{i=1}^{k} \log_N(N_i) \times H_{\mathit{norm}}(\mathfrak{C}_{F_i})$$

Theorem 43 is proved by Proof 44 in Section A.5. Following this theorem, the Inverse Normalized Entropy is a quasi-Σ-decomposable quality measure.

As mentioned in Section 8.8.3.2 the commonly used normalization factor N of a clustering with n entities is the n-th bell number B_n. Nevertheless, for comparing clusterings with comparable factorizations, i.e. they are all based on the factor sets $\{F_1, \ldots, F_k\}$, using the normalization factor $N = \prod_{i=1}^{k} N_i = \prod_{i=1}^{k} B_{n_i}$ is also suitable. By setting $N = \prod_{i=1}^{k} B_{n_i}$ and setting $N_i = B_{n_i}, i \in \{1, \ldots, k\}$, the equation given by Theorem 43 can be further transformed into

$$\mathit{invH}_{\mathit{norm}}(\mathfrak{C}) = \sum_{i=1}^{k} \log_N(B_{n_i}) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})$$

because $\sum_{i=1}^{k} \log_N(B_{n_i}) = \log_N(\prod_{i=1}^{k} B_{n_i}) = \log_N(N) = 1$.

Certain Density / Answer Decisiveness If factors are considered as choice points, Certain Density and Answer Decisiveness are already defined in a factor-based way. Nevertheless, the Certain Density and the Answer Decisiveness of an uncertain clustering based on its non-factorized representation (a single choice point) can be computed in a factor-based fashion as well because we only require its Number of Alternatives and the maximal probability of all alternatives of the non-factorized representation that both can be computed based on the factors.

As presented above, the Number of Alternatives is a Π-decomposable quality measure and hence can be computed as defined in Equation 8.64. Since we only consider factorizations into absolutely independent factors, the maximal probability of all alternatives is Π-decomposable, too. Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering, let $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ be a range-disjoint factorization of $\mathfrak{C}$ where $\mathfrak{C}_{F_i} = (\Gamma_{F_i}, Pr_{F_i})$. The probability of an alternative $\mathcal{C} \in \Gamma$ is equivalent to multiplying the probabilities of all factor alternatives that need to be integrated in order to reconstruct $\mathcal{C}$, i.e. $Pr(\mathcal{C}) = \prod_{i=1}^{k} Pr_{F_i}(\pi_{F_i}(\mathcal{C}))$. Therefore, the maximal probability of all alternatives of Γ can be simply computed as:

$$\mathbf{max}_{\mathcal{C}\in\Gamma} Pr(\mathcal{C}) = \prod_{i=1}^{k} \mathbf{max}_{\mathcal{C}'\in\Gamma_{F_i}} Pr_{F_i}(\mathcal{C}') \tag{8.69}$$

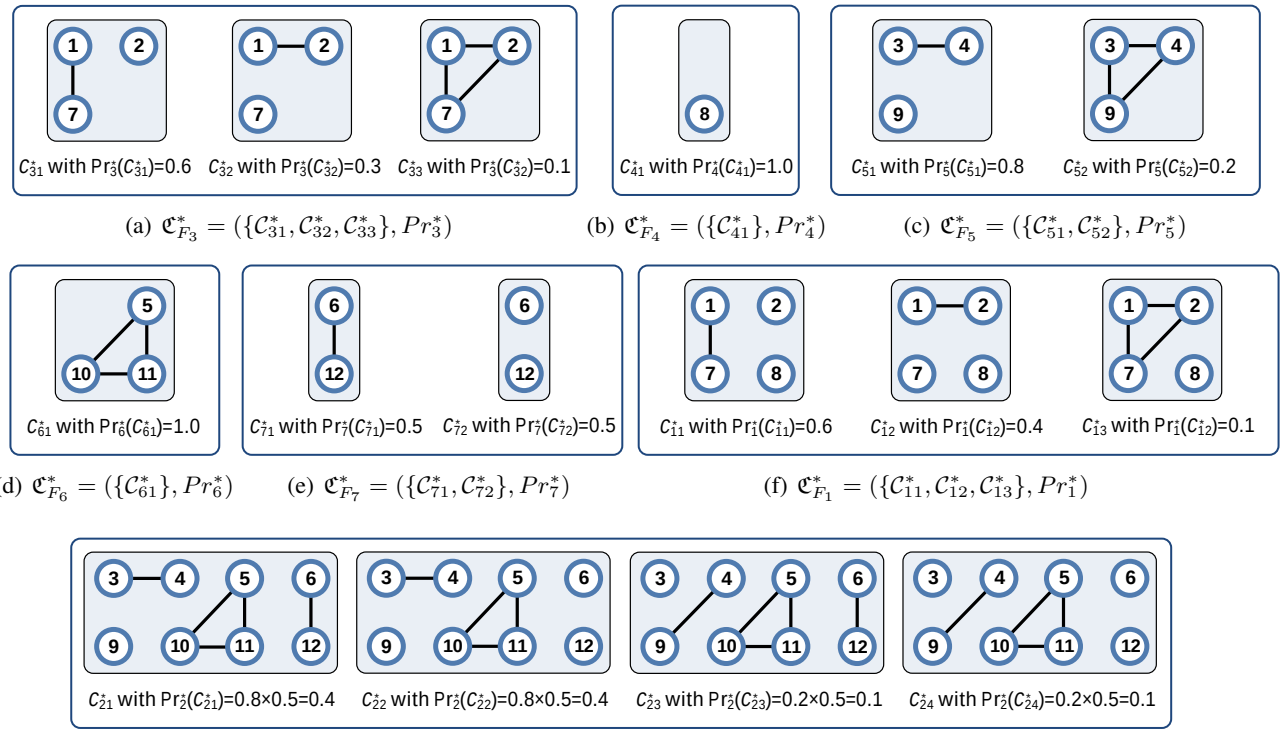
As a consequence, the Answer Decisiveness of $\mathfrak{C}$ with respect to its non-factorized representation can be computed as:

$$ADec_{\mathfrak{C}}(\{\mathfrak{C}\}) = \frac{\prod_{i=1}^{k} \mathbf{max}_{\mathcal{C}\in\Gamma_{F_i}} Pr_{F_i}(\mathcal{C})}{(2 - \prod_{i=1}^{k} \mathbf{max}_{\mathcal{C}'\in\Gamma_{F_i}} Pr_{F_i}(\mathcal{C}')) \times \log_2\left(\mathbf{max}(2, \prod_{i=1}^{k} |\Gamma_{F_i}|)\right)} \tag{8.70}$$

The same approach can be used if we need to compute the Answer Decisiveness with respect to a factorization that was transformed because the original factorization was not comparable to the factorization of another uncertain clustering (recall Section 8.8.3). This holds because each factor set of the original factorization is always a subset of one factor set of the transformed factorization (recall transformation is done by computing a coarser partition of the considered range), and therefore for each factor of the transformed factorization there is a set of factors of the original factorization that forms a range-disjoint factorization of it (see Theorem 25). As a consequence, we can compute the number of alternatives as well as the maximal probability of each factor of the transformed factorization based on the factors of the original factorization and hence do not need to perform the required transformation for real.

Example 219 *For illustration, we consider a probabilistic clustering $\mathfrak{C}^*$ that is given by a range-disjoint factorization $\mathcal{F}^*$ into the five factors $\mathfrak{C}^*_{F_3}$ to $\mathfrak{C}^*_{F_7}$ that are depicted in Figure 8.35(a) to Figure 8.35(e). The corresponding factor sets are $\mathcal{F}^{set*} = \{F_3, F_4, F_5, F_6, F_7\}$ with $F_3 = \{e_1, e_2, e_7\}$, $F_4 = \{e_8\}$, $F_5 = \{e_3, e_4, e_9\}$, $F_6 = \{e_5, e_{10}, e_{11}\}$, and $F_7 = \{e_6, e_{12}\}$.*

Now let us assume that we want to compare the certainty of this clustering with the certainty of the sample clustering $\mathfrak{C}$ from Figure 8.33(a) by computing the Answer Decisiveness of their factorized representations. Since both factorizations are defined on different factor sets they are not comparable. Nevertheless, because each factor set of $\mathcal{F}^{set*} = \{F_3, F_4, F_5, F_6, F_7\}$ is a subset of a factor set of $\mathcal{F}^{set} = \{F_1, F_2\}$, we can achieve comparability by transforming the factorization $\mathcal{F}^*$ into a factorization $\mathcal{F}^*_2$ with the factor sets $F_1 = \{e_1, e_2, e_7, e_8\}$ and $F_2 = \{e_3, e_4, e_5, e_6, e_9, e_{10}, e_{11}, e_{12}\}$. For that purpose we need to integrate the factors $\mathfrak{C}^*_{F_3}$ and $\mathfrak{C}^*_{F_4}$ (factor set F_1) as well as the factors $\mathfrak{C}^*_{F_3}$, $\mathfrak{C}^*_{F_5}$ and $\mathfrak{C}^*_{F_6}$



(a) $\mathfrak{C}^*_{F_3} = (\{C^*_{31}, C^*_{32}, C^*_{33}\}, Pr^*_3)$ (b) $\mathfrak{C}^*_{F_4} = (\{C^*_{41}\}, Pr^*_4)$ (c) $\mathfrak{C}^*_{F_5} = (\{C^*_{51}, C^*_{52}\}, Pr^*_5)$

(d) $\mathfrak{C}^*_{F_6} = (\{C^*_{61}\}, Pr^*_6)$ (e) $\mathfrak{C}^*_{F_7} = (\{C^*_{71}, C^*_{72}\}, Pr^*_7)$ (f) $\mathfrak{C}^*_{F_1} = (\{C^*_{11}, C^*_{12}, C^*_{13}\}, Pr^*_1)$

(g) $\mathfrak{C}^*_{F_2} = (\{C^*_{21}, C^*_{22}, C^*_{23}, C^*_{24}\}, Pr^*_2)$

Figure 8.35.: Sample for a factor-based computation of Answer Decisiveness

(factor set F_2). The resultant factors $\mathfrak{C}^*_{F_1} = \mathfrak{C}^*_{F_3} \boxtimes \mathfrak{C}^*_{F_4}$ and $\mathfrak{C}^*_{F_2} = (\mathfrak{C}^*_{F_5} \boxtimes \mathfrak{C}^*_{F_6}) \boxtimes \mathfrak{C}^*_{F_7}$ are presented in Figure 8.35(f) and Figure 8.35(g). Note that the set $\{\mathfrak{C}^*_{F_3}, \mathfrak{C}^*_{F_4}\}$ is a range-disjoint factorization of $\mathfrak{C}^*_{F_1}$ and that the set $\{\mathfrak{C}^*_{F_5}, \mathfrak{C}^*_{F_6}, \mathfrak{C}^*_{F_7}\}$ is a range-disjoint factorization of $\mathfrak{C}^*_{F_2}$.

These integrations, however, only need to be performed theoretically because we can compute all values of $\mathfrak{C}^*_{F_1}$ and $\mathfrak{C}^*_{F_2}$ directly from the original factors $\mathfrak{C}^*_{F_3}$ to $\mathfrak{C}^*_{F_7}$. The Number of Alternatives of $\mathfrak{C}^*_{F_1}$ is the Number of Alternatives of $\mathfrak{C}^*_{F_3}$ times the Number of Alternatives of $\mathfrak{C}^*_{F_4}$ and hence is three. Accordingly, the Number of Alternatives of $\mathfrak{C}^*_{F_2}$ can be computed by multiplying the Number of Alternatives of $\mathfrak{C}^*_{F_5}$, $\mathfrak{C}^*_{F_6}$, and $\mathfrak{C}^*_{F_7}$ and therefore is four. The maximal probability of $\mathfrak{C}^*_{F_1}$ is the maximal probability of $\mathfrak{C}^*_{F_3}$ times the maximal probability of $\mathfrak{C}^*_{F_4}$ and hence results in 0.6. Accordingly, the maximal probability of $\mathfrak{C}^*_{F_2}$ can be computed by multiplying the maximal probabilities of $\mathfrak{C}^*_{F_5}$, $\mathfrak{C}^*_{F_6}$, and $\mathfrak{C}^*_{F_7}$ and hence is 0.4. Since these four values are sufficient to compute the Answer Decisiveness of $\mathfrak{C}^*$ with respect to the factorization $\mathcal{F}^*_2$, we can compute this quality score based on the factors of the original factorization $\mathcal{F}^*$ and thus can compute it in an efficient way if all factors of $\mathcal{F}^*$ are of a manageable size. In contrast, the sizes of the factors of the transformed factorization $\mathcal{F}^*_2$ do not affect computation time.

Precision The Expected Pair Precision of a probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ can be considered as the expected similarity of two pairwise independent but data equivalent[11] probabilistic clusterings

[11] We consider two probabilistic values to be data equivalent if they have the same set of alternatives and ascribe the same probability mass to each of these alternatives. Note that two data equivalent probabilistic values do not need to describe the same uncertain event but can describe different independent events, i.e. the 'true' alternative of the first value does not need to depend on the 'true' alternative of the second value.

$\mathfrak{C}' = (\Gamma, Pr)$ and $\mathfrak{C}^* = (\Gamma, Pr)$, i.e. $\mathit{EPPrec}(\mathfrak{C}) = \mathit{EXP}(sim(\mathfrak{C}', \mathfrak{C}^*))$. Therefore, whether or not the Expected Pair Precision of a probabilistic clustering can be computed in a factor-based fashion depends on the used measure of similarity. If we use the Rand Index a factor-based computation is possible because the Rand Index is a quasi-Σ-decomposable quality measure (see Appendix A.6).

Theorem 44 *Let $\mathfrak{C}'$ and $\mathfrak{C}^*$ be two independent but data equivalent probabilistic clusterings of $n = |rng(\mathfrak{C}')| = |rng(\mathfrak{C}^*)|$ entities and let $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ be a range-disjoint factorization of both with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The expected Rand Index $\mathit{EXP}(\mathit{RI}(\mathfrak{C}', \mathfrak{C}^*))$ can be computed as:*

$$\mathit{EXP}(\mathit{RI}(\mathfrak{C}', \mathfrak{C}^*)) = \frac{1}{\binom{n}{2}} \times \Big(\sum\nolimits_{i=1}^{k} (\mathit{EXP}(|M(\mathfrak{C}'_{F_i}) \cap M(\mathfrak{C}^*_{F_i})|) + \mathit{EXP}(|U(\mathfrak{C}'_{F_i}) \cap U(\mathfrak{C}^*_{F_i})|)) + |U(\mathcal{F}^{set})| \Big)$$

Theorem 44 is proved by Proof 45 in Section A.5.

Finally, we consider the Average Decision Certainty. Due to all entity pairs whose entities belong to different factor sets are certain UNMATCHES, we can conclude the following theorem.

Theorem 45 *Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering of $n = |rng(\mathfrak{C})|$ entities and let $\mathcal{F}$ be a range-disjoint factorization of $\mathfrak{C}$ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Average Decision Certainty of $\mathfrak{C}$ can be computed as:*

$$\mathit{AvgDecC}(\mathfrak{C}) = \frac{1}{\binom{n}{2}} \times \left(\sum\nolimits_{i=1}^{k} \sum\nolimits_{\{e_r, e_s\} \in \mathit{Pairs}(\mathfrak{C}_{F_i})} \mathit{DecC}(\mathfrak{C}_{F_i}, \{e_r, e_s\}) - \sum\nolimits_{C \in \mathcal{F}^{set}} \binom{|C|}{2} \right) + 1$$

Theorem 45 is proved by Proof 46 in Section A.5. Following this theorem, the Average Decision Certainty is a quasi-Σ-decomposable quality measure and hence can be computed in a factor-based fashion.

Example 220 *For illustration, we reconsider the sample probabilistic clustering $\mathfrak{C}$ and its two factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$ from Figure 8.33(a). Factor $\mathfrak{C}_{F_1}$ has five decision of maximal certainty and one decision of zero certainty (entity pair $\{e_7, e_8\}$). Factor $\mathfrak{C}_{F_2}$ has twenty-one decisions of certainty 1.0 and has seven decisions of certainty 0.2. The number of inter-factor certain* UNMATCHES *is $66 - |\mathit{Pairs}(F_1)| - |\mathit{Pairs}(F_2)| = 66 - 6 - 28 = 32$. If we consider the non-factorized representation of $\mathfrak{C}$ we have the same seven decisions with certainty 0.2 and the same decision with certainty 0. Consequently, the average decision certainty of $\mathfrak{C}$ can be computed in a factor-based fashion and results in $(5 + 21 + 7 \times 0.2 + 32)/66 = 0.9$.*

Chapter 9

Experimental Evaluation

In this chapter, we present the results of some experimental evaluations that we conducted to prove the concepts of the approaches and methods that we have proposed in this thesis. For that purpose, we conducted an experiment for comparing the Probabilistic Monge-Elkan Similarity with the Earth Mover's Distance, conducted two experiments for proving the concepts of the description-based detection approach and the world-based detection approach respectively, conducted an experiment for evaluating the In-Depth Pruning technique, and conducted an experiment to prove the concept of indeterministic duplicate detection. We start with an initial remark in Section 9.1 and then continue with a presentation of the prototypical implementation of our duplicate elimination system called HaDES in Section 9.2. Thereafter we describe the characteristics of the used test databases in Section 9.3 and present the results of the conducted experiments in the remaining sections.

9.1. Initial Remark

A detection of duplicates is required in a variety of highly diverse application domains such as bioinformatics, forensics, risk management, or business intelligence. Moreover, in each of these application domains the considered databases can have different characteristics, e.g. large vs. small, dirty vs. clean, certain vs. uncertain. Finally, for each duplicate detection scenario another set of quality requirements can be given. As a consequence, for each combination of application domain, data characteristics, and quality requirements another process configurations can be valuable. For all these reasons, the experimental results that are provided in this chapter cannot be used to determine which of the resolution methods or which of the resolution moments generally perform best. Furthermore, these results are only marginally useful to conclude whether the description-based detection approach is superior to the world-based detection approach or vice versa. In contrast, we used these experimental evaluations to prove the individual concepts and hence to show that these concepts serve the purpose for which they have been designed. For space reasons, we restrict this chapter to the experiments that proves the main contributions of this thesis. Experimental results on some of the blocking methods that we have proposed in Section 7.4.4 can be found in [PWFR12]. Moreover, some additional experimental results on indeterministic duplicate detection can be found in [PvKR13] and [PR12].

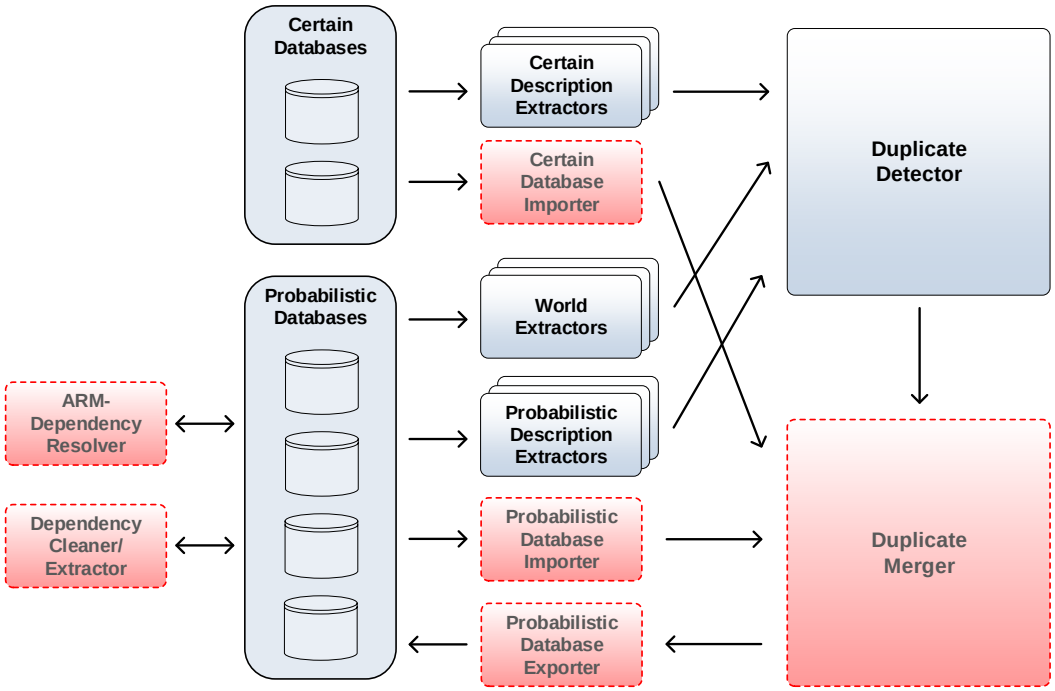


Figure 9.1.: Planned architecture of HaDES (red colored components are not implemented yet)

9.2. HaDES - A Prototypical Implementation

In order to evaluate the approaches and techniques that we have proposed in this thesis, we implemented a prototype that we call HAmburg Duplicate Elimination System (short HaDES). HaDES is implemented in Java and its planned architecture is depicted in Figure 9.1. The input to the system is either a certain database or a probabilistic database. If the input database is probabilistic and if it contains incorrect ARM-dependencies and/or if it contains entity dependencies an ARM-Dependency Resolver and a Dependency Cleaner/Extractor prepare the database before the actual duplicate elimination process starts. Duplicate detection is performed by comparing entity descriptions. For that reason, we implemented a variety of extractors that extracts such descriptions from the input database. The extracted descriptions are passed to the Duplicate Detector that either performs the world-based detection approach or performs the description-based detection approach. The result of the Duplicate Detector is either a certain clustering (deterministic duplicate detection) or a probabilistic clustering (indeterministic duplicate detection) and serves as input to the Duplicate Merger that consolidates the entities of each duplicate cluster. Since the Duplicate Merger works on the original database entities and not on their descriptions, a Database Importer (certain or probabilistic) fetches the database entities from the input database and provides it to the Duplicate Merger. After the merging process is completed, a Probabilistic Database Exporter exports the produced database entities into a new instance of the considered database system. Of course, if HaDES is used to perform a deterministic duplicate elimination process on a certain database and if it does not incorporate uncertainty into the elimination result, the elimination output can be a certain database as well.

In this thesis, we focus on duplicate detection. For that reason we neither implemented the Duplicate Merger nor the Database Importer nor the Database Exporter so far, but consider it as a major goal of future research (see Section 11.2.10). Moreover, the experiments that are presented in the remainder of this chapter only consider a single certain database table or consider a single BID-table without incorrect ARM-dependencies as input. Therefore, the ARM-Dependency Resolver and the Dependency Cleaner/Extractor are currently not implemented and we consider such an implementation as future work as well.

9.2.1. Description Extractors

Since we only consider single database tables (certain tables or BID-tables) as experimental input, the implemented extractors only extract attribute descriptions, but do not extract relationship descriptions (see Section 5.1.2). Moreover, they are only designed to extract attribute descriptions from single tables and therefore do not consider inheritance hierarchies (see Section 5.3) within the extraction process.

Obviously, the extraction process depends on the given database system and depends on the used detection approach. For the latter reason, we distinguish between the three different types of extractors that are described below. Due to the former reason, for each type several concrete instances can be implemented.

- **Certain Description Extractor:** Indeterministic duplicate detection can be performed on a certain database (or certain data file respectively). Therefore, the currently implemented prototype contains two Certain Description Extractors: One for HSQL-databases and one for CSV-files. Of course, HaDES has been designed in a generic way and therefore can be simply extended by further Certain Description Extractors that are defined on other formats.

- **World Extractor:** In the world-based detection approach, we perform duplicate detection on a set of worlds. For that purpose, we need a World Extractor that generates a set of worlds from the input database and then extracts a certain description per database entity from each of these worlds. For our experiments we implement two World Extractors that are both defined on HSQL-databases. The first World Extractor generates a set of worlds by performing a Monte-Carlo Simulation. The second World Extractor performs the dissimilar worlds selection strategy that we have proposed for blocking purposes in Section 7.4.4.1.

- **Probabilistic Description Extractor:** In the description-based detection approach, we match database entities by matching their probabilistic descriptions. As a consequence, we require an extractor that realizes the extraction approach that we have described in Section 7.4.2. In HaDES we implement such a Probabilistic Description Extractor for HSQL-databases. Since the input is a single BID-database (each database entity corresponds to an x-tuple) and because we use all attributes for duplicate detection, this extractor simply generates one description alternatives per x-tuple alternative.

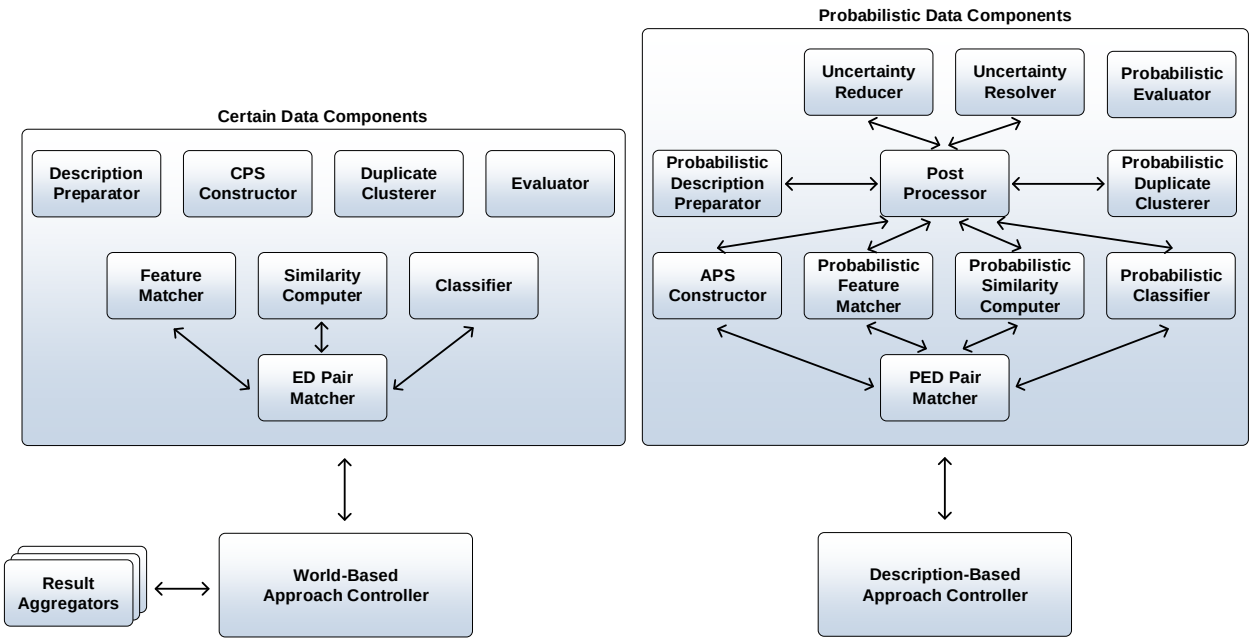


Figure 9.2.: Architecture of the Duplicate Detector

9.2.2. Duplicate Detector

The architecture of the Duplicate Detector is depicted in Figure 9.2. The Duplicate Detector contains two controller components: One for the world-based detection approach and one for the description-based detection approach. Whereas the first controller almost only interacts with components that are required for duplicate detection in certain databases (the only exceptions are the Result Aggregators), the second controller interacts with components that have been particularly conceptualized for working on probabilistic entity descriptions.

9.2.2.1. Certain Data Components

The certain data components correspond to the phases of a conventional duplicate detection process. For each component we implemented a set of methods that the component can use to accomplish its task.

- **Description Preparator:** In the case of the Description Preparator we only implemented two simple preparation activities that convert a data value into lower case or upper case respectively. More complex preparation activities like a detection of outliers or a splitting/merging of attributes has currently not been implemented. Nevertheless, because of the generic implementation of the individual components and because of the modularity of the Detector's architecture such activities can be subsequently added easily.

- **CPS Constructor:** For the CPS Constructor we implemented several blocking methods including Standard Blocking, the Sorted Neighborhood Method, and an extended variant of the Suffix-Array based Blocking [dVKCC11]. In contrast to the other detection phases, the CPS Constructor is used for certain entity descriptions as well as probabilistic entity descriptions. In the probabilistic

case, the blocking method can be combined with one of the certain key-based approaches Key-per-Entity, Key-per-Alternative, or Key-per-Representative that we have described in Section 7.4.4.

- **Entity Description Pair Matcher:** The Entity Description Pair Matcher (short ED Pair Matcher) performs a pairwise matching process according to HaDDeF (see Section 5.2) and therefore controls a subsequent execution of the Feature Matcher, the Similarity Computer, and the Classifier.
- **Feature Matcher:** The Feature Matcher computes a feature score from the given entity descriptions by pairwise comparing attribute values or related entity sets[1] respectively. For comparing purposes we provide a variety of similarity measures and impact measures for several domains.
- **Similarity Computer:** The Similarity Computer gets the feature score as input and computes a similarity-impact pair as output. For computing entity similarity we implemented the two distance-based decision models that we have presented in Section 4.3.6.1. Whereas the first computes the weighted average similarity of all compared features, the second computes a vector-based similarity. Rule-based decision models, probabilistic decision models, learning-based decision models, and clustering-based decision models have not been implemented so far, but can be incorporated into HaDES with only less additional effort.
- **Classifier:** The Classifier performs a simple thresholding to convert a similarity score into a duplicate decision.
- **Duplicate Clusterer:** The Duplicate Clusterer gets all pairwise duplicate decisions as input and computes a duplicate clustering as output. For that purpose we implemented several clustering approaches including Partitioning based on Connected Components, Partitioning based on Centers, some variants of Partitioning based on Merge-Centers [HCML09, NH10], and Markov Clustering [vD09].
- **Evaluator:** The Evaluator provides several methods in order to evaluate the quality of the candidate pair spaces, the pairwise matching results, and the final duplicate clusterings. Up until now we implemented the Number of Pair True Positives, the Number of Pair False Positives, the Number of Pair False Negatives, the Pair Recall, the Pair Precision, and the Pair F_1-score.

9.2.2.2. World-Based Approach Controller

Recall from Section 7.3 that the world-based detection approach performs duplicate detection until a particular moment in each of the generated worlds separately, then aggregates the worlds' detection results, and finally performs the remaining detection phases on the aggregation output. Thus, all detection phases are performed on certain input data and the World-Based Approach Controller utilizes the certain data components to accomplish its goal. The only exceptions are the Result Aggregators that are used to resolve data uncertainty by consolidating the detection outputs of the individual worlds. For our experiments, we implemented several Result Aggregators per moment at which consolidation can be performed. Among these aggregators are one that computes the most probable input value, one that

[1] Notice that although we currently have not implemented an extractor that extracts relationship information, the Feature Matcher is able to process relationship information in the form of entity sets.

aggregates a set of feature scores by selecting the maximal value per vector position, one that aggregates a set of similarity scores by computing the expected similarity value and the expected impact value, one that computes the most probable matching class, one that aggregates a set of clusterings by computing the clustering that has the minimal/maximal number of clusters, and one that selects the clustering that has the maximal similarity to all other input clusterings. Of course, additional Result Aggregators can be incorporated at discretion.

9.2.2.3. Probabilistic Data Components

As described in Section 7.4 we match probabilistic entity descriptions by reusing concepts for matching certain entity descriptions. Thus, the most of the probabilistic data components represent a probabilistic version of a certain data component and therefore extend them in two ways. (i) they enable a processing of probabilistic input values according to the Uncertain Value Theory (see Section 7.4.6), and (ii) they allow an incorporation of process uncertainty into the detection result by executing several certain data implementations (see Section 7.8). In addition to these components, we require some specific components that can be used to manipulate the probabilistic outputs of the individual detection phases.

- **Uncertainty Resolver:** The Uncertainty Resolver transforms a probabilistic value into a certain value by utilizing one of the provided Result Aggregators. Among the implemented resolution methods are the expected similarity score, the maximal-θ aggregation, and the Probabilistic Monge-Elkan Similarity.

- **Uncertainty Reducer:** The task of this component is to reduce the uncertainty of a probabilistic value by decreasing its number of alternatives. For uncertainty reduction we implemented a deciding method (see Section 7.6.2.2) that selects the k most probable alternatives and discards the remaining ones. A mediating reduction method has not been implemented so far, but can be subsequently incorporated into HaDES with only less effort.

- **Post Processor:** The job of the Post Processor is to apply a sequence of post processes to the outputs of the individual detection phases. The most important post processes are the resolution and the reduction of data uncertainty. For this purpose, it either invokes the Uncertainty Resolver or it invokes the Uncertainty Reducer. Of course, in the pairwise matching phases, the Post Processor works on matching cubes instead of probabilistic values and therefore can execute a sequence of resolution/reduction processes that each is applied to some dimensions of the considered matching cube. In our current implementation, we restrict post processing to a resolution/reduction of single dimensions or whole cubes.

- **Probabilistic Description Preparator:** The Probabilistic Description Preparator is the probabilistic equivalent of the conventional Description Preparator. It gets a set of probabilistic entity descriptions as input and produces a set of probabilistic entity descriptions as output by performing a sequence of preparation activities on each of the description alternatives separately. For this purpose, the Probabilistic Description Preparator utilizes the conventional Description Preparator. Since the Probabilistic Description Preparator can introduce preparation uncertainty, for each input description alternative several output description alternatives can be produced. After preparation,

the uncertainty of the prepared entity descriptions can be resolved or can be reduced by utilizing the Post Processor.

- **Probabilistic Entity Description Pair Matcher:** The Probabilistic Entity Description Pair Matcher (short PED Pair Matcher) controls a subsequent execution of the APS Constructor, the Probabilistic Feature Matcher, the Probabilistic Similarity Computer, and the Probabilistic Classifier. Recall that the Duplicate Clusterer cannot deal with probabilistic input values or matching cubes. For that reason, the result of the PED Pair Matcher is always a certain duplicate decision and we need to perform uncertainty resolution at any moment of the pairwise matching process if the input are probabilistic entity descriptions (i.e. if uncertainty has not been resolved at an earlier moment). For reducing run time, the matching process can be performed iteratively. For our experiment, we exemplary implemented In-Depth Pruning for the expected entity similarity and the Probabilistic Monge-Elkan Similarity.
- **APS Constructor:** The task of the APS Constructor is to construct the alternative description pair space. Since we only consider entity independent databases as experimental input, the implemented APS Constructor always computes the naive pair space.
- **Probabilistic Feature Matcher:** The Probabilistic Feature Matcher utilizes the conventional Feature Matcher for computing the feature scores of the alternative description pairs. Because we consider an incorporation of process uncertainty, the result of the Feature Matcher is a feature cube instead of a probabilistic feature score. Finally, the Probabilistic Feature Matcher utilizes the Post Processor in order to resolve and/or to reduce some dimensions of the feature cube.
- **Probabilistic Similarity Computer:** The Probabilistic Similarity Computer gets a feature cube as input and produces a similarity cube as output. For computing the individual similarity scores of the output cube, the Probabilistic Similarity Computer utilizes the conventional Similarity Computer. Like the Probabilistic Feature Matcher, the Probabilistic Similarity Computer can finally utilize the Post Processor to resolve or to reduce some dimensions of the produced similarity cube.
- **Probabilistic Classifier:** The Probabilistic Classifier gets a similarity cube as input and computes a decision cube as output. For computing the individual duplicate decisions of the decision cube, the Probabilistic Classifier utilizes the conventional Classifier. Since the result of the Probabilistic Classifier must be a certain duplicate decision, the produced decision cube is finally resolved by utilizing the Post Processor.
- **Probabilistic Duplicate Clusterer:** The Probabilistic Duplicate Clusterer gets a set of pairwise duplicate decisions as input and either computes a single duplicate clustering (deterministic approach) or computes a probabilistic clustering (indeterministic approach). The first is accomplished by conventional clustering approaches and in this case the Probabilistic Duplicate Clusterer utilizes the conventional Duplicate Clusterer. For the latter we implemented the clustering approach that we have proposed in Section 8.7. Of course, a probabilistic clustering can be also achieved by performing several conventional clustering approaches separately and thus by incorporating clustering uncertainty into the detection result. In this case, the Probabilistic Duplicate Clusterer utilizes the conventional Duplicate Clusterer for several times under different settings.

- **Probabilistic Evaluator:** As its certain equivalent, the Probabilistic Evaluator evaluates the quality of the candidate pair spaces, the pairwise matching results, and the final duplicate clusterings. Since we also consider probabilistic outcomes we implemented several measures of the four quality semantics that we have proposed in Section 8.8.

9.2.2.4. Description-Based Approach Controller

The Description-Based Approach Controller gets the extracted probabilistic entity descriptions as input and computes a duplicate clustering (certain or probabilistic) as output. In the first step, the controller optionally reduces or resolves the uncertainty of the probabilistic entity descriptions by invoking the Post Processor. For instance resolving uncertainty by selecting the most probable alternative per description corresponds to processing only the most probable world of the input database if the used description extractor produces another description alternative for each possible instance of one entity. Then it utilizes the Probabilistic Description Preparator to prepare the probabilistic entity descriptions and utilizes the CPS Constructor to compute the candidate pair space. In the next step, the controller invokes the PED Pair Matcher per candidate pair. The resultant duplicate decision are then pushed to the Probabilistic Duplicate Clusterer that computes a certain clustering or a factorized probabilistic clustering respectively. For validation purposes the controller can invoke the Probabilistic Evaluator after each phase.

9.2.2.5. Implementation Details

As described above, we did not implement all the approaches and techniques that we have described in this thesis. Table 9.1 gives an overview which of these approaches and techniques are actually implemented in the version of HaDES that we used for conducting our experimental evaluation.

9.3. Probabilistic Test Databases

In the three of the five following experiments we need to process probabilistic databases. However, large unclean probabilistic real-life databases that are labeled, i.e. each true duplicate pair is exactly known, were not accessible to us. For that reason, we produced some synthetic databases. For data generation, we used the probabilistic data generator *ProbDataGen* that has been developed by Wingerath and Friedrich [WF10]. To make the generated databases as realistic as possible, *ProbDataGen* uses real-life data from an existing certain database. For that purpose Wingerath and Friedrich extracted data values of about $300,000$ movies from the online movie database IMDb[2] with the Java application JMDb[3] and stored the data to an HSQLDB[4]. With *ProbDataGen* it is possible to choose among several HSQL databases holding certain movie data to generate a probabilistic movie database with duplicates, where the user can make several adjustments, e.g. the number of duplicates, the maximal number of alternatives per x-tuple, or the data's degree of dirtiness where we measure the dirtiness of a database by the average expected Levenshtein Similarity between the true duplicates. Obviously, the lower the expected similarity between

[2]The Internet Movie Database (http://www.imdb.com)
[3]Java Movie Database (http://www.jmdb.de)
[4]HyperSQL DataBase (http://hsqldb.org)

HaDDeF	
Extraction of Attribute Descriptions	YES
Extraction of Relationship Descriptions	NO
Impact Values	YES
Belief Map	YES
Multi-Table Memberships	NO

Aggregation Methods	
Aggregation of Possible Worlds	NO
Aggregation of Entity Descriptions	NO
Aggregation of Feature Scores	YES
Aggregation of Similarity Scores	YES
Aggregation of Duplicate Decisions	YES
Aggregation of Duplicate Clusterings	YES

Deterministic Detection Approaches	
World-based Detection Approach	YES
Description-based Detection Approach	YES
Certain Key-based Blocking	YES
Probabilistic Key-based Blocking	NO

Detection of Scattered Duplicates	
Probabilistic Monge-Elkan Similarity	YES

Matching Cost Optimization	
Hard Attribute Match Pruning	YES
Hard Alternative Match Pruning	YES
Alternative Pair Space Reduction	NO
Soft Alternative Match Pruning	YES
Hybrid Alternative Match Pruning	YES

Probabilistic Data Preparation	
ARM-dependency Resolution	NO
Entity Dependency Cleaning/Extraction	NO

Incorporation of Process Uncertainty	
Matching Cubes	YES
Resolution of Cube Dimensions	YES
Hard Alternative Match Pruning	YES
Soft Alternative Match Pruning	YES
Hybrid Alternative Match Pruning	YES

Indeterministic Deduplication	
Databases Exportation	NO
Rewriting of Aggregate Queries	NO
Duplicate Clustering with Uncertainties	YES
Quality Evaluation Measures	YES

Table 9.1.: Overview which of the proposed approaches and techniques are implemented in HaDES

the true duplicates the dirtier is the corresponding database and the more difficult is a detection of all true duplicates.

ProbDataGen is restricted to a generation of BID-tables. For that reason, each database entity is represented by an x-tuple and each of the entity's possible instances corresponds to a single x-tuple alternative. Moreover it does not contain entity correlations. The schema of the generated movie BID-table consists of four attributes. The first attribute *'title'* describes the movie titles where each title usually consists of several words. The second attribute *'studio'* describes the names of the studios that produced the corresponding movies. The third attribute *'director'* describes the directors of the movies and is therefore a name attribute whose values usually consists of a first name and a last name. The last attribute *'year'* describes the production year of the movies and is typically a numerical value. In the absence of relationship information, the description of an entity corresponds to its attribute description, i.e. the relationship description of each database entity is empty. Since each of the four attributes can be useful for detecting duplicate x-tuples, we consider x-tuples and probabilistic entity descriptions to be synonymously within the context of our experiment. Moreover, because none of the four attributes incorrectly depends on the membership to the considered movie table, the database schema does not contain incorrect ARM-dependencies.

database	N_X	N_A	N_A^{max}	N_A^{avg}	N_{Dup}	sim_{Dup}^{exp}	sim_{Dup}^{max}
$db_{A_1}^{10} - db_{A_5}^{10}$	$102,692$	$558,947 - 561,403$	10	$5.44 - 5.47$	$4,380$	$0.715 - 0.718$	$0.809 - 0.812$
$db_{B_1}^{10} - db_{B_5}^{10}$	$102,692$	$558,598 - 561,783$	10	$5.44 - 5.47$	$4,380$	$0.768 - 0.772$	$0.858 - 0.863$
$db_{C_1}^{10} - db_{C_5}^{10}$	$102,692$	$559,617 - 560,996$	10	$5.45 - 5.46$	$4,380$	$0.803 - 0.809$	$0.885 - 0.889$
$db_{D_1}^{10} - db_{D_5}^{10}$	$102,692$	$559,685 - 561,467$	10	$5.45 - 5.47$	$4,380$	$0.859 - 0.863$	$0.947 - 0.954$
$db_{E_1}^{10} - db_{E_5}^{10}$	$102,692$	$558,578 - 562,170$	10	$5.44 - 5.47$	$4,380$	$0.906 - 0.908$	0.994
$db_{F_1}^{10} - db_{F_5}^{10}$	$102,692$	$559,334 - 560,478$	10	$5.45 - 5.46$	$4,380$	$0.949 - 0.951$	1.0
$db_{D_{small}}^{10}$	$10,266$	$56,649$	10	5.52	412	0.867	0.936
$db_{D_{small}}^{50}$	$10,266$	$259,564$	50	25.28	412	0.818	0.951
$db_{D_{small}}^{100}$	$10,266$	$521,796$	100	50.83	412	0.792	0.954
$db_{D_{small}}^{150}$	$10,266$	$773,491$	150	75.35	412	0.773	0.956

Table 9.2.: Characteristics of the generated test databases

To improve the reliability of our experimental results further on, we used a standard data setting for generating the individual test databases. The characteristics of this standard setting are adopted from the characteristics of a real-life CD-dataset[5] where the true duplicates are exactly known. We adjusted the percentage of duplicates, the average duplicate cluster size, and the average similarity between the true duplicates of the generated databases to the corresponding values of this real-life data set. In most experiments we need to vary a particular data characteristic. For that reason, some of the generated test databases differ from this standard setting, but each of them differ in at most one characteristic. In each experiment, we used the test databases that completely agree to the standard setting as a fixed point and then vary a particular characteristic by using the corresponding test databases that differ from the standard setting in exactly this characteristic. We think, these adjustments made our experiments quite realistic, even though synthetic databases are used.

In order to minimize the influence of randomness on the experimental results we generated five databases for the most of the considered database characteristics and then always computed the average result from these five databases The only exception are the databases that we generated for evaluating detection quality under a varying number of x-tuple alternatives because processing a database where each x-tuple has hundred or more alternatives is extremely time consuming if no match cost optimization technique is used.

The generated databases as well as their characteristics are listed in Table 9.2 where N_X refers to the total number of x-tuples (and thus database entities), N_A refers to the total number of x-tuple alternatives, N_A^{max} refers to the maximal number of alternatives per x-tuple, N_A^{avg} refers to the average number of alternatives per x-tuple, N_{Dup} refers to the total number of true duplicates, sim_{Dup}^{exp} refers to the average expected Levenshtein Similarity between all true duplicates, and sim_{Dup}^{max} refers to the average maximal Levenshtein Similarity between all true duplicates. A database name $db_{Z_*}^n$ indicates that the database's dirtiness is of level Z and each of the database's x-tuples has at most n alternatives. The dirtiness level D corresponds to the standard setting that we have mentioned above.

[5] http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cd_datasets.html

The first six database characteristics $db^{10}_{A_*}$ to $db^{10}_{F_*}$ agree in their number of x-tuple alternatives, but disagree in the similarity between their true duplicates. Whereas the database characteristic $db^{10}_{A_*}$ is the dirtiest one, the database characteristic $db^{10}_{F_*}$ is the cleanest one.

The last four databases $db^{10}_{D_{small}}$, $db^{50}_{D_{small}}$, $db^{100}_{D_{small}}$, and $db^{150}_{D_{small}}$ were all generated with the same setting for data dirtiness that we have used to generated the five databases $db^{10}_{D_1}$ to $db^{10}_{D_5}$, but are each generated with a different setting for the maximal number of alternatives per x-tuple. As a consequence, these databases largely agree in their dirtiness[6], but disagree in their uncertainty if we measure the uncertainty of a database by its number of possible worlds (recall that in a BID-database the number of possible worlds is computed by multiplying the number of alternatives per x-tuple).

9.4. Experiment: Probabilistic Monge-Elkan Similarity

The goal of this experiment is to compare the Probabilistic Monge-Elkan Similarity proposed in Section 7.5.4 with the Earth Mover's Distance that we have presented in Section 7.5.3.1.

9.4.1. Set-Up

In this experiment we aimed to compare the Probabilistic Monge-Elkan Similarity with the Earth Mover's Distance. However, while the first is a similarity measure, the second is a distance measure. For that reason, we used the Probabilistic Monge-Elkan Distance $\textit{PME-Dst}(x, y) = 1 - \textit{PME-Sim}(x, y)$ instead of the Probabilistic Monge-Elkan Similarity throughout this experiment. Since the Earth Mover's Distance is symmetric and the Probabilistic Monge-Elkan Distance is not, we considered four variants of the latter. The first variant only computes the distance in one direction and hence is asymmetric. We denote this variant by the abbreviation $\textit{PME-Dst}_{\text{asym}}$. The three other variants compute the distance in both directions and then use the minimal directed distance (variant $\textit{PME-Dst}_{\text{min}}$), the maximal directed distance (variant $\textit{PME-Dst}_{\text{max}}$), or the average directed distance (variant $\textit{PME-Dst}_{\text{avg}}$) as final output value.

We implemented all these four variants in Java and compared them with an implementation of the Earth Mover's Distance that is provided by the open available Java library *JFastEMD*[7] and is based on the algorithm by Pele and Werman [PW09] which has been proven to be one of the fastest algorithms for computing the Earth Mover's Distance.

This experiment was performed on a machine with an Intel Core i5-3317U Processor, 8GB main memory, and a 64-bit version of the operating system *Windows 7*.

- **Evaluation Measures:** We compared the four variants of the Probabilistic Monge-Elkan Distance with the Earth Mover's Distance with respect to two different criteria. The first criterion was the runtime because we expected that the Probabilistic Monge-Elkan Distance is much faster to compute than the Earth Mover's Distance. The second criterion was the closeness between the scores that are computed by both measures. To quantify this closeness we (i) computed the average Deviation between the corresponding scores and (ii) computed a Correlation Coefficient between both

[6] Note that although we used the same generation setting, these five databases do not completely agree in their dirtiness because the average expected duplicate similarity decreases with a growing number of alternatives per x-tuple. The average maximal duplicate similarity, however, is highly similar for all of them.

[7] https://github.com/telmomenezes/JFastEMD

measures. For the purpose of the latter we used the Correlation Coefficient that has been described by Cha [Cha07] and is defined as follows: Given a set of 'reference' *pmf*s $R = \{r_1, \ldots, r_k\}$ and a single 'query' *pmf* q, the Correlation Coefficient quantifies the semantical correlation between two distance measures d_x and d_y as:

$$cCoeff(d_x, d_y) = \frac{\sum_{i=1}^{|R|}(d_x(r_i, q) - \overline{d_x}) \cdot (d_y(r_i, q) - \overline{d_y})}{\sqrt{\sum_{i=1}^{|R|}(d_x(r_i, q) - \overline{d_x})^2 \cdot \sum_{i=1}^{|R|}(d_y(r_i, q) - \overline{d_y})^2}} \tag{9.1}$$

where $\overline{d_x} = \frac{\sum_{i=1}^{|R|} d_x(r_i,q)}{|R|}$.

The Correlation Coefficient approaches 1 if the compared distance measures are semantically similar and is close to 0 if the semantics of these measures are highly dissimilar.

- **Probability Mass Functions:** To compare both distance measures under different circumstances we generated a synthetic set of *pmf*s that are each defined on a numerical domain. For evaluating the interdependencies between the characteristics of the input data and the results of the individual evaluation measures, we generated *pmf*s with different domain sizes following different probability distributions. As domain sizes we choose 5, 10, 25, 50, 75, and 100. For probability computation, we utilized the Zipf distribution and the Binomial distribution because they are commonly used in many application domains.

  The Zipf distribution is defined as

$$P(x) = \frac{x^{-(\rho+1)}}{\zeta(\rho+1)} \tag{9.2}$$

  and has a single parameter ρ that is a positive real number[8]. Based on the Zipf distribution we generated 51 *pmf*s by constantly increasing the parameter ρ from 0 (almost a uniform distribution)[9] to 5 (almost a certain value) by intervals of 0.1.

  The Binomial distribution is defined as

$$P(x) = \binom{n}{x} \times p^x \times (1-p)^{n-x} \tag{9.3}$$

  and has the two parameters n and p where n models the number of trials of independent yes/no experiments and p models the probability that a particular trial is successful. Based on the Binomial distribution we generated 36 *pmf*s where n was always set to the size of the considered domain. In contrast to n, every *pmf* was generated by using another value for the parameter p. For this purpose, we used a selection of values[10] that we considered to be most meaningful because they implicated probability distributions that are more or less different. The smallest selected value was 0.5 (the broadest bell-shaped curve possible) and the largest selected value was 1.0 (a certain value).

[8] Note that in contrast to a commonly used definition of the semantically equivalent Zeta distribution the parameter ρ is not restricted to values greater than one.

[9] Actually we use 0.01 instead of 0 because the Zipf distribution is undefined for $\rho = 0$.

[10] For the sake of completeness the exact set of selected values is: {0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 0.991, 0.992, 0.993, 0.994, 0.995, 0.996, 0.997, 0.998, 0.999, 1.0}.

To ensure that the most probable domain elements are different in the majority of the generated *pmf*s even if the underlying generation settings were quite similar, we introduced a *pmf*-specific offset that finally shifted the generated probabilities from small numbers to large numbers where an overflow guaranteed that the considered domain remained unchanged. The offset was computed for each of the generated *pmf*s randomly. Note that using this shift did not mean that all the generated *pmf*s were automatically highly dissimilar because the finally used distance between two domain elements did not correspond to the numerical distance of these values, but was separately defined by a ground distance matrix (see below).

- **Ground Distance Matrices:** To simulate the distance between the individual domain elements, we generated ten ground distance matrices by random. The Earth Mover's Distance is only symmetric if the used ground distance matrix is symmetric. For that reason, we only considered symmetric distance matrices in this experiment.

9.4.2. Execution

Since comparing *pmf*s that are defined on different domains is meaningless, we paired each of the 87 *pmf*s per domain size and matched these pairs by using the Earth Mover's Distance as well as by using the different variants of the Probabilistic Monge-Elkan Distance.

In order to investigate the interdependencies between the used distribution types and the evaluation results we formed four groups of *pmf* pairs per domain size. The first group contained all pairs of *pmf*s, the second group contained all pairs of Zipf distributions, the third group contained all pairs of Binomial distributions, and the last group contained all pairs where one *pmf* is a Zipf distribution and the other *pmf* is a Binomial distribution (we refer to this group as 'mixed' in all figures and tables). As the *pmf*s of a particular group we consider all *pmf*s that belong to at least one pair of this group.

The runtime of the compared distance measures was evaluated once for each of these groups. The closeness of the matching results was then evaluated per group in two ways. First by computing the average Deviation of the distance values that were computed for all *pmf* pairs of the considered group and second by computing the average Correlation Coefficient of this group.

For each group we computed one Correlation Coefficient per *pmf* that belongs to this group. In this case, we selected the considered *pmf*s as 'query' *pmf* and selected a particular subset of the group's *pmf*s as 'reference' *pmf*s. In the case of the first three groups each of the group's *pmf*s was considered as 'reference' *pmf*[11]. In contrast, in the case of the 'mixed' group we selected all Zipf distributions as 'reference' *pmf*s if the 'query' *pmf* was binomial and selected all Binomial distributions as 'reference' *pmf*s if the 'query' *pmf* was a Zipf distribution.

dist. measure	runtime [ns]	*cCoeff*			Deviation		
	avg	min	max	avg	min	max	avg
EMD	3144493	1.0	1.0	1.0	0.0	0.0	0.0
PME-Dst$_{\text{asym}}$	80079	0.864	1.0	0.975	0.0	0.388	0.041
PME-Dst$_{\text{min}}$	173897	0.967	1.0	0.995	0.0	0.160	0.020
PME-Dst$_{\text{max}}$	180694	0.781	1.0	0.959	0.0	0.388	0.062
PME-Dst$_{\text{avg}}$	177735	0.935	1.0	0.986	0.0	0.198	0.041

Table 9.3.: Runtime, Correlation Coefficient, and Deviation for all considered distance measures computed for all pairs of considered *pmf*s

9.4.3. Results

We start with considering the experimental results on a coarse level of granularity. For that purpose, we aggregated the results of the first group (all pairs of *pmf*s) of all domain sizes. This implicated a consideration of 454140[12] distance computations. The aggregated values are presented in Table 9.3 for each of the five considered distance measures. As it can be seen, the average runtime of the Earth Mover's Distance (short *EMD*) was much larger than the runtime of the different variants of the Probabilistic Monge-Elkan Distance. For instance, the average runtime of the Earth Mover's Distance was almost 40 times larger than the average runtime of the asymmetric variant of the Probabilistic Monge-Elkan Distance and was around 18 times larger than the average runtime of the symmetric variants of the Probabilistic Monge-Elkan Distance. Obviously, the average runtimes of the symmetric variants were approximately twice as large as the average runtime of the asymmetric variant.

Of course, since we consider this measure as benchmark, the Correlation Coefficient and the Deviation of the Earth Mover's Distance were always 1.0 and 0.0 respectively. The minimal Correlation Coefficient as well as the average Correlation Coefficient of the Probabilistic Monge-Elkan Distance was worst if we used the the maximum operator for aggregating the directed distances and was best if we used the minimum operator for distance aggregation. The maximum Correlation Coefficient was 1.0 for all variants. This, however, is actually an obvious result because we compared each *pmf* with itself. Although the worst minimal Correlation Coefficient was down to 0.781, the average Correlation Coefficient of all considered variants was above 0.95 and therefore generally indicates a high semantical correlation between the Earth Mover's Distance and the Probabilistic Monge-Elkan Distance. Similar observations resulted from rating the semantical closeness in terms of Deviation. Again using the minimal directed distance turned out to be most fitting and using the maximal directed distance turned out to be least fitting. Moreover, the average Deviation of all four variants was very low (below 0.065) although in the

[11] For processing reasons we included the 'query' *pmf* into the set of 'reference' *pmf*s because in this case the set of 'reference' *pmf*s was always the same for each 'query' *pmf* of the considered group and the value $\overline{d_x}$ only needed to be computed once. Of course, by doing so each *pmf* was matched with itself. However, the number of matching identical *pmf*s was only small compared to the whole number of matches and therefore influenced the final Correlation Coefficient only to a small extent.

[12] For each of the six domain sizes, we computed the distance between each two of the 87 *pmf*s by using ten different ground distance matrices. In summary, this made $6 \times 87 \times 87 \times 10 = 454140$ distance computations.

worst cases the Deviation was up 0.388. Nonetheless, the largest Deviation that resulted from using the minimal directed distance was 0.160 which is not much for 454140 computed distance values.

The previously considered results show that all four variants of the Probabilistic Monge-Elkan Distance produced similar distance values and had a similar runtime (the only exception was the asymmetric variant which was twice as fast as the remaining three variants). The same holds for considerations on finer levels of granularity as for example by considering runtime, Correlation Coefficient, and Deviation per domain size or per group of *pmf*s. Nevertheless, despite all these similarities the variant that computes the minimal directed distance had the greatest semantical correlation to the Earth Mover's Distance. For that reason, we restrict the following considerations to this variant.

The corresponding results are depicted in Figure 9.3. Figure 9.3(a) presents the average runtime of the Earth Mover's Distance and the average runtime of Probabilistic Monge Elkan Distance with respect to the different domain sizes. It can be simply seen that the average runtime of the Earth Mover's Distance increased much faster with a growing number of domain elements than the average runtime of the Probabilistic Monge Elkan Distance.

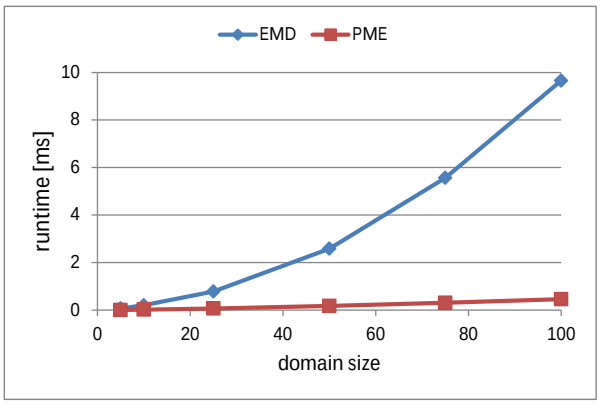
Figure 9.3(b) presents the average speed-up that was achieved by using the Probabilistic Monge Elkan Distance instead of the Earth Mover's Distance for the different combinations of distribution types and domain sizes. As it can be seen, the speed-up increased remarkably with a growing domain size with respect to the most groups. The only exception occurred if the compared *pmf*s are both binomial distributed. In this case, the speed-up only changed marginally with a growing number of domain elements.

The average Correlation Coefficient of the Probabilistic Monge Elkan Distance with respect to the different groups of distribution types and domain sizes is depicted in Figure 9.3(c). In most cases, the Correlation Coefficient only changed marginally with a varying domain size, but decreased strikingly with a growing number of domain elements if we compared two Binomial distributions. Nevertheless, even in this case the coefficient was around 0.987 if we considered domains with hundred elements and the increment of decrease diminished the larger the considered domains were. In contrast, the semantical correlation between the Probabilistic Monge Elkan Distance and the Earth Mover's Distance was best if the compared *pmf*s were both Zipf distributions.

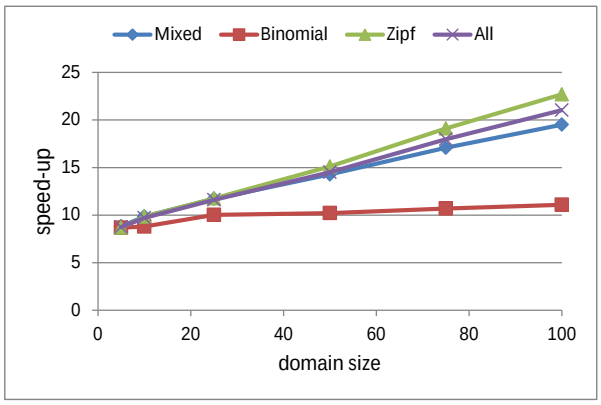
Finally, the average Deviation of the Probabilistic Monge Elkan Distance with respect to the different groups of distribution types and domain sizes is depicted in Figure 9.3(d). The average Deviation increased notably if the domain size grew from 5 to 50, but only increased marginally if the domain size grew from 50 to 100. Again, the group of binomial distributed *pmf* pairs had the lowest semantical correlation and the group of zipf distributed *pmf* pairs had the highest semantical correlation.

9.4.4. Conclusion

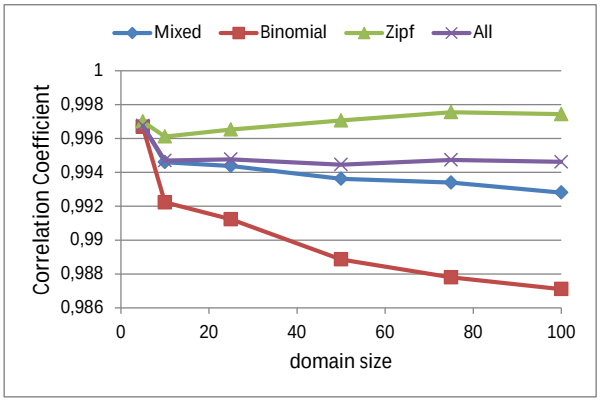
The experimental results show that the Probabilistic Monge-Elkan Distance required a much lower runtime than the Earth Mover's Distance while producing similar distance scores. The increase in runtime and the semantical correlation considerably decreased if the compared *pmf*s are both binomial distributed. Nevertheless, even in this case the Probabilistic Monge-Elkan Distance was around ten times faster than the Earth Mover's Distance and the semantical difference was still less than 0.035 in terms of the average Deviation and was still less than 0.02 in terms of the Correlation Coefficient. From this



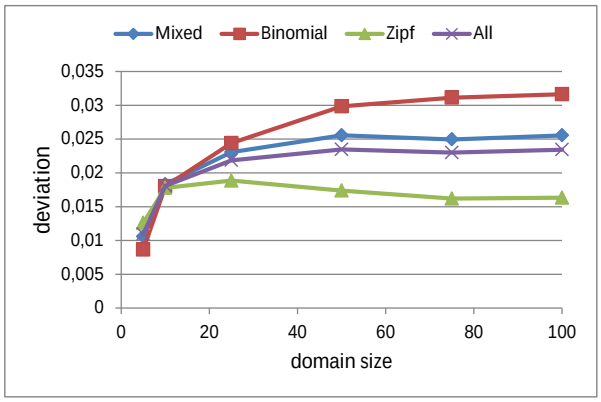
(a) Avg. runtime of EMD and PME



(b) Avg. speed-up of PME compared to EMD



(c) Avg. Correlation Coefficient of PME compared to EMD



(d) Avg. Deviation of PME compared to EMD

Figure 9.3.: Experimental results from comparing *PME-Dst*$_{\text{min}}$ with the Earth Mover's Distance

follows that the Probabilistic Monge-Elkan Distance can be considered a fast alternative to the Earth Mover's Distance.

9.5. Experiment: Description-based Detection Approach

The goal of this experiment was to prove the concept of the description-based detection approach. Obviously, the quality of the detection approach depends on the used resolution method, but also depends on the dirtiness and the uncertainty of the considered source database. For that reason, we divided this experiment into two sub-experiments. In the first sub-experiment, we evaluated the detection quality of several resolution methods under a varying dirtiness of the source database. In the second sub-experiment, we evaluated the detection quality of several resolution methods under a varying uncertainty of the source database where we consider uncertainty in terms of Number of Alternatives.

9.5.1. Set-Up

In both sub-experiments we used the synthetic test databases that we have described in Section 9.3. Moreover, we used a simple configuration for the certain data components of the Duplicate Detector. The candidate pair space was reduced by combining the top-10 strategy of Key-per-Alternative (see

Method	Description	Moment of Resolution
MProbED	selects the most probable alternative description per database entity	data preparation phase
MProbPair	selects the most probable alternative description pair per candidate pair	APS construction phase
MaxFS	computes the maximal value per position of the feature score vectors	feature matching phase
ExpSim	computes the expected similarity value and the expected impact value of the similarity score	similarity computation phase
PME	computes the Probabilistic Monge-Elkan Similarity and the expected impact value of the similarity score	similarity computation phase
MaxSim	computes the maximal similarity value and the expected impact value of the similarity score	similarity computation phase
MProbDec	computes the most probable matching class, the expected similarity, and the expected impact value of the duplicate decision	classification phase

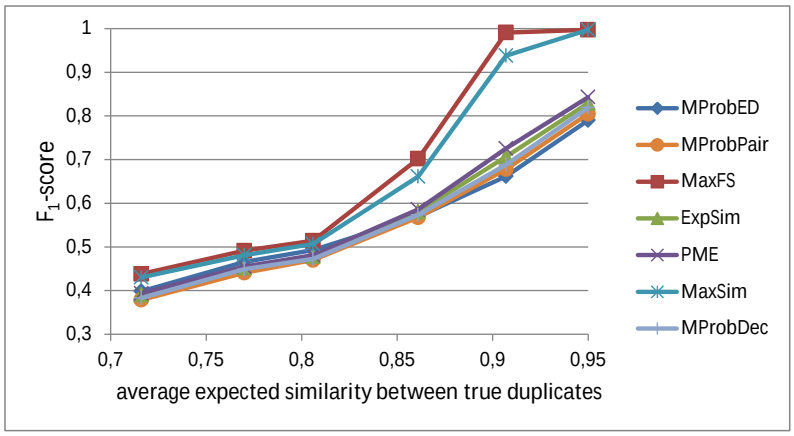
Table 9.4.: Used methods for uncertainty resolution

Section 7.4.4) with the improved version of the Suffix Array based Blocking that has been proposed by de Vries et al. [dVKCC11]. To enable a fair comparison, we used a specific blocking configuration per test database so that the constructed candidate pair spaces were always of a similar size. For matching two attribute values in the feature matching phase, we used the Levenshtein Similarity (see Section 4.3.5) The similarity between two certain entity descriptions was computed by a distance-based decision model that simply calculates the weighted average similarity between the values in the four attributes *'title'*, *'studio'*, *'director'*, and *'year'*. More exact, the similarity between two descriptions alternatives d_1 and d_2 was computed as:

$$\begin{aligned} sim(d_1, d_2) \quad = \quad & 0.6 \times sim(d_1[title], d_2[title]) \; + \; 0.1 \times sim(d_1[studio], d_2[studio]) \\ & + \; 0.2 \times sim(d_1[director], d_2[director]) \; + \; 0.1 \times sim(d_1[year], d_2[year]) \end{aligned}$$

For classification we used the threshold that leads to the best F_1-score of the pairwise matching results and therefore did not produce a class of POSSIBLE MATCHES. Each of the considered resolution methods resolves uncertainty before the end of the pairwise matching process. For that reason, we removed an unknown from this experimental evaluation by omitting the clustering phase and by evaluating detection quality based on the pairwise made duplicate decisions instead.

For uncertainty resolution, we used different methods at different moments. All these methods are listed and described in Table 9.4. The first two resolution methods semantically correspond to a simple approach that only processes the most probable world of the input database. The difference in both methods is that the first one performs resolution before the candidate pair space is constructed and the second one performs resolution after the candidate pair space is constructed. The consequence is that we used all description alternatives for computing candidate pairs in the case of the latter, but could only



(a) Correlation between detection quality and data dirtiness for several resolution methods

Data Dirtiness	***MProbED***	***MProbPair***	***MaxFS***	***ExpSim***	***PME***	***MaxSim***	***MPDec***
A (0.716)	0.39852	0.37887	0.43838	0.39037	0.39371	0.43040	0.38316
B (0.770)	0.46554	0.44087	0.49139	0.45196	0.45505	0.48116	0.44896
C (0.806)	0.49291	0.46991	0.51422	0.47822	0.48091	0.50683	0.47272
D (0.861)	0.56927	0.56743	0.70198	0.58042	0.58618	0.66107	0.57323
E (0.907)	0.66167	0.67783	0.99084	0.70680	0.72536	0.93803	0.68812
F (0.950)	0.78997	0.80494	0.99724	0.82938	0.84316	0.99714	0.81912

(b) Exact F_1-scores of the different resolution methods for the different levels of data dirtiness

Figure 9.4.: Quality of several uncertainty resolution methods under a varying dirtiness of the source database

used a single description alternative for computing candidate pairs in the case of the first. As we have described in [PWFR12], using one blocking key value per description alternative is much more effective than using one blocking key value per entity. For that reason, the experimental detection processes that used the latter resolution method worked on a more complete candidate pair space than the experimental detection processes that used the first resolution method.

For evaluating detection quality, we used the Pair F_1-score in both sub-experiments. Finally, both sub-experiments were performed on a machine with an Intel(R) 3.1GHz quad-core processor, 8GB main memory, and a 64-bit version of the operating system *Windows 7*.

9.5.2. Execution

In the first sub-experiment we wanted to evaluate the correlation between detection quality and source data dirtiness. For that reason, we performed one detection process for each of the aforementioned process configurations on each of the test databases $db^{10}_{A_1} - db^{10}_{A_5}$, $db^{10}_{B_1} - db^{10}_{B_5}$, $db^{10}_{C_1} - db^{10}_{C_5}$, $db^{10}_{D_1} - db^{10}_{D_5}$, $db^{10}_{E_1} - db^{10}_{E_5}$, and $db^{10}_{F_1} - db^{10}_{F_5}$ where data dirtiness gradually increases from $db^{10}_{A_*}$ to $db^{10}_{F_*}$.

The goal of the second sub-experiment is to evaluate the correlation between detection quality and number of description alternatives. Therefore, we performed one detection process for each of the aforementioned process configurations on each of the test databases $db^{10}_{D_{small}}$, $db^{50}_{D_{small}}$, $db^{100}_{D_{small}}$, and $db^{150}_{D_{small}}$.

9.5.3. Results

The results of the first sub-experiment are graphically depicted in Figure 9.4(a) and are exactly listed in Table 9.4(b). Obviously, the dirtier the input database the lower was the detection quality for each of the considered process configurations.

Interestingly, the differences between the individual process configurations was low for data of poor quality and increased the more the cleaner the input database became. With respect to the considered test databases, the detection processes that resolved uncertainty by computing the maximal value per feature score position (*MaxFS*) or that resolved uncertainty by computing the maximal similarity between the compared entities (*MaxSim*) respectively provided the best detection results. The detection quality of the remaining detection processes was close for all degrees of data dirtiness, but the differences became larger for the cleanest databases. From this group, the process that used the Probabilistic Monge-Elkan Similarity performed best. The second best of this group was the process that resolved uncertainty by computing the Expected Entity Similarity (*ExpSim*). The third best process was the one that computed the most probable matching class. Close to this process followed the process that resolved uncertainty by selecting the most probable alternative description pair. The lowest detection quality was provided by the detection process that resolved uncertainty by selecting the most probable description alternative per entity. Although the detection quality of the individual processes from this group was close together, the differences between using the Probabilistic Monge-Elkan Similarity and selecting the most probable alternative description pair (or most probable description alternative respectively) was still noticeable. For instance, for the cleanest databases (type F) the first process achieved a Pair F_1-score of 0.843 and the latter processes achieved a Pair F_1-score of 0.79 (MProbED) or 0.805 (MProbPair) respectively. This seems not much if we consider the Pair F_1-score, but already implicates a production of 300 more Pair True Positives while producing 162 less Pair False Positives and 301 less False Negatives compared to selecting the most probable entity description and implicates a production of 204 more Pair True Positives while producing 135 less Pair False Positives and 204 less False Negatives compared to selecting the most probable description pair. Thus compared to these two resolution methods, using the Probabilistic Monge-Elkan Similarity led to a detection of more 'true' duplicates while decreasing the number of Pair False Decisions and hence achieved a better detection quality with respect to any possible quality requirement.

The results of the second sub-experiment are graphically depicted in Figure 9.5. For reasons of clarity and comprehensibility we drop the results from the two resolution methods *MaxSim* and *MProbDec* from this presentation. The first one is dropped because it is based on the same idea than *MaxFS*, i.e. both compute the maximal possible similarity, and its quality was always a bit smaller than this of *MaxFS*. Similar holds for *MProbDec* that always achieved similar results than *ExpSim*. Nevertheless, for both resolution methods we observed that their quality increased with a growing number of alternatives per entity description.

For time reasons, we only processed one database per considered characteristics. However, although we could not completely avoid that these results are affected by randomness, a trend is clearly evident. Whereas the detection quality that resulted from using one of the resolution methods *MaxFS*, *ExpSim*, and *PME* increased with a growing number of description alternatives (much stronger for *MaxFS* than for *ExpSim* and *PME*), the detection quality that resulted from using one of the trivial resolution methods

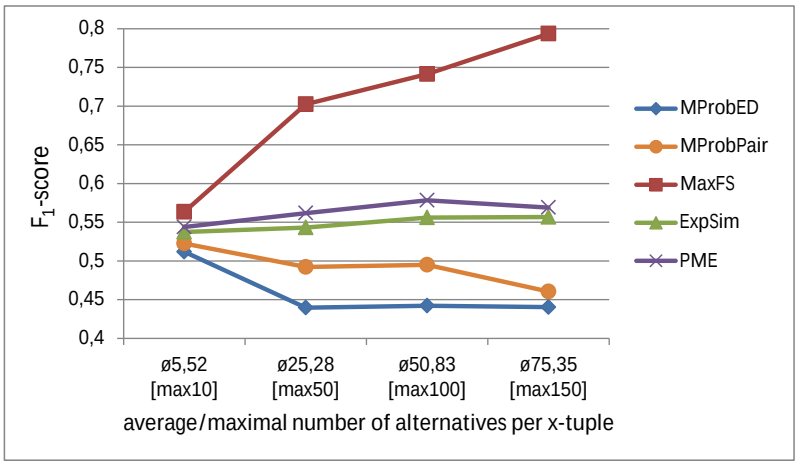


Figure 9.5.: Quality of several uncertainty resolution methods under a varying uncertainty of the source database

MProbED and *MProbPair* decreased with a growing number of description alternatives. As a consequence, whereas the former profited from the additional information that was provided by the additional alternatives, the latter suffered from this circumstances because it became much more unlikely that the selected description alternative (or the selected alternative description pair respectively) was a good representation for the complete set of description alternatives (or alternative description pairs respectively).

9.5.4. Conclusion

The results of this experiments show that the choice of the used resolution method becomes more important the cleaner the considered database is and becomes more important the more alternatives per entity description have been extracted. The first conclusion is somewhat surprising because we actually expected the contrary, i.e. we expected that the differences between the individual resolution methods are the greater the dirtier the considered databases are, because in such cases the demarcation between duplicates and non-duplicates becomes much more difficult. The by far best results were provided by the two resolution methods that computed the maximal similarity on attribute level and description level respectively. Thus, another interesting observation is that the considered resolution methods could benefit from many alternatives per description, but actually could not profit from the provided probabilities. The first one is logical because the more alternatives per description are available the more reasoned is the made resolution. The latter one shows that in our test databases the probabilities do not provide useful information which is likely reasoned by the circumstance that the probabilities were assigned to the individual alternatives randomly. One goal of future research is therefore to expand our experimental evaluations on more and maybe more realistic test databases. However, we think that the results of this experiment already serve as a proof of concept because we were able to detect a large amount of duplicates. Moreover, the description-based detection approach has not been designed for probabilistic databases exclusively, but can be also used for incomplete databases.

9.6. Experiment: Matching Cost Optimization

In Section 7.6.1.2 we propose the In-Depth Pruning technique that reduces the costs of matching two probabilistic entity descriptions by pruning unnecessary matchings of alternative description pairs. The goal of this experiment is to evaluate the effectiveness of In-Depth Pruning under different circumstances.

It is notorious that the effectiveness of In-Depth Pruning depends on several factors such as the characteristic of the processed database or the used matching process. For that reason, we divided this experiment into three sub-experiments. In the first sub-experiment, we evaluated the effectiveness of In-Depth Pruning under a varying dirtiness of the source database. In the second sub-experiment, we evaluated the effectiveness of In-Depth Pruning under a varying uncertainty of the source database. Finally, in the last sub-experiment we evaluated pruning effectiveness under a varying complexity of the process that is used for matching two description alternatives.

9.6.1. Set-Up

In all three sub-experiments we used the synthetic test databases that we have described in Section 9.3. We principally used the same configurations for the certain data components as in the previous experiment (see Section 9.5). However, the iterative process of matching two probabilistic descriptions can only be prematurely stopped if from the Best Match Scenario and the Worst Match Scenario same consequences result. For simulating non-trivial consequences, we additionally performed the clustering phase in this experiment and used a variant of Partitioning based on Merged Centers [HCML09] that gets the Duplicate-Pair Graph as input. Since this clustering method requires the exact similarity scores for all MATCHES, alternative description pair matchings could only be pruned if the compared entities were certainly an UNMATCH and hence if the similarity between their descriptions was definitely lower than the similarity threshold that we used for classification.

As mentioned in Section 9.2, we implemented In-Depth Pruning only exemplarily for two resolution methods so far, namely the Expected Entity Similarity and the Probabilistic Monge-Elkan Similarity. For that reason, we restricted our considerations in this experiment to these two resolution methods.

In the third sub-experiment, we studied the correlation between pruning effectiveness and matching complexity. The complexity of matching two certain entity descriptions in turn depends on several factors like the complexity of the used matching process or the number of features (in our case attributes) that need to be compared. Since the number of available attributes was restricted to four, we decided to vary the complexity of the used matching process. The feature matching phase is by far the most expensive part of the pairwise matching process and the spectrum of available configurations ranges from using a single less complex measure like the Levenshtein Similarity to aggregating the results of several complex similarity measures such as the Monge-Elkan Similarity or the Extended Jaccard Coefficient. For that reason, we varied the complexity of matching two certain descriptions by using different similarity measures for attribute value comparison. In order to achieve a cardinal scale of matching complexity, we decided to simulate a set of similarity measures with different computation complexities instead of using some actual measures. For simulation purposes, we used the Levenshtein Similarity and computed it for a number of fixed times (parameter t_{Lev}) on the same attribute values. By doing so, the complexity of the feature matching phase was changed by changing the number of times

two attribute values were compared by using the Levenshtein Similarity, i.e. setting $t_{Lev} = 3$ means to compute the Levenshtein Similarity between two attribute values for three times instead of only one time. In summary, using the setting $t_{Lev} = 3$ simulates a scenario in which we compare two attribute values by using a similarity measure that is three times as much complex as the Levenshtein Similarity, but also simulates a scenario in which we need to compare three times as many attribute value pairs by using the Levenshtein Similarity.

At a first glance, the time saving that is achieved by In-Depth Pruning is linear to the number of pruned alternative description pair matchings. However, In-Depth Pruning needs to compute a Best Match Scenario and needs to compute a Worst Match Scenario for every of the matched alternative description pairs and therefore implicates a computational overhead compared to the naive matching approach. For that reason, we measured pruning effectiveness in terms of pruned alternative description pair matchings as well as in terms of time saving. For measuring the first, we computed the average number of alternative description pairs that needed to be matched for matching a pair of probabilistic entity descriptions with and without pruning and then calculated the amount of pruned matchings by dividing the first by the latter. For measuring time saving, we computed the average time that is required for matching a pair of probabilistic entity descriptions with and without pruning and then calculated the achieved speed-up by dividing the second by the first.

All three sub-experiments were performed on a machine with an Intel(R) 3.1GHz quad-core processor, 8GB main memory, and the 64-bit version of the operating system *Windows 7*.

9.6.2. Execution

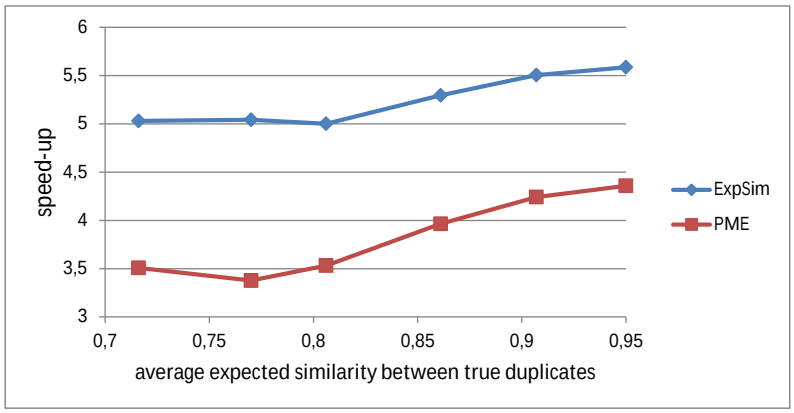
In the first sub-experiment, we wanted to evaluate the effectiveness of In-Depth Pruning depending on the dirtiness of the source databases. For that reason, we performed one detection process with pruning and one detection process without pruning for each of the aforementioned process configurations on each of the test databases $db^{10}_{A_1} - db^{10}_{A_5}$, $db^{10}_{B_1} - db^{10}_{B_5}$, $db^{10}_{C_1} - db^{10}_{C_5}$, $db^{10}_{D_1} - db^{10}_{D_5}$, $db^{10}_{E_1} - db^{10}_{E_5}$, and $db^{10}_{F_1} - db^{10}_{F_5}$ where data dirtiness gradually increases from $db^{10}_{A_*}$ to $db^{10}_{F_*}$.

In the second sub-experiment, we evaluated to which extent the number of description alternatives influences pruning effectiveness. For that purpose, we performed one detection process with pruning and one detection process without pruning for each of the aforementioned process configurations on each of the test databases $db^{10}_{D_{small}}$, $db^{50}_{D_{small}}$, $db^{100}_{D_{small}}$, and $db^{150}_{D_{small}}$.

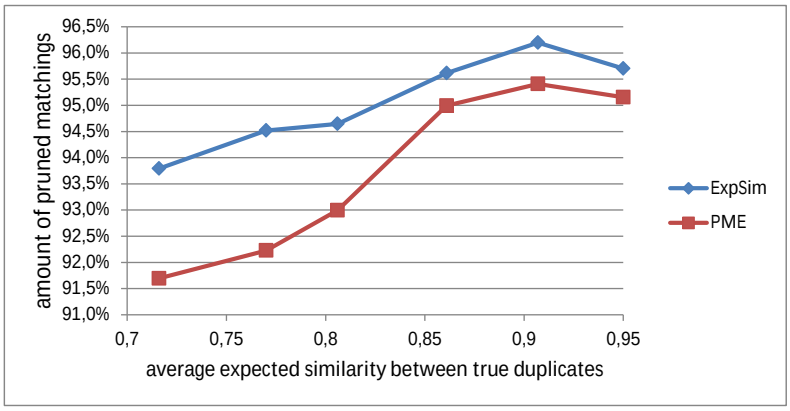
In the last sub-experiment, we restricted evaluation to the test databases $db^{10}_{D1} - db^{10}_{D5}$, but increased the parameter t_{Lev} from 1 to 6.

9.6.3. Results

The results of the first sub-experiment are presented in Figure 9.6(a) (speed-up) and Figure 9.6(b) (amount of pruned alternative description pair matchings). As we can see, the speed-up in runtime marginally increased with a growing database cleanness and was always between $5 - 5.6$ for the Expected Entity Similarity and was always between $3.4 - 4.4$ for the Probabilistic Monge-Elkan Similarity.



(a) speed-up in computation time



(b) Amount of pruned alternative description pair matchings

Figure 9.6.: Correlation between pruning effectiveness and source data dirtiness

The amount of pruned alternative description pair matchings increased with a shrinking dirtiness of the source database, but was already around 91.7% for the Probabilistic Monge-Elkan Similarity and was already around 93.8% for Expected Entity Similarity even in the case of the dirtiest database.

From the results of the second sub-experiment we could not retrieve any patterns or trends. Thus, the correlation between pruning effectiveness and data uncertainty did not become clear by considering our test databases. Since we cannot make any meaningful conclusions from these results we omit their presentation from this thesis.

The results of the third sub-experiment are depicted in Figure 9.7. As anticipated, the speed-up in computation time increased from 4 to 8.7 (Probabilistic Monge-Elkan Similarity) and increased from 5 to 11.6 (Expected Entity Similarity) with a growing complexity of the used matching process. Of course, since we only changed the costs of the pairwise matching process, but did not changed the number of matchings, the amount of pruned alternative description pair matchings was constantly 95.6% for the Expected Entity Similarity and was constantly 94.4% for the Probabilistic Monge-Elkan Similarity.

9.6.4. Conclusion

Two of the three sub-experiments that we have presented above showed that using In-Depth Pruning can decrease the complexity of matching two probabilistic entity descriptions to a large extent. Of

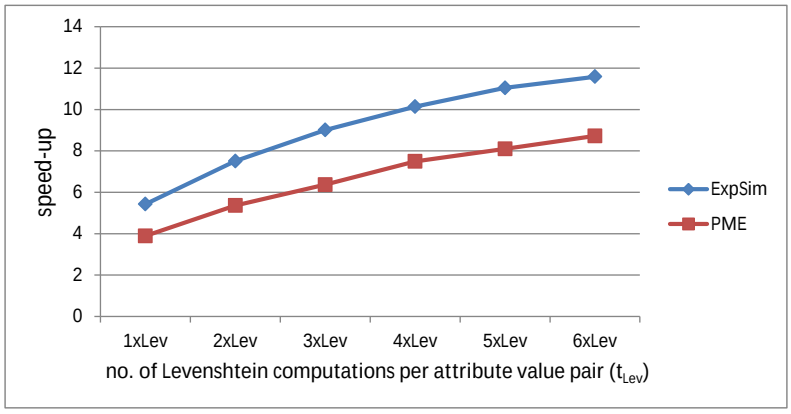


Figure 9.7.: Correlation between pruning effectiveness and complexity of the used matching process

course, pruning was the more lucrative the less computational overhead was implicated by an iterative performance of the corresponding resolution method. For that reason, pruning was more effective for the Expected Entity Similarity than for the Probabilistic Monge-Elkan Similarity. Moreover, the results of these evaluations indicate that pruning becomes more valuable the cleaner the input database is or the more time consuming the used matching process is.

9.7. Experiment: World-based Detection Approach

The goal of this experiment was to prove the concept of the world-based detection approach. Obviously, the quality of the detection approach depends on the used process configuration, but also depends on the dirtiness and the uncertainty of the considered source database. For that reason, we divided this experiment into three sub-experiments.

In the first sub-experiment, we evaluated the detection quality of several aggregation methods under a varying world generation method and a varying number of generated worlds on databases of a fixed dirtiness and a fixed uncertainty.

In the second sub-experiment, we evaluated the detection quality of a single aggregation method under a varying world generation method, a varying number of generated worlds, and a varying dirtiness of the source database.

In the third sub-experiment, we evaluated the detection quality of a single aggregation method under a varying world generation method, a varying number of generated worlds, and a varying uncertainty of the source database where we consider uncertainty in terms of Number of Alternatives per x-tuple.

9.7.1. Set-Up

In all three sub-experiments we used the synthetic test databases that we have described in Section 9.3. Moreover, we used the same configurations for the certain data components as in the previous two experiments (see Section 9.5). Since one of the considered aggregation methods resolves uncertainty on clustering level, we took the clustering phase into account and used the same clustering method as in the previous experiment.

Method	Description	Moment of Consolidation
MaxFS	computes the maximal value per position of the feature score vectors	feature matching phase
ExpSim	computes the expected similarity value and the expected impact value of the similarity score	similarity computation phase
MaxSim	computes the maximal similarity value and the expected impact value of the similarity score	similarity computation phase
MProbDec	computes the most probable matching class, the expected similarity, and the expected impact value of the duplicate decision	classification phase
Rand	selects the clustering that has the maximal Rand Index to all other clusterings	duplicate clustering phase

Table 9.5.: Aggregation methods that are used for consolidating the results of the individual worlds

In addition to the parameters of the certain data components, the world-based detection approach has four parameters. The first parameter is the algorithm that is used for generating worlds, the second parameter is the number of generated worlds, the third parameter is the moment of consolidation, and the fourth parameter is the aggregation method that is used for consolidation. In this experiment we tested different settings for each of these parameters. For world generation, we used the Monte-Carlo Simulation (short MC) and the Dissimilar Worlds selection strategy (short DW) that we have proposed in Section 7.4.4.1. As consolidation moments we considered the end of the feature matching phase, the end of the similarity computation phase, the end of the classification phase, and the end of the clustering phase. For consolidation we used the aggregation methods that are listed in Table 9.5.

Recall that the Dissimilar Worlds selection strategy can only generate as many different worlds as the largest x-tuple has alternatives. For this reason, we combined this world generation method only with at most ten generated worlds in each of the sub-experiments.

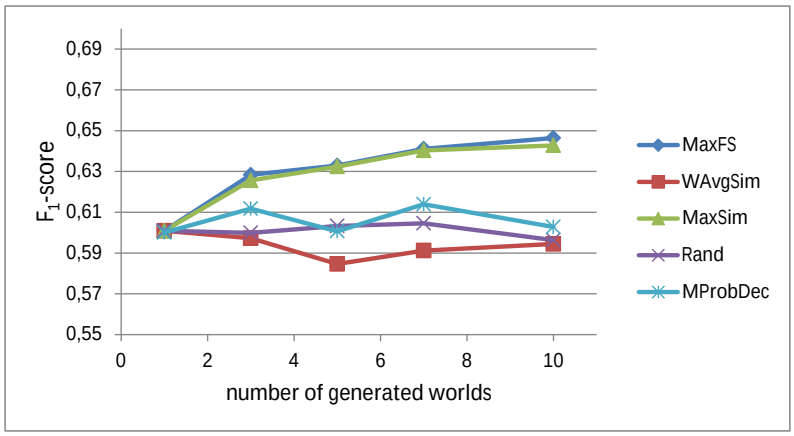
All sub-experiments were performed on a machine with an Intel 2.27GHz double-core processor, 20GB main memory, and a 64-bit version of the operating system *Suse Enterprise Server 11*.

9.7.2. Execution

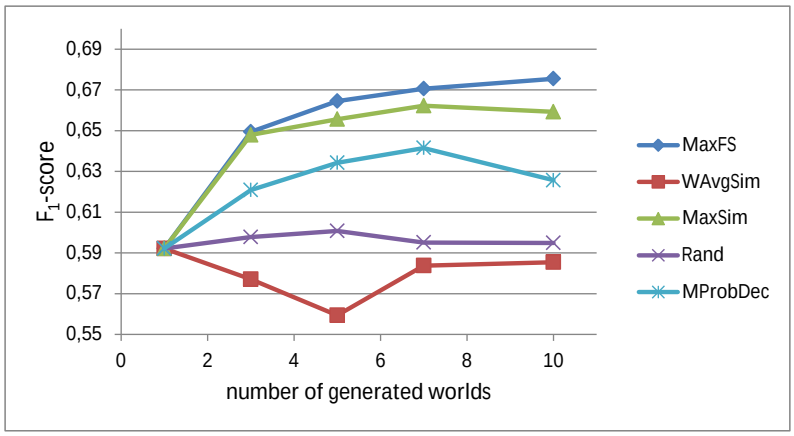
In the first sub-experiment, we only considered the standard databases $db^{10}_{D_1}$ to $db^{10}_{D_5}$ and evaluated both world generation methods with all the aforementioned aggregation methods for different numbers of generated worlds.

In the second sub-experiment, we selected one of the process configurations that performed well in the first sub-experiment and evaluated it in combination with both world generation methods for each of the test databases $db^{10}_{A_1} - db^{10}_{A_5}$, $db^{10}_{B_1} - db^{10}_{B_5}$, $db^{10}_{C_1} - db^{10}_{C_5}$, $db^{10}_{D_1} - db^{10}_{D_5}$, $db^{10}_{E_1} - db^{10}_{E_5}$, and $db^{10}_{F_1} - db^{10}_{F_5}$ and different numbers of generated worlds.

In the third sub-experiment, we selected the same process configurations that we have used in the second sub-experiment and evaluated it in combination with both world generation methods for each of the test databases $db^{10}_{D_{small}}$, $db^{50}_{D_{small}}$, $db^{100}_{D_{small}}$, and $db^{150}_{D_{small}}$ and different numbers of generated worlds.



(a) Monte-Carlo Simulation

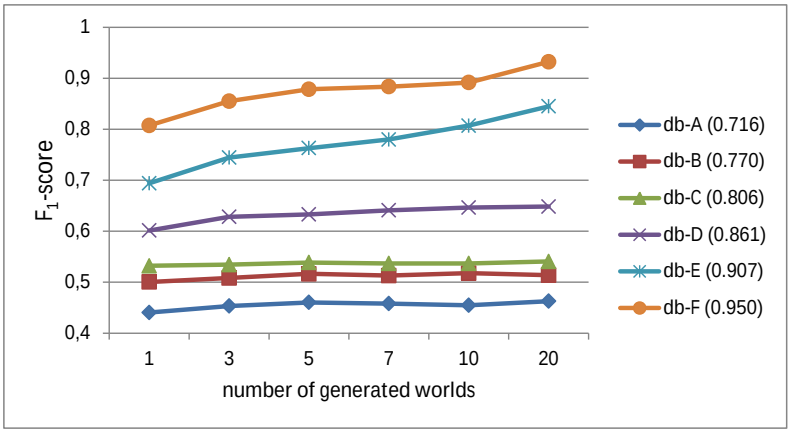


(b) Dissimilar Worlds selection strategy

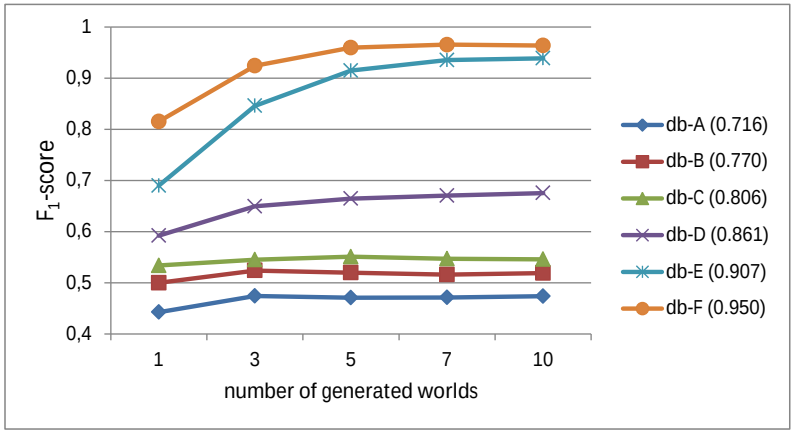
Figure 9.8.: Quality of several aggregation methods under a varying number of generated worlds

9.7.3. Results

The results of the first sub-experiment are depicted in Figure 9.8(a) (Monte-Carlo Simulation) and Figure 9.8(b) (Dissimilar Worlds). As we can see, increasing the number of generated worlds was not profitable for all aggregation methods. In the case of using the Monte-Carlo Simulation, only the two aggregation methods that computes the maximal similarity (*MaxFS* and *MaxSim*) profited from a large number of processed worlds. This is logical because we could already observed in the experiment on the description-based detection approach that using the maximal similarity led to good detection results and obviously the maximal similarity was the greater the more worlds were processed. In the case of the Dissimilar Worlds selection strategy, besides *MaxFS* and *MaxSim* the aggregation method that computes the most probable duplicate decision per entity pair (*MProbDec*) additionally profited from an increasing number of generated worlds. However, as we can see in Figure 9.8(b) its quality increased as long as the number of worlds grows up to seven, but decreased when we processed more than seven worlds At a first glance, this result looks somewhat curious. However, by taking a closer look on the used process configuration we see that it is a logical result. Recall that the Dissimilar Worlds selection strategy takes the ith most probable alternative per x-tuple for generating the ith world, but falls back to the most probable alternative for each x-tuple that has less than i alternatives. Moreover, the probability that an



(a) Monte-Carlo Simulation



(b) Dissimilar Worlds selection strategy

Figure 9.9.: Quality of the aggregation method *MaxFS* under a varying dirtiness of the database and a varying number of generated worlds

entity pair is classified as a MATCH is computed by the amount of worlds in which these entities are classified as a MATCH. In the considered databases the average number of alternatives per x-tuple is between $5.45 - 5.47$. As a consequence, the closer the number of generated worlds came to ten, for the more x-tuples the most probable alternative was selected, and the greater became the probability of the matching classes that were provided by the first world. Note, if we would generate more than ten worlds, all additional worlds would correspond to the first world and the greater the number of processed worlds the more probable would become the matching classes that are provided by this world.

Processing multiple worlds was more lucrative for *MaxFS* and *MaxSim* if these worlds were generated by using the Dissimilar Worlds strategy than by using the Monte-Carlo Simulation. This could be explained by the circumstance that the worlds generated by the Monte-Carlo Simulation were very similar to each other and therefore processing multiple worlds did not brought much advantage. In contrast, the Dissimilar Worlds selection strategy was even designed for producing a set of dissimilar worlds and each of these worlds potentially increased the maximal similarity between two database entities. It is obvious that this difference is based on the circumstance that aggregating detection results by computing the maximal similarity performed best on the considered test databases and the maximal similarity becomes

larger if the processed worlds are more dissimilar. However, it cannot be excluded that the Monte-Carlo Simulation would perform better for test databases for which the Weigthed Average Entity Similarity (*WAvgSim*) or the Probabilistic Monge-Elkan Similarity are the best methods for world consolidation.

For the second sub-experiment we selected the aggregation method *MaxFS* because it became apparent in the first sub-experiment that this method profited very well from processing multiple worlds. The results of the second sub-experiment are depicted in Figure 9.9(a) (Monte-Carlo Simulation) and Figure 9.9(b) (Dissimilar Worlds) where each line correspond to another type of data dirtiness (the number in the brackets refer to the corresponding average expected similarities between the true duplicates).

As we can see, for both world generation methods the Pair F_1-score increased the more with a growing number of generated worlds the cleaner the input database was. In the dirtiest cases (types A, B, and C), processing multiple worlds only brought less advantage for both world generation methods. In contrast, for the cleanest databases (types E and F) processing multiple worlds increased detection quality to a large extent. This increase was more remarkable if we used the Dissimilar Worlds selection strategy than if we used the Monte-Carlo Simulation although we processed twenty worlds in the case of the latter. This experiment showed that using multiple worlds can be very valuable for clean databases. For instance, the detection quality that resulted from processing the databases of dirtiness type E with using the Dissimilar Worlds selection strategy increased from 0.69 (one world) to 0.94 (ten worlds) which is an impressive gain in detection quality.

The goal of the last sub-experiment was to evaluate the correlation between detection quality, number of generated worlds, and the uncertainty of the input database. For that purpose we selected the aggregation method *MaxFS* and compute the Pair F_1-score for each test database for different number of generated worlds. However, although the quality increased with a growing number of generated worlds for all considered test databases, we did not observe any pattern or trend in these results. Thus, the correlation between detection quality, number of generated worlds, and data uncertainty did not become clear by considering our test databases. Since we cannot make any meaningful conclusions from these results we omit their presentation from this thesis.

9.7.4. Conclusion

In general, the results of this experiment demonstrated that the world-based detection approach is a useful tool for detecting duplicates in probabilistic databases. Moreover, two of the three sub-experiments showed that processing multiple worlds can considerably increase the quality of the detection process. This increase was particularly remarkable if uncertainty in the cleanest databases was resolved by computing the maximal similarity of attribute values or entities and if worlds were generated by using the Dissimilar Worlds selection strategy.

9.8. Experiment: Indeterministic Duplicate Detection

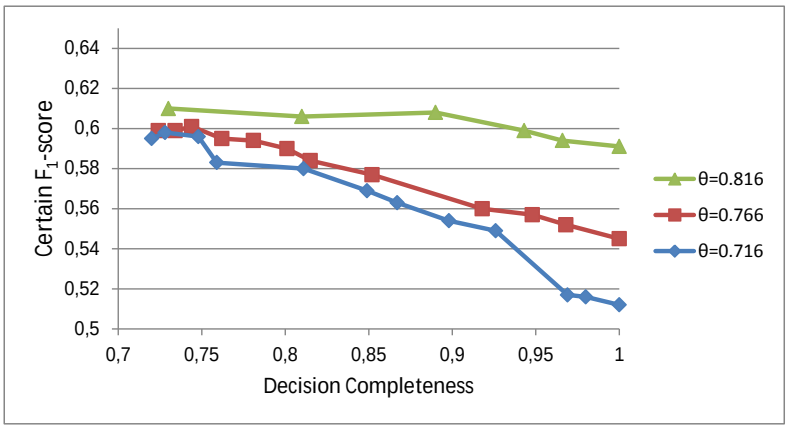
The idea behind the concept of indeterministic duplicate detection is to introduce a trade-off between decision correctness and decision certainty. Thus, the goal of this experiment was to prove that our indeterministic detection approach is indeed suitable for increasing decision correctness by reducing decision certainty and hence to illustrate that the concept of indeterministic duplicate detection can be useful in

many detection scenarios. Of course, to which extent decision correctness can be actually increased depends on several factors. One of these factors is the quality of the matching process that is used for computing pairwise duplicate decisions. If this process has a good quality the most decisions will be correct and a further improvement of decision correctness is difficult. In contrast, if the performed detection process is of a poor quality, the number of Pair False Decisions will be large and handling the most uncertain of these decisions in an indeterministic way is assumed to increase decision correctness. In order to demonstrate such a scenario, we conducted several test runs in which we changed the configuration of the similarity threshold that demarcates the set of MATCHES from the set of UNMATCHES under a varying setting of our indeterministic detection approach.

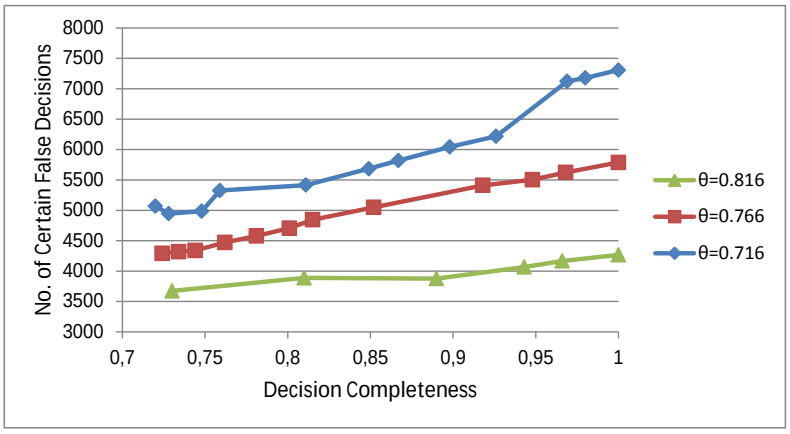
9.8.1. Set-Up

As detection input, we considered a certain database. For this purpose, we used the most probable world of the probabilistic test database $db^{10}_{D_1}$. For computing pairwise duplicate decisions, we used the same configurations for the certain data components as in the previous experiments (see Section 9.5). In order to compute probabilistic duplicate clusterings from these decisions we used our clustering approach that we have proposed in Section 8.7. To enable an efficient computation, we combined an (α, β)-restriction with the HC-restriction that we both have described in Section 8.7.5. For resolving inconsistent *partial M-graphs* we used a simple approach that performs a conventional clustering method with three different threshold settings on this graph. For this purpose, we used the three probability thresholds 0.4, 0.5, and 0.6 and constructed for each of these threshold another Duplicate-Pair Graph. Then we used a variant of Partitioning based on Merged Centers [HCML09] to compute a clustering from each of these graphs. We decided to use more than one probability threshold because otherwise large *partial M-graphs* with many uncertain edges would produce only a single clustering and hence would produce many false decisions. By doing so the underlying idea of the indeterministic detection approach would be ad absurd um.

In this experiment, we restricted the indeterministical handling of duplicate decisions on the set of POSSIBLE MATCHES. In order to utilize process configurations of different quality, we used three different threshold settings. In the first case, we simulated a perfect setting by choosing the similarity threshold $\theta_{Best} = 0.816$ that led to the best Pair F_1-score of the pairwise duplicate decisions In the second and third case, we simulated two worse settings by choosing the non-optimal similarity thresholds $\theta_{Worse} = 0.766$ and $\theta_{Worst} = 0.716$ respectively. For each threshold θ_i, we created a set of POSSIBLE MATCHES by setting the two thresholds $\theta_{U/P}$ and $\theta_{P/M}$ that demarcates the set of MATCHES, UNMATCHES, and POSSIBLE MATCHES to $\theta_{U/P} = \theta_i - 0.15$ and $\theta_{P/M} = \theta_i + 0.15$ respectively. In order to vary the trade-off between decision correctness and decision certainty, we combined each of these three threshold settings with different values for α and β of the used (α, β)-restriction. For mapping similarities to probabilities we used the method that is described in Equation 8.12 and that defines the matching probability of two entities based on the distance of their similarity to the two thresholds $\theta_{U/P}$ and $\theta_{P/M}$. As quality semantics we used the Certain World Semantics because it is most intuitive to the reader. For measuring decision correctness we used the Certain Pair F_1-score and the Number of Certain Pair False Decisions. To measure decision certainty we used the Decision Completeness with setting $\alpha = 1$, $\beta = 0$, and $\gamma = 1$ and hence did not take the Certain Pair True Negatives into account.



(a) Certain F_1-score



(b) Number of Certain Pair False Decisions

Figure 9.10.: Quality of several indeterministic duplicate detection results computed under the Certain World Semantics

All sub-experiments were performed on a machine with an Intel Core i5-3317U Processor, 8GB main memory, and a 64-bit version of the operating system *Windows 7*.

9.8.2. Execution

We conducted several test runs for each of the three threshold settings described above. We start with a run that used $\alpha = \beta = 0.5$ in the (α, β)-restriction and therefore corresponded to a conventional duplicate detection process that uses a single threshold to classify all entity pairs into the set of MATCHES and the set of UNMATCHES. Then we decreased α step by step (note, we always chose $\beta = 1 - \alpha$) and hence gradually decreased decision certainty.

9.8.3. Results

The results of the experiment are depicted in Figure 9.10(a) (Certain Pair F_1-score) and Figure 9.10(b) (Number of Certain False Decisions). Obviously, in the case where we set $\alpha = 0.5$ only a single duplicate clustering was produced and the detection process that used the perfect similarity threshold performed much better in both terms of decision correctness than the detection processes that used one of the other

Settings	$\|M_c\|$	$\|P\|$	$\|p\text{TP}^{CWS}\|$	$\|p\text{FP}^{CWS}\|$	$\|p\text{FN}^{CWS}\|$	$p\text{Rec}^{CWS}$	$p\text{Prec}^{CWS}$	pF_1^{CWS}	$DecComp_{1,0,1}$
$\alpha = 0.5$	10600	0	3836	6764	544	0.876	0.362	0.512	1.0
$\alpha = 0.495$	10469	194	3831	6638	540	0.876	0.366	0.516	0.98
$\alpha = 0.49$	10413	337	3819	6594	530	0.878	0.367	0.517	0.969
$\alpha = 0.48$	9498	760	3787	5711	506	0.882	0.399	0.549	0.926
$\alpha = 0.47$	9304	1052	3759	5545	499	0.883	0.404	0.554	0.898
$\alpha = 0.46$	9078	1390	3753	5325	494	0.884	0.413	0.563	0.867
$\alpha = 0.45$	8963	1589	3758	5205	480	0.887	0.419	0.569	0.849
$\alpha = 0.44$	8677	2017	3744	4933	482	0.886	0.431	0.58	0.811
$\alpha = 0.43$	8573	2718	3725	4848	478	0.886	0.435	0.583	0.759
$\alpha = 0.42$	8324	3180	3723	4601	467	0.889	0.447	0.595	0.72
$\alpha = 0.41$	8192	2758	3680	4512	478	0.886	0.449	0.596	0.748
$\alpha = 0.40$	8156	3043	3677	4479	466	0.888	0.451	0.598	0.728

Table 9.6.: The detection quality that was produced by the test runs using the threshold $\theta_{Worst} = 0.716$

two thresholds. However, the decision correctness that resulted from using the individual threshold settings became closer in terms of the Certain Pair F_1-score as well as in terms of the Number of Certain False Decisions the more of the duplicate decisions were handled indeterministically. Considering a decision certainty of 0.73 the correctness of the certain duplicate decisions was nearly equal for all three thresholds although two of them were poorly set up. Thus, the indeterministic duplicate detection approach was able to compensate the poor choices of the used similarity threshold.

In order to get an impression on the different characteristics of the indeterministic detection results, we list the results of the test runs that used threshold $\theta_{Worst} = 0.716$ in Table 9.6 and Table 9.7. The first table describes the quality of the individual runs and contains the sizes of the sets of certain MATCHES (M_c), POSSIBLE MATCHES (P), Certain Pair True Positives ($p\text{TP}^{CWS}$), Certain Pair False Positives ($p\text{FP}^{CWS}$), and Certain Pair False Negatives ($p\text{FN}^{CWS}$). Moreover it contains the values of the Certain Pair Recall ($p\text{Rec}^{CWS}$), the Certain Pair Precision ($p\text{Prec}^{CWS}$), the Certain Pair F_1-score (pF_1^{CWS}), and the Decision Completeness ($DecComp_{1,0,1}$). The second table describes the complexity of these runs by listing the number of possible worlds (N_{PW}), the number of produced factors (N_F), the number of possible clusters ($N_{\mathfrak{C}}$) which corresponded to the number of database entities that needed to be stored in the output database, and the total number of variable-value pairs (N_{var}) that needed to be stored in the World-Table if we use negation for factors with two possible clusterings.

As we can see the number of possible worlds, the number of required variable-value pairs and the number of output entities increased with an increasing number of POSSIBLE MATCHES. In contrast, the number of factors decreased with a growing number of POSSIBLE MATCHES which is intuitive because the more edges of the initial *M-graph* are uncertain into the less *partial M-graphs* this graph could be decomposed. Moreover, we can observe that the Certain Pair Recall did not change much with a growing uncertainty, but the Certain Pair Precision considerably increased with a growing number of indeterministically handled decisions. This observation actually reflects the idea behind indeterministic

Settings	N_{PW}	N_F	$N_{\mathfrak{E}}$	N_{var}
$\alpha = 0.5$	1	97513	97806	0
$\alpha = 0.495$	5.6×10^{15}	97492	97900	49
$\alpha = 0.49$	5.9×10^{29}	97460	97981	97
$\alpha = 0.48$	6.1×10^{60}	97391	98175	217
$\alpha = 0.47$	2.7×10^{89}	97316	98344	315
$\alpha = 0.46$	8.2×10^{118}	97244	98513	420
$\alpha = 0.45$	5.7×10^{145}	97177	98659	512
$\alpha = 0.44$	3.3×10^{175}	97102	98840	625
$\alpha = 0.43$	3.1×10^{198}	97052	98980	707
$\alpha = 0.42$	7.9×10^{219}	96987	99115	788
$\alpha = 0.41$	5.3×10^{243}	96915	99246	870
$\alpha = 0.40$	2.4×10^{271}	96839	99395	976

Table 9.7.: The complexity of the conducted test runs that used the threshold $\theta_{Worst} = 0.716$

duplicate detection because the goal of this concept is to model only those decisions deterministically the detection process is confident about.

An interesting observation is that decreasing α did not necessarily increased the number of POSSIBLE MATCHES and did not necessarily increased uncertainty. For instance the result of using $\alpha = 0.42$ was more uncertain than the result of using $\alpha = 0.41$. This can be explained by the fact that we generate *W-graphs* by using the HC-restriction mapping $f^{\text{HC}}_{MG \mapsto WG}$ and in this case adding an additional uncertain edge can lead to the circumstance that more of the generated *W-graphs* become inconsistent. Another interesting aspect is that the number of modeled possible worlds is already above 10^{15} if we only model 194 entity pairs in an indeterministic way (i.e. $|P| = 194$). These numbers perfectly illustrate why a factorization of the deduplication result into small factors is inalienable. Recall that without factorization, we would require to store one variable-value pair in the World-Table per possible world and it is obvious that storing 2.4×10^{271} of these pairs is not practical.

It is interesting to note that even in the most uncertain case, the number of POSSIBLE MATCHES is not very large (around 3000). This number is small enough that it can be manually reviewed by a domain expert. Of course, such reviews need time, especially if each decision should be made with confidence. This again demonstrates that an indeterministic duplicate detection result can be perfectly used as a prompt answer that is instantly queryable. The indeterministically modeled decisions can then be resolved by the expert from time to time.

9.8.4. Conclusion

The results of this experiment demonstrate that our indeterministic duplicate detection approach is suitable to compute an adequate trade-off between decision certainty and decision correctness. Moreover, these results show that the correctness of the certain duplicate decisions became much more robust against a bad configuration of the detection process if we admit a small amount of uncertainty. At this point we have to comment that the used measure for decision certainty, i.e. the Decision Completeness,

was very harsh because it did not take the Certain Pair True Negatives into account. The most other certainty measures such as the Answer Decisiveness only decreased to a small extent, i.e. even when the Decision Completeness were down to 0.73 the Answer Decisiveness of the factorized representation were still above 0.99. However, we decided to use the Decision Completeness because it is tailor-made for the Certain World Semantics and it is more intuitive to the reader.

Another important conclusion is that in all test runs, modeling the indeterministic detection result could be accomplished with a moderate level of complexity. For instance, we required less than thousand variable-value pairs of the World-Table in each of the conducted runs. Moreover, the number of resultant database entities was only marginally larger than the number of entities that resulted from the deterministic run. More exact, the increase of output entities was less than $1,589$ (which corresponds to 1.6%) in each test run.

Chapter 10

Related Work

In this chapter, we present related work and discuss the differences between these works and the contributions of this thesis. We start with related work on deterministic duplicate detection in certain databases in Section 10.1. Then we continue with related work on deterministic duplicate detection in probabilistic databases in Section 10.2 and consider related work on indeterministic duplicate detection in Section 10.3. Finally, we present research that is related to this thesis by any other mean in Section 10.4.

10.1. Deterministic Duplicate Detection in Certain Databases

Deterministic duplicate detection in certain relational databases has been considered in a plethora of works. The most of these approaches focus on detecting duplicates within a single database table (for overviews see [EIV07, NH10, Tal11, Chr12]). Nevertheless, a variety of newer proposals [PMM+02, MW04, PD04, DHM05, BG06, SD06, WN06b, BG07b, BG07c, HSM08, RDG11, HNST12] utilize information on relationships between database entities in order to improve the effectiveness and/or the efficiency of the detection process. In our HaDDeF framework we adopted the concept of description-based entity matching from Herschel et al. [WN06b, NH10]. Nevertheless, the approach that we have proposed for modeling relationship information by using database queries is more generic than any of the modeling approaches that are known to us. As a consequence, our approach enables a more flexible and more powerful modeling of entity relationships which can be useful for duplicate detection in complex databases. Moreover, to the best of our knowledge there is no detection approach that considers cases where one database entity belongs to multiple entity types and thus is represented by tuples in several entity tables. Such a scenario, however, is typical if the considered database schema contains inheritance hierarchies and is especially interesting in probabilistic databases where the membership of entities to entity types can be uncertain. Finally, we do not know a detection approach that uses a concept which is similar to the presented idea of impact values. On the other hand we do not consider the concept of collective duplicate detection in this thesis and instead consider it as a topic for future research.

10.2. Deterministic Duplicate Detection in Probabilistic Databases

Lian and Cheng [LC10, LC11b] propose an approach for performing similarity joins on mutually independent probabilistic sets. As join criterion they use a probability threshold in addition to a similarity threshold and join all two probabilistic sets that are more similar than the given similarity threshold with a probability that is greater than the given probability threshold. We already discussed this simple uncertainty resolution method as well as its drawbacks in Section 7.2.6. Similarity join computing is often considered as a light version of duplicate detection. However, a serious shortcoming of using such a similarity join approach for duplicate detection is that a similarity of zero is assumed for all possible worlds in which one of the two compared entities does not exist. Recall that we extensively discussed this challenge in Section 7.1.3.2 and that this circumstance was the main reason for introducing world impact values. Finally, using a similarity join approach for duplicate detection is only meaningful for single BID-tables, but probabilistic databases can consists of several tables and can contain complex tuple correlations. Of course, the same deficits hold for other similarity join approaches including those that are described by Kriegel et al. [KKPR06].

Feng et al. [FWLG13] present an approach for deterministic duplicate detection in single BID-tables without maybe blocks that is based on the similarity join principle and that uses the same resolution method than has been used by Lian and Cheng [LC10]. This means that they match two x-tuples in two steps. First they compute the accumulative probability that the similarity between the x-tuples' alternatives is above a given similarity threshold. Then they classify these x-tuples as a MATCH if this probability is above a given probability threshold and classify them as an UNMATCH else. For similarity computation, they convert the x-tuple alternatives into q-grams and use the Jaccard Coefficient as similarity measure. For reducing runtime they use pruning techniques that are similar to the techniques that are used by Lian and Cheng. To combine the pairwise matching results they use a density-based clustering approach that utilizes a probabilistic distance measure. In general, this approach has a strong correlation to the approach of performing similarity joins on probabilistic sets that has been proposed by Lian and Cheng [LC10] and therefore suffers from similar shortcomings. Moreover, Feng et al. do not consider uncertainty on table membership and therefore do not respond to many of these challenges that we have addressed in this thesis.

Entity resolution for probabilistic data has been considered by Ayat et al. [NAV12]. Nevertheless, they consider entity resolution as a kind of entity identification where a single *query entity* is matched with a complete database in order to find the database's entity that is most similar to the query entity. Ayat et al. adapt this entity identification problem to BID-tables by searching for the BID-table's entity that is most probably most similar to the query entity, i.e. to find the most-probable match. For similarity computation, Ayat et al. distinguish between context-free similarity measures and context-sensitive similarity measures. Whereas the results of context-free similarity measures only depend on the compared data (e.g. the Levenshtein Similarity), the results of context-sensitive similarity measures also depend on the values of the remaining database entities (e.g. a similarity measure that uses TF-IDF for weighting the individual tokens). They present a PTIME algorithm for computing the most-probable match if a context-free similarity measure is used and propose a Monte-Carlo approach for approximating the most-probable match if a context-sensitive similarity measure is used. For improving performance they

additionally propose a context-sensitive similarity measure that is particularly suitable for their purpose because it requires only less set-up time. Moreover, they propose a parallel version of their Monte-Carlo algorithm using the MapReduce framework.

In [AAAV13] Ayat et al. propose a decentralized version of their entity resolution approach that works on a BID-table which is fragmented over a large number of nodes in a distributed system.

Since Ayat et al. rather consider entity identification than duplicate detection, their proposed methods are only partially suitable for detecting duplicate database entities. Moreover, they restrict their considerations to single BID-tables and therefore do not address any of the challenges that emerge in the presence of multi-table memberships or tuple correlations. In contrast, a parallel computation of duplicate detection in a single database as well as detecting duplicates within a distributed database are topics that we do not have addressed in the main part of this thesis, but consider them as important goals of future research (see Section 11.2).

In [MBGM06] Menestrina et al. propose a generic entity resolution approach that follows the principles of the Stanford Entity Resolution Framework and that works on tuples that are ascribed with confidence values (recall from Section 4.3.10.2 that SERF approaches perform the matching phase and the merging phase in an interleaved manner). In this approach they abstract from particular matching methods and abstract from particular merging methods, but instead consider them as black-boxes. Nevertheless, they assume that the used matching method does not take confidence values into account, but instead makes its duplicate decisions only based on the similarity of the compared attribute values. However, although probabilities were not very valuable in our experiments that we have described in Section 9 we think that in other domains probabilities can provide useful information that can help to increase the quality of the made duplicate decisions. Moreover, since Menestrina et al. consider all input tuples to be independent[1] their input database corresponds to a TI-database. More complex representation systems are not covered by their approach.

In summary, there is currently no research that addresses a deterministic detection of duplicates in complex probabilistic databases as we do it in this thesis.

10.3. Indeterministic Duplicate Detection

Although many research studies focus on duplicate detection only a handful of proposals considers the challenge of indeterministic duplicate detection. To the best of our knowledge the studies presented by Beskales et al. [BSIBD09, BSI+10, Bes12], Ioannou et al. [INNV10, INNV11, Ioa11], and de Keijzer and van Keulen [vKdKA05, dKvKL06, dKvK07b, dKvK08, dK08, vKdK09, dK10b] are the only proposals that handle uncertain duplicate decisions in an indeterministic way. Since each of these studies takes another path for addressing the problem of modeling uncertain duplicate decisions, we will discuss them in more detail.

[1] This circumstance is not explicitly stated in this work, but they merge probabilities by computing their product and no contrary assumption on tuple correlations is made.

10.3.1. Modeling and Querying Possible Repairs in Duplicate Detection

Beskales et al. [BSIBD09, BSI+10, Bes12] consider duplicate detection as a data cleaning task and present an approach for modeling *possible duplication repairs* along with the process configurations that produce these repairs within a probabilistic database. The underlying idea of this approach is that the user becomes enabled to specify a cleaning requirement, i.e. the used process configuration, at query time instead of cleaning time. Moreover, by using several configurations in the query condition she can specifically ask for uncertainty in the cleaning result.

Although duplicate detection processes typically have several configuration parameters, Beskales et al. restrict their considerations to a single parameter, i.e. the similarity threshold that is used to demarcate the set of MATCHES from the set of UNMATCHES. They consider this parameter as a random variable where in the discrete case each plausible setting is ascribed with a probability. Of course, in the continuous case a probability density function is used instead.

To compactly store a set of possible repairs, they propose a representation system called *U-clean Relation* that is tailor-made for their purpose. A U-clean relation is defined as a set of c-records where each c-record represent a possible cluster. In addition to the attributes of the considered input database table, a U-clean relation contains the two additional attributes C and P. Attribute C stores for every c-record the identifiers of the input tuples that form the duplicate cluster that is represented by this c-record. In contrast, attribute P stores for every c-record the parameter settings of the detection algorithm that produce the duplicate cluster which is represented by this c-record. For instance, if the parameter setting is considered as a real-valued threshold, attribute P contains a set of intervals.

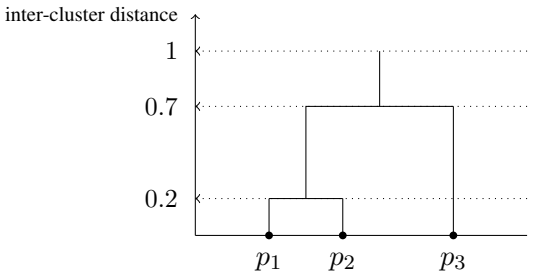
It is interesting to note that because the configuration parameter is considered as a random variable, attribute P has a the same functionality as the conditions in pc-tables. Thus, given a specific parameter setting we can compute all c-records that belong to the setting's corresponding repair, i.e. the repair contains each c-record whose value in the attribute P contains the considered parameter setting. Since each parameter setting is associated with a probability, we can compute a probability for each c-record based on its value in attribute P.

For constructing a U-clean relation from an unclean input database table, Beskales et al. utilize hierarchical clustering algorithms. They start with one c-record for each input tuple and then create a new c-record for each new cluster that results from merging two existing clusters. The initial c-records only contain a single value in attribute C, i.e. the identifier of the corresponding input tuple, and contain the complete parameter domain in attribute P. If a new c-record r_3 is created by merging two existing c-records r_1 and r_2 because of the parameter setting τ, its value in attribute C is set to the union of the corresponding values of r_1 and r_2. Its value in attribute P is set to all values of the parameter domain that are equal to or larger than τ. The values of r_3 in the remaining attributes, i.e. the attributes of the unclean input table, are computed by merging the corresponding values of the input tuples that are represented by r_3. Since each input tuple can only be represented by one c-record per possible parameter setting, the values of r_1 and r_2 in the attribute P are changed by capping them with τ, i.e. they are not allowed to coexist with c-record r_3.

Finally, Beskales et al. present mechanisms for evaluating database queries on U-clean relations according to the possible worlds semantics, and propose some new query types that can be used to retrieve uncertainty aspects from the probabilistic cleaning results. If uncertainty is not resolved by the use of

ID	name	price
p1	X	5
p2	Y	10
p3	Z	15

(a) Unclean input table *'Product'*



(b) Hierarchical clustering

ID	name	price	C	P
r1	X	5	{p1}	[0,0.2]
r2	Y	10	{p2}	[0,0.2]
r3	Z	15	{p3}	[0,0.7]
r4	XY	7.5	{p1,p2}	[0.2,0.7]
r5	XYZ	10	{p1,p2,p3}	[0.7,1.0]

(c) U-clean relation *'Product^C'*

price	C	P
5	{p1}	[0,0.2]
7.5	{p1,p2}	[0.2,0.7]
10	{p2} ∨ {p1,p2,p3}	[0,0.2] ∪ [0.7,1.0]
15	{p3}	[0,0.7]

(d) U-clean relation *'Price^C'*

Figure 10.1.: Example for generating and querying U-clean relations

aggregation, from querying a U-clean relation another U-clean relation results. This U-clean relation contains one c-record per possible query answer. Interestingly in querying a U-clean relation the values of attribute P are used in the same way than the tuple conditions are used in querying a pc-database. Therefore, if two c-records r_1 and r_2 provide the same query answer r_3, i.e. r_3 results from each possible world that contains r_1 or r_2, the value of r_3 in P corresponds to the set union of $r_1[P]$ and $r_2[P]$. In contrast, if c-record r_3 results from joining the two c-records r_1 and r_2, r_3 belongs to the query answer of every possible world that contains r_1 and r_2. Thus, in this case the value of r_3 in P corresponds to the set intersection of $r_1[P]$ and $r_2[P]$.

Example 221 *For illustrating the idea of U-clean relations, we consider the unclean database table* 'Product' *that is presented in Figure 10.1(a) and consider a normalized distance threshold with the continuous range* $[0, 1]$. *Moreover, we consider the hierarchical clustering result that is depicted in Figure 10.1(b). The resultant U-clean relation 'Product^C' is presented in Figure 10.1(c) and contains five c-records. The lowest distance between two input tuples is* 0.2. *For that reason, for all threshold settings within the range* $[0, 0.2]$ *each input tuple form its own cluster (c-records* r_1, r_2, *and* r_3*). For a threshold setting within the range* $[0.2, 0.7]$ *we have two duplicate clusters: One that contains* p_1 *and* p_2 *(c-record* r_4*) and one that contains* p_3 *(c-record* r_3*). Finally, for a threshold setting within the range* $[0.7, 1.0]$ *we only have a single duplicate cluster that contains all three input tuples (c-record* r_5*).*

For illustrating query evaluation, we consider a query that requests for all product prices. Since it is only interested in the prices, it eliminates duplicate answers. The result of this query is the U-clean relation 'Price^C' that is presented in Figure 10.1(d). Note, because c-record r_2 *as well as c-record* r_5 *contains the value* 10 *in the attribute* 'price'*, the price* 10 *is an answer for all threshold settings within the range* $[0, 0.2]$ *and is an answer for all threshold settings within the range* $[0.7, 1.0]$.

The benefit of the approach proposed by Beskales et al. is that their representation system allows a connection between the used settings of the similarity/distance threshold and the modeled decision uncertainty so that a threshold setting can be selected afterwards which is unprovided by our approach. However, it should not be too difficult to adapt our approach to a similar concept. For this purpose we need to create an extra table of the output database. This table contains one tuple per considered setting of the similarity threshold and this tuple is conditioned by the same variable-value pairs than the possible clusters that result from using this threshold setting. By doing so, we only need to preserve the connection between the possible clustering and the used similarity threshold settings during the duplicate detection process. A trivial option is to run a fix number of detection processes that each uses another threshold setting. Another option is to adapt the HC-restriction strategy that we have proposed in Section 8.7.5 in a way that we store for each *W-graph* the threshold setting that leads to the generation of this *W-graph*.

One main characteristics of the approach proposed by Beskales et al. is that they use a hierarchical clustering algorithm to compute the uncertain detection output. As presented in Section 8.7.5, our duplicate clustering approach for computing indeterministic duplicate detection results can be adapted to hierarchical clusterings, too. In contrast, since our approach is not restricted to hierarchical clusterings, it is more general.

Moreover, their approach is restricted to incorporating the uncertainty on only a single configuration parameter into the detection output whereas in our description-based detection approach any kind of process uncertainty can be incorporated into the detection result. Furthermore, we consider probabilistic databases as input and therefore address an additional source of uncertainty. Finally, they use a representation system that is tailor-made for their purpose. Using this system is particularly valuable to query a probabilistic cleaning result, but seems to be limited in many other application scenarios. In contrast, our approach is based on using conventional representation systems such as BID-databases or pc-databases. As we think, this increases the reusability of the resultant database, especially if deduplication is considered as a step in a data integration process.

10.3.2. Entity-Aware Query Processing in the Presence of Linkage

An interesting approach for indeterministic duplicate detection has been proposed by Ioannou et al. [INNV10, INNV11, Ioa11]. In order to capture decision uncertainty, they store possible linkages in the output database. For this purpose, they propose a new representation system called *probabilistic linkage database* that combines the ideas of probabilistic databases and linkage databases. A probabilistic linkage database contains a set of entities that can be connected by linkages where each linkage can be ascribed with a probability. Moreover, each of the entities' attribute values can be ascribed with a probability. From the theoretical viewpoint, a probabilistic linkage database models a probability distribution over a set of *possible linkage worlds* (short *possible l-worlds*) where a particular possible l-world can be computed by choosing a particular linkage specification, i.e. by selecting a subset of all stored linkages. Since computing a possible l-world only means to resolve uncertainty on linkages but not on attribute values, a possible l-world represents a probability distribution over a set of possible worlds each a conventional database instance with linkages. A particular possible world is computed by selecting a fraction of all uncertain attribute values.

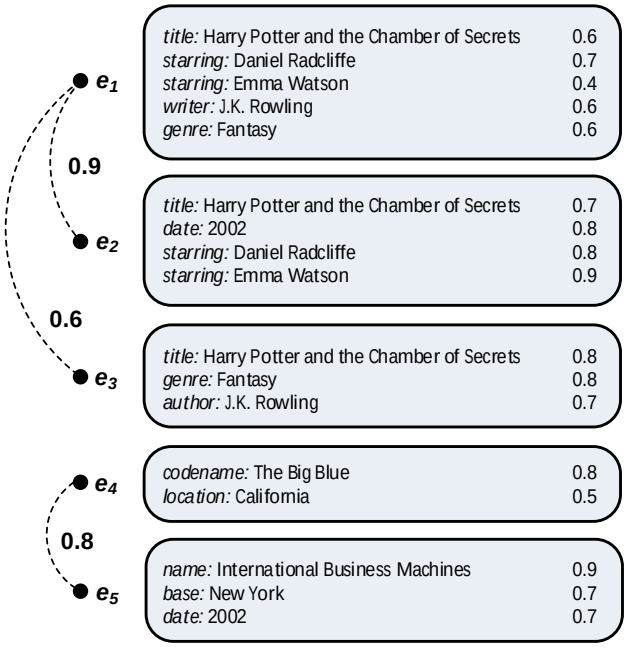


Figure 10.2.: Probabilistic linkage database *PLDB* (adopted from [INNV10])

The result of posing a query to a probabilistic linkage database is defined according to the possible worlds semantics. Thus, an entity is a possible query answer if it satisfies the query conditions and if it belongs to the conventional database instance that results from merging linked entities within at least one possible world of at least one possible l-world. For evaluating queries on probabilistic linkage databases they developed a mechanism that uses the linkage information of the database in order to decide at query time (on the fly) which of the query relevant tuples are duplicates and have to be merged. For this purpose, the mechanism enumerates all possible worlds, merges the linked entities by a predefined merge function, computes the probabilities of all entities, and then returns all entities (along with their probabilities) that satisfy the query conditions. Of course, an enumeration of all possible l-worlds of a probabilistic linkage database typically does not scale well. Nevertheless, the most entities are not linked with each other. For that reason, the query mechanism initially divides the input database into its independent factors and then processes each of these factors independently. Since a query typically only addresses a small part of the database, the mechanism is able to massively reduce query evaluation time by only considering factors that contain at least one entity per query condition that satisfies this query condition. Furthermore, the mechanism uses some additional query-specific properties to reduce the evaluation effort further on.

Example 222 *For illustrating the idea of probabilistic linkage databases we consider a simple example that is presented in [INNV10]. The corresponding probabilistic linkage database PLDB* $= \langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$ *is depicted in Figure 10.2 and contains the five database entities* $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$ *as well as the three linkages* $\mathcal{L} = \{\{e_1, e_2\}, \{e_1, e_3\}, \{e_4, e_5\}\}$*. Each database entity describes a movie and has several attributes. The value in each attribute can be uncertain. For instance, the probability that movie* e_1 *is based on a novel written by 'J.K. Rowling' is* 0.6*. The probabilities of the individual attributes are described by the probability mass function* p^a*, i.e.* $p^a(e_1[\textit{writer}] = \text{'J.K. Rowling'}) = 0.6$*. Each*

linkage is ascribed with a probability and represents the case that the two linked entities are duplicates. For example, the two entities e_4 *and* e_5 *are duplicates with a probability of* 0.8*. The probabilities of the individual linkages are described by the probability mass function* p^l*, i.e.* $p^l(e_4, e_5) = 0.8$*.*

The probability of a linkage describes the probability that the two linked entities are duplicates. Thus, if we consider the linkage information of a probabilistic linkage database as a undirected weighted graph where entities are nodes and linkages are edges that each is weighted with the linkage's probability, we see that there is a high similarity between the approach that has been proposed by Ioannou et al. and the clustering approach that we have proposed in Section 8.7. However, there are several differences:

- In probabilistic linkage databases, data uncertainty can only be modeled by ascribing probabilities to the individual attribute values. Thus, modeling correlations between individual attribute values or modeling correlations between individual entities is not possible. Moreover, this representation system does not even allow to model several mutually exclusive values per attribute. As a consequence, compared to BID-databases or pc-databases, probabilistic linkage databases have a low modeling power.

- The query mechanism proposed by Ioannou et al. only supports simple queries that consists of projections and selections. In contrast, our approach is based on the usage of conventional probabilistic representation systems for which efficient querying has been researched in an exhaustive way (see Section 3.5). Thus, complex queries with joins or subqueries can be efficiently performed on the deduplicated database. Moreover, as we have demonstrated in Section 8.5.2 even several types of aggregate queries can be efficiently evaluated on indeterministically deduplicated databases if these databases only contain entity correlations that originate from modeling uncertain decisions.

- Using a probabilistic linkage database or using a conventional probabilistic database results in two fundamentally different approaches for handling uncertain decisions in deduplication. Ioannou et al. pairwise match the given database entities offline, store the most uncertain duplicate decisions, i.e. linkages, persistently in a probabilistic linkage database, and finally construct the possible worlds of the deduplciated database at query time. In contrast, like Beskales et al. [BSIBD09] we do not store the individual matching results (entity similarity or matching probability) persistently, but instantly construct a possible world space based on the made duplicate decisions, and finally store this world space within a probabilistic database. As a consequence, in the approach proposed by Ioannou et al. queries are evaluated on stored linkages, but in our approach queries are evaluated on stored possible worlds.

- The query mechanism proposed by Ioannou et al. selects a subset of all linkages and then computes duplicate clusters by building the transitive closure of the selected linkages. This approach, however, poses a fundamental problem because it disregards negative information (certain non-duplicate decisions), which makes the detection result more inaccurate and which makes an introduction of negative expertise during the offline performed matching phase impossible.

  We want to illustrate this shortcoming based on the probabilistic linkage database that is presented in Figure 10.2. Moreover, let us assume that the entities e_2 and e_3 are not linked because during

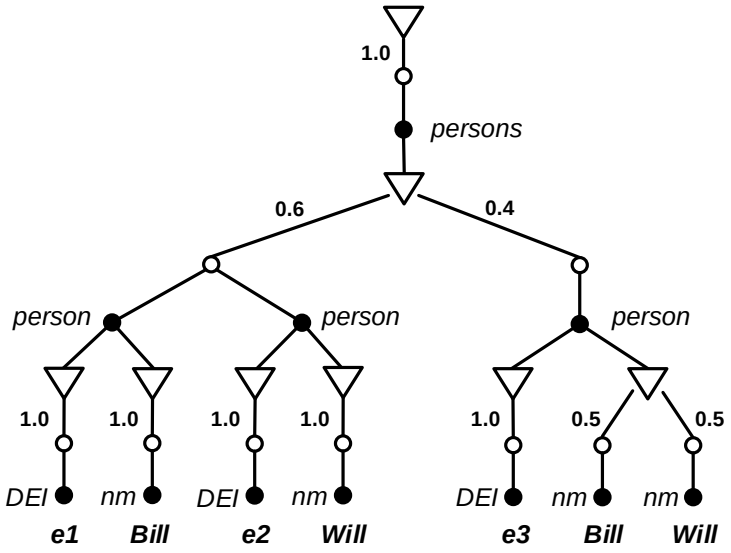


Figure 10.3.: Probabilistic XML tree

the offline performed entity matching a domain expert knew with absolute confidence that both entities are no duplicates ($\Rightarrow p^l(e_2, e_3) = 0$). However, from performing the transitive closure for the linkage specification $\mathcal{L}^{sp} = \mathcal{L}$ the possible l-world $W = \{f_\mu(\{e_1, e_2, e_3\}), e_4, e_5\}$ results (compare this l-world with the possible l-world I_2 that is considered in Example 2 in [INNV10]). In this possible l-world the entities e_2 and e_3 belong to the same duplicate cluster which contradicts the introduced knowledge that they certainly refer to different real-world entities. Thus, introducing the negative knowledge of domain experts during the offline performed matching phase was in vain. To enable an incorporation of negative information on duplicate decisions, our clustering approach considers all matching results (whether possible or not) for constructing the *M-graph*, then creates all possible combinations of uncertain edges (i.e. worlds), and finally remove impossible ones.

10.3.3. Data Integration with Probabilistic XML

In [vKdKA05, dKvKL06, dKvK07b, dKvK08, vKdK09, dK10b] de Keijzer and van Keulen propose an XML approach for modeling indeterministic deduplication results. For this purpose, they introduce a probabilistic XML representation system that uses *probability nodes* and *possibility nodes* to model complex correlations between uncertain XML elements. A sample probabilistic XML tree is presented in Figure 10.3. In such a tree, a probability node (symbol $\triangledown$) models a probabilistic value and a possibility node (symbol $\circ$) models an alternative of such a probabilistic value. As a consequence, each probability node has solely possibility nodes as child nodes and each possibility node has a probability node as parent node. Furthermore, each possibility node has ordinary XML nodes (symbol $\bullet$) as child nodes. Because possibility nodes corresponds to alternatives of a probabilistic values, they are ascribed with probabilities. Moreover, because each two sibling possibility nodes represent mutual exclusive events

and because the case of non-existence is represented by an empty possibility node, the probabilities of all child nodes of each probability node sum up to one.

Example 223 *For illustration, we reconsider the simple probabilistic XML tree that is presented in Figure 10.3. The root of this three is a probability node with only one child node. Thus at this level, the tree contains no uncertainty. The corresponding subtree represents a sequence of persons. Since the root of this subtree has two possibility nodes as child nodes, the tree models uncertain information on the correct instance of this sequence. In the first case (probability* 0.6*), the sequence contains two persons (*e_1 *and* e_2*). In the second case (probability* 0.4*), the sequence consists of a single person (*e_3*). The information on the two persons within the first possible sequence is certain. In contrast, the value for the attribute* 'nm' *of the single person in the second possible sequence is uncertain and is either 'Bill' (probability* 0.5*) or 'Will' (probability* 0.5*). In conclusion, this probabilistic XML tree models two possible worlds.*

It is obvious that the presented possible world space can be considered as an indeterministic deduplication result, because we are uncertain whether we have the two distinct persons e_1 *and* e_2 *with different names (probability* 0.6*) or we have one person* $e_3 = f_\mu(e_1, e_2)$ *that has two equally probable alternative name values (probability* 0.4*).*

As application area they consider an integration scenario in which only two sources are integrated at a time. Thus, their detection of duplicates actually corresponds to a coupling of XML elements. Recall from Section 4.3.10.5 that data coupling does not produce duplicate clusters, but instead produces a 1:1 mapping from elements of the first source to elements of the second source. Thus, whereas our clustering approach computes a set of possible duplicate clusterings, their approach considers a computation of all plausible 1:1 mappings between the elements of both sources. These mappings as well as the corresponding matching probabilities are computed by a rule engine that is able to utilizes knowledge rules in order to meaningfully reduce the number of plausible 1:1 mappings. Finally, they propose techniques for exploiting user feedback on query results for improving the factual correctness of the indeterministic coupling results.

An essential contribution of this thesis is our study about the meaning of detection quality in the presence of uncertain duplicate decisions. In this context, we used several existing uncertainty measures such as the Shannon Entropy [Tan01] or the Answer Decisiveness [dKvK07a] for rating the uncertainty of a probabilistic duplicate clustering. An adaptation of measures for decision correctness to sets of uncertain duplicate decisions, however, has only been considered by de Keijzer and van Keulen [dKvK07a]. Nevertheless, they restrict these considerations to a measure that computes the expected Pair Recall and a measure that computes a kind of expected Pair Precision[2]. Both measures correspond to an adaptation of Pair Recall and Pair Precision under the aggregated world semantics. However, they do not consider the dependency between the meaning of quality and the intended use of the database. As a consequence, they do not distinguish between the different quality semantics that we have presented in Section 8.8.1. Finally, we do not know a proposal that considers a factor-based computation of quality measures that is comparable to the approaches we have contributed in Section 8.8.5.

[2] Actually this measure computes the expected Number of True Positives divided by the expected Number of Positives. This computation does not exactly correspond to the expected Pair Precision, but can be computed without enumerating all possible worlds.

10.4. Further Related Work

A usage of uncertain databases in general and probabilistic databases in particular within the context of data integration has been considered in several publications. However, the most of these works focus on uncertainty in schema matching [Gal06a, MG07, GMSS09, DHY09, SDH09, Gal11b, Gal11a, AAAV12]. Especially Magnani and Montesi [MRMM05, MR06, MM07, MM08b, MM09b, MM09a, MM10a, MM10b] provided a lot of useful insights into this matching challenge. Cali and Lukasiewicz [CL08] consider schema matching uncertainty in data integration for the Semantic Web.

Agrawal et al [ASUW10] propose an approach for integrating uncertain databases that has been extended to probabilistic databases by Sadri [Sad12]. Nevertheless, both approaches focus on an answering LAV-queries and abstract from details on detecting duplicate database entities. In this context, Borhanian and Sadri [BS13] propose a new representation system that is particularly suitable for admitting efficient integration of probabilistic databases.

Beskales et al. [BSI08] presented an approach for computing the top k probable nearest neighbors in uncertain databases. Since duplicate cluster are not restricted to two entities, i.e. each entity can be a duplicate of several other entities, such a computation seems only less valuable for detecting duplicates in probabilistic databases. However, it may be useful in data coupling or entity identification scenarios. Similar holds for the graph-based approach that is proposed by Potamias et al. [PBGK10].

Miller et al. [AFM06, HM09] propose approaches for creating probabilistic databases from unclean certain databases by merging duplicate entities using a conflict repairing strategy. Other approaches that produce probabilistic output data by using conflict repairing strategies are presented by DeMichiel [DeM89] and Tseng [T$^+$92, TCY93]. Lim et al. [LSS96b] propose a similar approach that is based on the Dempster-Shafer theory instead of probability theory.

Chapter 11

Conclusion & Outlook

In this chapter, we conclude this thesis and recap its contributions in Section 11.1. Finally, we give an outlook on open challenges in Section 11.2.

11.1. Conclusion

In many application areas such as astronomy [SCH09], forensics [SKSM08, HHvK13], information extraction [GS06, DBS09, WMF+10, vKH11], data integration [DHY09, HM09, BBC+11, Gal11a, ZCJC13], bioinformatics [DGL+09, LS09, ZLGZ10, PBGK10], business intelligence [SWF+09], or risk management [AJP+10, JXW+11, XBE+09], there is a growing demand for support for flexible and robust handling of uncertain data. Although data cleaning activities can be potentially used to resolve uncertainty in the data, at the same time imperfect circumstances continuously produce new uncertain data. For that reason, flexible and robust handling of uncertain data means to regard uncertainty as a 'fact of life' and all data management functions should be able to support uncertainty in their input as well as their output [MM10a, vK12]. In order to meet such demands, probabilistic databases [Gre09, SBH+09, SD09, AKO09, SORK11] have been developed by the database research community in recent years. They do so, by attributing probabilities to a set of mutually exclusive database instances and offer functions to allow reflecting these probabilities in all CRUD operations.

As conventional databases that handle certain data, a probabilistic database lacks consistency if it contains several database entities that refer to the same real-world entity as duplicates can lead to incorrect query results. Moreover, it can be beneficial to integrate data from several uncertain sources [Len02, LN06, HDI12] because each source can contain data that is not present in the others and certainty as well as correctness can be increased by combining overlapping data. Of course, an integration is only meaningful if we are able to link the data from the individual sources that describe the same real-world entity. As a consequence, duplicate detection is an important component for cleaning an inconsistent database and for integrating data from several autonomous sources. However, conventional duplicate detection approaches [EIV07, NH10, Chr12] are only designed for processing database entities that are described by certain values and certainly belong to the considered universe of discourse. In probabilistic databases, however, each database entity can have several alternative values per attribute

and the membership of an entity to a universe can be questionable. For all these reasons, conventional duplicate detection approaches cannot be used for probabilistic databases without adaptation.

In this thesis, we tackled the problem of duplicate detection in probabilistic relational databases where we identified the following challenges:

Type mismatch

The problem of duplicate detection is entity-based because we search for database entities that refer to the same real-world entity. However, the relational data model in general and its probabilistic extensions in particular are only partially entity-based because not every database tuple represents a real-world entity and it can be the case that a real-world entity needs to be represented by tuples in several tables. This type mismatch becomes even more significant in the presence of uncertainty because in this case a real-world entity maybe needs to be represented by multiple tuples even within the same table. Thus, in (probabilistic) relational databases the semantical concept of database entities is not evident. Existing proposals on duplicate detection are typically restricted to single database tables so that the aforementioned type mismatch does not matter. Moreover, the small number of detection approaches that have been developed with multi-table databases in mind only consider information on relationships between real-world entities, but do not consider cases where one real-world entity is represented by tuples in several tables as it can happen if the underlying database schema contains inheritance hierarchies. As a consequence, the type mismatch between the tuple-based relational data model and the entity-based problem of duplicate detection needs to be resolved before we can define a detection approach for complex probabilistic relational databases.

Representation heterogeneity

A probabilistic database is theoretically defined as a probability distribution over a set of conventional database instances that are called *possible worlds*. In practice, however, a separate storage of all these possible worlds is not feasible and a more compact representation is required. To this day, a variety of probabilistic representation systems has been defined and most of these systems differ in their compactness, their modeling power, and their representation/query complexity. The existence of different representation systems cumbers duplicate detection because the same information can be represented in different ways. Therefore, our detection approach has to deal with a type of representation heterogeneity that is not known from the processing in certain databases.

Complex correlations between entities, tuples, and attribute values

Many probabilistic representation systems do not only allow a specification of uncertainty on attribute value level, but also allow a specification of uncertainty on tuple level. Moreover, several representation systems enable a modeling of correlations between the uncertainty of different values or the uncertainty of different tuples. From this it follows that (real-world) entities can be represented by several tuples which belong to different, maybe semantically disjoint, tables. For example, a person that is either a student or a professor is represented by a tuple in the table *'Student'*, but is also represented by a tuple in the table *'Professor'*. Nevertheless, tuple correlations are not restricted to tuples representing the same (real-world) entity, but can also concern tuples that represent different (real-world) entities. For instance, we can model the information that a

person is only a student if another person is a student as well. All these circumstances make duplicate detection in probabilistic databases especially interesting if we consider complex databases that consists of multiple tables. However, in such cases a probabilistic database cannot only contain errors in its data values and in its probabilities, but can also contain errors in the existence of tuples and can contain errors in the correlations between tuples (representing the same entity or representing different entities) which can considerably complicate a detection of duplicates.

Incorrect dependencies from table membership

In the presence of tuple-level uncertainty the existence of an attribute value is always conditioned by the existence of its corresponding tuple. However, if the corresponding database table represents a particular type of entities, e.g. the type of all students, the existence of a tuple only models the membership of a real-world entity to this entity type, e.g. the entity is a student, but the existence of some of the entity's attribute values can actually be independent from this membership, e.g. a person has an age regardless she is a student or not. Since this aspect comes not into effect in certain relational databases, it has not been considered in conventional approaches for relational schema design. As a consequence, if the schema of a probabilistic database has been designed by using such a conventional designing approach, it can happen that the existence of attribute values incorrectly depend on table memberships.

Deterministic duplicate detection vs. indeterministic duplicate detection

Duplicate decisions are often tainted with uncertainty because we cannot generally infer real-world equivalence from high data similarity, but can only use the latter as an indication of the first. This is especially so if the input data is uncertain and/or an appropriate configuration of the duplicate detection process is unknown. Nevertheless, if we use a probabilistic database as deduplication output it seems natural to store emerging decision uncertainty within the output database rather than to resolve it. However, many probabilistic representation systems are not powerful enough to model uncertainty on duplicate decisions. For that reason, we have to distinguish between deterministic duplicate detection approaches and indeterministic duplicate detection approaches. Whereas each of the former produces a single duplicate clustering as a result by resolving uncertainty on duplicate decisions at some moment of the detection process, each of the latter incorporates decision uncertainty into the detection result by producing a probabilistic duplicate clustering as output. Both detection concepts pose a set of further challenges because it is not clear at which moments we can resolve decision uncertainty by which methods and it is not clear in which way we can efficiently compute and represent a probabilistic duplicate clustering.

In this thesis, we addressed all the above challenges. In doing so, we focus on developing a generic approach that allows an adjustment to individual needs and hence can be used in a variety of application domains. The contributions of this thesis can therefore be summarized as follows.

Entity-based interpretation of (probabilistic) relational data

Due to the aforementioned type mismatch between the entity-based problem of duplicate detection and the tuple-based concept of relational databases, we introduced an entity-based interpretation of certain relational databases and extended this interpretation to probabilistic relational databases

by using the possible worlds semantics. In this context, we discussed different approaches for modeling inheritance hierarchies within compact probabilistic representation systems.

Framework for deterministic duplicate detection in certain databases

Most research on duplicate detection focuses on single-table databases. Moreover, although these proposals have many commonalities they differ in some fundamental aspects. For that reason, we introduced a description-based framework for duplicate detection in certain databases that we then used as starting point for developing approaches for duplicate detection in probabilistic databases. A characteristic feature of this framework is that we extract an entity description per database entity and then match two entities by matching their descriptions. In the context of this framework we proposed the concept of impact values and presented an approach for describing and matching database entities in multi-table databases.

Uncertain Value Theory

In this thesis, we addressed uncertainty at different levels and within different contexts. For illustration, we consider data uncertainty in form of attribute-level uncertainty or tuple-level uncertainty, but also consider uncertainty on process configurations. To create a formal baseline for all these considerations, we introduced a formalism that we called *Uncertain Value Theory*.

Generic approaches for deterministic duplicate detection in probabilistic databases

We identified two intuitive solutions for conceptualizing a generic approach that accomplishes deterministic duplicate detection results.

The first approach is inspired by the idea of the possible worlds semantics and separately performs duplicate detection on a set of sample worlds until a particular moment, then resolves uncertainty by aggregating the results of the individual worlds, and finally performs the remaining duplicate detection phases on the aggregated result. Since each of the processed worlds corresponds to a certain database instance, the individual detection phases can be performed by using conventional duplicate detection methods.

The second approach does not consider uncertainty on database level, but considers uncertainty on the level of entity descriptions. For this purpose, it extracts a probabilistic descriptions per database entity from the input database and then matches two database entities by matching their probabilistic descriptions. Two probabilistic entity descriptions are in turn matched by pairwise matching their description alternatives that each corresponds to a certain entity description and hence can be matched by using conventional matching methods including similarity measures or decision models. From the results of matching the individual pairs of description alternatives a single matching output is derived (and hence resolves uncertainty) by using an aggregation method whereby aggregation can be performed at different moments of the matching process.

Since uncertainty can be resolved (or reduced) at different moments of the detection process by using one of several conceivable aggregation methods, both approaches enable the user to define a detection process which meets her individual needs.

Similarity measure for discrete probability distributions

A probabilistic entity description corresponds to a discrete probability distributions. As a conse-

quence, it seems obvious to reuse existing similarity measures for probability distributions in order to match probabilistic entity descriptions. A well-known and frequently used similarity measure for probability distributions is the Earth Mover's Distance. Nevertheless, although a variety of efficient algorithms for this distance measure is available it remains a complex optimization problem and hence is often too expensive for our purpose. For that reason, we developed a new similarity measure which is conceptually based on the Monge-Elkan Similarity and that we therefore coined the *Probabilistic Monge-Elkan Similarity*. We demonstrated with several experimental evaluations that the Probabilistic Monge-Elkan Similarity has a high semantical correlation to the Earth Mover's Distance, but is much faster to compute.

Probabilistic data preparation

To deal with incorrect entity correlations, we proposed a preparation technique that extracts and removes all entity correlations from the probabilistic input database, derives a set of matching activities from the extracted correlations, and finally provides these activities to the detection process. Furthermore, we developed a preparation technique that removes incorrect dependencies from attributes to table memberships by transforming the database schema and by exporting the instance data from the original schema to the transformed schema.

Incorporation of process uncertainty

The effectiveness of a duplicate detection process depends on several factors such as the considered application domain or the quality of the input data. For that reason, finding an appropriate process configuration can become difficult and used configurations are often tainted with uncertainty. We discussed several sources of uncertainty that can emerge within a duplicate detection process and propose the concept of matching cubes in order to incorporate process uncertainty into the description-based detection approach.

Indeterministic duplicate detection

For incorporating decision uncertainty into the detection result we introduced the concept of indeterministic duplicate detection. The result of an indeterministic duplicate detection process is a probabilistic clustering. However, a naive representation of such a probabilistic clustering is not very efficient and becomes already impractical if only a few number of pairwise duplicate decisions are uncertain. For that reason, we studied the aspect of factorizing a probabilistic clustering into its independent components. Then, we presented approaches for modeling indeterministic deduplication results within probabilistic conditional tables and BID-tables that are both well studied probabilistic representation systems. Furthermore, we discussed different ways for processing indeterministic deduplication results in general and discussed the challenge of evaluating aggregate queries on indeterministically deduplicated databases in particular. Moreover, we proposed a clustering approach that directly computes a factorized representation of a probabilistic duplicate clustering based on a set of pairwise duplicate decisions. Finally, we studied the meaning of detection quality in the presence of uncertain duplicate decisions. For that purpose, we conceptualize four quality semantics, proposed several quality measures for each of these semantics, and presented methods to efficiently compute these quality measures based on the factorized representation of a probabilistic duplicate clustering.

Prototypical implementation & experimental evaluation

We implemented a prototypical duplicate detection system and proved the concepts of our detection approaches by conducting a set of experiments.

The contributions that are provided by this thesis enable the user to define a duplicate detection process that meets her particular requirements and hence can be used to clean a single probabilistic database or to meaningful integrate several probabilistic databases in any application domain that is intended by the user.

11.2. Open Challenges & Future Research

Although we studied the different aspects of duplicate detection in probabilistic databases in this thesis extensively there remains a set of open challenges.

11.2.1. Compact Representation of Probabilistic Entity Descriptions

In this thesis, we consider a probabilistic entity description as a probability distribution over a set of description alternatives. Without reduction from each possible entity instance another description alternative can be extracted. Of course, if we consider database entities as single x-tuples, the number of description alternatives is restricted to the number of x-tuple alternatives which is definitely manageable in practice because otherwise the original x-tuple could not be stored within a BID-table. In more compact representation systems such as AOR?-databases or in multi-table databases where independence between different attribute values (or relationships) of the same entity is assumed, the number of possible instances of a single database entity can become extremely large. For illustration recall that the single A-tuple t_1 from the motivating example that we have presented in Section 1.2.1 had already 484 possible instances even its corresponding database schema only consists of eight attributes.

In such cases, modeling probabilistic entity descriptions within the possible worlds representation can be to inefficient and a more compact representation system need to be used instead. Obviously, we can adopt the representation system from the source database to the probabilistic descriptions because if some values or some tuples are independent in this database their information (attribute as well as relationship) will be independent in the probabilistic entity description as well. However, we also need to adapt the matching process to the compactly represented descriptions and such an adaptation is not straightforward if we strive for a matching process that is independent from the actually used representation system, i.e. processing compactly represented descriptions leads to the same detection result than processing their possible worlds representation. On the one hand, duplicate detection is only heuristic by nature and thus representation independence is not a mandatory requirement. On the other hand, such an independence is nice to have because its simplifies the usage of the detection approach. In this context it is important to note that whether or not a probabilistic description can be lossless represented within a particular system depends on the system of the input database. For instance, if the input database is a BID-table, but we compactly represent probabilistic entity descriptions by the usage of attribute-ORs all attribute value dependencies are lost by extracting the compactly represented descriptions. This leads to the circumstance that the duplicate detection approach either has to provide an armory of systems that can

be used to represent probabilistic entity descriptions or the user has to accept that useful information will be lost in some detection scenarios.

11.2.2. Querying Indeterministic Deduplication Results

As presented in Section 8.5.2, querying an indeterministic deduplication result can become critical if aggregate functions are used. Nevertheless, we also presented some first ideas for solving this problem in BID-databases by using the concept of query rewriting. A goal of future research is to study this challenge and its possible solutions in more detail (for BID-databases as well as pc-databases). Such a study is two-part. First, we need to identify which combinations of aggregate queries and aggregation semantics can be generally computed in an efficient way. Second, we need develop query mechanisms that realize such an efficient computation for the individual representation systems.

11.2.3. Decision Models for Probabilistic Entity Descriptions

In this thesis, we use conventional duplicate detection algorithms to match probabilistic entity descriptions. Nevertheless, instead of matching each pair of description alternatives separately, it can be useful to design matching algorithms that work on all these pairs collectively. For instance, it could be valuable to train a learning-based decision model that does not classify individual alternative description pairs, but directly classifies a complete probabilistic description pair.

11.2.4. Duplicate Merging in Uncertain Databases

In this thesis, we restrict our consideration to the detection of duplicates. Nevertheless, deduplicating a database also includes a merging of the detected duplicates. Existing researches on duplicate merging [BN08, BBB+05, MA06, ZH11, WBGM09] focus on certain databases. Moreover, they restrict to merging single tuples. Thus, to the best of our knowledge a merging of complex database entities, i.e. entities that are represented by tuples in several tables, or a merging of relationship information has not been considered so far, but are essential steps towards a deduplication in complex databases.

Merging approaches for certain databases focus on the resolution/repairing of attribute value conflicts and the resolution of uncertainties that are caused by null values. In contrast, in uncertain databases we also have to take attribute-level uncertainty and tuple-level uncertainty into account. Moreover, a merging approach has to deal with correlations between the different values and/or tuples that represent the same entity and has to deal with correlations between different entities. Thus, even for merging duplicate entities within a single pc-table, we can identify three main challenges:

- **Merging of Instance Data:** The first challenge of duplicate merging is to merge the sets of possible instances of the duplicate database entities. Recall that in the context of a single pc-table, each database entity corresponds to an x-tuple and its set of possible instances corresponds to its set of alternatives. The alternative sets of duplicate x-tuples can be merged in several ways. In the incomplete case where x-tuple alternatives are not ascribed with probabilities we can identify two extreme merging approaches. For the purpose of explanation, we distinguish between positive information (an ordinary tuple is an alternative of the considered x-tuple) and negative information (an ordinary tuple is not an alternative of the considered x-tuple). The first approach is to mistrust

the negative information that is provided by an individual x-tuple if it conflicts with some positive information that is provided by another x-tuple. As a consequence, we merge several sets of alternatives by building their set union. The second approach is to trust the negative information that is provided by the individual x-tuples regardless of the positive information that is provided by the other x-tuples. In this case, we merge the given sets of x-tuple alternative by computing their intersection. Of course, the second merging approach is only applicable if the intersection is not empty and hence if the duplicate x-tuples do not contradict each other, i.e. it exists at least one possible reality in which they have the same instance.

The result of the first approach is likely more correct than the result of the second because we do not make unverified assumptions and we take all plausible values into consideration. On the contrary, the result of the second approach is more certain than the result of the first because it contains less number of alternatives. Note that the latter approach exactly corresponds to the strategy of using the non-null value in the merge of certain data tuples [BN08].

The idea behind the first approach is that each x-tuple can contain positive information that was not available for creating the others. This approach corresponds to the Open World Assumption because we assume that an alternative can be missed in an x-tuple for no particular reason (maybe the creator of this x-tuples was not aware of the chance that this alternative is possible). In contrast, the idea behind the second approach is that each x-tuple can contain negative information that was not available for creating the others. This approach corresponds to the Closed World Assumption because we assume that each missing alternative is missing for a particular reason, i.e. the creator of the corresponding x-tuple was sure that this alternative is not possible.

Example 224 *For illustration, we consider two x-tuples that are defined on a schema with a single attribute. The first x-tuple has the three alternative values 'A', 'B', and 'C' in this attribute and the second x-tuple has the three alternative values 'A', 'B', and 'D' in this attribute. If we use the Open World Assumption or if we mistrust the information that is provided by the individual x-tuples, we conclude that the 'true' value of this attribute is likely 'A', 'B', 'C', or 'D'. In contrast, if we use the Closed World Assumption and if we trust the individual x-tuples, we can conclude that 'C' and 'D' are certainly not the 'true' value because the first is excluded by the second x-tuple and the second is excluded by the first x-tuple. As a consequence, we can conclude that the 'true' value must be 'A' or 'B'.*

Obviously these merging approaches becomes more complicated if the alternative values are ascribed with probabilities. For instance the intersection of all duplicate x-tuples can be a single alternative that has a low probability in all these x-tuples and therefore is likely not the 'true' instance. For demonstration, does it make sense to take the alternative 'B' as merging output if the first x-tuple corresponds to the probability distribution $(\text{'A'}: 0.9, \text{'B'}: 0.1)$ and the second x-tuple corresponds to the probability distribution $(\text{'C'}: 0.8, \text{'B'}: 0.2)$? In general, a consideration of probabilities begs the question in which cases two probabilistic x-tuples agree in their information and in which cases they contradict each other.

Recall that we have illustrated a mix of both approaches in the motivating example in Section 1.3.

- **Merging of Membership Information:** A second challenge in merging duplicate entities in probabilistic databases is a merging of membership probabilities because these probabilities can be contrary and it is often not clear which merge represents reality best. For instance, one database entity belongs to the table *'Student'* with probability 0.8, but another entity of the same duplicate cluster belongs to this table only with probability 0.6.

  Moreover, the uncertainty on the table membership of an entity can be strongly correlated with the uncertainty on some attribute values of this entity. For example a membership to a database table *'Senior'* is strongly correlated to the value in the attribute *'age'* if only persons that are older than 50 years are considered to be seniors. In such cases, we have to co-ordinate the merging of attribute values and the merging membership information.

  The problem of merging membership information becomes even more complicated if we want to integrate data from different sources and the membership information of the source tuples can only be hardly compared because they are defined within different contexts. For example, let us assume that we integrate a database on horror movies and a database on science fiction movies in order to obtain a database on movies. In this case, we need to derive the probability that an entity is a movie from the probability that it is a horror movie and the probability that it is a science fiction movie. Of course, if we trust both sources we can say that this probability must be at least as great than the maximal probability of both corresponding source entities because each horror movie and each science fiction movie is a movie. In addition, if we assume that the considered entity cannot be a movie from any other genre we can compute an upper bound by summing up the probabilities from the two source entities. Computing a point probability, however, is not simply made. Moreover, this example illustrates another problem that is generally present in integrating probabilistic data. Data integration is usually performed under the Open World Assumption. In such cases, however, we can only compute a lower bound on membership probabilities, but can never compute an upper bound because it could be the case that the considered entity is missing in some of the given sources.

- **Merging of Entity Correlations:** A final challenge is the merging of entity correlations because it also affects entities outside the considered duplicate cluster. For instance, let us consider the three database entities e_1, e_2, and e_3 where the first two entities belong to the same duplicate cluster and the last entity belongs to another duplicate cluster. Moreover, let us assume that in the input database the existence of e_1 includes the existence of e_3, but the existence of e_2 excludes the existence of e_3. In this case it is not clear in which way the existence of the merged entity $f_\mu(e_1, e_2)$ should be correlated to the existence of e_3 because the existence of the first can either include the existence of the second or can exclude the existence of the second but cannot both. Obviously, such a situation becomes more complicated if more than three entities are involved and if the original correlations between these entities form a complex pattern.

Of course, a solution to these problems becomes even more challenging if we extend our consideration from single tables to multi-table databases. For instance the merged membership to one table must be consistent to the merged membership to another table if the membership to both tables are correlated, e.g.

a person cannot be a master student with probability 0.8 if she is a student with probability 0.5 because a membership to the table *'Master Student'* includes a membership to the table *'Student'*.

We present some first reflections about challenges and solutions on merging duplicates in probabilistic databases in [PR10c, PR11]. Moreover, we consider a merging of duplicate tuples in fuzzy databases in [PR10b].

11.2.5. Quality of Probabilistic Databases

Duplicate elimination is a process that does not only increase the consistency of a database, but can also increase the intensional completeness of a database and its correctness. Moreover, data integration can increase the extensional completeness of a database. The increase in intensional completeness and the increase in accuracy are caused by the circumstance that each of the detected duplicates can contribute some information that is not contributed by the others (completeness), i.e. one duplicate entity has a value in an attribute that is filled by NULL for the other entities, and that combining the information provided by the individual duplicates increases the chance of detecting the correct value for each of the considered attributes (correctness).

Of course, depending on the used merge function the same holds for probabilistic databases. Nevertheless, to quantify such increases in quality we need to adapt measures for completeness and measures for correctness from certain databases to probabilistic databases. For that purpose, we principally can adopt the same four quality semantics that we have proposed in Section 8.8.1. We make some first steps towards adapting completeness measures from certain databases to probabilistic databases in [Pan09a, PR09, PR10a].

11.2.6. Collective Duplicate Detection in Probabilistic Databases

In the detection approaches that we have presented in this thesis, we match each two database entities independently. However, in existing research on duplicate detection in certain databases several approaches have been proposed that use the matching result of one entity pair to affect the matching results of some related entity pairs. By doing so the effectiveness of the detection result can often be considerably increased. If matching uncertainty is resolved before affections are computed, existing approaches for collective duplicate detection can be adopted to probabilistic source data with only less additional effort. The main challenge is to adapt such collective approaches from deterministic duplicate detection to indeterministic duplicate detection because in the case of the latter a definition of meaningful affections is not trivial.

11.2.7. Parallelization

Duplicate detection in general and duplicate detection in probabilistic databases in particular have a high potential for parallelization. First, in iterative detection approaches the pairwise duplicate decisions are made independently and therefore can be simply allocated to different processors. The same holds for matching the alternatives of two probabilistic entity descriptions. In the Probabilistic Monge-Elkan Similarity, we separately compare each alternative of the first value with all alternatives of the second value and therefore can compute them in parallel. In the world-based detection approach, all possible

worlds are processed separately and thus can be distributed to different machines. Finally, the factors of a probabilistic clustering and thus their corresponding *matching-graphs* are mutual independent. As a consequence, the efficiency of a factor-based quality computation and the efficiency of the clustering approach that we have presented in Section 8.7 can be massively improved by the use of parallel computing.

11.2.8. Duplicate Detection in Distributed Databases

In this thesis, we restrict our consideration to databases that are stored in single nodes. However, in many application areas the amount of collected data becomes larger and larger and the relevance of *big data* is growing fast in today's information systems. For these size reasons as well as for reasons of availability, databases are often distributed over a large number of nodes and sending all the distributed data to a single node for the purpose of detecting duplicates does not scale well or is even impractical. As a consequence, we require for detection approaches that can be efficiently performed on distributed databases.

11.2.9. Probabilistic Monge-Elkan Similarity

As we have experimental validated in Section 9.4, the Probabilistic Monge-Elkan Similarity can be considered as an efficient alternative to the Earth Mover's Distance. Today, the Earth Mover's Distance is used in several application domain such as multimedia information retrieval and pattern recognition in general and content-based image retrieval [PWR89, RTG00], video event recognition [XC08], music recommendation [Log04, PFW05], face swapping [BKD+08], image matching [GD04, GD05], or speaker clustering [SF06] in particular. Moreover, it has been used in mining probabilistic databases [XZTY12]. It stands to reason that the Probabilistic Monge-Elkan Similarity could be useful in some of these domains as well. For that reason, we strive to evaluate the usability (in terms of effectiveness and efficiency) of the Probabilistic Monge-Elkan Similarity within in some of these application domains.

11.2.10. Further Development of HaDES

Until now, we did not implement all of the techniques that we have proposed in this thesis. For that reason, we aim to extend the current version of HaDES by the following components:

- **Database Extractors** that are designed for complex databases and therefore extract relationship descriptions and take inheritance hierarchies in the extraction process into account.
- An **ARM-Dependency Resolver** that gets an unclean probabilistic database as input and produces a clean pc-database as output by resolving some of the given ARM-dependencies as described in Section 7.7.2.
- A **Dependency Cleaner/Extractor** that extracts entity dependencies from the input database and uses context information to derive matching activities from these dependencies.
- **Probabilistic Database Importers** that can be used to import the original database entities for merging purposes. Of course, we require a particular importer per representation system.

- **Probabilistic Database Exporters** that can be used to store a deterministic deduplication result or to store an indeterministic deduplication result within a probabilistic database. Obviously, we require a particular exporter per representation system.
- A **Duplicate Merger** that merges duplicate database entities. Since duplicate merging in probabilistic databases has only been marginally studied so far (see future goal above), we aim to start with a basic implementation that simply merges a set of independent duplicate x-tuples by computing the set union of their sets of alternatives. This implementation should then be extended to more sophisticated merge functions step by step.

11.2.11. Experimental Evaluations

The experiments that we have presented in this thesis are restricted to single BID-tables. Several of the challenges that we have discussed in this thesis and therefore some of the proposed techniques are only valuable if the deduplicated database contains entity correlations and/or has a complex schema with relationship information and inheritance hierarchies. One goal of future research is therefore to expand the experimental evaluations from single BID-tables to more complex databases. Of course, this in turn requires a generation of more complex test databases.

Moreover, some of the experiments that we presented in this thesis were conducted on synthetic databases that are generated by ascribing probabilities to x-tuple alternatives randomly. Many of the proposed resolution methods, however, are based on the assumption that these probabilities provide meaningful information and hence assume that they can draw useful inferences from the alternatives' probabilities about their correctness. Indeed in reality probabilities typically follow specific patterns, e.g. a bell-shaped curve that represents the uncertain output of a measurement. In this case, the individual resolution methods can conclude more information from these probabilities and it can be expected that the different use cases for which the individual resolution methods are valuable become more clear. For that reason, we strive for experiments that are conducted by using labeled test databases that are not synthetic but real or strive at least for experiments that are conducted by using synthetic test databases whose probability distributions follow more realistic patterns.

Appendix A

Proofs

A.1. Proofs: Clustering Background

In this section, we prove the theorems of Section 2.2.

Proof of Theorem 1

Theorem 1 *The clustering $\mathcal{C}_{ij} = \mathcal{C}_i \boxtimes \mathcal{C}_j$ is a cluster-disjoint clustering, iff $\mathcal{C}_i$ and $\mathcal{C}_j$ are cluster-disjoint clusterings and either their ranges are disjoint or the elements of the ranges' overlap only form clusters that belongs to $\mathcal{C}_i$ and $\mathcal{C}_j$, i.e. $\forall C_1 \in \mathcal{C}_i : \forall C_2 \in \mathcal{C}_j : C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$.*

Theorem 1 can be proved as follows:

Proof 13 *($\Rightarrow$)*

- *Case 1: Assumption: $\mathcal{C}_{ij}$, $\mathcal{C}_i$ and $\mathcal{C}_j$ are cluster-disjoint, but $\exists C_1 \in \mathcal{C}_i, \exists C_2 \in \mathcal{C}_j : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.*
  Due to $\mathcal{C}_{ij} = \mathcal{C}_i \cup \mathcal{C}_j \Rightarrow \exists C_1, C_2 \in \mathcal{C}_{ij} : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
  $\Rightarrow \mathcal{C}_{ij}$ is not cluster-disjoint which is a contradiction to the initially assumption.
  $\Rightarrow$ if $\mathcal{C}_{ij}$ is cluster-disjoint, the overlap of the ranges of $\mathcal{C}_i$ and $\mathcal{C}_j$ only include clusters that belong to both.

- *Case 2: Assumption: $\mathcal{C}_{ij}$ is cluster-disjoint but $\mathcal{C}_i$ is not.*
  Due to $\mathcal{C}_i$ is not cluster-disjoint $\Rightarrow \exists C_1, C_2 \in \mathcal{C}_i : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
  Due to $\mathcal{C}_{ij} = \mathcal{C}_i \cup \mathcal{C}_j \Rightarrow \exists C_1, C_2 \in \mathcal{C}_{ij} : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
  $\Rightarrow \mathcal{C}_{ij}$ is not cluster-disjoint which is a contradiction to the initially assumption.
  $\Rightarrow$ if $\mathcal{C}_{ij}$ is cluster-disjoint, $\mathcal{C}_i$ is cluster-disjoint as well.

($\Leftarrow$)

- *Case 1: Assumption: $\mathcal{C}_i$ and $\mathcal{C}_j$ are cluster-disjoint and $rng(\mathcal{C}_i) \otimes rng(\mathcal{C}_j)$, but $\mathcal{C}_{ij}$ is not cluster-disjoint.*
  Due to $\mathcal{C}_{ij}$ is not cluster-disjoint $\Rightarrow \exists C_1, C_2 \in \mathcal{C}_{ij} : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.

Due to $\mathcal{C}_{ij} = \mathcal{C}_i \cup \mathcal{C}_j \Rightarrow \exists C_1, C_2 \in \mathcal{C}_i \cup \mathcal{C}_j : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
$rng(\mathcal{C}_i) \circledast rng(\mathcal{C}_j) \Rightarrow \exists C_1, C_2 \in \mathcal{C}_i : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$ or $\exists C_1, C_2 \in \mathcal{C}_j : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
$\Rightarrow$ either $\mathcal{C}_i$ or $\mathcal{C}_j$ is not cluster-disjoint (or both) which is a contradiction to our initially assumption.
$\Rightarrow$ if $\mathcal{C}_i$ and $\mathcal{C}_j$ are cluster-disjoint clusterings and their ranges do not overlap, $\mathcal{C}_{ij}$ is cluster-disjoint as well.

- *Case 2: Assumption: $\mathcal{C}_i$ and $\mathcal{C}_j$ are cluster-disjoint and $\forall C_1 \in \mathcal{C}_i, \forall C_2 \in \mathcal{C}_j : C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$, but $\mathcal{C}_{ij}$ is not cluster-disjoint.*
  Due to $\mathcal{C}_{ij}$ is not cluster-disjoint $\Rightarrow \exists C_1, C_2 \in \mathcal{C}_{ij} : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
  Due to $\mathcal{C}_{ij} = \mathcal{C}_i \cup \mathcal{C}_j \Rightarrow \exists C_1, C_2 \in \mathcal{C}_i \cup \mathcal{C}_j : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
  Since all clusters that contain elements of $rng(\mathcal{C}_i) \cap rng(\mathcal{C}_j)$ belong to both clusterings $\Rightarrow$ $\exists C_1, C_2 \in \mathcal{C}_i : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$ or $\exists C_1, C_2 \in \mathcal{C}_j : C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
  $\Rightarrow$ either $\mathcal{C}_i$ or $\mathcal{C}_j$ is not cluster-disjoint (or both) which is a contradiction to our initially assumption.
  $\Rightarrow$ if $\mathcal{C}_i$ and $\mathcal{C}_j$ are cluster-disjoint clusterings and the overlap of their ranges only include clusters that belong to both, $\mathcal{C}_{ij}$ is cluster-disjoint as well.

Proof of Theorem 2

Theorem 2 *Let $\mathcal{C}$ be a clustering and $S_i, S_j \subseteq rng(\mathcal{C})$ be two disjoint subsets of the clustering's range. If there is no cluster $C \in \mathcal{C}$ its elements belong to S_i and S_j, i.e. $\forall C \in \mathcal{C} : C \circledast S_i \vee C \circledast S_j$, it holds: $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) = \pi_{S_i \cup S_j}(\mathcal{C})$.*

Theorem 2 can be proved as follows:

Proof 14 *Assumption: S_i and S_j are disjoint and $\forall C \in \mathcal{C} : C \circledast S_i \vee C \circledast S_j$, but $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) \neq \pi_{S_i \cup S_j}(\mathcal{C})$.*
Due to $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) \neq \pi_{S_i \cup S_j}(\mathcal{C})$
$\Rightarrow$ either $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C})$ has a cluster that does not belong to $\pi_{S_i \cup S_j}(\mathcal{C})$ or $\pi_{S_i \cup S_j}(\mathcal{C})$ has a cluster that does not belong to $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C})$.

- *Case 1: Assumption: $\exists C \in \pi_{S_i \cup S_j}(\mathcal{C}) : C \notin \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C})$.*
  Due to $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) = \pi_{S_i}(\mathcal{C}) \cup \pi_{S_j}(\mathcal{C})$
  $\Rightarrow \exists C \in \pi_{S_i \cup S_j}(\mathcal{C}) : C \notin \pi_{S_i}(\mathcal{C}) \wedge C \notin \pi_{S_j}(\mathcal{C})$.
  $\Rightarrow C \nsubseteq S_i \wedge C \nsubseteq S_j$.
  Due to $C \subseteq S_i \cup S_j \Rightarrow C \circledcirc S_i \wedge C \circledcirc S_j$.
  $\Rightarrow S_i$ and S_j overlap which is a contradiction to the initial assumption that S_i and S_j are disjoint.
  $\Rightarrow$ there is no cluster in $\pi_{S_i \cup S_j}(\mathcal{C})$ that is not in $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C})$.

- *Case 2: Assumption: $\exists C \in \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) : C \notin \pi_{S_i \cup S_j}(\mathcal{C})$.*
  Due to $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) = \pi_{S_i}(\mathcal{C}) \cup \pi_{S_j}(\mathcal{C})$
  $\Rightarrow \forall C \in \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) : C \in \pi_{S_i}(\mathcal{C}) \vee C \in \pi_{S_j}(\mathcal{C})$.

$\Rightarrow \forall C \in \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) \colon C \in \{S_i \cap C' \mid C' \in \mathcal{C}\} \vee C \in \{S_j \cap C' \mid C' \in \mathcal{C}\}$.
$\Rightarrow \forall C \in \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) \colon C \in \{S_i \cap C' \cup S_j \cap C' \mid C' \in \mathcal{C}\}$.
$\Rightarrow \forall C \in \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) \colon C \in \{(S_i \cup S_j) \cap C' \mid C' \in \mathcal{C}\}$.
$\Rightarrow \forall C \in \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) \colon C \in \pi_{S_i \cup S_j}(\mathcal{C})$ *which is a contradiction to the assumption that there is a cluster in the integration result that does not belong to the projection result.*
$\Rightarrow$ *there is no cluster in* $\pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C})$ *that is not in* $\pi_{S_i \cup S_j}(\mathcal{C})$.

$\Rightarrow \pi_{S_i}(\mathcal{C}) \boxtimes \pi_{S_j}(\mathcal{C}) = \pi_{S_i \cup S_j}(\mathcal{C})$ *which is a contradiction to the initial assumption that both clusterings are inequal.*

Proof of Theorem 3

Theorem 3 *Let* $\mathcal{C}$ *be a clustering and let* $S = \{S_1, \ldots, S_k\}$ *be a partition of* $rng(\mathcal{C})$. *If there is no cluster* $C \in \mathcal{C}$ *its element belongs to different partition classes, i.e.* $\forall C \in \mathcal{C} \colon \forall S_i, S_j \in S \colon C \otimes S_i \vee C \otimes S_j$, *it holds:* $\bigboxtimes_{S_i \in S}\big(\pi_{S_i}(\mathcal{C})\big) = \mathcal{C}$.

Theorem 3 can be proved as follows:

Proof 15 *The proof of Theorem 3 follows directly from Theorem 2, because each partition of* $rng(\mathcal{C})$ *can be build by binary dividing the given parition classes (starting from* $rng(\mathcal{C})$ *itself) further on.*

We prove it by the use of mathematical induction:

Basis:

Assumption: The theorem is true for $k = 2$, *i.e.* $S = \{S_1, S_2\}$.
Proof: Following Theorem 2 $\Rightarrow \pi_{S_1}(\mathcal{C}) \boxtimes \pi_{S_2}(\mathcal{C}) = \pi_{S_1 \cup S_2}(\mathcal{C}) = \pi_S(\mathcal{C}) = \mathcal{C}$.

Now, we have to show that the theorem is true for $S = \{S_1, \ldots, S_k, S_{k+1}\}$ *provided that it is true for* $S = \{S_1, \ldots, S_k\}$.

Inductive Step:

Assumption: The theorem is true for $S = \{S_1, \ldots, S_{k+1}\}$.
Proof:

$$
\begin{aligned}
& \textstyle\bigboxtimes_{i=1}^{k+1} \pi_{S_i}(\mathcal{C}) & & \\
\textit{(following Definition 11)} \quad & = & & \big(\textstyle\bigboxtimes_{i=1}^{k} \pi_{S_i}(\mathcal{C})\big) \boxtimes \pi_{S_{k+1}}(\mathcal{C}) \\
\textit{(following the assumption that the theorem holds for } k\textit{)} \quad & = & & \pi_{S - \{S_{k+1}\}}(\mathcal{C}) \boxtimes \pi_{S_{k+1}}(\mathcal{C}) \\
\textit{(following Theorem 2)} \quad & = & & \pi_{S - \{S_{k+1}\} \cup S_{k+1}}(\mathcal{C}) \\
& = & & \pi_S(\mathcal{C}) \\
& = & & \mathcal{C}
\end{aligned}
$$

A.2. Proofs: Properties of Uncertain Clusterings

In this section, we prove the theorems of Section 8.2.

Proof of Theorem 17

Theorem 17 *An incomplete clustering Γ_i is dominated by an incomplete clustering Γ_j, iff each possible clustering $\mathcal{C}_i \in \Gamma_i$ is dominated by a possible clustering $\mathcal{C}_j \in \Gamma_j$ and iff each possible clustering $\mathcal{C}_j \in \Gamma_j$ dominates a possible clustering $\mathcal{C}_i \in \Gamma_i$.*

Theorem 17 can be proved as follows:

Proof 16 *($\Rightarrow$)*
Here we can distinct between two cases:

- ***Case 1:*** *Assumption:* $\Gamma_i \prec \Gamma_j$, *but* $\exists \mathcal{C}_i \in \Gamma_i \colon \forall \mathcal{C}_j \in \Gamma_j \colon \neg(\mathcal{C}_i \prec \mathcal{C}_j)$.
  Due to $\Gamma_i \prec \Gamma_j$
  $\Rightarrow \Gamma_i = \pi_{rng(\Gamma_i)}(\Gamma_j)$.
  $\Rightarrow \Gamma_i = \{\pi_{rng(\Gamma_i)}(\mathcal{C}_j) \mid \mathcal{C}_j \in \Gamma_j\}$.
  $\Rightarrow \forall \mathcal{C}_i \in \Gamma_i \colon \exists \mathcal{C}_j \in \Gamma_j \colon \mathcal{C}_i = \pi_{rng(\Gamma_i)}(\mathcal{C}_j)$.
  $\Rightarrow \forall \mathcal{C}_i \in \Gamma_i \colon \exists \mathcal{C}_j \in \Gamma_j \colon \mathcal{C}_i \prec \mathcal{C}_j$ *which is a contradiction to the initial assumption. Thus, if an incomplete clustering Γ_i is dominated by an incomplete clustering Γ_j it follows that every alternative of Γ_i is dominated by an alternative of Γ_j.*

- ***Case 2:*** *Assumption:* $\Gamma_i \prec \Gamma_j$, *but* $\exists \mathcal{C}_j \in \Gamma_j \colon \forall \mathcal{C}_i \in \Gamma_i \colon \neg(\mathcal{C}_i \prec \mathcal{C}_j)$.
  Due to $\Gamma_i \prec \Gamma_j$
  $\Rightarrow \pi_{rng(\Gamma_i)}(\Gamma_j) = \Gamma_i$.
  $\Rightarrow \{\pi_{rng(\Gamma_i)}(\mathcal{C}_j) \mid \mathcal{C}_j \in \Gamma_j\} = \Gamma_i$.
  $\Rightarrow \forall \mathcal{C}_j \in \Gamma_j \colon \exists \mathcal{C}_i \in \Gamma_i \colon \pi_{rng(\Gamma_i)}(\mathcal{C}_j) = \mathcal{C}_i$.
  $\Rightarrow \forall \mathcal{C}_j \in \Gamma_j \colon \exists \mathcal{C}_i \in \Gamma_i \colon \mathcal{C}_i \prec \mathcal{C}_j$ *which is a contradiction to the initial assumption.*

Thus, if an incomplete clustering Γ_i is dominated by an incomplete clustering Γ_j it follows that every alternative of Γ_j dominates an alternative of Γ_i.

($\Leftarrow$)
Assumption: $\forall \mathcal{C}_i \in \Gamma_i \colon \exists \mathcal{C}_j \in \Gamma_j \colon \mathcal{C}_i \prec \mathcal{C}_j$ *and* $\forall \mathcal{C}_j \in \Gamma_j \colon \exists \mathcal{C}_i \in \Gamma_i \colon \mathcal{C}_i \prec \mathcal{C}_j$, *but* $\neg(\Gamma_i \prec \Gamma_j)$.
Due to $\neg(\Gamma_i \prec \Gamma_j) \Rightarrow \Gamma_i \neq \pi_{rng(\Gamma_i)}(\Gamma_j)$.
Here we can distinct between two cases:

- ***Case 1:*** Γ_i *contains an alternative that is not contained in* $\pi_{rng(\Gamma_i)}(\Gamma_j)$.
  $\Rightarrow \exists \mathcal{C}_i \in \Gamma_i \colon \nexists \mathcal{C}_j \in \Gamma_j \colon \mathcal{C}_i = \pi_{rng(\Gamma_i)}(\mathcal{C}_j)$.
  $\Rightarrow \exists \mathcal{C}_i \in \Gamma_i \colon \nexists \mathcal{C}_j \in \Gamma_j \colon \mathcal{C}_i \prec \mathcal{C}_j$ *which is a contradiction to the initial assumption.*

- ***Case 2:*** *$\pi_{rng(\Gamma_i)}(\Gamma_j)$ contains an alternative that is not contained in Γ_i.*
  $\Rightarrow \exists \mathcal{C}_j \in \Gamma_j \colon \nexists \mathcal{C}_i \in \Gamma_i \colon \mathcal{C}_i = \pi_{rng(\Gamma_i)}(\mathcal{C}_j)$.
  $\Rightarrow \exists \mathcal{C}_j \in \Gamma_j \colon \nexists \mathcal{C}_i \in \Gamma_i \colon \mathcal{C}_i \prec \mathcal{C}_j$ which is a contradiction to the initial assumption.

Thus, if every alternative of an incomplete clustering Γ_i is dominated by an alternative of an incomplete clustering Γ_j and every alternative of Γ_j dominates an alternative of Γ_i it follows that $\Gamma_i \prec \Gamma_j$.

Proof of Theorem 18

Theorem 18 *A probabilistic clustering $\mathfrak{C}_i = (\Gamma_i, Pr_i)$ is dominated by a probabilistic clustering $\mathfrak{C}_j = (\Gamma_j, Pr_j)$, iff Γ_i is dominated by Γ_j and it holds that $\forall \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$.*

Theorem 18 can be proved as follows:

Proof 17 *($\Rightarrow$)*
It is trivial that $\mathfrak{C}_i$ can only be dominated by $\mathfrak{C}_j$ if Γ_i is dominated by Γ_j. For that reason, we only need to prove that $\mathfrak{C}_i$ can only be dominated by $\mathfrak{C}_j$ if $\forall \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$.

Assumption: $\mathfrak{C}_i \prec \mathfrak{C}_j$, but $\exists \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) \neq \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$.
Due to $\mathfrak{C}_i \prec \mathfrak{C}_j$, it holds that $\mathfrak{C}_i = \pi_{rng(\mathfrak{C}_i)}(\mathfrak{C}_j)$.
$\Rightarrow \forall \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} = \pi_{rng(\mathfrak{C}_i)}(\mathcal{C}')} Pr_j(\mathcal{C}')$.
Due to $\mathcal{C} = \pi_{rng(\mathfrak{C}_i)}(\mathcal{C}') \Rightarrow \mathcal{C} \prec \mathcal{C}'$.
$\Rightarrow \forall \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$ which is a contradiction to the initial assumption.
Thus, if $\mathfrak{C}_i$ is dominated by $\mathfrak{C}_j$ it follows that $\forall \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$.

($\Leftarrow$)
Assumption: $\Gamma_i \prec \Gamma_j$ and $\forall \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$, but $\neg(\mathfrak{C}_i \prec \mathfrak{C}_j)$.
Due to $\neg(\mathfrak{C}_i \prec \mathfrak{C}_j)$
$\Rightarrow \mathfrak{C}_i \neq \pi_{rng(\mathfrak{C}_i)}(\mathfrak{C}_j)$.
Due to $\Gamma_i \prec \Gamma_j$ and therefore $\Gamma_i = \pi_{rng(\Gamma_i)}(\Gamma_j)$
$\Rightarrow \exists \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) \neq \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} = \pi_{rng(\mathfrak{C}_i)}(\mathcal{C}')} Pr_j(\mathcal{C}')$ which is a contradiction to the initial assumption.
Thus, if Γ_i is dominated by Γ_j and if $\forall \mathcal{C} \in \Gamma_i \colon Pr_i(\mathcal{C}) = \sum_{\mathcal{C}' \in \Gamma_j, \mathcal{C} \prec \mathcal{C}'} Pr_j(\mathcal{C}')$ it follows that $\mathfrak{C}_i$ is dominated by $\mathfrak{C}_j$.

Proof of Theorem 19

Theorem 19 *The incomplete clustering $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$ is a cluster-disjoint clustering, iff Γ_i and Γ_j are cluster-disjoint clusterings and either their ranges are disjoint or the elements of the ranges' overlap only form clusters that belongs to all possible clusterings of Γ_i and belongs to all possible clusterings of Γ_j, i.e. $\forall \mathcal{C}_i \in \Gamma_i \colon \forall \mathcal{C}_j \in \Gamma_j \colon \forall C_1 \in \mathcal{C}_i \colon \forall C_2 \in \mathcal{C}_j \colon C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$.*

Theorem 19 can be proved as follows:

Proof 18 $(\Rightarrow)$
It is trivial that Γ_{ij} *can only be cluster-disjoint if* Γ_i *and* Γ_j *are cluster-disjoint. For that reason, we only need to prove that* Γ_{ij} *can only be cluster-disjoint if* $\forall \mathcal{C}_i \in \Gamma_i \colon \forall \mathcal{C}_j \in \Gamma_j \colon \forall C_1 \in \mathcal{C}_i \colon \forall C_2 \in \mathcal{C}_j \colon C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$.

Assumption: $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$ *is cluster-disjoint, but* $\exists \mathcal{C}_i \in \Gamma_i \colon \exists \mathcal{C}_j \in \Gamma_j \colon \exists C_1 \in \mathcal{C}_i \colon \exists C_2 \in \mathcal{C}_j \colon C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
Due to $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$
$\Rightarrow \exists \mathcal{C}_{ij} \in \Gamma_{ij} \colon \mathcal{C}_{ij} = \mathcal{C}_i \boxtimes \mathcal{C}_j$.
$\Rightarrow \exists \mathcal{C}_{ij} \in \Gamma_{ij} \colon C_1, C_2 \in \mathcal{C}_{ij}$.
$\Rightarrow \exists \mathcal{C}_{ij} \in \Gamma_{ij} \colon \mathcal{C}_{ij}$ *is not cluster-disjoint because* C_1 *and* C_2 *overlap.*
$\Rightarrow$ Γ_{ij} *is not cluster-disjoint which is a contradiction to the initial assumption. Thus, if* $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$ *is cluster-disjoint it follows that* Γ_i *and* Γ_j *are cluster-disjoint and* $\forall \mathcal{C}_i \in \Gamma_i \colon \forall \mathcal{C}_j \in \Gamma_j \colon \forall C_1 \in \mathcal{C}_i \colon \forall C_2 \in \mathcal{C}_j \colon C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$.

$(\Leftarrow)$
Assumption: Γ_i *and* Γ_j *are cluster-disjoint and* $\forall \mathcal{C}_i \in \Gamma_i \colon \forall \mathcal{C}_j \in \Gamma_j \colon \forall C_1 \in \mathcal{C}_i \colon \forall C_2 \in \mathcal{C}_j \colon C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$, *but* $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$ *is not cluster-disjoint.*
Due to Γ_{ij} *is not cluster-disjoint*
$\Rightarrow \exists \mathcal{C}_{ij} \in \Gamma_{ij} \colon \mathcal{C}_{ij}$ *is not cluster-disjoint.*
$\Rightarrow \exists \mathcal{C}_{ij} \in \Gamma_{ij} \colon \exists C_1, C_2 \in \mathcal{C}_{ij} \colon C_1 \circledcirc C_2 \wedge C_1 \neq C_2$.
Due to Γ_i *and* Γ_j *are cluster-disjoint,* C_1 *and* C_2 *must originate from different input clusterings.*
$\Rightarrow \exists \mathcal{C}_i \in \Gamma_i \colon C_1 \in \mathcal{C}_i \wedge \exists \mathcal{C}_j \in \Gamma_j \colon C_2 \in \mathcal{C}_j$.
$\Rightarrow \exists \mathcal{C}_i \in \Gamma_i \colon \exists \mathcal{C}_j \in \Gamma_j \colon \exists C \in \mathcal{C}_i \colon \exists C' \in \mathcal{C}_j \colon C \circledcirc C' \wedge C \neq C'$ *which is a contradiction to the initial assumption.*

Thus, if Γ_i *and* Γ_j *are cluster-disjoint and* $\forall \mathcal{C}_i \in \Gamma_i \colon \forall \mathcal{C}_j \in \Gamma_j \colon \forall C_1 \in \mathcal{C}_i \colon \forall C_2 \in \mathcal{C}_j \colon C_1 \circledcirc C_2 \Rightarrow C_1 = C_2$ *it follows that* $\Gamma_{ij} = \Gamma_i \boxtimes \Gamma_j$ *is cluster-disjoint.*

Proof of Theorem 20

Theorem 20 *Let* Γ *be an incomplete clustering and let* $S_i, S_j \subseteq rng(\Gamma)$ *be two disjoint subsets of the clustering's range. It holds that* $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) = \pi_{S_i \cup S_j}(\Gamma)$ *iff the two clusterings* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are mutually independent with respect to* Γ, *i.e. iff* $\forall \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \forall \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \exists \mathcal{C} \in \Gamma \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ *and* $\forall \mathcal{C} \in \Gamma \colon \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$.

Theorem 20 can be proved as follows:

Proof 19 *(*$\Rightarrow$*)*
Assumption: $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) = \pi_{S_i \cup S_j}(\Gamma)$, *but* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are not mutually independent with respect to* Γ.
Here we can distinct between two cases:

- ***Case 1:*** *Assumption:* $\exists \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \nexists \mathcal{C} \in \Gamma \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$.
  $\Rightarrow \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \nexists \mathcal{C} \in \pi_{S_i \cup S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$.
  $\Rightarrow \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \notin \pi_{S_i \cup S_j}(\Gamma)$.
  $\Rightarrow \pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) \neq \pi_{S_i \cup S_j}(\Gamma)$ *which is a contradiction to the initial assumption.*

- ***Case 2:*** *Assumption:* $\exists \mathcal{C} \in \Gamma \colon \nexists \mathcal{C}_i \in \pi_{S_i}(\Gamma), \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$.
  $\Rightarrow \exists \mathcal{C}' \in \pi_{S_i \cup S_j}(\Gamma) \colon \nexists \mathcal{C}_i \in \pi_{S_i}(\Gamma), \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}'$.
  $\Rightarrow \exists \mathcal{C}' \in \pi_{S_i \cup S_j}(\Gamma) \colon \mathcal{C}' \notin \pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma)$.
  $\Rightarrow \pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) \neq \pi_{S_i \cup S_j}(\Gamma)$ *which is a contradiction to the initial assumption.*

Thus, if $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) = \pi_{S_i \cup S_j}(\Gamma)$ *it follows that* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are mutually independent with respect to* Γ.

(⇐)
Assumption: $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) \neq \pi_{S_i \cup S_j}(\Gamma)$*, but* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are mutually independent with respect to* Γ.
Since $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) \neq \pi_{S_i \cup S_j}(\Gamma)$*, we can distinct between two cases:*

- ***Case 1:*** *Assumption:* $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma)$ *has an alternative that does not belong to* $\pi_{S_i \cup S_j}(\Gamma)$.
  $\Rightarrow \exists \mathcal{C} \in \pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) \colon \mathcal{C} \notin \pi_{S_i \cup S_j}(\Gamma)$.
  $\Rightarrow \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \notin \pi_{S_i \cup S_j}(\Gamma)$.
  $\Rightarrow \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \nexists \mathcal{C} \in \pi_{S_i \cup S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$.
  $\Rightarrow \exists \mathcal{C}_i \in \pi_{S_i}(\Gamma) \colon \exists \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \nexists \mathcal{C} \in \Gamma \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ *which is a contradiction to the initial assumption that* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are mutually independent with respect to* Γ.

- ***Case 2:*** *Assumption:* $\pi_{S_i \cup S_j}(\Gamma)$ *has an alternative that does not belong to* $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma)$.
  $\Rightarrow \exists \mathcal{C} \in \pi_{S_i \cup S_j}(\Gamma) \colon \mathcal{C} \notin \pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma)$.
  $\Rightarrow \exists \mathcal{C} \in \pi_{S_i \cup S_j}(\Gamma) \colon \nexists \mathcal{C}_i \in \pi_{S_i}(\Gamma), \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j = \mathcal{C}$.
  $\Rightarrow \exists \mathcal{C} \in \Gamma \colon \nexists \mathcal{C}_i \in \pi_{S_i}(\Gamma), \mathcal{C}_j \in \pi_{S_j}(\Gamma) \colon \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ *which is a contradiction to the initial assumption that* $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are independent with respect to* Γ.

Thus, if $\pi_{S_i}(\Gamma)$ *and* $\pi_{S_j}(\Gamma)$ *are mutually independent with respect to* Γ *it follows that* $\pi_{S_i}(\Gamma) \boxtimes \pi_{S_j}(\Gamma) = \pi_{S_i \cup S_j}(\Gamma)$.

A.3. Proofs: Factorization of Uncertain Clusterings

In this section, we prove the theorems of Section 8.3.

Proof of Theorem 21

Theorem 21 *Let* Γ *be an incomplete clustering. Every non range-disjoint factorization* $\mathcal{F}_i$ *of* Γ *can be further factorized into a range-disjoint factorization* $\mathcal{F}_j$ *of* Γ.

Theorem 21 can be proved as follows:

Proof 20 *Due to Theorem 19, a set of incomplete clusterings $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_n}\}$ with overlapping factor sets can only be a factorization of Γ, if each possible cluster of Γ that contains elements from different factor sets belongs to all possible clusterings of the corresponding factors, i.e. if holds that $\forall F_i, F_j \in \mathcal{F}^{set}: F_i \circledcirc F_j$ it follows that $\pi_{F_i \cap F_j}(\Gamma_{F_i}) = \pi_{F_i \cap F_j}(\Gamma_{F_j}) = \pi_{F_i \cap F_j}(\Gamma_{F_i}) \boxtimes \pi_{F_i \cap F_j}(\Gamma_{F_j})$. Consequently, this cluster information is certain and can be modeled by an extra factor with only one alternative. Therefore, any non range-disjoint factorization that has two factor sets F_i and F_j with $F_i \circledcirc F_j$ can be further factorized by creating the factors $\Gamma_{F_x} = \pi_{F_i - F_j}(\Gamma_{F_i})$, $\Gamma_{F_y} = \pi_{F_j - F_i}(\Gamma_{F_j})$, and $\Gamma_{F_z} = \pi_{F_i \cap F_j}(\Gamma_{F_i}) = \pi_{F_i \cap F_j}(\Gamma_{F_j})$.*

Proof of Theorem 22

Theorem 22 *A complete factorization is range-disjoint.*

Theorem 22 can be proved as follows:

Proof 21 *This proof follows directly from Theorem 21 because every non range-disjoint factorization can be factorized further on.*

Proof of Theorem 23

Theorem 23 *Let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of the incomplete clustering Γ, then the following statements are true:*

- *Each factor $\Gamma_F \in \mathcal{F}$ can be computed as: $\Gamma_F = \pi_F(\Gamma)$.*
- *The factor sets form a factorization of every alternative of Γ, i.e. for each $\mathcal{C} \in \Gamma$ the set $\{\pi_F(\mathcal{C}) \mid F \in \mathcal{F}^{set}\}$ is a factorization of $\mathcal{C}$.*
- *Each factor set is a subset of the range of Γ, i.e. $\forall F \in \mathcal{F}^{set}: F \subseteq rng(\Gamma)$*
- *Each element $e \in rng(\Gamma)$ is covered by exact one factor set, i.e. $\bigcup \mathcal{F}^{set} = rng(\Gamma)$*
- *The factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ form a partition of $rng(\Gamma)$ and hence form a cluster-disjoint clustering of $rng(\Gamma)$.*
- *Each two factors Γ_{F_i} and Γ_{F_j} are mutually independent with respect to Γ, i.e. $\forall \mathcal{C}_i \in \Gamma_{F_i}: \forall \mathcal{C}_j \in \Gamma_{F_j}: \exists \mathcal{C} \in \Gamma: \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$ and $\forall \mathcal{C} \in \Gamma: \exists \mathcal{C}_i \in \Gamma_{F_i}: \exists \mathcal{C}_j \in \Gamma_{F_j}: \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}$.*
- *Each factor $\Gamma_F \in \mathcal{F}$ is dominated by Γ, i.e. $\forall \mathcal{C}_F \in \Gamma_F: \exists \mathcal{C} \in \Gamma: \mathcal{C}_F \prec \mathcal{C}$ and $\forall \mathcal{C} \in \Gamma: \exists \mathcal{C}_F \in \Gamma_F: \mathcal{C}_F \prec \mathcal{C}$.*
- *No possible cluster of Γ contains elements from different factor sets, i.e. $\forall C \in possCl(\Gamma): \forall F_i, F_j \in \mathcal{F}^{set}: \neg(C \otimes F_i) \wedge \neg(C \otimes F_j) \Rightarrow F_i = F_j$.*

Theorem 23 can be proved as follows:

Proof 22 *We proof the given statements one after another.*

Statement 1:

Assumption: $\mathcal{F}$ is a factorization of Γ, i.e. $\boxtimes\mathcal{F} = \Gamma$, but $\exists\Gamma_F \in \mathcal{F}\colon \Gamma_F \neq \pi_F(\Gamma)$.
Here we can distinguish between two cases:

- *Case 1: Assumption: Γ_F has an alternative that does not belong to $\pi_F(\Gamma)$, i.e. $\exists\mathcal{C} \in \Gamma_F\colon \mathcal{C} \notin \pi_F(\Gamma)$.*
  $\Rightarrow \exists\mathcal{C} \in \Gamma_F\colon \nexists\mathcal{C}' \in \Gamma\colon \mathcal{C} \prec \mathcal{C}'$.
  $\Rightarrow \exists\mathcal{C} \in \boxtimes\mathcal{F}\colon \mathcal{C} \notin \Gamma$.
  $\Rightarrow \boxtimes\mathcal{F} \neq \Gamma$ which is a contradiction to the initial assumption that $\mathcal{F}$ is a range-disjoint factorization of Γ.

- *Case 2: Assumption: $\pi_F(\Gamma)$ has an alternative that does not belong to Γ_F, i.e. $\exists\mathcal{C} \in \pi_F(\Gamma)\colon \mathcal{C} \notin \Gamma_F$.*
  $\Rightarrow \exists\mathcal{C} \in \pi_F(\Gamma)\colon \nexists\mathcal{C}' \in \boxtimes\mathcal{F}\colon \mathcal{C} \prec \mathcal{C}'$.
  $\Rightarrow \exists\mathcal{C} \in \Gamma\colon \mathcal{C} \notin \boxtimes\mathcal{F}$.
  $\Rightarrow \boxtimes\mathcal{F} \neq \Gamma$ which is a contradiction to the initial assumption that $\mathcal{F}$ is a range-disjoint factorization of Γ.

Statement 2: *Assumption: $\mathcal{F}_\mathcal{C} = \{\pi_F(\mathcal{C}) \mid F \in \mathcal{F}^{set}\}$ is not a factorization of some clustering $\mathcal{C} \in \Gamma$.*
$\Rightarrow \mathcal{C}$ cannot be reconstructed by the integration operator, i.e. $\boxtimes\mathcal{F}_\mathcal{C} \neq \mathcal{C}$.
$\Rightarrow \mathcal{C} \notin \boxtimes\mathcal{F}$.
$\Rightarrow \boxtimes\mathcal{F} \neq \Gamma$ which is a contradiction to the initial assumption that $\mathcal{F}$ is a range-disjoint factorization of Γ.

Statement 3: *Assumption: $\mathcal{F}$ is a range-disjoint factorization of Γ, but the factor set $F \in \mathcal{F}^{set}$ is not a subset of $rng(\Gamma)$.*
$\Rightarrow \exists\Gamma_F \in \mathcal{F}\colon rng(\Gamma_F) \nsubseteq rng(\Gamma)$.
$\Rightarrow \exists\Gamma_F \in \mathcal{F}\colon \forall\mathcal{C} \in \Gamma_F\colon \nexists\mathcal{C}' \in \Gamma\colon C \prec \mathcal{C}'$.
$\Rightarrow \boxtimes\mathcal{F} \neq \Gamma$ which is a contradiction to the initial assumption that $\mathcal{F}$ is a range-disjoint factorization of Γ.

Statement 4: *Trivial, because otherwise some elements of $rng(\Gamma)$ would be missing in $\bigcup\mathcal{F}^{set} = rng(\boxtimes\mathcal{F})$ and Γ could not be reconstructed from $\mathcal{F}$. Moreover, each element of $rng(\Gamma)$ can only belong to one factor set because the factorization is range-disjoint.*

Statement 5:

- *due to Statement 3 it holds that $\bigcup\mathcal{F}^{set} \subseteq rng(\Gamma)$.*
- *due to Statement 4 it holds that $rng(\Gamma) \subseteq \bigcup\mathcal{F}^{set}$.*

From this follows that $\bigcup \mathcal{F}^{set} = rng(\Gamma)$ *and that* $\mathcal{F}^{set}$ *is a clustering of* $rng(\Gamma)$ *where the factor sets are the clusters. Since* $\mathcal{F}$ *is a range-disjoint factorization it additionally holds that all factor sets are disjoint. Consequently,* $\mathcal{F}^{set}$ *is a cluster-disjoint clustering of* $rng(\Gamma)$ *and thus a partition of* $rng(\Gamma)$.

Statement 6: *This proof follows directly from Theorem 20.*

Statement 7: *Since an incomplete clustering* Γ *is dominated by another incomplete clustering* Γ'*, if* $\Gamma = \pi_{rng(\Gamma)}(\Gamma')$ *the proof of this statement follows directly from Statement 1.*

Statement 8:

Assumption: $\mathcal{F}$ *is a factorization of* Γ*, but there is a possible cluster* $C \in possCl(\Gamma)$ *that overlaps with different factor sets in* $\mathcal{F}^{set}$.
$\Rightarrow$ *there is a possible clustering* $\mathcal{C}$ *of* Γ *that contains the cluster* C *that overlaps with different factor sets in* $\mathcal{F}^{set}$.
$\Rightarrow$ *following Theorem 3 and Definition 12 a partition of* $rng(\mathcal{C})$ *only forms a factorization of the certain clustering* $\mathcal{C}$ *if none of its clusters has elements from different partition classes. Thus,* $\mathcal{F}^{set}$ *cannot form a factorization of* $\mathcal{C}$.
$\Rightarrow$ *according to Statement 1,* $\mathcal{F}$ *is only a factorization of* Γ *if* $\mathcal{F}^{set}$ *forms a factorization of every alternative of* Γ.
$\Rightarrow$ $\mathcal{F}$ *is not a factorization of* Γ *which is a contradiction to the initial assumption.*

Proof of Theorem 24

Theorem 24 *Let* $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ *be a range-disjoint factorization of the probabilistic clustering* $\mathfrak{C} = (\Gamma, Pr)$ *where* $\forall i \in \{1, \ldots, k\}\colon \mathfrak{C}_{F_i} = (\Gamma_{F_i}, Pr_i)$*, then the following statements are true:*

- *Each factor* $\mathfrak{C}_{F_i} \in \mathcal{F}$ *can be computed as:* $\mathfrak{C}_{F_i} = \pi_{F_i}(\mathfrak{C})$.
- $\{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *is a range-disjoint factorization of* Γ.
- *The probabilities of the original clustering alternatives are preserved by the factorization, i.e.* $\forall \mathfrak{C}_{F_i} \in \mathcal{F}\colon \forall \mathcal{C}_i \in$ *C-All*$(rng(\mathfrak{C}_{F_i}))\colon Pr_i(\mathcal{C}_i) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \prec \mathcal{C}} Pr(\mathcal{C}) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \subseteq \mathcal{C}} Pr(\mathcal{C})$ *(the last equality holds because of range-disjointness).*
- *The factors are mutually independent with respect to* $\mathfrak{C}$*, i.e.* $\forall \mathfrak{C}_{F_i}, \mathfrak{C}_{F_j} \in \mathcal{F}\colon \forall \mathcal{C}_i \in$ *C-All*$(rng(\mathfrak{C}_{F_i}))\colon \forall \mathcal{C}_j \in$ *C-All*$(rng(\mathfrak{C}_{F_j}))\colon Pr_i(\mathcal{C}_i) \times Pr_j(\mathcal{C}_j) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}} Pr(\mathcal{C}) = \sum_{\mathcal{C} \in \Gamma, \mathcal{C}_i \boxtimes \mathcal{C}_j \subseteq \mathcal{C}} Pr(\mathcal{C})$.

Theorem 24 can be proved as follows:

Proof 23 *We proof the given statements one after another.*

Statement 1: *Assumption:* $\mathcal{F}$ *is a factorization of* $\mathfrak{C}$*, i.e.* $\boxtimes \mathcal{F} = \mathfrak{C}$*, but* $\exists \mathfrak{C}_F \in \mathcal{F}\colon \mathfrak{C}_F \neq \pi_F(\mathfrak{C})$.
According to Theorem 23 the probabilistic clusterings $\mathfrak{C}_F$ *and* $\pi_F(\mathfrak{C})$ *must share the same set of alternatives. Thus, they can be only non-equal if they differ in their probabilities.*

Assumption: $\mathcal{F}$ is a factorization of $\mathfrak{C}$, but $\exists \mathcal{C} \in \Gamma_F \colon Pr_F(\mathcal{C}) \neq \sum_{\mathcal{C}' \in \Gamma, \mathcal{C} = \pi_F(\mathcal{C}')} Pr(\mathcal{C}')$.
$\Rightarrow \exists \mathcal{C} \in \Gamma_F \colon Pr_F(\mathcal{C}) \neq \sum_{\mathcal{C}' \in \Gamma, \mathcal{C} \prec \mathcal{C}'} Pr(\mathcal{C}')$.
let $\mathfrak{C}_F = (\Gamma_{\mathcal{F}}, Pr_{\mathcal{F}})$ be the probabilistic clustering that results from integrating the factors of $\mathcal{F}$
$\Rightarrow \exists \mathcal{C} \in \Gamma_{\mathcal{F}}, : Pr_{\mathcal{F}}(\mathcal{C}) \neq \sum_{\mathcal{C}' \in \Gamma, \mathcal{C} \prec \mathcal{C}'} Pr(\mathcal{C}')$.
$\Rightarrow \exists \mathcal{C} \in \Gamma_{\mathcal{F}}, : Pr_{\mathcal{F}}(\mathcal{C}) \neq Pr(\mathcal{C})$.
$\Rightarrow \bigboxtimes \mathcal{F} \neq \mathfrak{C}$ *which is a contradiction to the initial assumption.*

Statement 2: *Assumption: $\bigboxtimes \mathcal{F} = \mathfrak{C}$, but $\bigboxtimes_{i=1}^{k} \Gamma_{F_i} \neq \Gamma$.*
This proof is trivial because two probabilistic clusterings can only be equivalent if they have the same set of alternatives. Thus, from $\bigboxtimes_{i=1}^{k} \Gamma_{F_i} \neq \Gamma$ it follows that $\bigboxtimes \mathcal{F} \neq \mathfrak{C}$ which is a contradiction to the initial assumption.

Statement 3: *This proof follows directly from the fact that $\mathfrak{C}_{F_i} = \pi_{F_i}(\mathfrak{C})$ (Statement 1) and the definition of the probabilistic version of the projection operator (Equation 8.3).*

Statement 4: *This proof follows directly from the definition of the probabilistic version of the integration operator (Equation 8.5) and the probabilistic version of the projection operator (Equation 8.3).*
Let $\mathfrak{C}_{ij} = (\Gamma_{ij}, Pr_{ij}) = \mathfrak{C}_{F_i} \boxtimes \mathfrak{C}_{F_j}$. Due to the integration operator holds:
$\forall \mathcal{C} \in rng(\mathfrak{C}_{F_i} \boxtimes \mathfrak{C}_{F_j}) \colon Pr_i(\pi_{F_i}(\mathcal{C})) \times Pr_j(\pi_{F_j}(\mathcal{C})) = Pr_{ij}(\mathcal{C})$.
$\Rightarrow \forall \mathcal{C}_i \in rng(\mathfrak{C}_{F_i}) \colon \forall \mathcal{C}_j \in rng(\mathfrak{C}_{F_j}) \colon Pr_i(\pi_{F_i}(\mathcal{C})) \times Pr_j(\pi_{F_j}(\mathcal{C})) = Pr_{ij}(\mathcal{C})$.

Due to the projection operator holds:
$\forall \mathcal{C} \in rng(\pi_{F_i \cup F_j}(\mathfrak{C})) \colon \sum_{\mathcal{C}' \in \Gamma, \mathcal{C} = \pi_{F_i \cup F_j}(\mathcal{C}')} Pr(\mathcal{C}') = Pr_{ij}(\mathcal{C})$.
$\Rightarrow \forall \mathcal{C} \in rng(\pi_{F_i \cup F_j}(\mathfrak{C})) \colon \sum_{\mathcal{C}' \in \Gamma, \mathcal{C} \prec \mathcal{C}'} Pr(\mathcal{C}') = Pr_{ij}(\mathcal{C})$.
$\Rightarrow \forall \mathcal{C} \in rng(\mathfrak{C}_{F_i} \bigboxtimes \mathfrak{C}_{F_j}) \colon \sum_{\mathcal{C}' \in \Gamma, \mathcal{C} \prec \mathcal{C}'} Pr(\mathcal{C}') = Pr_{ij}(\mathcal{C})$.
$\Rightarrow \forall \mathcal{C}_i \in rng(\mathfrak{C}_{F_i}) \colon \forall \mathcal{C}_j \in rng(\mathfrak{C}_{F_j}) \colon \sum_{\mathcal{C}' \in \Gamma, \mathcal{C}_i \boxtimes \mathcal{C}_j \prec \mathcal{C}'} Pr(\mathcal{C}') = Pr_{ij}(\mathcal{C})$.

Proof of Theorem 25

Theorem 25 *Let Γ be an incomplete clustering and let $\mathcal{F}_m$ and $\mathcal{F}_n$ be two range-disjoint factorizations of Γ. If $\mathcal{F}_m$ is a finer factorization of Γ than $\mathcal{F}_n$, the factors $\{\Gamma_{F_m} \mid \Gamma_{F_m} \in \mathcal{F}_m, rng(\Gamma_{F_m}) \subseteq rng(\Gamma_F)\}$ form a range-disjoint factorization of the factor $\Gamma_F \in \mathcal{F}_n$.*

Theorem 25 can be proved as follows:

Proof 24 *Let $\mathcal{F}_m^{set}$ be the set of factor sets of $\mathcal{F}_m$ and let $\mathcal{F}_n^{set}$ be the set of factor sets of $\mathcal{F}_n$. Moreover, let $F = rng(\Gamma_F)$ and let $\mathcal{F}_F^{set} = \{rng(\Gamma_{F_m}) \mid \Gamma_{F_m} \in \mathcal{F}_m, rng(\Gamma_{F_m}) \subseteq rng(\Gamma_F)\} = \{F_1, \ldots, F_k\}$. Since $\mathcal{F}_m$ is a finer factorization of Γ than $\mathcal{F}_n$, $\mathcal{F}_m^{set}$ is a finer partition than $\mathcal{F}_n^{set}$. Thus, each factor set of $\mathcal{F}_m^{set}$ is a subset of some factor set of $\mathcal{F}_n^{set}$ and because both factorizations are range-disjoint each factor set of $\mathcal{F}_m^{set}$ that overlaps with F is a subset of it. Consequently, it holds that $\mathcal{F}_F^{set}$ is a partition of F and thus it holds $\bigcup_{i=1}^{k} F_i = F$.*

The set $\mathcal{F}_F = \{\Gamma_{F_m} \mid \Gamma_{F_m} \in \mathcal{F}_m, rng(\Gamma_{F_m}) \subseteq rng(\Gamma_F)\} = \{\Gamma_1, \ldots, \Gamma_k\}$ is a factorization of Γ_F, if the latter can be reconstructed by integrating the elements of the first. By using the equivalence between F and $\bigcup_{i=1}^{k} F_i$ that we have proved above, this in turn can be proved as follows:

$$\begin{aligned} \boxtimes\mathcal{F}_F &= \boxtimes_{i=1}^{k}\Gamma_{F_i} \\ \textit{due to Theorem 23} &= \boxtimes_{i=1}^{k}\pi_{F_i}(\Gamma) \\ \textit{due to Theorem 20} &= \pi_{\bigcup_{i=1}^{k} F_i}(\Gamma) = \pi_F(\Gamma) = \Gamma_F \end{aligned}$$

A.4. Proofs: Duplicate Clustering with Uncertainty

In this section, we prove the theorems of Section 8.7.

Proof of Theorem 26

Theorem 26 *A* W-graph $G = (N, E, P)$ *is consistent, if and only if G is equivalent to its transitive closure, i.e. $G = G^*$.*

Theorem 26 can be proved as follows:

Proof 25 *($\Rightarrow$) Assumption: $G \neq G^*$, but G is consistent.*
Due to $G \neq G^*$
$\Rightarrow \exists n_i, n_j, n_k \in N\colon \{n_i, n_j\}, \{n_i, n_k\} \in E \wedge \{n_j, n_k\} \notin E$.
Let W be the world that is represented by G.
$\Rightarrow \exists e_i, e_j, e_k \in W\colon \omega(e_i) = \omega(e_j) \wedge \omega(e_i) = \omega(e_k) \wedge \omega(e_j) \neq \omega(e_k)$.
$\Rightarrow$ *the world W is impossible.*
$\Rightarrow$ *G is inconsistent which is a contradiction to the initial assumption.*
Therefore, a consistent W-graph *is always equivalent to its transitive closure.*

($\Leftarrow$) Assumption: G is inconsistent, but $G = G^*$.
Let W be the world that is represented by G.
Due to G is inconsistent
$\Rightarrow$ *the world W is impossible.*
$\Rightarrow \exists e_i, e_j, e_k \in W\colon \omega(e_i) = \omega(e_j) \wedge \omega(e_i) = \omega(e_k) \wedge \omega(e_j) \neq \omega(e_k)$.
$\Rightarrow \exists n_i, n_j, n_k \in N\colon \{n_i, n_j\}, \{n_i, n_k\} \in E \wedge \{n_j, n_k\} \notin E$.
$\Rightarrow$ *$G \neq G^*$ which is a contradiction to the initial assumption.*
Therefore, a W-graph *that is equivalent to its transitive closure is always consistent.*

Proof of Theorem 27

Theorem 27 *An* M-graph $M = (N, E, \gamma)$ *is inconsistent with respect to the* W-graph *computation mapping $f^{Naive}_{MG \mapsto WG}$ if and only if it contains at least one $\triangle$-conflict.*

We prove Theorem 27 step by step. First, we consider statement that an *M-graph* with a $\triangle$-*conflict* is per se inconsistent.

Lemma 1 *An* M-graph $M = (N, E, \gamma)$ *that has at least one* $\triangle$*-conflict is inconsistent with respect to the* W-graph *generation mapping* $f^{Naive}_{MG \mapsto WG}$.

Lemma 1 can be proved as follows:

Proof 26 *Assumption:* M *has a* $\triangle$*-conflict, but is consistent with respect to* $f^{Naive}_{MG \mapsto WG}$.
Due to M *has a* $\triangle$*-conflict*
$\Rightarrow \exists n_i, n_j, n_k \in N : \{n_i, n_j\}, \{n_i, n_k\} \in E^+ \wedge \{n_j, n_k\} \in E^-$.
$\Rightarrow \forall G = (N, E_G, Pr_G) \in f^{Naive}_{MG \mapsto WG}(M)$: $\{n_i, n_j\}, \{n_i, n_k\} \in E_G \wedge \{n_j, n_k\} \notin E_G$.
$\Rightarrow \forall G \in f^{Naive}_{MG \mapsto WG}(M)$: $G \neq G^*$.
$\Rightarrow \forall G \in f^{Naive}_{MG \mapsto WG}(M)$: *G is inconsistent.*
$\Rightarrow$ *M is inconsistent which is a contradiction to the initial assumption.*
Therefore, an M-graph *is always inconsistent with respect to* $f^{Naive}_{MG \mapsto WG}$ *if it contains at least one* $\triangle$*-conflict.*

Then we consider the relationship between the consistency of *M-graphs* that do not nave uncertain edges and $\triangle$*-conflicts*.

Lemma 2 *An* M-graph $M = (N, E, \gamma)$ *with no uncertain edge, i.e.* $E = E^!$*, is inconsistent with respect to* $f^{Naive}_{MG \mapsto WG}$ *if and only if it contains at least one* $\triangle$*-conflict.*

Lemma 2 can be proved as follows:

Proof 27 *(*$\Rightarrow$*) Assumption:* M *is inconsistent with respect to* $f^{Naive}_{MG \mapsto WG}$*, but does not contain a* $\triangle$*-conflict.*
Due to M *has no uncertain edge*
$\Rightarrow$ *from* $f^{Naive}_{MG \mapsto WG}(M)$ *only a single* W-graph $G = (N, E_G, Pr_G)$ *results where* $E_G = E^+$*, i.e.* G *is generated from* M *by removing all negative edges.*
Due to M *is inconsistent with respect to* $f^{Naive}_{MG \mapsto WG} \Rightarrow$ *from* $f^{Naive}_{MG \mapsto WG}(M)$ *no consistent* W-graph *result.*
$\Rightarrow$ *the single* W-graph G *is inconsistent.*
$\Rightarrow G \neq G^*$ *(following Theorem 26).*
$\Rightarrow \exists n_i, n_j, n_k \in N : \{n_i, n_j\}, \{n_i, n_k\} \in E_G \wedge \{n_j, n_k\} \notin E_G$.
$\Rightarrow \exists n_i, n_j, n_k \in N : \{n_i, n_j\}, \{n_i, n_k\} \in E^+ \wedge \{n_j, n_k\} \in E^-$.
$\Rightarrow$ M *contains a* $\triangle$*-conflict which is a contradiction to the initial assumption.*
Thus, an M-graph *with no uncertain edge that is inconsistent with respect to* $f^{Naive}_{MG \mapsto WG}$ *contains at least one* $\triangle$*-conflict.*

($\Leftarrow$*) Assumption:* M *contains a* $\triangle$*-conflict, but is consistent with respect to* $f^{Naive}_{MG \mapsto WG}$.
This assumption is disproved by Lemma 1.

The question is now whether or not the insertion of a new $\triangle$-*conflict* can be avoided if we replace an uncertain edge by a certain one (positive or negative) .

Lemma 3 *Given a $\triangle$-conflict-free* M-graph $M = (N, E, \gamma)$. *An uncertain edge $e \in E^?$ can either be changed to a certain positive edge without causing a $\triangle$-conflict or can be changed to a negative edge without causing a $\triangle$-conflict.*

Lemma 3 can be proved as follows:

Proof 28 *Assumption: M is $\triangle$-conflict-free, but changing the uncertain edge $\{n_i, n_j\} \in E^?$ to a certain edge causes a $\triangle$-conflict regardless of whether $\{n_i, n_j\}$ is changed to a certain positive edge or is changed to a negative edge.*
Changing $\{n_i, n_j\}$ from $E^?$ to E^- causes a $\triangle$-conflict.
$\Rightarrow \exists n_k \in N : \{n_i, n_k\}, \{n_j, n_k\} \in E^+$.
Moreover, changing $\{n_i, n_j\}$ from $E^?$ to E^+ causes a $\triangle$-conflict.
$\Rightarrow \exists n_l \in N : \{n_i, n_l\} \in E^+ \wedge \{n_j, n_l\} \in E^-$.
Since $n_k \neq n_l$ we have to distinguish between two cases:
(Case 1)

Assumption: $\{n_k, n_l\} \in E^+$.
$\Rightarrow$ the three edges $\{n_k, n_l\}, \{n_j, n_k\} \in E^+$ and $\{n_j, n_l\} \in E^-$ form a $\triangle$-conflict.
$\Rightarrow$ M already contained a $\triangle$-conflict before changing $\{n_i, n_j\}$ which is a contradiction to the initial assumption.

(Case 2)

Assumption: $\{n_k, n_l\} \in E^-$.
$\Rightarrow$ the three edges $\{n_i, n_k\}, \{n_i, n_l\} \in E^+$ and $\{n_k, n_l\} \in E^-$ form a $\triangle$-conflict.
$\Rightarrow$ M already contained a $\triangle$-conflict before changing $\{n_i, n_j\}$ which is a contradiction to the initial assumption.

Consequently, the assumption is not true and in a $\triangle$-conflict-free M-graph *an uncertain edge can always be changed to a certain edge (positive or negative) without causing a new $\triangle$-conflict.*

Using these lemmas we can prove Theorem 27 as follows:

Proof 29 *($\Rightarrow$) Assumption: M is inconsistent with respect to $f^{Naive}_{MG \mapsto WG}$, but does not contain a $\triangle$-conflict.*
M is $\triangle$-conflict-free $\Rightarrow$ we can change each uncertain edge of M to a certain edge one after another without causing a single $\triangle$-conflict (Lemma 3).
$\Rightarrow$ the resultant M-graph *M' has no uncertain edge and does not have a $\triangle$-conflict.*
$\Rightarrow$ the resultant M-graph *M' is consistent with respect to $f^{Naive}_{MG \mapsto WG}$ (Lemma 2).*
$\Rightarrow$ there is at least one consistent W-graph *that result from $f^{Naive}_{MG \mapsto WG}(M)$, i.e. the* W-graph *that contains the positive edges from M'.*
$\Rightarrow$ M is not inconsistent with respect to $f^{Naive}_{MG \mapsto WG}$ which is a contradiction to the initial assumption.
Thus, each M-graph *that is inconsistent with respect to $f^{Naive}_{MG \mapsto WG}$ contains at least one $\triangle$-conflict.*

This direction can be alternatively proved as follows:
($\Rightarrow$) Assumption: M is inconsistent with respect to $f^{Naive}_{MG \mapsto WG}$, but does not contain a $\triangle$-conflict.
M is inconsistent $\Rightarrow$ every possible sequence of changing all uncertain edges to certain edges must

result in an inconsistent M-graph.
Following Lemma 2, an M-graph *without an uncertain edge is only inconsistent if it contains a* $\triangle$*-conflict.*
$\Rightarrow$ *every possible sequence of changing all uncertain edges to certain edges must produce a* $\triangle$*-conflict.*
$\Rightarrow$ *there is no sequence of changing all uncertain edges to certain edges that does not produce a* $\triangle$*-conflict.*
$\Rightarrow$ *following Lemma 3 such a sequence only does not exist if the initial* M-graph *already has a* $\triangle$*-conflict which is a contradiction to the initial assumption.*
Thus, each M-graph *that is inconsistent with respect to* $f^{Naive}_{MG\mapsto WG}$ *contains at least one* $\triangle$*-conflict.*

($\Leftarrow$*) Assumption:* M *contains a* $\triangle$*-conflict, but is consistent with respect to* $f^{Naive}_{MG\mapsto WG}$*.*
This assumption is disproved by Lemma 1.

Proof of Theorem 28

Theorem 28 *Let* $M{=}(N, E.\gamma)$ *be an* M-graph *and let* $f_{dec}(M){=}\{M_1, \ldots, M_k\}$ *be its set of* partial M-graphs *where* $M_i{=}(N_i, E_i, \gamma)$ *for each* $i \in \{1, \ldots, k\}$*. The set* $f^{Naive}_{MG\mapsto WG}(M)$ *can be reconstructed from the sets* $\{f^{Naive}_{MG\mapsto WG}(M_1), \ldots, f^{Naive}_{MG\mapsto WG}(M_k)\}$ *by using the* W-graph *integration operator* ${}^{G}\boxtimes$ *that is defined as:*

$${}^{G}\boxtimes_{i=1}^{k}\, f^{Naive}_{MG\mapsto WG}(M_i) = \{(\bigcup_{i=1}^{k} N_i, \bigcup_{i=1}^{k} E_{G_i}, \prod_{i=1}^{k} Pr_{G_i}) \mid \forall i{\in}\{1, \ldots, k\}\colon G_i \in f^{Naive}_{MG\mapsto WG}(M_i)\}$$

where $G_i = (N_i, E_{G_i}, Pr_{G_i})$ *for each* $i \in \{1, \ldots, k\}$.

Theorem 28 can be proved as follows:

Proof 30 *Because each node of* N *belong to exact one* partial M-graph*, it holds:*

$$N = \bigcup_{i=1}^{k} N_i$$

Because each edge of E *belong to exact one* partial M-graph*, it holds:*

$$E = \bigcup_{i=1}^{k} E_i \qquad E^{+} = \bigcup_{i=1}^{k} E_i^{+} \qquad E^{?} = \bigcup_{i=1}^{k} E_i^{?}$$

Moreover it holds:

$$\begin{aligned}
\forall K \subseteq E^{?}\colon Pr(K) &= \prod_{e\in K} \gamma(e) \prod_{e\in E^{?}-K} (1-\gamma(e)) \\
&= \prod_{i=1}^{k} \big(\prod_{e\in K\cap E_i^{?}} \gamma(e) \prod_{e\in (E^{?}-K)\cap E_i^{?}} (1-\gamma(e))\big) \\
&= \prod_{i=1}^{k} \big(\prod_{e\in K\cap E_i^{?}} \gamma(e) \prod_{e\in E_i^{?}-(K\cap E_i^{?})} (1-\gamma(e))\big) \\
&= \prod_{i=1}^{k} Pr(K \cap E_i^{?})
\end{aligned}$$

$$
\begin{aligned}
f^{Naive}_{MG\mapsto WG}(M) &= \{(N, E^+ \cup K, Pr(K)) \mid K \subseteq E^?\} \\
&= \{(N, E^+ \cup K, Pr(K)) \mid K \subseteq \bigcup_{i=1}^{k} E_i^?\} \\
&= \{(\bigcup_{i=1}^{k} N_i, \bigcup_{i=1}^{k} E_i^+ \cup K, \prod_{i=1}^{k} Pr(K \cap E_i^?)) \mid K \subseteq \bigcup_{i=1}^{k} E_i^?\} \\
&= \{(\bigcup_{i=1}^{k} N_i, \bigcup_{i=1}^{k} E_i^+ \cup \bigcup_{i=1}^{k} (K \cap E_i^?), \prod_{i=1}^{k} Pr(K \cap E_i^?)) \mid K \subseteq \bigcup_{i=1}^{k} E_i^?\}
\end{aligned}
$$

by setting $K_i = K \cap E_i^?$

$$
\begin{aligned}
&= \{(\bigcup_{i=1}^{k} N_i, \bigcup_{i=1}^{k} E_i^+ \cup \bigcup_{i=1}^{k} K_i, \prod_{i=1}^{k} Pr(K_i)) \mid \forall i \in \{1, \ldots, k\} \colon K_i \subseteq E_i^?\} \\
&= \{(\bigcup_{i=1}^{k} N_i, \bigcup_{i=1}^{k} (E_i^+ \cup K_i), \prod_{i=1}^{k} Pr(K_i)) \mid \forall i \in \{1, \ldots, k\} \colon K_i \subseteq E_i^?\}
\end{aligned}
$$

due to $f^{Naive}_{MG\mapsto WG}(M_i) = \{(N_i, E_i^+ \cup K_i, Pr(K_i)) \mid K_i \subseteq E_i^?\}$ *for each* $i \in \{1, \ldots, k\}$

$$
\begin{aligned}
&= \{(\bigcup_{i=1}^{k} N_i, \bigcup_{i=1}^{k} E_{G_i}, \prod_{i=1}^{k} Pr_{G_i}) \mid \forall i \in \{1, \ldots, k\} \colon G_i \in f^{Naive}_{MG\mapsto WG}(M_i)\} \\
&\quad \textit{where } G_i = (N_i, E_{G_i}, Pr_{G_i}) \textit{ for each } i \in \{1, \ldots, k\}
\end{aligned}
$$

Proof of Theorem 29

Theorem 29 *Let* $f^{Naive}_{MG\mapsto Cl}$ *be the mapping that maps an* M-graph *to its probabilistic clustering by using the* W-graph *computation mapping* $f^{Naive}_{MG\mapsto WG}$*. Moreover, let* M *be an* M-graph *and let* $f_{dec}(M) = \{M_1 \ldots, M_k\}$ *be its set of* partial M-graphs*. The set of probabilistic clusterings that results from applying* $f_{MG\mapsto Cl}$ *to each* partial M-graph *of* M *separately is a factorization of the probabilistic clustering that results from applying* $f^{Naive}_{MG\mapsto Cl}$ *to* M *itself, i.e.*

$$f^{Naive}_{MG\mapsto Cl}(M) = \boxtimes_{i=1}^{k} f^{Naive}_{MG\mapsto Cl}(M_i)$$

Theorem 29 can be proved as follows:

Proof 31 *The proof follows directly from Theorem 28 because in the case of range-disjoint factors the* W-graph *integration operator is the graph equivalent of the clustering integration operator. This equivalence in turn is obvious because the probability of a reconstructed graph is exactly defined as the probability of a reconstructed clustering and in the case of range-disjointness a union of nodes and a union of edges corresponds to the union of clusters.*

Proof of Theorem 30

Theorem 30 *Let* M *be an arbitrary* M-graph*, it holds that:* $f^{HC}_{MG\mapsto WG}(M) \subseteq f^{Naive}_{MG\mapsto WG}(M)$

Theorem 30 can be proved as follows:

Proof 32 *This proof directly follows from the definitions of the mappings $f^{Naive}_{MG\mapsto WG}$ and $f^{HC}_{MG\mapsto WG}$ because both mappings only compute* W-graphs *by selecting all positive edges and by selecting a subset of all uncertain edges to produce a single* W-graph. *Due to there is no set of uncertain edges that is not considered by $f^{Naive}_{MG\mapsto WG}$, $f^{HC}_{MG\mapsto WG}$ cannot compute a* W-graph *that is not computed by $f^{Naive}_{MG\mapsto WG}$. From this follows that $f^{HC}_{MG\mapsto WG}(M) \subseteq f^{Naive}_{MG\mapsto WG}(M)$.*

Proof of Theorem 32

Theorem 32 *Let $M = (N, E, \gamma)$ be an* M-graph *and let $f_{dec}(M) = \{M_1, \ldots, M_k\}$ be its set of* partial M-graphs*, it holds: $f^{HC}_{MG\mapsto WG}(M) \subseteq {}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i) \subseteq f^{Naive}_{MG\mapsto WG}(M)$.*

Theorem 32 can be proved as follows:

Proof 33 *We start with proving the inclusion ${}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i) \subseteq f^{Naive}_{MG\mapsto WG}(M)$.*
According to Theorem 32 it holds that $f^{HC}_{MG\mapsto WG}(M) \subseteq f^{Naive}_{MG\mapsto WG}(M)$. As a consequence, it holds that $f^{HC}_{MG\mapsto WG}(M_i) \subseteq f^{Naive}_{MG\mapsto WG}(M_i)$ for every $i \in \{1, \ldots, k\}$. From this follows that ${}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i) \subseteq {}^{G}\bigboxtimes_{i=1}^{k} f^{Naive}_{MG\mapsto WG}(M_i)$. According to Theorem 28 it holds that $f^{Naive}_{MG\mapsto WG}(M) = {}^{G}\bigboxtimes_{i=1}^{k} f^{Naive}_{MG\mapsto WG}(M_i)$. From this follows that ${}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i) \subseteq f^{Naive}_{MG\mapsto WG}(M)$.

We continue with proving the inclusion $f^{HC}_{MG\mapsto WG}(M) \subseteq {}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i)$.
Assumption: $f^{HC}_{MG\mapsto WG}(M)$ has a W-graph *G that does not belong to ${}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i)$.*
Since M can be decomposed into the partial M-graphs *$\{M_1, \ldots, M_k\}$ there is no non-negative edge in M whose nodes belong to different* partial M-graphs*. From this follows that no* W-graph *in $f^{HC}_{MG\mapsto WG}(M)$ can contain an edge that connects nodes that belong to different* partial M-graphs*. From this follows that the* W-graph *G can be decomposed into the node-disjoint subgraphs G_1 to G_k so that each of these subgraphs is node equivalent to one of the* partial M-graphs*. If G does not belong to ${}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i)$ there must be at least one subgraph of G that does not belong to the set of* W-graphs *that is computed from the subgraph's node equivalent* partial M-graph*, i.e. $\exists i \in \{1, \ldots, k\} \colon G_i \notin f^{HC}_{MG\mapsto WG}(M_i)$. Thus, let us assume there is a subgraph $G_x = (N_x, E^*, Pr^*)$ of G that does not belong to the set $f^{HC}_{MG\mapsto WG}(M_x)$ of the node equivalent* partial M-graph *$M_x = (N_x, E_x, \gamma)$. Since M_x and G_x are node equivalent and M_x contains all edges of M that connects nodes in N_x, E^* must be a subset of E_x.*

Let γ^* be the lowest weight of all edges in E that belongs to E^*. The method $f^{HC}_{MG\mapsto WG}$ computes W-graphs *by adding edges in decreasing order of their weights. Thus, the fact that G_x does not belong to $f^{HC}_{MG\mapsto WG}(M_x)$ means that either G_x has an edge $e \in E^*$ with $\gamma(e) \geq \gamma^*$ that is not in E_x or M_x has an edge $e \in E^*$ with $\gamma(e) \geq \gamma^*$ that is not in E^*. The first case can be excluded because $E^* \subseteq E_x$. Thus, let us assume that M_x has an edge $e \in E_x$ with $\gamma(e) \geq \gamma^*$ that is not in E^*. Since M_x only contain edges that are contained in M and because G only belong to $f^{HC}_{MG\mapsto WG}(M)$ if it contains all edges that have a weight greater than γ^*, edge e must belong to G as well. If e belongs to M_x it must belong to the node equivalent subgraph of G which is G_x. Thus, e must belong to G_x which is a contradiction to the made assumption that there is an edge $e \in E_x$ with $\gamma(e) \geq \gamma^*$ that is not in E^*. From this follows that G_x belongs to $f^{HC}_{MG\mapsto WG}(M_x)$ which in turn is a contradiction to the assumption that G_x does not belong to $f^{HC}_{MG\mapsto WG}(M_x)$. As a consequence, G belongs to ${}^{G}\bigboxtimes_{i=1}^{k} f^{HC}_{MG\mapsto WG}(M_i)$ which is a contradiction to the*

initial assumption. From this follows that $f^{HC}_{MG\mapsto WG}(M)$ *does not have a* W-graph *that does not belong to* ${}^{G}\boxtimes^{k}_{i=1}\ f^{HC}_{MG\mapsto WG}(M_i)$ *and therefore it holds that* $f^{HC}_{MG\mapsto WG}(M) \subseteq {}^{G}\boxtimes^{k}_{i=1}\ f^{HC}_{MG\mapsto WG}(M_i)$.

A.5. Proofs: Factor-based Quality Computation

In this section, we prove the theorems of Section 8.8.5.

Proof of Theorem 33

Theorem 33 *Let* $\mathcal{C}$ *be a duplicate clustering and let* $\mathcal{C}_{gold}$ *be the corresponding gold standard, if* $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ *is a factor set that forms a range-disjoint factorization* $\mathcal{F}_C$ *of* $\mathcal{C}$ *and forms a range-disjoint factorization* $\mathcal{F}_{\mathcal{C}_{gold}}$ *of* $\mathcal{C}_{gold}$*, i.e.* $\boxtimes^{k}_{i=1}\pi_{F_i}(\mathcal{C}) = \mathcal{C}$ *and* $\boxtimes^{k}_{i=1}\pi_{F_i}(\mathcal{C}_{gold}) = \mathcal{C}_{gold}$*, the Closest Cluster Recall of* $\mathcal{C}$ *with respect to* $\mathcal{C}_{gold}$ *can be computed as:*

$$ccRec(\mathcal{C}, \mathcal{C}_{gold}) = \sum\nolimits_{i=1}^{k} \frac{|\pi_{F_i}(\mathcal{C}_{gold})|}{|\mathcal{C}_{gold}|} \times ccRec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))$$

Theorem 33 can be proved as follows:

Proof 34

$$ccRec(\mathcal{C}, \mathcal{C}_{gold}) = \frac{1}{|\mathcal{C}_{gold}|} \times \sum\nolimits_{C_q \in \mathcal{C}_{gold}} \boldsymbol{max}_{\mathcal{C}_p \in \mathcal{C}} sim(C_p, C_q)$$

due to each cluster of $\mathcal{C}_{gold}$ *and each cluster of* $\mathcal{C}$ *belongs to exactly one factor*

$$\begin{aligned}
&= \frac{1}{|\mathcal{C}_{gold}|} \times \sum\nolimits_{i=1}^{k} \sum\nolimits_{C_q \in \pi_{F_i}(\mathcal{C}_{gold})} \boldsymbol{max}_{\mathcal{C}_p \in \pi_{F_i}(\mathcal{C})} sim(C_p, C_q) \\
&= \sum\nolimits_{i=1}^{k} \frac{1}{|\mathcal{C}_{gold}|} \times \Big(\sum\nolimits_{C_q \in \pi_{F_i}(\mathcal{C}_{gold})} \boldsymbol{max}_{\mathcal{C}_p \in \pi_{F_i}(\mathcal{C})} sim(C_p, C_q) \Big) \\
&= \sum\nolimits_{i=1}^{k} \left(\frac{|\pi_{F_i}(\mathcal{C}_{gold})|}{|\mathcal{C}_{gold}|} \times \frac{\sum_{C_q \in \pi_{F_i}(\mathcal{C}_{gold})} \boldsymbol{max}_{\mathcal{C}_p \in \pi_{F_i}(\mathcal{C})} sim(C_p, C_q)}{|\pi_{F_i}(\mathcal{C}_{gold})|} \right) \\
&= \sum\nolimits_{i=1}^{k} \frac{|\pi_{F_i}(\mathcal{C}_{gold})|}{|\mathcal{C}_{gold}|} \times ccRec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))
\end{aligned}$$

Proof of Theorem 34

Theorem 34 *Let QM be a* Θ*-decomposable quality measure and let* $\mathcal{F}$ *be a range-disjoint factorization of an incomplete clustering* Γ *with the factor sets* $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$*, the incomplete version of QM can be computed as:*

$$QM^{PWS}(\Gamma, \mathcal{C}_{gold}) = \Theta^{k}_{i=1} f_A\big(\pi_{F_i}(\Gamma), \pi_{F_i}(\mathcal{C}_{gold})\big)$$

Theorem 34 can be proved as follows:

Proof 35

$$\begin{aligned}
QM^{PWS}(\Gamma, \mathcal{C}_{gold}) &= \{QM(\mathcal{C}, \mathcal{C}_{gold}) \mid \mathcal{C} \in \Gamma\} \\
&\overset{Def.\ 75}{=} \left\{\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big) \mid \mathcal{C} \in \Gamma\right\} \\
&\overset{Eq.\ 6.2}{=} \Theta_{i=1}^{k}\{f_A\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big) \mid \mathcal{C} \in \Gamma\} \\
&\overset{Eq.\ 8.1}{=} \Theta_{i=1}^{k}\{f_A\big(\mathcal{C}, \pi_{F_i}(\mathcal{C}_{gold})\big) \mid \mathcal{C} \in \pi_{F_i}(\Gamma)\} \\
&\overset{Theo.\ 4}{=} \Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\Gamma), \pi_{F_i}(\mathcal{C}_{gold})\big)
\end{aligned}$$

Proof of Theorem 35

Theorem 35 *Let QM be a quasi-Θ-decomposable quality measure and let $\mathcal{F}$ be a range-disjoint factorization of an incomplete clustering Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, the incomplete version of QM can be computed as:*

$$QM^{PWS}(\Gamma, \mathcal{C}_{gold}) = f_C\Big(\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\Gamma), \pi_{F_i}(\mathcal{C}_{gold})\big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big)$$

Theorem 35 can be proved as follows:

Proof 36

$$\begin{aligned}
QM^{PWS}(\Gamma, \mathcal{C}_{gold}) &= \{QM(\mathcal{C}, \mathcal{C}_{gold}) \mid \mathcal{C} \in \Gamma\} \\
&\overset{Def.\ 76}{=} \left\{f_C\Big(\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big) \mid \mathcal{C} \in \Gamma\right\} \\
&\overset{Theo.\ 4}{=} f_C\left(\left\{\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big) \mid \mathcal{C} \in \Gamma\right\}, f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\right) \\
&\overset{Eq.\ 6.2}{=} f_C\Big(\Theta_{i=1}^{k}\{f_A\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big) \mid \mathcal{C} \in \Gamma\}, f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big) \\
&\overset{Eq.\ 8.1}{=} f_C\Big(\Theta_{i=1}^{k}\{f_A\big(\mathcal{C}, \pi_{F_i}(\mathcal{C}_{gold})\big) \mid \mathcal{C} \in \pi_{F_i}(\Gamma)\}, f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big) \\
&\overset{Theo.\ 4}{=} f_C\Big(\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\Gamma), \pi_{F_i}(\mathcal{C}_{gold})\big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big)
\end{aligned}$$

Proof of Theorem 36

Theorem 36 *Let QM be a quality measure that is Θ-decomposable with the operator $\Theta \in \{\Sigma, \Pi\}$ and the function f_A. Moreover, let $\mathfrak{C}$ a probabilistic clustering that is represented by the range-disjoint factorization $\mathcal{F}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, it holds that:*

$$EXP\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{gold})\big) = \Theta_{i=1}^{k} EXP\Big(f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big)$$

Theorem 36 can be proved as follows:

Proof 37

$$\begin{aligned} EXP\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{gold})\big) &\overset{\text{Def. 75}}{=} EXP\Big(\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big) \\ &\overset{\text{Eq. 8.51 and Eq. 8.52}}{=} \Theta_{i=1}^{k} EXP\Big(f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big) \end{aligned}$$

Proof of Theorem 37

Theorem 37 *Let QM be a quality measure that is quasi-Θ-decomposable with the operator $\Theta \in \{\Sigma, \Pi\}$ and the functions f_A, f_B, and f_C. Moreover, let $\mathfrak{C}$ a probabilistic clustering that is represented by the range-disjoint factorization $\mathcal{F}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. If f_B produces a numerical value and if f_C is either the sum operator or the multiplication operator, i.e. $f_C(x, y) = x + y$ or $f_C(x, y) = x \times y$, it holds that:*

$$EXP\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{gold})\big) = f_C\Big(\Theta_{i=1}^{k} EXP\Big(f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big)$$

Theorem 37 can be proved as follows:

Proof 38

$$\begin{aligned} EXP\big(QM^{PWS}(\mathfrak{C}, \mathcal{C}_{gold})\big) &\overset{\text{Def. 76}}{=} EXP\Big(f_C\Big(\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big)\Big) \\ &\overset{\text{Eq. 8.53}}{=} f_C\Big(EXP\Big(\Theta_{i=1}^{k} f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big) \\ &\overset{\text{Eq. 8.51 and Eq. 8.52}}{=} f_C\Big(\Theta_{i=1}^{k} EXP\Big(f_A\big(\pi_{F_i}(\mathfrak{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)\Big), f_B(\mathcal{F}^{set}, \mathcal{C}_{gold})\Big) \end{aligned}$$

Note, the second equality results from Equation 8.53 as follows: Since $\mathcal{F}^{set}$ and $\mathcal{C}_{gold}$ are certain clusterings, the score resultant from f_B is a real number and f_C is either the sum operator or the multiplication operator, computing the function $f_C(f_A(x), f_B(y, z))$ corresponds to adding a constant real number to a random variable or to multiplying a constant real number to a random variable.

Proof of Theorem 38

Theorem 38 *Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering and let $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ be a range-disjoint factorization of $\mathfrak{C}$, the set $\{\widehat{\mathfrak{C}_F} \mid \mathfrak{C}_F \in \mathcal{F}\}$ is a range-disjoint factorization of the incomplete clustering $\widehat{\mathfrak{C}}$.*

Theorem 38 can be proved as follows:

Proof 39 *According to Definition 64, the set of incomplete clusterings* $\{\widehat{\mathfrak{C}_{F_1}}, \ldots, \widehat{\mathfrak{C}_{F_k}}\}$ *is a factorization of the incomplete clustering* $\widehat{\mathfrak{C}}$*, if the latter can be reconstructed from the first, i.e. if* $\widehat{\mathfrak{C}} = \bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$.

Let $\mathfrak{C}_{F_i} = (\Gamma_{F_i}, Pr_{F_i})$ *for each* $i \in \{1, \ldots, k\}$*. Assumption 1:* $\widehat{\mathfrak{C}}$ *contains a clustering* $\mathcal{C}$ *that does not belong to* $\bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$.
Due to $\mathcal{C} \notin \bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$
$\Rightarrow$ *there is a factor set* $F_j \in \mathcal{F}^{set}$ *so that* $\pi_{F_j}(\mathcal{C}) \notin \widehat{\mathfrak{C}_{F_j}}$.
$\Rightarrow$ *there is a clustering* $\mathcal{C}' \in \Gamma_{F_j}$ *so that* $Pr_{F_j}(\mathcal{C}') > Pr_{F_j}(\pi_{F_j}(\mathcal{C}))$.
$\Rightarrow$ *the clustering* $\mathcal{C}^* = \big(\bigboxtimes_{i=1, i\neq j}^{k} \pi_{F_i}(\mathcal{C})\big) \boxtimes \mathcal{C}'$ *has a greater probability than* $\mathcal{C}$ *because* $Pr(\mathcal{C}^*) = \prod_{i=1, i\neq j}^{k} Pr_{F_i}(\pi_{F_i}(\mathcal{C})) \times Pr_{F_j}(\mathcal{C}') > \prod_{i=1}^{k} Pr_{F_i}(\pi_{F_i}(\mathcal{C})) = Pr(\mathcal{C})$.
$\Rightarrow$ *the clustering* $\mathcal{C}$ *cannot belong to* $\widehat{\mathfrak{C}}$ *which is a contradiction to our initial assumption.*
Consequently, $\widehat{\mathfrak{C}}$ *cannot contain a clustering that does not belong to* $\bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$*, i.e.* $\widehat{\mathfrak{C}} \subseteq \bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$.

Assumption 2: $\bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$ *contains a clustering* $\mathcal{C}$ *that does not belong to* $\widehat{\mathfrak{C}}$.
Due to $\mathcal{C} \notin \widehat{\mathfrak{C}}$
$\Rightarrow$ *there is a clustering* $\mathcal{C}' \in \Gamma$ *so that* $Pr(\mathcal{C}') > Pr(\mathcal{C})$.
Since the probability of an integrated clustering results from multiplying the probabilities of its factors
$\Rightarrow \prod_{i=1}^{k} Pr_{F_i}(\pi_{F_i}(\mathcal{C}')) > \prod_{i=1}^{k} Pr_{F_i}(\pi_{F_i}(\mathcal{C}))$.
$\Rightarrow$ *there is at least one factor set* $F \in \mathcal{F}^{set}$ *so that* $Pr_F(\pi_F(\mathcal{C}')) > Pr_F(\pi_F(\mathcal{C}))$.
$\Rightarrow$ *the clustering* $\pi_F(\mathcal{C})$ *cannot belong to* $\widehat{\mathfrak{C}_F}$.
$\Rightarrow$ *the clustering* $\mathcal{C}$ *cannot belong to* $\bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$ *(Theorem 3) which is a contradiction to the initial assumption.*
Consequently, $\bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$ *cannot contain a clustering that does not belong to* $\widehat{\mathfrak{C}}$*, i.e.* $\widehat{\mathfrak{C}} \supseteq \bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$.

From $\widehat{\mathfrak{C}} \subseteq \bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$ *and* $\widehat{\mathfrak{C}} \supseteq \bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$ *follows that* $\widehat{\mathfrak{C}} = \bigboxtimes_{i=1}^{k} \widehat{\mathfrak{C}_{F_i}}$.

Proof of Theorem 39

Theorem 39 *Let* Γ *be an incomplete clustering and let* $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *be a range-disjoint factorization of* Γ*, the set* $\{\sigma_{maxCl}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ *is a range-disjoint factorization of the incomplete clustering* $\sigma_{maxCl}(\Gamma)$.

Theorem 39 can be proved as follows:

Proof 40 *According to Definition 64, the set of incomplete clusterings* $\{\sigma_{maxCl}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ *is a factorization of the incomplete clustering* $\sigma_{maxCl}(\Gamma)$*, if the latter can be reconstructed from the first, i.e. if* $\sigma_{maxCl}(\Gamma) = \bigboxtimes_{i=1}^{k} \big(\sigma_{maxCl}(\Gamma_{F_i})\big)$.

Assumption 1: $\sigma_{maxCl}(\Gamma)$ *contains a clustering* $\mathcal{C}$ *that does not belong to* $\bigboxtimes_{i=1}^{k} \big(\sigma_{maxCl}(\Gamma_{F_i})\big)$.
Because all factor sets are disjoint, the number of clusters of clustering $\mathcal{C}$ *can be computed as* $|\mathcal{C}| = \sum_{i=1}^{k} |\pi_{F_i}(\mathcal{C})|$.
Since the factors are mutually independent with respect to Γ *and* $\mathcal{C} \notin \bigboxtimes_{i=1}^{k} \big(\sigma_{maxCl}(\Gamma_{F_i})\big)$
$\Rightarrow$ *there must be a factor set* $F_j \in \mathcal{F}^{set}$ *so that there is a clustering* $\mathcal{C}' \in \Gamma_{F_j}$ *with* $|\mathcal{C}'| > |\pi_{F_j}(\mathcal{C})|$.

$\Rightarrow$ *the clustering* $\mathcal{C}^* \in \Gamma$ *with* $\mathcal{C}^* = \left(\boxtimes_{i=1,i\neq j}^{k} \pi_{F_i}(\mathcal{C})\right) \boxtimes \mathcal{C}'$ *has more clusters than* $\mathcal{C}$.
$\Rightarrow$ *the clustering* $\mathcal{C}$ *cannot belong to* $\sigma_{maxCl}(\Gamma)$ *which is a contradiction to the initial assumption.*
Consequently, $\sigma_{maxCl}(\Gamma)$ *cannot contain a clustering that does not belong to* $\boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$ *and hence* $\sigma_{maxCl}(\Gamma) \subseteq \boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$.

Assumption 2: $\boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$ *contains a clustering* $\mathcal{C}$ *that does not belong to* $\sigma_{maxCl}(\Gamma)$.
Due to $\mathcal{C} \notin \sigma_{maxCl}(\Gamma)$
$\Rightarrow$ *there is a clustering* $\mathcal{C}' \in \Gamma$ *with* $|\mathcal{C}'| > |\mathcal{C}|$.
Since all factor sets are disjoint
$\Rightarrow$ *there is a factor set* $F \in \mathcal{F}^{set}$ *so that* $|\pi_F(\mathcal{C}')| > |\pi_F(\mathcal{C})|$.
$\Rightarrow$ $\pi_F(\mathcal{C})$ *cannot belong to the set* $\sigma_{maxCl}(\Gamma_F)$.
$\Rightarrow$ $\mathcal{C}$ *cannot belong to* $\boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$ *which is a contradiction to the initial assumption.*
Consequently, $\boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$ *cannot contain a clustering that does not belong to* $\sigma_{maxCl}(\Gamma)$ *and hence* $\sigma_{maxCl}(\Gamma) \supseteq \boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$.

From $\sigma_{maxCl}(\Gamma) \subseteq \boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$ *and* $\sigma_{maxCl}(\Gamma) \supseteq \boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$ *follows that* $\sigma_{maxCl}(\Gamma) = \boxtimes_{i=1}^{k}\left(\sigma_{maxCl}(\Gamma_{F_i})\right)$.

Proof of Theorem 40

Theorem 40 *Let* Γ *be an incomplete clustering and let* $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *be a range-disjoint factorization of* Γ*, the set* $\{\sigma_{minCl}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ *is a range-disjoint factorization of the incomplete clustering* σ_{minCl}.

Theorem 40 can be proved as follows:

Proof 41 *Following Definition 64, the set of incomplete clusterings* $\{\sigma_{minCl}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ *is a factorization of the incomplete clustering* $\sigma_{minCl}(\Gamma)$*, if the latter can be reconstructed from the first, i.e. if* $\sigma_{minCl}(\Gamma) = \boxtimes_{i=1}^{k}\left(\sigma_{minCl}(\Gamma_{F_i})\right)$.

Assumption 1: $\sigma_{minCl}(\Gamma)$ *contains a clustering* $\mathcal{C}$ *that does not belong to* $\boxtimes_{i=1}^{k}\left(\sigma_{minCl}(\Gamma_{F_i})\right)$.
Since all factor sets are disjoint, the number of clusters of clustering $\mathcal{C}$ *can be computed as* $|\mathcal{C}| = \sum_{i=1}^{k} |\pi_{F_i}(\mathcal{C})|$.
Since the factors are mutually independent with respect to Γ *and* $\mathcal{C} \notin \boxtimes_{i=1}^{k}\left(\sigma_{minCl}(\Gamma_{F_i})\right)$
$\Rightarrow$ *there must be a factor set* $F_j \in \mathcal{F}^{set}$ *so that there is a clustering* $\mathcal{C}' \in \Gamma_{F_j}$ *with* $|\mathcal{C}'| < |\pi_{F_j}(\mathcal{C})|$.
$\Rightarrow$ *the clustering* $\mathcal{C}^* \in \Gamma$ *with* $\mathcal{C}^* = \left(\boxtimes_{i=1,i\neq j}^{k} \pi_{F_i}(\mathcal{C})\right) \boxtimes \mathcal{C}'$ *has less clusters than* $\mathcal{C}$.
$\Rightarrow$ *the clustering* $\mathcal{C}$ *cannot belong to* $\sigma_{minCl}(\Gamma)$ *which is a contradiction to the initial assumption.*
Consequently, $\sigma_{minCl}(\Gamma)$ *cannot contain a clustering that does not belong to* $\boxtimes_{i=1}^{k}\left(\sigma_{minCl}(\Gamma_{F_i})\right)$ *and hence* $\sigma_{minCl}(\Gamma) \subseteq \boxtimes_{i=1}^{k}\left(\sigma_{minCl}(\Gamma_{F_i})\right)$.

Assumption 2: $\boxtimes_{i=1}^{k}\left(\sigma_{minCl}(\Gamma_{F_i})\right)$ *contains the clustering* $\mathcal{C}$ *that does not belong to* $\sigma_{minCl}(\Gamma)$.
Due to $\mathcal{C} \notin \sigma_{minCl}(\Gamma)$

$\Rightarrow$ *there is a clustering* $\mathcal{C}' \in \Gamma$ *with* $|\mathcal{C}'| < |\mathcal{C}|$.
Since all factor sets are disjoint
$\Rightarrow$ *there is a factor set* $F \in \mathcal{F}^{set}$ *so that* $|\pi_F(\mathcal{C}')| < |\pi_F(\mathcal{C})|$.
$\Rightarrow$ $\pi_F(\mathcal{C})$ *cannot belong to the set* $\sigma_{minCl}(\Gamma_F)$.
$\Rightarrow$ $\mathcal{C}$ *cannot belong to* $\bigtimes_{i=1}^{k}\big(\sigma_{minCl}(\Gamma_{F_i})\big)$ *which is a contradiction to the initial assumption.*
Consequently, $\bigtimes_{i=1}^{k}\big(\sigma_{minCl}(\Gamma_{F_i})\big)$ *cannot contain a clustering that does not belong to* $\sigma_{minCl}(\Gamma)$ *and hence* $\sigma_{minCl}(\Gamma) \supseteq \bigtimes_{i=1}^{k}\big(\sigma_{minCl}(\Gamma_{F_i})\big)$.

From $\sigma_{minCl}(\Gamma) \subseteq \bigtimes_{i=1}^{k}\big(\sigma_{minCl}(\Gamma_{F_i})\big)$ *and* $\sigma_{minCl}(\Gamma) \supseteq \bigtimes_{i=1}^{k}\big(\sigma_{minCl}(\Gamma_{F_i})\big)$ *follows that* $\sigma_{minCl}(\Gamma) = \bigtimes_{i=1}^{k}\big(\sigma_{minCl}(\Gamma_{F_i})\big)$.

Proof of Theorem 41

Theorem 41 *Let* Γ *be an incomplete clustering and let* $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *be a range-disjoint factorization of* Γ *with the factor sets* $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$*, it holds:*

$$\begin{aligned} M_c(\Gamma) &= \bigcup_{i=1}^{k} M_c(\Gamma_{F_i}) \\ P(\Gamma) &= \bigcup_{i=1}^{k} P(\Gamma_{F_i}) \\ U_c(\Gamma) &= \Big(\bigcup_{i=1}^{k} U_c(\Gamma_{F_i})\Big) \cup U(\mathcal{F}^{set}) \end{aligned}$$

Theorem 41 can be proved as follows:

Proof 42 *We start with the set of certain* MATCHES. *An entity pair* $\{e_r, e_s\}$ *is a certain* MATCH *in* Γ*, if it is a* MATCH *in every* $\mathcal{C} \in \Gamma$. *Since entities are restricted to single factors and pairs whose entities belong to different factors cannot be a* MATCH *in any alternative of* Γ*, the pair* $\{e_r, e_s\}$ *is a* MATCH *in every* $\mathcal{C} \in \Gamma$*, if it is a* MATCH *in every alternative of its corresponding factor and hence is a certain* MATCH *in this factor. This can be formally proved as:*

$$M_c(\Gamma) = \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \forall \mathcal{C} \in \Gamma\colon \{e_r, e_s\} \in M(\mathcal{C})\}$$

due to clusters and thus MATCHES *are restricted to single factors, it follows:*

$$\begin{aligned} &= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists F \in \mathcal{F}^{set}\colon \forall \mathcal{C} \in \Gamma\colon \{e_r, e_s\} \in M(\pi_F(\mathcal{C}))\} \\ &= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists \Gamma_F \in \mathcal{F}\colon \forall \mathcal{C} \in \Gamma_F\colon \{e_r, e_s\} \in M(\mathcal{C})\} \\ &= \bigcup_{i=1}^{k} \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma_{F_i}), \forall \mathcal{C} \in \Gamma_{F_i}\colon \{e_r, e_s\} \in M(\mathcal{C})\} \\ &= \bigcup_{i=1}^{k} M_c(\Gamma_{F_i}) \end{aligned}$$

We proceed with the set of POSSIBLE MATCHES. *An entity pair* $\{e_r, e_s\}$ *is a* POSSIBLE MATCH *in* Γ*, if it is an* UNMATCH *in some* $\mathcal{C} \in \Gamma$ *and is a* MATCH *in some* $\mathcal{C}' \in \Gamma$. *Since* MATCHES *are restricted to single factors, an entity pair* $\{e_r, e_s\}$ *can only be an* UNMATCH *in some* $\mathcal{C} \in \Gamma$ *and a* MATCH *in some*

$\mathcal{C}' \in \Gamma$, if it is a POSSIBLE MATCH *in its corresponding factor. This can be formally proved as:*

$$P(\Gamma) = \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists \mathcal{C}_p, \mathcal{C}_q \in \Gamma \colon \{e_r, e_s\} \in M(\mathcal{C}_p) \wedge \{e_r, e_s\} \in U(\mathcal{C}_q)\}$$

due to MATCHES *are restricted to single factors, it follows:*

$$\begin{aligned}
&= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists F \in \mathcal{F}^{set} \colon \exists \mathcal{C}_p, \mathcal{C}_q \in \Gamma \colon \\
&\qquad\qquad \{e_r, e_s\} \in M(\pi_F(\mathcal{C}_p)) \wedge \{e_r, e_s\} \in U(\pi_F(\mathcal{C}_q))\} \\
&= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists \Gamma_F \in \mathcal{F} \colon \exists \mathcal{C}_p, \mathcal{C}_q \in \Gamma_F \colon \\
&\qquad\qquad \{e_r, e_s\} \in M(\mathcal{C}_p) \wedge \{e_r, e_s\} \in U(\mathcal{C}_q)\} \\
&= \bigcup_{i=1}^{k} \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma_{F_i}), \exists \mathcal{C}_p, \mathcal{C}_q \in \Gamma_{F_i} \colon \{e_r, e_s\} \in M(\mathcal{C}_p) \wedge \{e_r, e_s\} \in U(\mathcal{C}_q)\} \\
&= \bigcup_{i=1}^{k} P(\Gamma_{F_i})
\end{aligned}$$

Finally, we consider the set of UNMATCHES*. An entity pair $\{e_r, e_s\}$ is a certain* UNMATCH *in Γ, if it is an* UNMATCH *in every $\mathcal{C} \in \Gamma$. Since entities are restricted to single factors, the pair $\{e_r, e_s\}$ is an* UNMATCH *in every $\mathcal{C} \in \Gamma$, if e_r and e_s belong to different factor sets or if it is an* UNMATCH *in every alternative of the entities' corresponding factor and hence is a certain* UNMATCH *in this factor. This can be formally proved as:*

$$U_c(\Gamma) = \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \forall \mathcal{C} \in \Gamma \colon \{e_r, e_s\} \in U(\mathcal{C})\}$$

e_r and e_s either both belong to the same factor set or they belong to different factor sets.
Since in the second case they are automatically an UNMATCH *in every alternative $\mathcal{C} \in \Gamma$, it follows:*

$$\begin{aligned}
&= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), (\exists F \in \mathcal{F}^{set} \colon \{e_r, e_s\} \in F, \forall \mathcal{C} \in \Gamma \colon \{e_r, e_s\} \in U(\pi_F(\mathcal{C}))) \\
&\quad \vee (\exists F_i, F_j \in \mathcal{F}^{set} \colon e_r \in F_i, e_s \in F_j, F_i \neq F_j)\} \\
&= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), (\exists \Gamma_F \in \mathcal{F} \colon \{e_r, e_s\} \in F, \forall \mathcal{C} \in \Gamma_F \colon \{e_r, e_s\} \in U(\mathcal{C})) \\
&\quad \vee (\exists F_i, F_j \in \mathcal{F}^{set} \colon e_r \in F_i, e_s \in F_j, F_i \neq F_j)\} \\
&= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists \Gamma_F \in \mathcal{F} \colon \forall \mathcal{C} \in \Gamma_F \colon \{e_r, e_s\} \in U(\mathcal{C})\} \\
&\quad \cup \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists F_i, F_j \in \mathcal{F}^{set} \colon e_r \in F_i, e_s \in F_j, F_i \neq F_j\}
\end{aligned}$$

due to all entities that belong to different factor sets are UNMATCHES *of the clustering $\mathcal{F}^{set}$, it follows:*

$$\begin{aligned}
&= \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma), \exists \Gamma_F \in \mathcal{F} \colon \forall \mathcal{C} \in \Gamma_F \colon \{e_r, e_s\} \in U(\mathcal{C})\} \cup U(\mathcal{F}^{set}) \\
&= \bigcup_{i=1}^{k} \{\{e_r, e_s\} \mid \{e_r, e_s\} \in \textit{Pairs}(\Gamma_{F_i}), \forall \mathcal{C} \in \Gamma_{F_i} \colon \{e_r, e_s\} \in U(\mathcal{C})\} \cup U(\mathcal{F}^{set}) \\
&= \bigcup_{i=1}^{k} U_c(\Gamma_{F_i}) \cup U(\mathcal{F}^{set})
\end{aligned}$$

Proof of Theorem 42

Theorem 42 *Let Γ be an incomplete clustering, let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$, and let $\mathcal{C}_{gold}$ be the corresponding gold standard, it holds:*

$$\begin{aligned}
|p\text{TP}^{CWS}(\Gamma, \mathcal{C}_{gold})| &= \sum\nolimits_{i=1}^{k} |p\text{TP}^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))| \\
|p\text{FP}^{CWS}(\Gamma, \mathcal{C}_{gold})| &= \sum\nolimits_{i=1}^{k} |p\text{FP}^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))| \\
|p\text{FN}^{CWS}(\Gamma, \mathcal{C}_{gold})| &= \Big(\sum\nolimits_{i=1}^{k} |p\text{FN}^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))|\Big) + |p\text{FN}(\mathcal{F}^{set}, \mathcal{C}_{gold})|
\end{aligned}$$

Theorem 42 can be proved as follows:

Proof 43 *We start with the set of Certain Pair True Positives. An entity pair $\{e_r, e_s\}$ is a Certain Pair True Positive in Γ, if it is a Pair True Positive in every $\mathcal{C} \in \Gamma$. Since entities are restricted to single factors and all entity pairs whose entities belong to different factors are certain* UNMATCHES*, the pair $\{e_r, e_s\}$ is a Pair True Positive in every $\mathcal{C} \in \Gamma$ if it is a Pair True Positive in every alternative of its corresponding factor and hence if it is a Certain Pair True Positive in this factor. This can be formally proved as:*

$$\begin{aligned}
pTP^{CWS}(\Gamma, \mathcal{C}_{gold}) &= M_c(\Gamma) \cap M(\mathcal{C}_{gold}) \\
&= \Big(\bigcup\nolimits_{i=1}^{k} M_c(\Gamma_{F_i})\Big) \cap M(\mathcal{C}_{gold}) \\
&= \bigcup\nolimits_{i=1}^{k} \big(M_c(\Gamma_{F_i}) \cap M(\mathcal{C}_{gold})\big)
\end{aligned}$$

due to Γ_{F_i} only shares entities with $\pi_{F_i}(\mathcal{C}_{gold})$, it follows:

$$\begin{aligned}
&= \bigcup\nolimits_{i=1}^{k} \big(M_c(\Gamma_{F_i}) \cap M(\pi_{F_i}(\mathcal{C}_{gold}))\big) \\
&= \bigcup\nolimits_{i=1}^{k} pTP^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))
\end{aligned}$$

Consequently, the Number of Certain Pair True Positives can be computed as:

$$|pTP^{CWS}(\Gamma, \mathcal{C}_{gold})| = |\bigcup\nolimits_{i=1}^{k} pTP^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))|$$

due to all factor sets are disjoint, it follows:

$$= \sum\nolimits_{i=1}^{k} |pTP^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))|$$

We proceed with the set of Certain Pair False Positives. An entity pair $\{e_r, e_s\}$ is a Certain Pair False Positive in Γ, if it is a Pair False Positive in every $\mathcal{C} \in \Gamma$. Since MATCHES *are restricted to single factors, the same holds for Pair False Positives. Consequently, an entity pair $\{e_r, e_s\}$ is a Certain Pair False Positive in Γ, if it is a Certain Pair False Positive in its corresponding factor. This can be formally*

proved as:

$$\begin{aligned} pFP^{CWS}(\Gamma, \mathcal{C}_{gold}) &= M_c(\Gamma) \cap U(\mathcal{C}_{gold}) \\ &= \left(\bigcup_{i=1}^{k} M_c(\Gamma_{F_i})\right) \cap U(\mathcal{C}_{gold}) \\ &= \bigcup_{i=1}^{k} \left(M_c(\Gamma_{F_i}) \cap U(\mathcal{C}_{gold})\right) \end{aligned}$$

due to Γ_{F_i} only shares entities with $\pi_{F_i}(\mathcal{C}_{gold})$, it follows:

$$\begin{aligned} &= \bigcup_{i=1}^{k} \left(M_c(\Gamma_{F_i}) \cap U(\pi_{F_i}(\mathcal{C}_{gold}))\right) \\ &= \bigcup_{i=1}^{k} pFP^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold})) \end{aligned}$$

Consequently, the Number of Certain Pair False Positives can be computed as:

$$\begin{aligned} |pFP^{CWS}(\Gamma, \mathcal{C}_{gold})| &= |\bigcup_{i=1}^{k} pFP^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))| \end{aligned}$$

due to all factor sets are disjoint, it follows:

$$\begin{aligned} &= \sum_{i=1}^{k} |pFP^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))| \end{aligned}$$

Finally, we consider the set of Certain Pair False Negatives. An entity pair $\{e_r, e_s\}$ is a Certain Pair False Negative in Γ, if it is a Pair False Negative in every $\mathcal{C} \in \Gamma$. Since the entities of UNMATCHES *in Γ can belong to different factor sets, we have to distinguish between intra-factor Pair False Negatives and inter-factor Pair False Negatives. Since all entity pairs whose entities belong to different factor sets are certain* UNMATCHES*, each inter-factor Pair False Negatives is automatically certain. In contrast, an intra-factor Pair False Negative is only certain in Γ if it is a Certain Pair False Negative in its corresponding factor. This can be formally proved as:*

$$\begin{aligned} pFN^{CWS}(\Gamma, \mathcal{C}_{gold}) &= U_c(\Gamma) \cap M(\mathcal{C}_{gold}) \\ &= \left(\bigcup_{i=1}^{k} U_c(\Gamma_{F_i}) \cup U(\mathcal{F}^{set})\right) \cap M(\mathcal{C}_{gold}) \\ &= \bigcup_{i=1}^{k} \left(U_c(\Gamma_{F_i}) \cap M(\mathcal{C}_{gold})\right) \cup \left(U(\mathcal{F}^{set}) \cap M(\mathcal{C}_{gold})\right) \end{aligned}$$

due to Γ_{F_i} only shares entities with $\pi_{F_i}(\mathcal{C}_{gold})$, it follows:

$$\begin{aligned} &= \bigcup_{i=1}^{k} \left(U_c(\Gamma_{F_i}) \cap M(\pi_{F_i}(\mathcal{C}_{gold}))\right) \cup \left(U(\mathcal{F}^{set}) \cap M(\mathcal{C}_{gold})\right) \\ &= \bigcup_{i=1}^{k} \left(pFN^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))\right) \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold}) \end{aligned}$$

Consequently, the Number of Certain Pair False Negative can be computed as:

$$|pFN^{CWS}(\Gamma, \mathcal{C}_{gold})| = |\bigcup_{i=1}^{k} \left(pFN^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))\right) \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})|$$

due to $U(\mathcal{F}^{set})$ only contains pairs with entities that belong to different factors, it follows:

$$= |\bigcup_{i=1}^{k} \left(pFN^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))\right)| + |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})|$$

due to all factor sets are disjoint, it follows:

$$= \sum_{i=1}^{k} \left(|pFN^{CWS}(\Gamma_{F_i}, \pi_{F_i}(\mathcal{C}_{gold}))|\right) + |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})|$$

Proof of Theorem 43

Theorem 43 *Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering, let N be the normalization factor of $\mathfrak{C}$, and let $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ be a factorization of $\mathfrak{C}$ where $\mathfrak{C}_{F_i} = (\Gamma_{F_i}, Pr_{F_i})$ and N_i is the normalization factor of $\mathfrak{C}_{F_i}$, the Inverse Normalized Entropy of $\mathfrak{C}$ can be computed as:*

$$invH_{norm}(\mathfrak{C}) = 1 - \sum_{i=1}^{k} \log_N(N_i) \times H_{norm}(\mathfrak{C}_{F_i})$$

Theorem 43 can be proved as follows:

Proof 44

$$\begin{aligned} invH_{norm}(\mathfrak{C}) &= 1 - H_{norm}(\mathfrak{C}) = 1 - \frac{H(\mathfrak{C})}{H_{max(\mathfrak{C})}} = 1 - \frac{\sum_{i=1}^{k} H(\mathfrak{C}_{F_i})}{H_{max(\mathfrak{C})}} \\ &= 1 - \sum_{i=1}^{k} \frac{H(\mathfrak{C}_{F_i})}{H_{max(\mathfrak{C})}} = 1 - \sum_{i=1}^{k} \frac{H_{max(\mathfrak{C}_{F_i})}}{H_{max(\mathfrak{C})}} \times \frac{H(\mathfrak{C}_{F_i})}{H_{max(\mathfrak{C}_{F_i})}} \\ &= 1 - \sum_{i=1}^{k} \frac{\log_2(N_i)}{\log_2(N)} \times H_{norm}(\mathfrak{C}_{F_i}) \\ &= 1 - \sum_{i=1}^{k} \log_N(N_i) \times H_{norm}(\mathfrak{C}_{F_i}) \end{aligned}$$

If we use the normalization factors $N = \prod_{i=1}^{k} B_{n_i}$ *and* $N_i = B_{n_i}, i \in \{1, \ldots, k\}$*, this equation can be further transformed to:*

$$\begin{aligned}
\mathit{invH}_{\mathit{norm}}(\mathfrak{C}) &= 1 - \sum_{i=1}^{k} \log_N(N_i) \times H_{\mathit{norm}}(\mathfrak{C}_{F_i}) \\
&= 1 - \sum_{i=1}^{k} \log_N(N_i) \times (1 - \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})) \\
&= 1 - \sum_{i=1}^{k} \big(\log_N(N_i) - \log_N(N_i) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})) \big) \\
&= 1 - \left(\sum_{i=1}^{k} \log_N(N_i) - \sum_{i=1}^{k} \log_N(N_i) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})) \right) \\
&= 1 - \left(\log_N(\prod_{i=1}^{k} N_i) - \sum_{i=1}^{k} \log_N(N_i) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})) \right)
\end{aligned}$$

due to it holds that $\prod_{i=1}^{k} N_i = N$ *(each cluster-disjoint clustering of range* $rng(\mathfrak{C})$ *needs to be reconstructed by a combination of one cluster-disjoint clustering per range* $rng(\mathfrak{C}_{F_i}), i \in \{1, \ldots, k\}$*)*

$$\begin{aligned}
&= 1 - \left(\log_N(N) - \sum_{i=1}^{k} \log_N(N_i) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})) \right) \\
&= 1 - \left(1 - \sum_{i=1}^{k} \log_N(N_i) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})) \right) \\
&= 1 - 1 + \sum_{i=1}^{k} \log_N(N_i) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i})) \\
&= \sum_{i=1}^{k} \log_N(N_i) \times \mathit{invH}_{\mathit{norm}}(\mathfrak{C}_{F_i}))
\end{aligned}$$

Proof of Theorem 44

Theorem 44 *Let* $\mathfrak{C}'$ *and* $\mathfrak{C}^*$ *be two independent but data equivalent probabilistic clusterings of* $n = |rng(\mathfrak{C}')| = |rng(\mathfrak{C}^*)|$ *entities and let* $\mathcal{F} = \{\mathfrak{C}_{F_1}, \ldots, \mathfrak{C}_{F_k}\}$ *be a range-disjoint factorization of both with* $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$*. The expected Rand Index* $EXP(RI(\mathfrak{C}', \mathfrak{C}^*))$ *can be computed as:*

$$EXP(RI(\mathfrak{C}', \mathfrak{C}^*)) = \frac{1}{\binom{n}{2}} \times \big(\sum_{i=1}^{k} (EXP(|M(\mathfrak{C}'_{F_i}) \cap M(\mathfrak{C}^*_{F_i})|) + EXP(|U(\mathfrak{C}'_{F_i}) \cap U(\mathfrak{C}^*_{F_i})|)) + |U(\mathcal{F}^{set})| \big)$$

Theorem 44 can be proved as follows:

Proof 45

$$
\begin{aligned}
&\mathit{EXP}(\mathit{RI}(\mathfrak{C}', \mathfrak{C}^*)) \\
&\quad = \mathit{EXP}\Big(\frac{1}{\binom{n}{2}} \times \big(\sum_{i=1}^{k}(|M(\mathfrak{C}'_{F_i}) \cap M(\mathfrak{C}^*_{F_i})| + |U(\mathfrak{C}'_{F_i}) \cap U(\mathfrak{C}^*_{F_i})|) + |U(\mathcal{F}^{set})|\big)\Big) \\
&\quad \textit{because } \mathit{EXP}(c \times X) = c \times \mathit{EXP}(X) \textit{ for } c \in \mathbb{R} \textit{ and a random variable } X \\
&\quad = \frac{1}{\binom{n}{2}} \times \mathit{EXP}\Big(\big(\sum_{i=1}^{k}(|M(\mathfrak{C}'_{F_i}) \cap M(\mathfrak{C}^*_{F_i})| + |U(\mathfrak{C}'_{F_i}) \cap U(\mathfrak{C}^*_{F_i})|) + |U(\mathcal{F}^{set})|\big)\Big) \\
&\quad \textit{because } \mathit{EXP}(X + c) = \mathit{EXP}(X) + c \textit{ for } c \in \mathbb{R} \textit{ and a random variable } X \\
&\quad = \frac{1}{\binom{n}{2}} \times \Big(\mathit{EXP}\big(\sum_{i=1}^{k}(|M(\mathfrak{C}'_{F_i}) \cap M(\mathfrak{C}^*_{F_i})| + |U(\mathfrak{C}'_{F_i}) \cap U(\mathfrak{C}^*_{F_i})|)\big) + |U(\mathcal{F}^{set})|\Big) \\
&\quad \textit{because } \mathit{EXP}\Big(\sum_{i=1}^{n} X_i\Big) = \sum_{i=1}^{n} \mathit{EXP}(X_i) \textit{ for a set of random variables } \{X_1, \ldots, X_n\} \\
&\quad = \frac{1}{\binom{n}{2}} \times \Big(\sum_{i=1}^{k} \mathit{EXP}\big(|M(\mathfrak{C}'_{F_i}) \cap M(\mathfrak{C}^*_{F_i})| + |U(\mathfrak{C}'_{F_i}) \cap U(\mathfrak{C}^*_{F_i})|\big) + |U(\mathcal{F}^{set})|\Big) \\
&\quad \textit{because } \mathit{EXP}\Big(\sum_{i=1}^{n} X_i\Big) = \sum_{i=1}^{n} \mathit{EXP}(X_i) \textit{ for a set of random variables } \{X_1, \ldots, X_n\} \\
&\quad = \frac{1}{\binom{n}{2}} \times \Big(\big(\sum_{i=1}^{k} \mathit{EXP}(|M(\mathfrak{C}'_{F_i}) \cap M(\mathfrak{C}^*_{F_i})|) + \mathit{EXP}(|U(\mathfrak{C}'_{F_i}) \cap U(\mathfrak{C}^*_{F_i})|)\big) + |U(\mathcal{F}^{set})|\Big)
\end{aligned}
$$

Note, the first equality is a consequence of Theorem 59.

Proof of Theorem 45

Theorem 45 *Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering of $n = |rng(\mathfrak{C})|$ entities and let $\mathcal{F}$ be a range-disjoint factorization of $\mathfrak{C}$ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Average Decision Certainty of $\mathfrak{C}$ can be computed as:*

$$
\mathit{AvgDecC}(\mathfrak{C}) = \frac{1}{\binom{n}{2}} \times \left(\sum_{i=1}^{k} \sum_{\{e_r, e_s\} \in \mathit{Pairs}(\mathfrak{C}_{F_i})} \mathit{DecC}(\mathfrak{C}_{F_i}, \{e_r, e_s\}) - \sum_{C \in \mathcal{F}^{set}} \binom{|C|}{2}\right) + 1
$$

Theorem 45 can be proved as follows:

Proof 46 *All entity pairs whose entities belong to different factor sets are certain* UNMATCHES. *Thus, the certainty of the decision on an entity pair $\{e_r, e_s\}$ can be defined as:*

$$
\mathit{DecC}(\mathfrak{C}, \{e_r, e_s\}) = \begin{cases} 1, & \textit{if } e_r \textit{ and } e_s \textit{ belong to different factor sets,} \\ \mathit{DecC}(\mathfrak{C}_{F_i}, \{e_r, e_s\}), & \textit{if } e_r \textit{ and } e_s \textit{ belong to the same factor set } F_i. \end{cases}
$$

The accumulative certainty of all pairs whose entities belong to the same factor set (intra factor pairs) can be computed as:

$$
\mathit{IntraFactorPairCertainty} = \sum_{i=1}^{k} \sum_{\{e_r, e_s\} \in \mathit{Pairs}(\mathfrak{C}_{F_i})} \mathit{DecC}(\mathfrak{C}_{F_i}, \{e_r, e_s\})
$$

The accumulative certainty of all pairs whose entities belong to different factor sets (inter factor pairs) is equal to the number of these pairs which in turn is equal to the number of all entity pairs minus the intra factor pairs. Since each factor set corresponds to a cluster in the clustering $\mathcal{F}^{set}$ and the number of pairs in a cluster C is defined as $\binom{|C|}{2}$, the accumulative certainty of inter factor pairs can be computed as:

$$\mathit{InterFactorPairCertainty} = \binom{n}{2} - \sum_{i=1}^{k} \binom{|F_i|}{2} = \binom{n}{2} - \sum_{C \in \mathcal{F}^{set}} \binom{|C|}{2}$$

Consequently, we can use the original definition from Equation 8.47 and replace the sum over the certainty of all entity pairs by the sum of the accumulative certainty of all intra factor pairs and the accumulative certainty of all inter factor pairs:

$$\begin{aligned}
\mathit{AvgDecC}(\mathfrak{C}) &= \frac{1}{\binom{n}{2}} \times \sum_{\{e_r,e_s\} \in \mathit{Pairs}(\mathfrak{C})} \mathit{DecC}(\mathfrak{C}, \{e_r, e_s\}) \\
&= \frac{1}{\binom{n}{2}} \times \left(\sum_{i=1}^{k} \sum_{\{e_r,e_s\} \in \mathit{Pairs}(\mathfrak{C}_{F_i})} \mathit{DecC}(\mathfrak{C}_{F_i}, \{e_r, e_s\}) + \binom{n}{2} - \sum_{C \in \mathcal{F}^{set}} \binom{|C|}{2} \right) \\
&= \frac{1}{\binom{n}{2}} \times \left(\sum_{i=1}^{k} \sum_{\{e_r,e_s\} \in \mathit{Pairs}(\mathfrak{C}_{F_i})} \mathit{DecC}(\mathfrak{C}_{F_i}, \{e_r, e_s\}) - \sum_{C \in \mathcal{F}^{set}} \binom{|C|}{2} \right) + 1
\end{aligned}$$

A.6. Proofs: Decomposability of Quality Measures

In this section, we prove the assertions that are listed in Table 8.10. For this purpose, we formulate some theorems and prove them instantly.

Set of MATCHES The set of MATCHES is not a quality measure in the traditional sense of quality. However, if no gold standard is available the set of MATCHES (or the number of MATCHES respectively) can be used to compare several duplicate detection process that have been performed on the same input database. Whereas a detection result with a large number of MATCHES indicates a detection process that tends to produce many false positives, a detection result with only a few number of MATCHES indicates a detection process that tends to produce many false negatives. Of course, since the number of MATCHES depends on the considered input database, it is only a useful quality measure if several detection results can be compared.

Theorem 46 *Let $\mathcal{C}$ be a crisp clustering and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The set of* MATCHES *is a $\bigcup$-decomposable measure because it can be computed as:*

$$M(\mathcal{C}) = \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}))$$

and hence by setting $\Theta = \bigcup$ and $f_A(x, y) = M(x)$.

Theorem 46 can be proved as follows:

Proof 47

$$\begin{aligned}
M(\mathcal{C}) &= M(\bigcup_{i=1}^{k} \pi_{F_i}(\mathcal{C}))) \\
\textit{because all factor sets are disjoint} &= \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}))
\end{aligned}$$

Set of Pair True Positives A Pair Positive (true or false) is always restricted to a single cluster and therefore is always restricted to a single factor. For that reason, the set of Pair True Positives is a $\cup$-decomposable quality measure.

Theorem 47 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The set of Pair True Positives is a $\cup$-decomposable quality measure because it can be computed as:*

$$pTP(\mathcal{C}, \mathcal{C}_{gold}) = \bigcup_{i=1}^{k} pTP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))$$

and hence by setting $\Theta = \bigcup$ and $f_A(x, y) = pTP(x, y)$.

Theorem 47 can be proved as follows:

Proof 48

$$\begin{aligned}
pTP(\mathcal{C}, \mathcal{C}_{gold}) &= M(\mathcal{C}) \cap M(\mathcal{C}_{gold}) \\
&\overset{\text{Theo. 46}}{=} \Big(\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}))\Big) \cap M(\mathcal{C}_{gold}) \\
&= \bigcup_{i=1}^{k} \Big(M(\pi_{F_i}(\mathcal{C})) \cap M(\mathcal{C}_{gold})\Big) \\
&= \bigcup_{i=1}^{k} \Big(M(\pi_{F_i}(\mathcal{C})) \cap M(\pi_{F_i}(\mathcal{C}_{gold}))\Big) \\
&= \bigcup_{i=1}^{k} pTP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))
\end{aligned}$$

Set of Pair False Positives A Pair Positive (true or false) is always restricted to a single cluster and therefore is always restricted to a single factor. For that reason, the set of Pair False Positives is a $\cup$-decomposable quality measure.

Theorem 48 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The set of Pair False Positives is a $\cup$-decomposable quality measure because it can be computed as:*

$$pFP(\mathcal{C}, \mathcal{C}_{gold}) = \bigcup_{i=1}^{k} pFP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))$$

and hence by setting $\Theta = \bigcup$ and $f_A(x, y) = pFP(x, y)$.

Theorem 48 can be proved as follows:

Proof 49

$$\begin{aligned}
pFP(\mathcal{C}, \mathcal{C}_{gold}) &= M(\mathcal{C}) - M(\mathcal{C}_{gold}) \\
&\overset{\text{Theo. 46}}{=} \Big(\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}))\Big) - M(\mathcal{C}_{gold}) \\
&= \bigcup_{i=1}^{k} \Big(M(\pi_{F_i}(\mathcal{C})) - M(\mathcal{C}_{gold})\Big) \\
&= \bigcup_{i=1}^{k} \Big(M(\pi_{F_i}(\mathcal{C})) - M(\pi_{F_i}(\mathcal{C}_{gold}))\Big) \\
&= \bigcup_{i=1}^{k} pFP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))
\end{aligned}$$

Set of Pair False Negatives We do not know a ∪-decomposable function to compute the set of Pair False Negatives and because Pair False Negatives are not restricted to single factors it can be doubt that such a function exists. Nevertheless, as illustrated in Section 8.8.5.1 it is quasi-∪-decomposable.

Theorem 49 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The set of Pair False Negatives is a quasi-∪-decomposable quality measure because it can be computed as:*

$$pFN(\mathcal{C}, \mathcal{C}_{gold}) = \big(\bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))\big) \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})$$

and hence by setting $\Theta = \bigcup$, $f_A(x, y) = pFN(x, y)$, $f_B(x, y) = pFN(x, y)$, and $f_C(x, y) = x \cup y$.

Theorem 49 can be proved as follows:

Proof 50

$$\begin{aligned}
pFN(\mathcal{C}, \mathcal{C}_{gold}) &= M(\mathcal{C}_{gold}) - M(\mathcal{C}) \\
&\overset{\text{Theo. 46}}{=} M(\mathcal{C}_{gold}) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C})) \\
&= \Big(\big(M(\mathcal{C}_{gold}) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold}))\big) \cup \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold}))\Big) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C})) \\
&\quad \textit{due to } M(\mathcal{C}_{gold}) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold})) \textit{ is disjoint with } \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C})) \\
&= \Big(\big(M(\mathcal{C}_{gold}) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold}))\big)\Big) \\
&\quad \cup \Big(\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold})) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}))\Big) \\
&\quad \textit{because all factor sets are disjoint} \\
&= \Big(\big(M(\mathcal{C}_{gold}) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold}))\big)\Big) \cup \Big(\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold})) - M(\pi_{F_i}(\mathcal{C}))\Big) \\
&= \Big(\big(M(\mathcal{C}_{gold}) - \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_{gold}))\big)\Big) \cup \bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})) \\
&\quad \textit{because } M(rng(F_i)) - M(\pi_{F_i}(\mathcal{C}_{gold})) \textit{ is disjoint with } M(\mathcal{C}_{gold}) \textit{ for every } i \in \{1, \ldots, k\} \\
&= \Big(\big(M(\mathcal{C}_{gold}) - \bigcup_{i=1}^{k} M(rng(F_i))\big)\Big) \cup \bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})) \\
&= \Big(\big(M(\mathcal{C}_{gold}) - M(\bigcup_{i=1}^{k} rng(F_i))\big)\Big) \cup \bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})) \\
&= \big(M(\mathcal{C}_{gold}) - M(\mathcal{F}^{set})\big) \cup \bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})) \\
&= pFN(\mathcal{F}^{set}, \mathcal{C}_{gold}) \cup \bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))
\end{aligned}$$

Set of Pair False Decisions The set of Pair False Decisions is the union of the set of Pair False Positives and the set of Pair False Negatives. Since we do not know a ∪-decomposable function for the latter, we do not know a ∪-decomposable function for the set of Pair False Decisions as well. Nevertheless, because the set of Pair False Positives is a ∪-decomposable quality measure and the set of Pair False Negatives is a quasi-∪-decomposable quality measure the set of Pair False Decisions is quasi-∪-decomposable.

Theorem 50 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The set of Pair False Decisions is a quasi-$\cup$-decomposable quality measure because it can be computed as:*

$$pFDec(\mathcal{C}, \mathcal{C}_{gold}) = \bigcup_{i=1}^{k} \left(pFDec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))\right) \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})$$

and hence by setting $\Theta = \bigcup$, $f_A(x, y) = pFDec(x, y)$, $f_B(x, y) = pFN(x, y)$, and $f_C(x, y) = x \cup y$.

Theorem 50 can be proved as follows:

Proof 51

$$\begin{aligned}
pFDec(\mathcal{C}, \mathcal{C}_{gold}) &= pFP(\mathcal{C}, \mathcal{C}_{gold}) \cup pFN(\mathcal{C}, \mathcal{C}_{gold}) \\
&= \bigcup_{i=1}^{k} \left(pFP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))\right) \cup \bigcup_{i=1}^{k} \left(pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))\right) \\
&\quad \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold}) \\
&= \bigcup_{i=1}^{k} \left(pFP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})) \cup pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))\right) \\
&\quad \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold}) \\
&= \bigcup_{i=1}^{k} \left(pFDec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))\right) \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})
\end{aligned}$$

Number of Pair True Positives The set of Pair True Positives is $\cup$-decomposable measure. For that reason, the Number of Pair True Positives is Σ-decomposable.

Theorem 51 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Number of Pair True Positives is a Σ-decomposable quality measure because it can be computed as:*

$$|pTP(\mathcal{C}, \mathcal{C}_{gold})| = \sum_{i=1}^{k} |pTP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))|$$

and hence by setting $\Theta = \Sigma$ and $f_A(x, y) = |pTP(x, y)|$.

Theorem 51 can be proved as follows:

Proof 52

$$\begin{aligned}
|pTP(\mathcal{C}, \mathcal{C}_{gold})| &= |\bigcup_{i=1}^{k} pTP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))| \\
&\quad \textit{because all factor sets are disjoint} \\
&= \sum_{i=1}^{k} |pTP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))|
\end{aligned}$$

Number of Pair False Positives The set of Pair False Positives is $\cup$-decomposable measure. For that reason, the Number of Pair False Positives is Σ-decomposable.

Theorem 52 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Number of Pair False Positives is a Σ-decomposable quality measure because it can be computed as:*

$$|pFP(\mathcal{C}, \mathcal{C}_{gold})| = \sum_{i=1}^{k} |pFP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))|$$

and hence by setting $\Theta = \Sigma$ and $f_A(x, y) = |pFP(x, y)|$.

Theorem 52 can be proved as follows:

Proof 53

$$\begin{aligned} |pFP(\mathcal{C}, \mathcal{C}_{gold})| &= |\bigcup_{i=1}^{k} pFP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))| \\ &\textit{because all factor sets are disjoint} \\ &= \sum_{i=1}^{k} |pFP(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))| \end{aligned}$$

Number of Pair False Negatives We do not know a Σ-decomposable function to compute the Number of Pair False Negatives, but because the set of Pair False Negatives is quasi-$\cup$-decomposable and because all factor sets are disjoint the Number of Pair False Negatives is a quasi-Σ-decomposable quality measure.

Theorem 53 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Number of Pair False Negatives is a quasi-Σ-decomposable quality measure because it can be computed as:*

$$|pFN(\mathcal{C}, \mathcal{C}_{gold})| = \big(\sum_{i=1}^{k} |pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))|\big) + |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})|$$

and hence by setting $\Theta = \Sigma$, $f_A(x, y) = |pFN(x, y)|$, $f_B(x, y) = |pFN(x, y)|$, and $f_C(x, y) = x + y$.

Theorem 53 can be proved as follows:

Proof 54

$$\begin{aligned} |pFN(\mathcal{C}, \mathcal{C}_{gold})| &= |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold}) \cup \bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))| \\ &\textit{because the set of Pair False Negatives of } \mathcal{F}^{set} \textit{ is disjoint to the set of Pair False Negatives of each factor} \\ &= |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})| + |\bigcup_{i=1}^{k} pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))| \\ &\textit{because all factor sets are disjoint} \\ &= |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})| + \sum_{i=1}^{k} |pFN(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))| \end{aligned}$$

Number of Pair False Decisions The Number of Pair False Decisions is the sum of the Number of Pair False Positives and the Number of Pair False Negatives. Since we do not know a Σ-decomposable function for the latter, we do not know a Σ-decomposable function for the Number of Pair False Decisions as well. Nevertheless, because the Number of Pair False Positives is a Σ-decomposable quality measure and the Number of Pair False Negatives is a quasi-Σ-decomposable quality measure the Number of Pair False Decisions is a quasi-Σ-decomposable quality measure.

Theorem 54 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Number of Pair False Decisions is a quasi-Σ-decomposable quality measure because it can be computed as:*

$$|pFDec(\mathcal{C}, \mathcal{C}_{gold})| = \Big(\sum_{i=1}^{k} |pFDec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))|\Big) + |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})|$$

and hence by setting $\Theta = \Sigma$, $f_A(x, y) = |pFDec(x, y)|$, $f_B(x, y) = |pFN(x, y)|$, and $f_C(x, y) = x + y$.

Theorem 54 can be proved as follows:

Proof 55

$$\begin{aligned}
|pFDec(\mathcal{C}, \mathcal{C}_{gold})| &= |\bigcup_{i=1}^{k} pFDec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})) \cup pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})| \\
&\textit{because the set of Pair False Negatives of } \mathcal{F}^{set} \textit{ is disjoint to the set of Pair False Negatives of each factor} \\
&= |\bigcup_{i=1}^{k} \big(pFDec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))\big)| + |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})| \\
&\textit{because all factor sets are disjoint} \\
&= \sum_{i=1}^{k} |pFDec(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold}))| + |pFN(\mathcal{F}^{set}, \mathcal{C}_{gold})|
\end{aligned}$$

Pair Recall An interesting case is the Pair Recall that is quasi-Σ-decomposable.

Theorem 55 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Pair Recall is a quasi-Σ-decomposable quality measure because it can be computed as:*

$$pRec(\mathcal{C}, \mathcal{C}_{gold}) = \frac{1}{|M(\mathcal{C}_{gold})|} \times \sum_{i=1}^{k} |pTP\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)|$$

and hence by setting $\Theta = \Sigma$, $f_A(x, y) = |pTP(x, y)|$, $f_B(x, y) = |M(y)|$, and $f_C(x, y) = \frac{x}{y}$.

Theorem 55 can be proved as follows:

Proof 56

$$\begin{aligned}
pRec(\mathcal{C}, \mathcal{C}_{gold}) &= \frac{|M(\mathcal{C}) \cap M(\mathcal{C}_{gold})|}{|M(\mathcal{C}_{gold})|} \\
&\overset{\text{Theo. 46}}{=} \frac{|\bigcup_{i=1}^{k} \big(M(\pi_{F_i}(\mathcal{C}))\big) \cap M(\mathcal{C}_{gold})|}{|M(\mathcal{C}_{gold})|} \\
&= \frac{1}{|M(\mathcal{C}_{gold})|} \times |\bigcup_{i=1}^{k} \big(M(\pi_{F_i}(\mathcal{C})) \cap M(\pi_{F_i}(\mathcal{C}_{gold}))\big)| \\
&\textit{because all factor sets are disjoint} \\
&= \frac{1}{|M(\mathcal{C}_{gold})|} \times \sum_{i=1}^{k} |M(\pi_{F_i}(\mathcal{C})) \cap M(\pi_{F_i}(\mathcal{C}_{gold}))| \\
&= \frac{1}{|M(\mathcal{C}_{gold})|} \times \sum_{i=1}^{k} |pTP\big(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})\big)|
\end{aligned}$$

Actually, Pair Recall can be also computed in a factor-based fashion without using any of the functions f_B and f_C, e.g. by

$$p\text{Rec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \sum_{i=1}^{k} p\text{Rec}(\pi_{F_i}(\mathcal{C}), \mathcal{C}_{\text{gold}})$$

Nevertheless, since we define some quality measures only for comparing clusterings with equal ranges and $\pi_{F_i}(\mathcal{C})$ and $\mathcal{C}_{\text{gold}}$ do not have equal ranges, we only consider it to be quasi-Σ-decomposable instead of Σ-decomposable (otherwise we need to reformulate the definition of Θ-decomposability).

Pair Precision Another interesting case is Pair Precision. The original function for computing Pair Precision (see Section 4.3.8) calculates the fraction of the Number of Pair True Positives to the total Number of Pair Positives (true or false). As a consequence, the denominator depends on the numerator and a pure factor-based consideration seems impossible. Notwithstanding, Pair Precision can be computed by using the Generalized Merge Distance [MWGM10] by setting the cost functions $f_s(x, y) = xy$ and $f_m(x, y) = 0$. Since both functions are operation order independent, Pair Precision is a quasi-Σ-decomposable quality measure (see Generalized Merge Distance below). This shows us that the property of quasi-Σ-decomposability does not only depend on the measure itself, but also depends on the different functions that can be used to compute this measure.

Pair F_1-score Currently, we do not know a function to compute the Pair F_1-score that is Θ-decomposable or quasi-Θ-decomposable.

Closest Cluster Recall Currently, we do not know a function to compute the Closest Cluster Recall that is Θ-decomposable or quasi-Θ-decomposable.

Closest Cluster Precision Currently, we do not know a function to compute the Closest Cluster Precision that is Θ-decomposable or quasi-Θ-decomposable.

Closest Cluster F_1-score Currently, we do not know a function to compute the Closest Cluster F_1-score that is Θ-decomposable or quasi-Θ-decomposable.

Basic Merge Distance The Basic Merge Distance is a special case of the Generalized Merge Distance that result from setting the two cost functions f_s and f_m to one. Since in this case f_m is a constant and a constant is operation order independent, this measure is quasi-Σ-decomposable (for explanation see Generalized Merge Distance).

Generalized Merge Distance The Generalized Merge Distance is an interesting measure because it can be used to compute several other measures as Pair Recall, Pair Precision, and the Variation of Information.

Theorem 56 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Generalized Merge Distance is a quasi-Σ-decomposable quality measure if the cost function f_m is operation order independent because in this case it can be computed as:*

$$GMD(\mathcal{C}, \mathcal{C}_{gold}) = \sum_{i=1}^{k} GMD(\pi_{F_i}(\mathcal{C}), \pi_{F_i}(\mathcal{C}_{gold})) + GMD(\bigcup_{i=1}^{k} \pi_{F_i}(\mathcal{C}_{gold}), \mathcal{C}_{gold})$$

and hence by setting $\Theta = \Sigma$, $f_A(x, y) = GMD(x, y)$, $f_B(x, y) = GMD(\bigcup_{c \in x} \pi_c(y), y)$, and $f_C(x, y) = x + y$.

Theorem 56 can be proved as follows:

Proof 57 *Menestrina et al. proved in [MWGM10] that there exists a minimum cost legal path from a clustering $\mathcal{C}$ to the clustering $\mathcal{C}_{gold}$ where all of the split operations precede the merge operations. The split operations only concerns Pair False Positives and the entities of a Pair False Positive always belong to the same cluster and hence always belong to the same factor. In contrast, the merge operations either concerns intra-factor Pair False Negatives or inter-factor Pair False Negatives. The merges that eliminates the intra-factor Pair False Negatives can be performed on each factor separately. In contrast, the inter-factor Pair False Negatives are the same for all alternatives of an uncertain clustering and the merge costs that are required to eliminate them can be computed only based on the factor sets and the gold standard itself.*

Thus, we can divide the transformation of $\mathcal{C}$ into $\mathcal{C}_{gold}$ into two parts. First we perform all split operations and all local merge operations on the individual factor separately and hence compute a cost value for each factor. Then we perform all global merge operations to eliminate the inter-factor Pair False Negatives. By assuming that all splits and all local merges have been already performed, we know that the input to the merging step has only Pair True Positives and neither Pair False Positives nor intra-factor Pair False Negatives. Thus, we can simulate the result of the factor-based computation step by projecting the gold standard on the individual factor sets (note that this division is not a factorization of the gold standard because it cannot reconstructed from the indivual projection results) and then compute the merging cost that is required for transforming the divided gold standard into the original gold standard. As a consequence, the splitting step can be performed on each factor separately and the merging step can be performed in two phases: First by processing the factors separately and then by only considering the factor sets and the gold standard. However, by doing so we need to reorder the merge operations and this is only legal if the cost function f_m is operation order independent *(for definition see [MWGM10]).*

Variation of Information Based on its actual definition it is hard to say whether or not the Variation of Information is Θ-decomposable or quasi-Θ-decomposable. However, it can be computed by the Generalized Merge Distance and therefore can be proved to be quasi-Σ-decomposable.

Theorem 57 *The Variation of Information is a quasi-Σ-decomposable quality measure.*

Theorem 57 can be proved as follows:

Proof 58 *According to Menestrina et al. [MWGM10], the Variation of Information can be computed by using the Generalized Merge Distance if we set the cost functions to $f_s(x,y) = f_m(x,y) = h(x+y) - h(x) - h(y)$ with $h(x) = \frac{x}{N}\log\left(\frac{x}{N}\right)$ where N is the total number of considered entities. As a consequence, we simply need to prove that these cost functions are operation order independent. According to Menestrina et al. [MWGM10], a cost function $f(x,y)$ is operation order independent, if $f(x,y) + f(x+y,z) = f(x,z) + f(x+z,y)$ for all x, y, z. For the given merge function holds:*

$$\begin{aligned} f_m(x,y) + f_m(x+y,z) &= h(x+y) - h(x) - h(y) + h(x+y+z) - h(x+y) - h(z) \\ &= h(x+z+y) - h(x) - h(z) - h(y) \\ &= h(x+z) - h(x) - h(z) + h(x+z+y) - h(x+z) - h(y) \\ &= f_m(x,z) + f_m(x+z,y) \end{aligned}$$

Of course, since both cost functions are identical, the same holds for the split function. As a consequence, f_s and f_m are both operation order independent and the Variation of Information is a quasi-Σ-decomposable quality measure.

Rand Index A measure that is often used to compare two clusterings is the Rand Index [Ran71]. Let $\mathcal{C}_p$ and $\mathcal{C}_q$ be two cluster-disjoint clusterings of a range with size n, the Rand-Index can be defined as:

$$\begin{aligned} RI(\mathcal{C}_p, \mathcal{C}_q) &= \frac{|M(\mathcal{C}_p) \cap M(\mathcal{C}_q)| + |U(\mathcal{C}_p) \cap U(\mathcal{C}_q)|}{Pairs(\mathcal{C}_p)} \\ &= \frac{1}{\binom{n}{2}} \times \left(|M(\mathcal{C}_p) \cap M(\mathcal{C}_q)| + |U(\mathcal{C}_p) \cap U(\mathcal{C}_q)|\right) \end{aligned}$$

We do not know a Σ-decomposable function to compute the Rand Index, but it is a quasi-Σ-decomposable quality measure.

Theorem 58 *Let $\mathcal{C}$ be a crisp clustering, let $\mathcal{C}_{gold}$ be the corresponding gold standard, and let $\mathcal{F}$ be a range-disjoint factorization of $\mathcal{C}$ with $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. The Rand Index between $\mathcal{C}$ and $\mathcal{C}_{gold}$ is a quasi-Σ-decomposable quality measure because it can be computed as:*

$$\begin{aligned} RI(\mathcal{C}, \mathcal{C}_{gold}) = \frac{1}{\binom{n}{2}} \times \Big(&\sum_{i=1}^{k} \left(|M(\pi_{F_i}(\mathcal{C})) \cap M(\pi_{F_i}(\mathcal{C}_{gold}))| + |U(\pi_{F_i}(\mathcal{C})) \cap U(\pi_{F_i}(\mathcal{C}_{gold}))|\right) \\ &+ |U(\mathcal{F}^{set}) \cap U(\mathcal{C}_{gold})| \Big) \end{aligned}$$

and hence by setting $\Theta = \Sigma$, $f_A(x, y) = |M(x) \cap M(y)| + |U(x) \cap U(y)|$, $f_B(x, y) = |U(x) \cap U(y)|$, and $f_C(x, y) = \frac{1}{\binom{n}{2}} \times (x + y)$.

Theorem 58 can be proved as follows:

Proof 59

$$RI(\mathcal{C}, \mathcal{C}_{gold}) \quad = \quad \frac{1}{\binom{n}{2}} \times \big(|M(\mathcal{C}) \cap M(\mathcal{C}_{gold})| + |U(\mathcal{C}) \cap U(\mathcal{C}_{gold})|\big)$$

because $\mathcal{F}^{set}$ forms a factorization of $\mathcal{C}$, all entity pairs that belong to different factor sets, i.e. $U(\mathcal{F}^{set})$, are UNMATCHES *in $\mathcal{C}$* $\Rightarrow U(\mathcal{C}) = \bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C})) \cup U(\mathcal{F}^{set})$

$$= \frac{1}{\binom{n}{2}} \times \Big(|\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C})) \cap M(\mathcal{C}_{gold})| + |\big(\bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C})) \cup U(\mathcal{F}^{set})\big) \cap U(\mathcal{C}_{gold})|\Big)$$

$$= \frac{1}{\binom{n}{2}} \times \Big(|\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C})) \cap M(\mathcal{C}_{gold})| + |\big(\bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C})) \cap U(\mathcal{C}_{gold})\big) \cup \big(U(\mathcal{F}^{set}) \cap U(\mathcal{C}_{gold})\big)|\Big)$$

because $\bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C}))$ *and* $U(\mathcal{F}^{set})$ *are disjoint*

$$= \frac{1}{\binom{n}{2}} \times \Big(|\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C})) \cap M(\mathcal{C}_{gold})| + |\bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C})) \cap U(\mathcal{C}_{gold})| + |U(\mathcal{F}^{set}) \cap U(\mathcal{C}_{gold})|\Big)$$

because all factor sets are disjoint

$$= \frac{1}{\binom{n}{2}} \times \Big(\sum_{i=1}^{k} |M(\pi_{F_i}(\mathcal{C})) \cap M(\mathcal{C}_{gold})| + \sum_{i=1}^{k} |U(\pi_{F_i}(\mathcal{C})) \cap U(\mathcal{C}_{gold})| + |U(\mathcal{F}^{set}) \cap U(\mathcal{C}_{gold})|\Big)$$

$$= \frac{1}{\binom{n}{2}} \times \Big(\sum_{i=1}^{k} \big(|M(\pi_{F_i}(\mathcal{C})) \cap M(\mathcal{C}_{gold})| + |U(\pi_{F_i}(\mathcal{C})) \cap U(\mathcal{C}_{gold})|\big) + |U(\mathcal{F}^{set}) \cap U(\mathcal{C}_{gold})|\Big)$$

$$= \frac{1}{\binom{n}{2}} \times \Big(\sum_{i=1}^{k} \big(|M(\pi_{F_i}(\mathcal{C})) \cap M(\pi_{F_i}(\mathcal{C}_{gold}))| + |U(\pi_{F_i}(\mathcal{C})) \cap U(\pi_{F_i}(\mathcal{C}_{gold}))|\big) + |U(\mathcal{F}^{set}) \cap U(\mathcal{C}_{gold})|\Big)$$

In contrast to the most other measures, the Rand Index is symmetric. In the case of a binary classification (as we have in duplicate detection), the Rand Index corresponds to the accuracy of the binary classification. Since it considers Pair True Positives and this number dominates the formula, it is usually only less suitable to rate the decision correctness of a clustering. Nevertheless, it can be used to compute the similarity between two clustering alternatives. Sometimes we even need the Rand Index for comparing two probabilistic clusterings (see Theorem 44). In this case, we can efficiently compute the Rand Index if both clusterings are factorized based on the same factor sets.

Theorem 59 *Let $\mathcal{C}_p$ and $\mathcal{C}_q$ be two cluster-disjoint clusterings of a range with size n and let the range partition $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$ form a factorization of $\mathcal{C}_p$ and form a factorization of $\mathcal{C}_q$, the Rand Index of $\mathcal{C}_p$ and $\mathcal{C}_q$ can be computed as:*

$$RI(\mathcal{C}_p, \mathcal{C}_q) = \frac{1}{\binom{n}{2}} \times \big(\sum_{i=1}^{k} (|M(\pi_{F_i}(\mathcal{C}_p)) \cap M(\pi_{F_i}(\mathcal{C}_q))| + |U(\pi_{F_i}(\mathcal{C}_p)) \cap U(\pi_{F_i}(\mathcal{C}_q))|) + |U(\mathcal{F}^{set})|\big)$$

Theorem 59 can be proved as follows:

Proof 60

$$RI(\mathcal{C}_p,\mathcal{C}_q) = 1/\binom{n}{2} \times \big(|M(\mathcal{C}_p) \cap M(\mathcal{C}_q)| + |U(\mathcal{C}_p) \cap U(\mathcal{C}_q)|\big)$$

$$RI(\mathcal{C}_p,\mathcal{C}_q) = \frac{1}{\binom{n}{2}} \times \big(|M(\mathcal{C}_p) \cap M(\mathcal{C}_q)| + |U(\mathcal{C}_p) \cap U(\mathcal{C}_q)|\big)$$

because $\mathcal{F}^{set}$ forms a factorization of $\mathcal{C}_p$ and $\mathcal{C}_q$, all entity pairs that belong to different factor sets, i.e. $U(\mathcal{F}^{set})$, are UNMATCHES *in $\mathcal{C}_p$ and $\mathcal{C}_q \Rightarrow U(\mathcal{C}) = \bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C})) \cup U(\mathcal{F}^{set})$ for $\mathcal{C} \in \{\mathcal{C}_p, \mathcal{C}_q\}$*

$$= \frac{1}{\binom{n}{2}} \times \Big(|\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_p)) \cap \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_q))| + |(\bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C}_p)) \cap \bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C}_q))) \cup U(\mathcal{F}^{set})|\Big)$$

$$= \frac{1}{\binom{n}{2}} \times \Big(|\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_p)) \cap \bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_q))| + |\bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C}_p)) \cap \bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C}_q))| + |U(\mathcal{F}^{set})|\Big)$$

because all factor sets are disjoint

$$= \frac{1}{\binom{n}{2}} \times \Big(|\bigcup_{i=1}^{k} M(\pi_{F_i}(\mathcal{C}_p)) \cap M(\pi_{F_i}(\mathcal{C}_q))| + |\bigcup_{i=1}^{k} U(\pi_{F_i}(\mathcal{C}_p)) \cap U(\pi_{F_i}(\mathcal{C}_q))| + |U(\mathcal{F}^{set})|\Big)$$

because all factor sets are disjoint

$$= \frac{1}{\binom{n}{2}} \times \Big(\sum_{i=1}^{k} |M(\pi_{F_i}(\mathcal{C}_p)) \cap M(\pi_{F_i}(\mathcal{C}_q))| + \sum_{i=1}^{k} |U(\pi_{F_i}(\mathcal{C}_p)) \cap U(\pi_{F_i}(\mathcal{C}_q))| + |U(\mathcal{F}^{set})|\Big)$$

$$= \frac{1}{\binom{n}{2}} \times \Big(\sum_{i=1}^{k} \big(|M(\pi_{F_i}(\mathcal{C}_p)) \cap M(\pi_{F_i}(\mathcal{C}_q))| + |U(\pi_{F_i}(\mathcal{C}_p)) \cap U(\pi_{F_i}(\mathcal{C}_q))|\big) + |U(\mathcal{F}^{set})|\Big)$$

Appendix B

Cluster-based Interpretation of the Certain World Semantics

In this chapter, we propose a cluster-based interpretation of the Certain World Semantics. For that purpose, we first present some clustering properties that are required for this interpretation in Section B.1. Then we consider in which way a cluster-based quality measure can be used in order to rate the decision correctness of the certain information of an uncertain clustering in Section B.2. Since an uncertain clustering is usually given in a factorized way, we finally consider a factor-based computation of the cluster-based interpretation in Section B.3.

B.1. Properties of Clusterings

Before considering the cluster-based interpretation of the Certain World Semantics we need to introduce some properties of uncertain clusterings and methods for uncertain clusterings that are useful for these considerations.

A clustering $\mathcal{C}$ represents information, in the following denoted as clustering information, on the relationships between the elements of its range. Each projection of $\mathcal{C}$ on one of its range's subsets contains a part of the clustering information of $\mathcal{C}$.

Definition 77 (Clustering Fact): *Let $\mathcal{C}$ be a clustering that is defined on the range $rng(\mathcal{C})$. A projection of $\mathcal{C}$ on the element set $S \subseteq rng(\mathcal{C})$ contains a part of the clustering information of $\mathcal{C}$ and is in the following denoted as a **clustering fact** of $\mathcal{C}$. Each clustering fact of $\mathcal{C}$ is a clustering that is dominated by $\mathcal{C}$. The set of all clustering facts of a clustering $\mathcal{C}$, i.e. all clusterings that are dominated by $\mathcal{C}$, is defined as:*

$$CF(\mathcal{C}) = \{\pi_S(\mathcal{C}) \mid S \subseteq rng(\mathcal{C})\}$$

A clustering fact $\mathcal{C}' \in CF(\mathcal{C})$ is called trivial, iff it has at most one element because the information that an element is in the same cluster as itself is a trivial information (similar holds for the empty set). A clustering fact $\mathcal{C}' \in CF(\mathcal{C})$ is called elementary, iff it has exact two elements, i.e. iff $|rng(\mathcal{C}')| = 2$.

The set of all elementary clustering facts of a clustering $\mathcal{C}$ corresponds to its set of pairwise decisions and hence can be also denoted as *Pairs*($\mathcal{C}$). Because each fact is a clustering itself, each fact has in turn a set of facts that we sometimes denote as *subfacts*.

Theorem 60 *The relation of being a clustering fact is transitive, i.e if $\mathcal{C}'$ is a clustering fact of $\mathcal{C}$ and $\mathcal{C}^*$ is a clustering fact of $\mathcal{C}'$, $\mathcal{C}^*$ is also a clustering fact of $\mathcal{C}$.*

Theorem 60 is proved by Proof 61 in Section B.4.

We will use the definition of a fact in Section 8.8.2.4 to define the certain information that is represented by an uncertain clustering. In duplicate detection, a clustering fact represents the information about the *'is duplicate'* relationship or the *'is no duplicate'* relationship between two or more entities. Consequently, if a clustering $\mathcal{C}_i$ is a fact of clustering $\mathcal{C}_j$, $\mathcal{C}_j$ contains all duplicate information that is contained in $\mathcal{C}_i$.

Example 225 *The clustering $\mathcal{C} = \{\langle a, b\rangle, \langle c\rangle, \langle d\rangle\}$ has the eleven non-trivial clustering facts CF_1 to CF_{11} that are listed below.*

$$
\begin{array}{lcllcl}
CF_1 & = & \{\langle a, b\rangle, \langle c\rangle, \langle d\rangle\} & CF_7 & = & \{\langle a\rangle, \langle c\rangle\} \\
CF_2 & = & \{\langle a, b\rangle, \langle c\rangle\} & CF_8 & = & \{\langle a\rangle, \langle d\rangle\} \\
CF_3 & = & \{\langle a, b\rangle, \langle d\rangle\} & CF_9 & = & \{\langle b\rangle, \langle c\rangle\} \\
CF_4 & = & \{\langle a\rangle, \langle c\rangle, \langle d\rangle\} & CF_{10} & = & \{\langle b\rangle, \langle d\rangle\} \\
CF_5 & = & \{\langle b\rangle, \langle c\rangle, \langle d\rangle\} & CF_{11} & = & \{\langle c\rangle, \langle d\rangle\} \\
CF_6 & = & \{\langle a, b\rangle\} & & &
\end{array}
$$

Theorem 61 *Let $\mathcal{C}$ be a cluster-disjoint clustering and let X be an arbitrary set of the same domain as $rng(\mathcal{C})$. Projecting the clustering fact set of $\mathcal{C}$ on X is equivalent to the clustering fact set of the projection of $\mathcal{C}$ on X, i.e. $\pi_X(CF(\mathcal{C})) = CF(\pi_X(\mathcal{C}))$.*

Theorem 61 is proved by Proof 62 in Section B.4.

As the information content of a clustering $\mathcal{C}$ (in symbols IC($\mathcal{C}$)) we consider the amount of clustering information that is represented by $\mathcal{C}$. Since we want to rate information on element relationships, we quantify the information content as the number of elementary clustering facts of $\mathcal{C}$:

$$\text{IC}(\mathcal{C}) = |Pairs(\mathcal{C})| = \frac{1}{2} \times |rng(\mathcal{C})| \times (|rng(\mathcal{C})| - 1) \tag{B.1}$$

Note, a trivial clustering does not have any information on element relationships. Therefore, it has zero information content.

Example 226 *For illustration, the information content of the clustering $\mathcal{C} = \{\langle a, b\rangle, \langle c\rangle, \langle d, e\rangle\}$ is $IC(\mathcal{C}) = 10$. In contrast, the accumulative information content of the two clusterings $\mathcal{C}_1 = \{\langle a, b\rangle, \langle c\rangle\}$ and $\mathcal{C}_2 = \{\langle d, e\rangle\}$ is only $IC(\mathcal{C}_1) + IC(\mathcal{C}_2) = 3 + 1 = 4$ because we do not have any information on the relationships between the elements of $\mathcal{C}_1$ and the elements of $\mathcal{C}_2$.*

For combining the elements of two clustering sets, we introduce the clustering set integration operator.

Definition 78 (Integration of Clustering Sets): *Two clustering sets CS_i and CS_j can be integrated to the single clustering set CS_{ij} (in symbols: $CS_{ij} = CS_i \otimes CS_j$) by pairwise integrating their elements: $CS_{ij} = \{\mathcal{C}_i \boxtimes \mathcal{C}_j \mid \mathcal{C}_i \in CS_i, \mathcal{C}_j \in CS_j\}$. The clustering set integration operator is reflexive, symmetric and associative.*

The n-ary integration of the k clustering sets $CS_1, \ldots, CS_k$ is defined as a subsequent execution of $k-1$ binary integrations, i.e.: $\bigotimes_{i=1}^{k} CS_i = ((CS_1 \otimes CS_2) \otimes \ldots) \otimes CS_k$. Note, the crisp version of the $\otimes$ operator is similar to the incomplete version of the $\boxtimes$ operator, but is semantical different because the first considers certain input sets and the second considers incomplete input values.

Often we do not need all the clusterings of a specific clustering set, e.g. all clustering facts, but only the clusterings that are not dominated by any other clustering of this set.

Definition 79 (Clustering Set Skyline): *Let CS be a set of clusterings. The skyline of CS is defined as the set $\overline{CS} \subseteq CS$ that contains all clusterings $\mathcal{C}_i \in CS$ that are not dominated by any other clustering $\mathcal{C}_j \in CS$:*

$$\overline{CS} = \{\mathcal{C}_i \in CS \mid \nexists \mathcal{C}_j \in CS \colon \mathcal{C}_i \neq \mathcal{C}_j, \mathcal{C}_i \prec \mathcal{C}_j\}$$

Example 227 *Consider the clustering set $CS = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ as shown below. Since $\mathcal{C}_2$ and $\mathcal{C}_4$ are dominated by $\mathcal{C}_1$, the skyline of CS results in $\overline{CS} = \{\mathcal{C}_1, \mathcal{C}_3\}$.*

$\mathcal{C}_1 = \{\langle a, b\rangle, \langle c\rangle\}$	$\mathcal{C}_1 = \{\langle a, b\rangle, \langle c\rangle\}$
$\mathcal{C}_2 = \{\langle b\rangle, \langle c\rangle\}$	$\mathcal{C}_3 = \{\langle c\rangle, \langle d\rangle\}$
$\mathcal{C}_3 = \{\langle c\rangle, \langle d\rangle\}$	
$\mathcal{C}_4 = \{\langle a, b\rangle\}$	
Clustering Set CS	*Skyline of CS*

In order to use the skyline concept for sets of clusters as well, we need to define the concept of domination for the domain of clusters. Since each cluster can be considered as an own clustering, this domination of clusters can be defined as follows: A single cluster C is dominated by a single cluster C' if the clustering $\mathcal{C} = \{C\}$ is dominated by the clustering $\mathcal{C}' = \{C'\}$, i.e. $C \subseteq C'$.

B.2. Measuring the Decision Correctness of Certain Clustering Information

As we have demonstrated in Section 8.8.2.4 computing the decision correctness of an uncertain clustering under the Certain World Semantics is quite simple for the pair-based interpretation because we can prove certainty for each of the entity pairs individually. However, as we will demonstrate in this section, computing the decision correctness of an uncertain clustering under the Certain World Semantics is much more difficult for the cluster-based interpretation.

As for the pair-based interpretation we first consider only absolute certainty and then relax the meaning of certainty afterwards. For that reason, the certain information of a probabilistic clustering $\mathfrak{C} = (\Gamma, Pr)$ is equivalent to the certain information of its incomplete clustering Γ and we therefore restrict our consideration to incomplete clusterings in the main part of this section.

The pair-based interpretation restricts the meaning of certainty to single pairs, but the cluster-based interpretation need to consider certainty in a broader way. For that reason, we consider the certain clustering information of an uncertain clustering as all its certain clustering facts.

Definition 80 (Certain Clustering Fact): *Let Γ be an incomplete clustering. A clustering fact of Γ is defined to be certain if it is a clustering fact of all alternatives of Γ. The set of all certain clustering facts of Γ is defined as:*

$$cCF(\Gamma) = \bigcap\nolimits_{\mathcal{C}\in\Gamma} CF(\mathcal{C})$$

It is obvious that if a fact $\mathcal{C}$ only dominates certain facts, $\mathcal{C}$ is certain itself.

Theorem 62 *Let Γ be an incomplete clustering. A clustering fact $\mathcal{C} \in CF(\Gamma)$ is certain, iff all of its subfacts are certain facts of Γ , i.e. $\mathcal{C} \in cCF(\Gamma) \Leftrightarrow CF(\mathcal{C}) \subseteq cCF(\Gamma)$.*

Theorem 62 is proved by Proof 63 in Appendix B.4. From Theorem 62 we can derive that a clustering fact is certain, iff all its elementary subfacts (i.e. pairwise duplicate decisions) are certain.

Because of Theorem 62, the set of certain clustering facts can be computed by using the apriori algorithm as it is known from frequent item set mining [AS94]. We start from the elementary facts that we can initially compute by considering every entity pair individually as we did it to compute the certain decisions in the pairwise interpretation. Then, the larger facts are computed in an iterative fashion. In the first iteration, we compute all 3-element facts. A 3-element fact is only certain, if all its elementary subfacts are certain. Then we proceed with the 4-element facts and so on until no new certain fact results from the considered iteration. Note, the difference to conventional algorithms for frequent itemset mining is that we do not have a minimal support threshold and that we do not need to count the frequency of an itemset in a set of transactions, but can directly conclude the certainty of a fact from the certainty of its subfacts.

The computation approach can be further optimized because we need not to consider all the n different $(n-1)$-element subfacts to check whether an n-element clustering is a certain fact of an incomplete clustering or not, but can always reduce the number of checked subfacts to three.

Theorem 63 *Let Γ be an incomplete clustering. The certainty of an n-element fact $\mathcal{C} \in CF(\Gamma)$ can be checked by checking the certainty of two of its $(n-1)$-element subfacts and by checking the certainty of one of its elementary subfacts.*

Theorem 63 is proved by Proof 64 in Appendix B.4.

To avoid that the same clustering information is considered for multiple times, we restrict quality computation to the skyline of the certain clustering facts which we call as the certain fact skyline $\overline{cCF(\Gamma)}$. The certain fact skyline of Γ can be computed simultaneously to the set of certain clusterings facts of Γ by managing a second list of facts. A n-element fact $\mathcal{C}$ is added to this list when it is detected to be certain and is removed from this list when any $n+1$-element fact is added to the list that has $\mathcal{C}$ as a subfact.

Lemma 4 *Let Γ be an incomplete clustering. The clusterings of the certain fact skyline of Γ are not necessarily range-disjoint, i.e. it can happen that $\exists \mathcal{C}_i, \mathcal{C}_j \in \overline{cCF(\Gamma)} \colon \mathcal{C}_i \circledcirc \mathcal{C}_j$.*

Clustering	Certain Fact Skyline $\overline{cCF}$	*CCSel*	*CSSel*
$\mathfrak{C}_2$	$CF_{21} = \{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_4, e_8\rangle\}$ $CF_{22} = \{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle\}$ $CF_{23} = \{\langle e_1, e_5\rangle, \langle e_7\rangle, \langle e_4, e_8\rangle\}$ $CF_{24} = \{\langle e_1, e_5\rangle, \langle e_7\rangle, \langle e_3\rangle\}$	$\{\langle e_1, e_5\rangle\}$	$\{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle, \langle e_4, e_8\rangle, \langle e_7\rangle\}$
$\mathfrak{C}_3$	$CF_{31} = \{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_4, e_8\rangle\}$ $CF_{32} = \{\langle e_1, e_5\rangle, \langle e_3\rangle, \langle e_7\rangle\}$ $CF_{33} = \{\langle e_1, e_5\rangle, \langle e_7\rangle, \langle e_4, e_8\rangle\}$	$\{\langle e_1, e_5\rangle\}$	$\{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle, \langle e_4, e_8\rangle, \langle e_7\rangle\}$
$\mathfrak{C}_4$	$CF_{41} = \{\langle e_1, e_5\rangle, \langle e_3\rangle, \langle e_8\rangle\}$ $CF_{42} = \{\langle e_1, e_5\rangle, \langle e_4\rangle, \langle e_7\rangle\}$ $CF_{43} = \{\langle e_2\rangle, \langle e_4\rangle\}$ $CF_{44} = \{\langle e_2\rangle, \langle e_8\rangle\}$ $CF_{45} = \{\langle e_4\rangle, \langle e_6\rangle\}$ $CF_{46} = \{\langle e_6\rangle, \langle e_8\rangle\}$	$\emptyset$	$\{\langle e_1, e_5\rangle, \langle e_2\rangle, \langle e_3\rangle, \langle e_4\rangle, \langle e_5\rangle, \langle e_6\rangle, \langle e_7\rangle, \langle e_8\rangle\}$

Table B.1.: The certain fact skylines, the Certain Cluster Selections, and the Certain Subcluster Selections of our sample probabilistic clusterings $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$

Lemma 4 can be proved by a simple example.

Example 228 *Consider the incomplete clustering* $\Gamma = \{\mathcal{C}_1, \mathcal{C}_2\}$ *with the two alternatives* $\mathcal{C}_1 = \{\langle e_1\rangle, \langle e_2, e_3\rangle\}$ *and* $\mathcal{C}_2 = \{\langle e_1\rangle, \langle e_2\rangle, \langle e_3\rangle\}$. *The only certain information of* Γ *is that* e_1 *is neither a duplicate to* e_2 *nor a duplicate to* e_3. *Consequently, the certain clustering facts of* Γ *are* $CF_1 = \{\langle e_1\rangle, \langle e_2\rangle\}$ *and* $CF_2 = \{\langle e_1\rangle, \langle e_3\rangle\}$. *Both facts are not dominated by any other fact and hence both belong to the certain fact skyline of* Γ. *Because they both contain the entity* e_1 *their ranges overlap.*

Nevertheless, although the ranges of two clusterings of the certain fact skyline can overlap, the entities of the overlaps always form the same clusters in both of the overlapping clusterings and hence integrating the elements of the certain fact skyline always results into a cluster-disjoint clustering.

Theorem 64 *Let* Γ *be a cluster-disjoint incomplete clustering. Integrating some elements of the certain fact skyline of* Γ *always results in a cluster-disjoint clustering.*

Theorem 64 is proved by Proof 65 in Appendix B.4.

As a consequence, the clustering that results from integrating some elements of the certain fact skyline can always be used as the input to a conventional cluster-based quality measure.

Example 229 *For illustration we reconsider the three probabilistic clusterings from the motivating example. The certain fact skyline of these clusterings are presented in Table B.1. As we can see the ranges of some facts of the same skyline overlap, e.g.* CF_{21} *and* CF_{22}, *but the overlap always concerns clusters that belong to both facts. For example,* CF_{21} *and* CF_{22} *contain the clusters* $\langle e_1, e_5\rangle$ *and* $\langle e_2, e_6\rangle$.

Cluster-based quality measures compare a single clustering with the gold standard. Consequently, we have two fundamental approaches to rate the quality of the certain fact skyline (and hence the certain information of its corresponding incomplete clustering). In the first approach, we construct a single clustering from the skyline, rate the quality of this clustering, and finally use the clustering's quality as the quality of the skyline ((*single-clustering approach*). In the second approach, we rate the quality of the skyline as the aggregated quality of all its elements. Thus, we first rate the quality of all clusterings of the skyline separately, and then aggregate the resultant quality scores to a single one (*multi-clustering approach*).

B.2.1. Single-Clustering Approach

As mentioned above, the idea of the single-clustering approach is to first construct a single clustering (called *Certain Clustering Component*, short CCC) that represents all the certain facts of the incomplete clustering Γ and then to use the quality of the Certain Clustering Component as the quality of the certain facts of Γ. Since the Certain Clustering Component sometimes does not contain every element of the considered range and the cluster-based measures for decision correctness are usually defined on clusterings with same ranges, we need to initially project the gold standard on the entities of the Certain Clustering Component. Therefore, let *QM* be a cluster-based measure of decision correctness, let Γ be an incomplete clustering, let $\mathcal{C}_{\text{gold}}$ be the corresponding gold standard, and let CCC be the Certain Clustering Component that has been constructed from the certain fact skyline of Γ, the decision correctness of the certain information of Γ is defined as:

$$QM^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) = QM(\mathsf{CCC}, \pi_{rng(\mathsf{CCC})}(\mathcal{C}_{\text{gold}})) \tag{B.2}$$

The problem with this approach is that constructing a single clustering that contains all and only all certain clustering facts of Γ is not possible, if none of the certain clustering facts has all other certain clustering facts as subfacts and hence if the certain fact skyline contains more than one element.

Example 230 *We will illustrate this circumstance by considering the two first facts CF_{43} and CF_{44} of $\mathfrak{C}_4$ (see Table B.1). Due to these facts, we have the information that e_2 and e_4 are certainly no duplicates and we have the certain information that e_2 and e_8 are certainly no duplicates. Combining the information of both clustering facts, however, would either implicate that e_4 and e_8 are certainly no duplicates (the clustering $\mathsf{CCC}_1 = \{\langle e_2\rangle, \langle e_4\rangle, \langle e_8\rangle\}$) or would implicate that e_4 and e_8 are certainly duplicates (the clustering $\mathsf{CCC}_2 = \{\langle e_2\rangle, \langle e_4, e_8\rangle\}$). Both implications, however, are incorrect because we cannot state with absolute certainty whether e_4 and e_8 are duplicates or not. As a consequence, there is no single clustering that contains the information of both facts without implicating information that is in none of these facts.*

For that reason, we have to make a compromise between certainty soundness (all clustering facts of the Certain Clustering Component are certain) and certainty completeness (the Certain Clustering Component contains all the certain clustering facts of Γ) in constructing the Certain Clustering Component.

Definition 81 (Certainty Soundness): *A clustering $\mathcal{C}$ is certainty sound with respect to an incomplete clustering Γ, if all its clustering facts are certain clustering facts of Γ, i.e.: $CF(\mathcal{C}) \subseteq cCF(\Gamma)$.*

Certain Clustering Component	ccRec	ccPrec	ccF_1	*IC*	*CPDecSound*			*CPDecComp*		
					$\mathfrak{C}_2$	$\mathfrak{C}_3$	$\mathfrak{C}_4$	$\mathfrak{C}_2$	$\mathfrak{C}_3$	$\mathfrak{C}_4$
$\{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_4, e_8\rangle\}$	1.0	1.0	1.0	15	1.0	1.0	0.6	0.63	0.68	0.6
$\{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle\}$	0.83	0.67	0.74	10	1.0	0.8	0.3	0.42	0.36	0.2
$\{\langle e_1, e_5\rangle, \langle e_7\rangle, \langle e_4, e_8\rangle\}$	1.0	1.0	1.0	10	1.0	1.0	0.8	0.42	0.45	0.53
$\{\langle e_1, e_5\rangle, \langle e_7\rangle, \langle e_3\rangle\}$	1.0	1.0	1.0	6	1.0	1.0	0.83	0.25	0.27	0.33
$\{\langle e_1, e_5\rangle, \langle e_3\rangle, \langle e_8\rangle\}$	1.0	1.0	1.0	6	0.83	0.83	1.0	0.21	0.23	0.4
$\{\langle e_1, e_5\rangle, \langle e_4\rangle, \langle e_7\rangle\}$	1.0	1.0	1.0	6	1.0	1.0	1.0	0.25	0.27	0.4
$\{\langle e_2\rangle, \langle e_4\rangle\}$	1.0	1.0	1.0	1	1.0	1.0	1.0	0.04	0.05	0.07
$\{\langle e_2\rangle, \langle e_8\rangle\}$	1.0	1.0	1.0	1	1.0	1.0	1.0	0.04	0.05	0.07
$\{\langle e_4\rangle, \langle e_6\rangle\}$	1.0	1.0	1.0	1	1.0	1.0	1.0	0.04	0.05	0.07
$\{\langle e_6\rangle, \langle e_8\rangle\}$	1.0	1.0	1.0	1	1.0	1.0	1.0	0.04	0.05	0.07
$\{\langle e_1, e_5\rangle\}$	1.0	1.0	1.0	1	1.0	1.0	1.0	0.04	0.05	0.07
$\{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle, \langle e_4, e_8\rangle, \langle e_7\rangle\}$	0.92	0.8	0.85	28	0.86	0.79	0.54	1.0	1.0	1.0
$\{\langle e_1, e_5\rangle, \langle e_2\rangle, \langle e_3\rangle, \langle e_4\rangle, \langle e_5\rangle, \langle e_6\rangle, \langle e_7\rangle, \langle e_8\rangle\}$	0.71	0.57	0.63	28	0.79	0.71	0.54	0.92	0.91	1.0

Table B.2.: The qualities and other properties of potential Certain Clustering Components of the three uncertain sample clusterings $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$.

Following Theorem 62, a clustering $\mathcal{C}$ is a certain fact of an incomplete clustering Γ, if all its subfacts are certain facts of Γ. Consequently, a clustering $\mathcal{C}$ is certainty sound with respect to an incomplete clustering Γ, if it is a certain fact of Γ and hence if it is dominated by at least one clustering of the certain fact skyline of Γ.

Definition 82 (Certainty Completeness): *A clustering $\mathcal{C}$ is certainty complete with respect to an incomplete clustering Γ, if all certain clustering facts of Γ are clustering facts of $\mathcal{C}$, i.e.: $CF(\mathcal{C}) \supseteq cCF(\Gamma)$.*

Consequently, a clustering $\mathcal{C}$ is certainty complete with respect to an incomplete clustering Γ, if it dominates all clusterings of the certain fact skyline of Γ.

One extreme is to take the certain fact of Γ that contains the most clustering information, i.e. the largest one, as the Certain Clustering Component. In this case, we can guarantee that all the facts that are represented by the CCC are certain in Γ and hence can ensure that CCC is certainty sound. The other extreme is to construct the CCC by integrating all the certain facts of Γ into a single clustering. In this case, we can guarantee that all the certain facts of Γ are included in the CCC and hence can ensure that CCC is certainty complete.

A compromise between these two extremes can be realized by selecting the largest clustering of the certain fact skyline and then to integrate it with other clusterings of the skyline as long as the resultant certainty soundness is tolerated by the user. For that purpose, however, we need measures for rating the certainty soundness and the certainty completeness of the resultant CCC in a more differentiated way, i.e. not only by 0 or 1 but by a real number between 0 and 1.

- **Certain Pairwise Decision Soundness:** The most intuitive way to rate the certainty soundness of a clustering $\mathcal{C}$ with respect to an incomplete clustering Γ is to compute the amount of pairwise

decisions in $\mathcal{C}$ that agree with a certain pairwise decision in Γ. The problem with this approach is that the resultant ratio is dominated by the number of certain UNMATCHES of Γ and the number of UNMATCHES of $\mathcal{C}$. Moreover, usually the most UNMATCHES of $\mathcal{C}$ will be certain UNMATCHES of Γ, i.e. $|U_c(\Gamma)| \ggg |M_c(\Gamma)|$, $|U(\mathcal{C})| \ggg |M(\mathcal{C})|$, and $U_c(\Gamma) \cap U(\mathcal{C}) \simeq U(\mathcal{C})$. Consequently, the considered amount will most often be extremely close to 1, even in cases where $\mathcal{C}$ and Γ disagree in the most of their certain MATCHES. For that reason, we introduce the weights $\alpha, \beta \in \mathbb{R}$ that enable a setting of the respective importance of the MATCHES and the UNMATCHES for each use case individually. As a consequence, we define the *Certain Pairwise Decision Soundness* of a certain clustering $\mathcal{C}$ with respect to an incomplete clustering Γ as:

$$\textit{CPDecSound}(\mathcal{C}, \Gamma) \quad = \quad \frac{\alpha|M(\mathcal{C}) \cap M_c(\Gamma)| + \beta|U(\mathcal{C}) \cap U_c(\Gamma)|}{\alpha|M(\mathcal{C})| + \beta|U(\mathcal{C})|} \tag{B.3}$$

- **Certain Pairwise Decision Completeness:** The certainty completeness of a certain clustering $\mathcal{C}$ with respect to an incomplete clustering Γ can be rated by the amount of certain pairwise decisions in Γ that agree with any pairwise decision in $\mathcal{C}$. For the same reasons as discussed for the Certain Pairwise Decision Soundness we introduce the two weights $\alpha, \beta \in \mathbb{R}$. Consequently, the *Certain Pairwise Decision Completeness* of a certain clustering $\mathcal{C}$ with respect to an incomplete clustering Γ is defined as:

$$\textit{CPDecComp}(\mathcal{C}, \Gamma) \quad = \quad \frac{\alpha|M(\mathcal{C}) \cap M_c(\Gamma)| + \beta|U(\mathcal{C}) \cap U_c(\Gamma)|}{\alpha|M_c(\Gamma)| + \beta|U_c(\Gamma)|} \tag{B.4}$$

Naturally, the computed quality score of the certain information of Γ is the more representative, the more certainty sound and the more certainty complete the used Certain Clustering Component is. Therefore, in cases we do not prefer soundness to certainty or vice versa, we use the harmonic mean of certainty soundness and certainty completeness to rate the reliability of the Certain Clustering Component and hence to rate the reliability of the final quality score that has been computed for the certain information of Γ:

$$\textit{Reliability}(\mathsf{CCC}, \Gamma) \quad = \quad \frac{2 \times \textit{CPDecSound}(\mathsf{CCC}, \Gamma) \times \textit{CPDecComp}(\mathsf{CCC}, \Gamma)}{\textit{CPDecSound}(\mathsf{CCC}, \Gamma) + \textit{CPDecComp}(\mathsf{CCC}, \Gamma)} \tag{B.5}$$

Example 231 *For illustration we consider some Certain Clustering Components that can be constructed for the three sample probabilistic clusterings $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$. These components along with their Closest Cluster Recall, their Closest Cluster Precision, their Closest Cluster F_1-score, their information content, and their Certain Pairwise Decision Soundness as well as their Certain Pairwise Decision Completeness with respect to all three probabilistic clusterings are listed in Table B.2. It is obvious that a clustering is absolutely certainty sound (i.e. Certain Pairwise Decision Soundness is one) with respect to a probabilistic clustering if it is an element of its certain fact skyline, for example the clustering $\mathsf{CCC}_1 = \{\langle e_2\rangle, \langle e_4\rangle\}$ is absolutely certainty sound with respect to $\mathfrak{C}_4$. In contrast, none of the certain fact skylines' elements is absolutely certainty complete, because all skylines consist of more than one element. The clustering $\mathsf{CCC}_2 = \{\langle e_1, e_5\rangle, \langle e_2, e_6\rangle, \langle e_3\rangle, \langle e_4, e_8\rangle, \langle e_7\rangle\}$ is the only component that is absolutely certainty complete with respect to all three sample clusterings. Considering these candidates, CCC_2 is the most reliable Certain Clustering Component of all three sample clusterings because it has a reliability with respect to*

$\mathfrak{C}_2$ *of* 0.925*, has a reliability with respect to* $\mathfrak{C}_3$ *of* 0.883*, and has a reliability with respect to* $\mathfrak{C}_4$ *of* 0.701*. The Closest Cluster* F_1*-score of* $\mathfrak{CCC}_2$ *is* 0.85*. Thus by selecting the most reliable Certain Clustering Component, all three sample clusterings are rated by the same quality and hence we cannot distinguish them.*

For clusterings with large ranges, computing the certain fact skyline can be very consuming in time and in space, because it contains each certain UNMATCH for at least one time. As a consequence, it is useful to have an approach that directly constructs the Certain Clustering Component without explicitly computing the certain fact skyline first. We identified two of such approaches:

- **Certain Cluster Selection:** Let Γ be an incomplete clustering. By using the method of *Certain Cluster Selection* (short *CCSel*), the Certain Clustering Component of Γ is defined as the set of all certain clusters:
$$\mathfrak{CCC} = \mathit{CCSel}(\Gamma) = \bigcap \Gamma \tag{B.6}$$
- **Certain Subcluster Selection:** Let Γ be an incomplete clustering. By using the method of *Certain Subcluster Selection* (short *CSSel*), the Certain Clustering Component of Γ is defined as the set of all certain subclusters that are not dominated by any other certain subcluster. Consequently, it corresponds to the skyline of all certain subclusters:
$$\mathfrak{CCC} = \mathit{CSSel}(\Gamma) = \overline{\mathit{CSC}} \text{ where } \mathit{CSC} = \{C_i \mid \forall \mathcal{C} \in \Gamma \colon \exists C_j \in \mathcal{C} \colon C_i \subseteq C_j\} \tag{B.7}$$

Example 232 *For demonstration, the Certain Clustering Components that result from applying the Certain Cluster Selection and that result from applying the Certain Subcluster Selection to the sample probabilistic clusterings* $\mathfrak{C}_2$*,* $\mathfrak{C}_3$*, and* $\mathfrak{C}_4$ *are listed in Table B.1. Since some of the clusterings of the certain fact skyline of* $\mathfrak{C}_4$ *have disjoint ranges, the Certain Cluster Selection only returns an empty set for* $\mathfrak{C}_4$*. Interestingly, the Certain Cluster Selection as well as the Certain Subcluster Selection each produces the same Certain Clustering Component for* $\mathfrak{C}_2$ *and* $\mathfrak{C}_3$*, although the certain fact skylines of* $\mathfrak{C}_2$ *and* $\mathfrak{C}_3$ *are different (i.e.* $\overline{\mathit{cCF}(\mathfrak{C}_3)} \subset \overline{\mathit{cCF}(\mathfrak{C}_2)}$*).*

The Certain Pairwise Decision Soundness and the Certain Pairwise Decision Completeness of these Certain Clustering Components with respect to the different probabilistic sample clusterings are listed in Table B.2. The same holds for the decision correctness of these components. As we have already seen in Example 231, the Certain Clustering Components constructed by the Certain Subcluster Selection have the highest reliability for each of the corresponding probabilistic clusterings. In contrast, the Certain Clustering Components constructed by the Certain Cluster Selection are only less certainty complete. Nevertheless, this observation cannot be generalized because the sample clusterings are rather small and their alternatives have only less clusters in common. In real-life duplicate detection scenarios, however, these characteristics can be completely different.

Both selection methods have some interesting properties.

Theorem 65 *Let* Γ *be a cluster-disjoint incomplete clustering. The Certain Cluster Selection of* Γ *is equivalent to the intersection of all clusterings of the certain fact skyline of* Γ*, i.e.* $\mathit{CCSel}(\Gamma) = \bigcap \overline{\mathit{cCF}(\Gamma)}$.

Theorem 65 is proved by Proof 66 in Appendix B.4.

Theorem 66 *Let Γ be a cluster-disjoint incomplete clustering. The Certain Subcluster Selection of Γ is equivalent to the union of all clusterings of the certain fact skyline of Γ, i.e. $CSSel(\Gamma) = \bigcup \overline{cCF(\Gamma)}$.*

Theorem 66 is proved by Proof 67 in Appendix B.4.

Since the union of certain clusterings corresponds to their integration, the Certain Subcluster Selection corresponds exactly to the second extreme of constructing the Certain Clustering Component that we have mentioned above. Thus, the clustering $CSSel(\Gamma)$ is always cluster-disjoint and absolutely certainty complete.

Theorem 67 *Let Γ be a cluster-disjoint incomplete clustering. The clusterings $CCSel(\Gamma)$ and $CSSel(\Gamma)$ are both cluster-disjoint.*

Theorem 67 is proved by Proof 68 in Appendix B.4.

Due to Theorem 67, the Certain Clustering Component that is produced by the Certain Cluster Selection as well as the Certain Clustering Component that is produced by the Certain Subcluster Selection can be used as input to a conventional cluster-based measure for decision correctness.

Theorem 68 *Let Γ be a cluster-disjoint incomplete clustering. The clustering $CCSel(\Gamma)$ is absolutely certainty sound with respect to Γ, i.e. $CF(CCSel(\Gamma)) \subseteq cCF(\Gamma)$.*

Theorem 68 is proved by Proof 69 in Appendix B.4.

Theorem 69 *Let Γ be a cluster-disjoint incomplete clustering. The clustering $CSSel(\Gamma)$ is absolutely certainty complete with respect to Γ, i.e. $CF(CSSel(\Gamma)) \supseteq cCF(\Gamma)$.*

Theorem 69 is proved by Proof 70 in Appendix B.4.

Theorem 70 *Let Γ be a cluster-disjoint incomplete clustering. The clustering $CCSel(\Gamma)$ is only maximally certainty sound with respect to Γ, i.e. there is no other clustering that is absolutely certainty sound with respect to Γ and dominates $CCSel(\Gamma)$, iff the certain fact skyline of Γ contains a single element.*

Theorem 70 is proved by Proof 71 in Appendix B.4.

Example 233 *For instance, the clustering $CCSel(\mathfrak{C}_2)$ is absolutely certainty sound with respect to $\mathfrak{C}_2$, but the clusterings CF_{21}, CF_{22}, CF_{23}, and CF_{24} are absolutely certainty sound with respect to $\mathfrak{C}_2$ as well. Moreover, they dominate $CCSel(\mathfrak{C}_2)$. For this reason, $CCSel(\mathfrak{C}_2)$ is not maximally certainty sound with respect to $\mathfrak{C}_2$.*

The advantage of both selection approaches is obvious because we can compute a Certain Clustering Component that is either absolute certainty sound or that is absolute certainty complete without computing the certain fact skyline first. Of course, we have to compute each possible cluster (or each possible subcluster respectively), but this number is much smaller than the number of clustering facts because they only contain MATCHES and not UNMATCHES.

B.2.2. Multi-Clustering Approach

The idea of the multi-clustering approach is to consider each element of the certain fact skyline as an individual clustering, to apply a cluster-based measure for decision correctness to each of these clusterings separately and to finally aggregate the resultant quality scores to a single one. Obviously, more informative facts should have a greater impact on the aggregated quality score than less informative facts. For that reason, we weight the quality of each fact by its information content. As aggregation function we use the weighted average.

Let *QM* be a cluster-based measure for decision correctness, let Γ an incomplete clustering, and let $\mathcal{C}_{\text{gold}}$ be the corresponding gold standard. By using the multi-clustering approach we rate the decision correctness of the certain information of Γ as:

$$QM^{CWS}(\Gamma, \mathcal{C}_{\text{gold}}) \quad = \quad \frac{\sum_{\mathcal{C} \in \overline{cCF(\Gamma)}} IC(\mathcal{C}) \times QM(\mathcal{C}, \pi_{rng(\mathcal{C})}(\mathcal{C}_{\text{gold}}))}{\sum_{\mathcal{C} \in \overline{CCF(\Gamma)}} IC(\mathcal{C})} \tag{B.8}$$

Optionally, instead of using the elements of the certain fact skyline, we can also aggregate the decision correctness of several Certain Clustering Components each weighted by their information content and their reliabilities.

Example 234 *Whereas all other clusterings of the certain fact skylines of* $\mathfrak{C}_2$*,* $\mathfrak{C}_3$ *and* $\mathfrak{C}_4$ *have the optimal Closest Cluster* F_1*-score of* 1.0*, the clustering* CF_{22} *of the certain fact skyline of* $\mathfrak{C}_2$ *has only a Closest Cluster* F_1*-score of* 0.74*. Consequently, the average Closest Cluster* F_1*-score of the certain fact skyline of* $\mathfrak{C}_3$ *and the average Closest Cluster* F_1*-score of the certain fact skyline of* $\mathfrak{C}_4$ *each results in* 1.0 *and the average Closest Cluster* F_1*-score of the certain fact skyline of* $\mathfrak{C}_2$ *only results in* 0.937*. Thus, as in the pair-based interpretation, the certain information of the clusterings* $\mathfrak{C}_3$ *and* $\mathfrak{C}_4$ *is rated to be absolutely decision correct and the certain information of the clustering* $\mathfrak{C}_2$ *contains some errors.*

The problem with this approach is that clusterings of the certain fact skyline can overlap and hence we weight the decision correctness of some clustering information more than the decision correctness of some other clustering information which can distort the reliability of the aggregated quality score. An exact computation of all overlaps, however, is usually extremely expensive.

B.2.3. Conclusion

As we can conclude from the above discussions, both cluster-based approaches (single-clustering as well as multi-clustering) are only suitable for quality computation to a limited extent because they are expensive to compute and the resultant scores are often only less reliable. Nevertheless, if a cluster-based measure for decision correctness is required, one of both approaches need to be used. In contrast, if such a need is not given, we recommend to use a pair-based measure for rating the decision correctness of a clustering's certain information.

B.2.4. Relaxing the Meaning of Certainty

Sometimes it can be useful to relax the requirement of absolute certainty by considering all clustering information to be certain that is more probable than $1-\tau$. As we have presented in Section 8.8.2.4, in the

pair-based interpretation such an extension does not pose a problem. In the cluster-based interpretation, however, such a relaxing complicates the definition and the computation of the certain fact skyline as well as the construction of a Certain Clustering Component enormously because some of the made theorems are not valid anymore. For example, Theorem 62 becomes invalid in the presence of tolerance, due to an integration of certain facts can result in an uncertain fact as illustrated in the following example.

Possible Clustering	**Prob**
$\mathcal{C}_1 = \{\langle e_1, e_2, e_3\rangle\}$	0.7
$\mathcal{C}_2 = \{\langle e_1, e_2\rangle, \langle e_3\rangle\}$	0.1
$\mathcal{C}_3 = \{\langle e_1, e_3\rangle, \langle e_2\rangle\}$	0.1
$\mathcal{C}_4 = \{\langle e_1\rangle, \langle e_2, e_3\rangle\}$	0.1

(a) The probabilistic clustering $\mathfrak{C}$

Clustering Fact	**Certain?**
$CF_1 = \{\langle e_1, e_2\rangle\}$	YES
$CF_2 = \{\langle e_1, e_3\rangle\}$	YES
$CF_3 = \{\langle e_2, e_3\rangle\}$	YES
$CF_4 = \{\langle e_1, e_2, e_3\rangle\}$	FALSE

(b) Fact certainty for $0.2 < \tau \leq 0.3$

Clustering Fact	**Certain?**
$CF_1 = \{\langle e_1, e_2\rangle\}$	YES
$CF_2 = \{\langle e_1, e_3\rangle\}$	YES
$CF_3 = \{\langle e_2, e_3\rangle\}$	YES
$CF_4 = \{\langle e_1, e_2, e_3\rangle\}$	YES

(c) Fact certainty for $\tau > 0.3$

Figure B.1.: Sample situation where a clustering fact is uncertain although all its subfacts are certain

Example 235 *Let $\mathfrak{C} = (\Gamma, Pr)$ be a probabilistic clustering with four alternatives $\Gamma = \{\mathcal{C}_1, \ldots, \mathcal{C}_4\}$ as presented in Figure B.1(a). All three entity pairs $\{e_1, e_2\}$, $\{e_1, e_3\}$, and $\{e_2, e_3\}$ (and hence the corresponding facts) are certain* MATCHES *if we use a tolerance $\tau > 0.2$, but the clustering fact $\{\langle e_1, e_2, e_3\rangle\}$ is only certain if we use a tolerance $\tau > 0.3$. Consequently, given a tolerance $0.2 < \tau \leq 0.3$, all three pairs are considered to be certain* MATCHES*, but the clustering that contains these elementary facts is not a certain fact. As a consequence, a clustering fact can be uncertain although all of its subfacts are certain.*

In such a situation the difference between the pair-based interpretation and the cluster-based interpretation becomes more clear because the pair-based interpretation does not distinguish between the two tolerance scenarios that are presented in Figure B.1(b) and Figure B.1(c). In contrast, the cluster-based interpretation makes a difference because it does not restrict to elementary clustering facts, and hence is more correct than the pair-based interpretation. This advantage, however, is a disadvantage at the same time because its increases the complexity of computing the certain fact skyline and increases the difficulty of constructing the Certain Clustering Component.

A direct consequence from the invalidity of Theorem 62 is that the set of all certain facts of a probabilistic clustering is not equivalent to the result of integrating all the sets of certain facts of its factors anymore (see Theorem 71 in Section B.3), because the probability of the integration result is the multiplication of the probabilities of the integration input and hence integrating two clusterings with a probability greater than $1-\tau$ can lead to a clustering with probability lower than $1-\tau$. Consequently, a factor-based computation of the certain fact skyline or the Certain Clustering Component becomes more difficult. For all these reasons, we abstain from relaxing certainty in this thesis.

B.3. Factor-based Computation

In the cluster-based interpretation a factor-based quality computation is a little bit more complicated than in the pair-based interpretation. We start with the certain clustering facts. A clustering is a certain fact of

an incomplete clustering, if all its factors are certain facts of the corresponding factors of the incomplete clustering.

Theorem 71 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. A clustering $\mathcal{C}$ is a certain fact of Γ, iff for each factor set $F \in \mathcal{F}^{set}$ the clustering $\pi_F(\mathcal{C})$ is a certain fact of Γ_F, i.e. $\mathcal{C} \in cCF(\Gamma) \Leftrightarrow \forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in cCF(\Gamma_F)$.*

Theorem 71 is proved by Proof 73 in Section B.4.

Since for all $\mathcal{C} \in \Gamma$ it holds that $\mathcal{C} = \boxtimes_{i=1}^{k} \pi_{F_i}(\mathcal{C})$, we can conclude that $cCF(\Gamma) \not\supset \bigotimes_{i=1}^{k} cCF(\Gamma_{F_i})$ because $\mathcal{C} \in cCF(\Gamma) \Rightarrow \forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in cCF(\Gamma_F)$ and we can conclude that $cCF(\Gamma) \not\subset \bigotimes_{i=1}^{k} cCF(\Gamma_{F_i})$ because $\forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in cCF(\Gamma_F) \Rightarrow \mathcal{C} \in cCF(\Gamma)$. As a consequence, the certain facts of the incomplete clustering Γ can be computed by integrating the certain facts of its factors, i.e. $cCF(\Gamma) = \bigotimes_{i=1}^{k} cCF(\Gamma_{F_i})$.

A clustering belongs to the certain fact skyline of an incomplete clustering if its projection on a factor set belongs to the certain fact skyline of the corresponding factor. Thus, the certain fact skyline of an incomplete clustering can be computed in a factor based fashion.

Theorem 72 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. A clustering $\mathcal{C}$ is an element of the certain fact skyline of Γ, iff for each factor set $F \in \mathcal{F}^{set}$ the clustering $\pi_F(\mathcal{C})$ is an element of the certain fact skyline of Γ_F, i.e. $\mathcal{C} \in \overline{cCF(\Gamma)} \Leftrightarrow \forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in \overline{cCF(\Gamma_F)}$.*

Theorem 72 is proved by Proof 74 in Section B.4.

Since for all $\mathcal{C} \in \Gamma$ it holds that $\mathcal{C} = \boxtimes_{i=1}^{k} \pi_{F_i}(\mathcal{C})$, we can conclude that $\overline{cCF(\Gamma)} \not\supset \bigotimes_{i=1}^{k} \overline{cCF(\Gamma_{F_i})}$ because $\mathcal{C} \in \overline{cCF(\Gamma)} \Rightarrow \forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in \overline{cCF(\Gamma_F)}$ and we can conclude that $\overline{cCF(\Gamma)} \not\subset \bigotimes_{i=1}^{k} \overline{cCF(\Gamma_{F_i})}$ because $\forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in \overline{cCF(\Gamma_F)} \Rightarrow \mathcal{C} \in \overline{cCF(\Gamma)}$. Thus, the certain fact skyline of the incomplete clustering Γ can be computed by integrating the certain fact skylines of its factors, i.e. $\overline{cCF(\Gamma)} = \bigotimes_{i=1}^{k} \overline{cCF(\Gamma_{F_i})}$.

As a consequence, the certain fact skyline of an uncertain clustering can be efficiently computed if this clustering is factorized into small factors.

Example 236 *For illustration, we reconsider the sample clustering $\mathfrak{C}$ and its factorization into $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$ as presented in Figure 8.33. The certain fact skylines of all three probabilistic clusterings are presented in Table B.3. It is simply to see that for each possible combination of elements of the certain fact skylines of $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$ there is an element in the certain fact skyline of $\mathfrak{C}$ that results by integrating them. For instance, the certain fact CF_1 results from integrating the certain facts $CF_{(F_1)1}$ and $CF_{(F_2)1}$. Due to the set $\overline{cCF(\mathfrak{C}_{F_1})}$ has two elements and the set $\overline{cCF(\mathfrak{C}_{F_2})}$ has twelve elements, the skyline $\overline{cCF(\mathfrak{C})}$ has twenty-four elements.*

The Certain Cluster Selection is restricted to single clusters and hence can be computed in a factor-based fashion.

$\overline{cCF}$ of factor $\mathfrak{C}_{F_1}$
$CF_{(F_1)1} = \{\langle e_1, e_2\rangle, \langle e_7\rangle\}$
$CF_{(F_1)2} = \{\langle e_1, e_2\rangle, \langle e_8\rangle\}$

$\overline{cCF}$ of factor $\mathfrak{C}_{F_2}$	
$CF_{(F_2)1} = \{\langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_6\rangle\}$	$CF_{(F_2)2} = \{\langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_{12}\rangle\}$
$CF_{(F_2)3} = \{\langle e_9\rangle, \langle e_4, e_{10}\rangle, \langle e_6\rangle\}$	$CF_{(F_2)4} = \{\langle e_9\rangle, \langle e_4, e_{10}\rangle, \langle e_{12}\rangle\}$
$CF_{(F_2)5} = \{\langle e_3\rangle, \langle e_5\rangle, \langle e_6\rangle\}$	$CF_{(F_2)6} = \{\langle e_3\rangle, \langle e_5\rangle, \langle e_{12}\rangle\}$
$CF_{(F_2)7} = \{\langle e_9\rangle, \langle e_5\rangle, \langle e_6\rangle\}$	$CF_{(F_2)8} = \{\langle e_9\rangle, \langle e_5\rangle, \langle e_{12}\rangle\}$
$CF_{(F_2)9} = \{\langle e_3\rangle, \langle e_{11}\rangle, \langle e_6\rangle\}$	$CF_{(F_2)10} = \{\langle e_3\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$
$CF_{(F_2)11} = \{\langle e_9\rangle, \langle e_{11}\rangle, \langle e_6\rangle\}$	$CF_{(F_2)12} = \{\langle e_9\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$

$\overline{cCF}$ of clustering $\mathfrak{C}$	
$CF_1 = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_6\rangle\}$	$CF_2 = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_6\rangle\}$
$CF_3 = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_{12}\rangle\}$	$CF_4 = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_{12}\rangle\}$
$CF_5 = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_9\rangle, \langle e_4, e_{10}\rangle, \langle e_6\rangle\}$	$CF_6 = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_9\rangle, \langle e_4, e_{10}\rangle, \langle e_6\rangle\}$
$CF_7 = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_9\rangle, \langle e_4, e_{10}\rangle, \langle e_{12}\rangle\}$	$CF_8 = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_9\rangle, \langle e_4, e_{10}\rangle, \langle e_{12}\rangle\}$
$CF_9 = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_3\rangle, \langle e_5\rangle, \langle e_6\rangle\}$	$CF_{10} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_3\rangle, \langle e_5\rangle, \langle e_6\rangle\}$
$CF_{11} = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_3\rangle, \langle e_5\rangle, \langle e_{12}\rangle\}$	$CF_{12} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_3\rangle, \langle e_5\rangle, \langle e_{12}\rangle\}$
$CF_{13} = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_9\rangle, \langle e_5\rangle, \langle e_6\rangle\}$	$CF_{14} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_9\rangle, \langle e_5\rangle, \langle e_6\rangle\}$
$CF_{15} = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_9\rangle, \langle e_5\rangle, \langle e_{12}\rangle\}$	$CF_{16} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_9\rangle, \langle e_5\rangle, \langle e_{12}\rangle\}$
$CF_{17} = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_3\rangle, \langle e_{11}\rangle, \langle e_6\rangle\}$	$CF_{18} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_3\rangle, \langle e_{11}\rangle, \langle e_6\rangle\}$
$CF_{19} = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_3\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$	$CF_{20} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_3\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$
$CF_{21} = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_9\rangle, \langle e_{11}\rangle, \langle e_6\rangle\}$	$CF_{22} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_9\rangle, \langle e_{11}\rangle, \langle e_6\rangle\}$
$CF_{23} = \{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_9\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$	$CF_{24} = \{\langle e_1, e_2\rangle, \langle e_8\rangle, \langle e_9\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$

Table B.3.: The certain fact skylines of the sample clustering $\mathfrak{C}$ and its two factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$

Theorem 73 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ, the set $\{CCSel(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ is a range-disjoint factorization of the certain clustering $CCSel(\Gamma)$.*

Theorem 73 is proved by Proof 75 in Section B.4.

From Theorem 73 follows that the Certain Cluster Selection of an incomplete clustering Γ that is given by the factorization $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ can be computed by integrating the Certain Cluster Selections of each of its factors, i.e. $CCSel(\Gamma) = \bigboxtimes_{i\in\{1,\ldots,k\}} CCSel(\Gamma_{F_i})$.

Similar holds for the Certain Subcluster Selection. If the factorization is range-disjoint, a cluster can only be the subcluster of another cluster from the same factor. Thus, the Certain Subcluster Selection can be computed in a factor-based fashion.

Theorem 74 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ, the set $\{CSSel(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ is a range-disjoint factorization of the certain clustering $CSSel(\Gamma)$.*

Theorem 74 is proved by Proof 76 in Section B.4.

Clustering	*CCSel*	*CSSel*
$\mathfrak{C}_{F1}$	$\{\langle e_1, e_2\rangle\}$	$\{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_8\rangle\}$
$\mathfrak{C}_{F2}$	$\emptyset$	$\{\langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_5\rangle, \langle e_6\rangle, \langle e_9\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$
$\mathfrak{C}$	$\{\langle e_1, e_2\rangle\}$	$\{\langle e_1, e_2\rangle, \langle e_7\rangle, \langle e_8\rangle, \langle e_3\rangle, \langle e_4, e_{10}\rangle, \langle e_5\rangle, \langle e_6\rangle, \langle e_9\rangle, \langle e_{11}\rangle, \langle e_{12}\rangle\}$

Table B.4.: The Certain Cluster Selection and the Certain Subcluster Selection of the sample clustering $\mathfrak{C}$ and its two factors $\mathfrak{C}_{F1}$ and $\mathfrak{C}_{F2}$

From Theorem 74 follows that the Certain Subcluster Selection of an incomplete clustering Γ that is given by the factorization $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ can be computed by integrating the Certain Subcluster Selections of each of its factors, i.e. $\mathit{CSSel}(\Gamma) = \bigboxtimes_{i\in\{1,\ldots,k\}} \mathit{CSSel}(\Gamma_{F_i})$.

As a consequence, the Certain Cluster Selection as well as the Certain Subcluster Selection can be computed based on the individual factors and hence can be computed efficiently if the factors are small.

Example 237 *For illustration we reconsider the sample factorization that is presented in Figure 8.33. The Certain Cluster Selection and the Certain Subcluster Selection of the probabilistic clustering $\mathfrak{C}$ as well as its two factors $\mathfrak{C}_{F_1}$ and $\mathfrak{C}_{F_2}$ are presented in Table B.4. The Certain Cluster Selection of the factor $\mathfrak{C}_{F_1}$ contains a single 2-element cluster. In contrast, the Certain Cluster Selection of the factor $\mathfrak{C}_{F_2}$ is the empty set. The Certain Subcluster Selection of $\mathfrak{C}_{F_1}$ contains three clusters and the Certain Subcluster Selection of $\mathfrak{C}_{F_2}$ contains seven clusters. It is easy to see that the Certain Cluster Selection (or the Certain Subcluster Selection respectively) of $\mathfrak{C}$ results in the union of the Certain Cluster Selection (or the Certain Subcluster Selection respectively) of its factors.*

B.4. Proofs

In this section, we prove the theorems of Appendix B.

Proof of Theorem 60

Theorem 60 *The relation of being a clustering fact is transitive, i.e if $\mathcal{C}'$ is a clustering fact of $\mathcal{C}$ and $\mathcal{C}^*$ is a clustering fact of $\mathcal{C}'$, $\mathcal{C}^*$ is also a clustering fact of $\mathcal{C}$.*

Theorem 60 can be proved as follows:

Proof 61 *This can be simply proven, because there must be two sets S^* and S' so that $\mathcal{C}^* = \pi_{S^*}(\mathcal{C}') = \pi_{S^*}(\pi_{S_2}(\mathcal{C}))$ and the range of $\mathcal{C}^*$, i.e. S^*, must be a subset of the range of $\mathcal{C}'$, i.e. S', that in turn must be a subset of the range of $\mathcal{C}$ it holds that $S^* \subseteq S' \subseteq rng(\mathcal{C})$ and $\pi_{S^*}(\pi_{S'}(\mathcal{C})) = \pi_{S^*}(\mathcal{C})$.*

Proof of Theorem 61

Theorem 61 *Let $\mathcal{C}$ be a cluster-disjoint clustering and let X be an arbitrary set of the same domain as $rng(\mathcal{C})$. Projecting the clustering fact set of $\mathcal{C}$ on X is equivalent to the clustering fact set of the projection of $\mathcal{C}$ on X, i.e. $\pi_X(\mathit{CF}(\mathcal{C})) = \mathit{CF}(\pi_X(\mathcal{C}))$.*

Theorem 61 can be proved as follows:

Proof 62

$$\begin{aligned}\pi_X(CF(\mathcal{C})) &= \{\pi_X(CF) \mid CF \in CF(\mathcal{C})\} \\ &= \{\pi_X(\pi_S(\mathcal{C})) \mid S \subseteq rng(\mathcal{C})\} \\ &= \{\pi_S(\pi_X(\mathcal{C})) \mid S \subseteq rng(\mathcal{C})\} \\ &= CF(\pi_X(\mathcal{C}))\end{aligned}$$

Proof of Theorem 62

Theorem 62 *Let Γ be an incomplete clustering. A clustering fact $\mathcal{C} \in CF(\Gamma)$ is certain, iff all of its subfacts are certain facts of Γ , i.e. $\mathcal{C} \in cCF(\Gamma) \Leftrightarrow CF(\mathcal{C}) \subseteq cCF(\Gamma)$.*

Theorem 62 can be proved as follows:

Proof 63 *($\Rightarrow$) Assumption 1: $\mathcal{C}$ is a certain fact of Γ, but its subfact $\mathcal{C}^* \in CF(\mathcal{C})$ is not a certain fact of Γ.*
$\mathcal{C}^*$ is not a certain fact of Γ.
$\Rightarrow$ there is an alternative $\mathcal{C}'$ of Γ that does not dominate $\mathcal{C}^*$.
$\Rightarrow$ $\mathcal{C}'$ does not dominate $\mathcal{C}$.
$\Rightarrow$ $\mathcal{C}$ is not a certain fact of Γ which is a contradiction to our assumption.
Consequently, if $\mathcal{C}$ is a certain fact of Γ, all its subfacts are certain facts of Γ as well.

($\Leftarrow$) Assumption 2: $\mathcal{C}$ is not a certain fact of Γ, but all its subfacts are certain facts of Γ.
$\mathcal{C}$ is not a certain fact of Γ.
$\Rightarrow$ there is an alternative $\mathcal{C}'$ of Γ that does not dominate $\mathcal{C}$, i.e. $\mathcal{C} \not\prec \mathcal{C}'$.
$\Rightarrow$ there is an entity pair $\{e_i, e_j\}$ that is a MATCH *in $\mathcal{C}$ but an* UNMATCH *in $\mathcal{C}'$ or that is an* UNMATCH *in $\mathcal{C}$ but a* MATCH *in $\mathcal{C}'$.*
$\Rightarrow$ the entity pair $\{e_i, e_j\}$ is not a certain fact of Γ which is a contradiction to our assumption that all subfacts of $\mathcal{C}$ are certain facts of Γ.
Consequently, if all its subfacts are certain facts of Γ, $\mathcal{C}$ is a certain fact of Γ as well.

Proof of Theorem 63

Theorem 63 *Let Γ be an incomplete clustering. The certainty of an n-element fact $\mathcal{C} \in CF(\Gamma)$ can be checked by checking the certainty of two of its $(n-1)$-element subfacts and by checking the certainty of one of its elementary subfacts.*

Theorem 63 can be proved as follows:

Proof 64 *Following Theorem 62 a clustering is a certain fact of* Γ *if all its elementary subfacts are certain duplicate decisions of* Γ*. Let* $\mathcal{C}$ *be a clustering with the range* $\{e_1, \ldots, e_k\}$ *and let* $\mathcal{C}'$ *be the subfact of* $\mathcal{C}$ *that has the range* $\{e_1, \ldots, e_{k-1}\}$ *If* $\mathcal{C}'$ *is a certain fact of* Γ*, all entity pairs in Pairs*$(\mathcal{C}')$ *are certain duplicatedecisions of* Γ*. Thus, we only need to check whether the entity pairs* $\{\{e_1, e_k\}, \ldots, \{e_{k-1}, e_k\}\}$ *are certain duplicate decisions of* Γ*. The certainty of the last* $k-2$ *pairs is implicated by the certainty of the* $(n-1)$*-element subfact of* $\mathcal{C}$ *that is named as* $\mathcal{C}^*$ *and has the range* $\{e_2, \ldots, e_k\}$*. Consequently, after checking the certainty of* $\mathcal{C}'$ *and* $\mathcal{C}^*$ *we only need to check the certainty of the dduplicate decision on the entity pair* $\{e_1, e_k\}$*. Therefore, the certainty of an* n*-element fact can be checked by checking the certainty of two its* $(n-1)$*-element subfacts and by checking the certainty of one of its elementary subfacts.*

Proof of Theorem 64

Theorem 64 *Let* Γ *be a cluster-disjoint incomplete clustering. Integrating some elements of the certain fact skyline of* Γ *always results in a cluster-disjoint clustering.*

Theorem 64 can be proved as follows:

Proof 65 *Let* $S = \{CF_1, \ldots, CF_k\}$ *be a set of clustering that belong to the certain fact skyline of* Γ*. Each clustering of the certain fact skyline of* Γ *is per definition cluster-disjoint if* Γ *is cluster-disjoint. Thus, the clustering* $\boxtimes_{i=1}^{k} CF_i$ *can only be not cluster-disjoint, if an entity of* $rng(\bigcup_{i=1}^{k} rng(CF_i))$ *belongs to different clusters in different clusterings of* S*.*
Assumption: There are two clusterings $\mathcal{C}$ *and* $\mathcal{C}'$ *in* $S \subseteq \overline{cCF(\Gamma)}$ *so that* $C \in \mathcal{C}$*,* $C' \in \mathcal{C}'$ *and the clusters* C *and* C' *are not equivalent but overlap.*
$\Rightarrow$ *the entities in* C *belong to the same cluster in each alternative of* Γ *and the entities in* C' *belong to the same cluster in each alternative of* Γ*.*
$\Rightarrow$ *there is an entity that is in the same cluster as the entities of* C *in each alternative of* Γ *and that is in the same cluster as the entities of* C' *in each alternative of* Γ*.*
$\Rightarrow$ *the entities of* C *and the entities of* C' *are in the same cluster in each alternative of* Γ*.*
$\Rightarrow$ *the clustering* $\{C \cup C'\}$ *is a certain fact of* Γ*.*
$\Rightarrow$ *the clusterings* $\mathcal{C}^* = \mathcal{C} - C \cup \{C \cup C'\}$ *and* $\mathcal{C}^{\#} = \mathcal{C}' - C' \cup \{C \cup C'\}$ *must belong to cCF*(Γ)*.*
The clustering $\mathcal{C}$ *is dominated by* $\mathcal{C}^*$ *and the clustering* $\mathcal{C}'$ *is dominated by* $\mathcal{C}^{\#}$*.*
$\Rightarrow$ *the clusterings* $\mathcal{C}$ *and* $\mathcal{C}'$ *cannot belong to the certain fact skyline (and hence cannot belong to* S*) which is a contradiction to our assumption.*
Consequently, two distinct clusterings of the certain fact skyline of a cluster-disjoint incomplete clustering cannot have non-equivalent but overlapping clusters and hence the clustering $\boxtimes_{i=1}^{k} CF_i$ *is cluster-disjoint.*

Proof of Theorem 65

Theorem 65: *Let Γ be a cluster-disjoint incomplete clustering. The Certain Cluster Selection of Γ is equivalent to the intersection of all clusterings of the certain fact skyline of Γ, i.e. $CCSel(\Gamma) = \bigcap \overline{cCF(\Gamma)}$.*

Theorem 65 can be proved as follows:

Proof 66 *Assumption 1: $CCSel(\Gamma)$ misses a cluster C that is in $\bigcap \overline{cCF(\Gamma)}$.*
C belongs to $\bigcap \overline{cCF(\Gamma)}$.
$\Rightarrow$ the entities in C are certainly duplicates and are certainly duplicates with no other entities.
$\Rightarrow$ the entities in C are in the same cluster in every alternative of Γ and they are not in the same cluster with any other entity in any alternative of Γ.
$\Rightarrow$ C belongs to every alternative of Γ.
$\Rightarrow$ C belongs to $CCSel(\Gamma)$ which is a contradiction to out initial assumption.
Consequently, $CCSel(\Gamma)$ contains each cluster of $\bigcap \overline{cCF(\Gamma)}$, i.e. $CCSel(\Gamma) \supseteq \bigcap \overline{cCF(\Gamma)}$.

Assumption 2: $CCSel(\Gamma)$ contains a cluster C that is not in $\bigcap \overline{cCF(\Gamma)}$.
C belongs to $CCSel(\Gamma)$.
$\Rightarrow$ the cluster belongs to every alternative of Γ.
$\Rightarrow$ the clustering $\mathcal{C} = \{C\}$ is a certain fact of Γ.
$\Rightarrow$ there is a clustering $\mathcal{C}' \in \overline{cCF(\Gamma)}$ that dominates $\mathcal{C}$.
$\Rightarrow$ there is a clustering $\mathcal{C}' \in \overline{cCF(\Gamma)}$ that contains C or that contains a cluster $C' \supset C$.
The first case is a contradiction to the initial assumption.
$\Rightarrow$ the clustering $\mathcal{C}'$ contains a cluster $C' \supset C$.
$\Rightarrow$ the cluster C' is a certain fact.
$\Rightarrow$ the cluster C' must be a subcluster of another cluster in every alternative of Γ.
$\Rightarrow$ the cluster $C \subset C'$ cannot belong to any alternative of Γ which is a contradiction to the initial assumption that C belongs to $CCSel(\Gamma)$.
Consequently, $CCSel(\Gamma)$ contains no cluster that does not belong to $\bigcap \overline{cCF(\Gamma)}$, i.e. $CCSel(\Gamma) \subseteq \bigcap \overline{cCF(\Gamma)}$
From $CCSel(\Gamma) \supseteq \bigcap \overline{cCF(\Gamma)}$ (Assumption 1) and $CCSel(\Gamma) \subseteq \bigcap \overline{cCF(\Gamma)}$ (Assumption 2) follows that $CCSel(\Gamma)$ is equivalent to $\bigcap \overline{cCF(\Gamma)}$.

Proof of Theorem 66

Theorem 66: *Let Γ be a cluster-disjoint incomplete clustering. The Certain Subcluster Selection of Γ is equivalent to the union of all clusterings of the certain fact skyline of Γ, i.e. $CSSel(\Gamma) = \bigcup \overline{cCF(\Gamma)}$.*

Theorem 66 can be proved as follows:

Proof 67 *Assumption 1: CSSel(Γ) misses a cluster C that is in $\bigcup\overline{cCF(\Gamma)}$.*
C belongs to $\bigcup\overline{cCF(\Gamma)}$.
$\Rightarrow$ the cluster is a certain fact of Γ.
$\Rightarrow$ the cluster's entities belong to the same cluster in every alternative of Γ.
$\Rightarrow$ the cluster is a certain subcluster in Γ.
$\Rightarrow$ the cluster is in CSSel(Γ) or there is a cluster C' in CSSel(Γ) that dominates C.
The first case is a contradiction to the initial assumption.
$\Rightarrow$ there is a cluster C' in CSSel(Γ) that dominates C.
$\Rightarrow$ for every clustering $\mathcal{C} \in \overline{cCF(\Gamma)}$ that contains C there is a clustering $\mathcal{C}' = \mathcal{C} - C \cup C'$ that is a certain fact of Γ as well.
$\Rightarrow$ $\mathcal{C}'$ is a certain fact of Γ and dominates $\mathcal{C}$.
$\Rightarrow$ $\mathcal{C}$ cannot belong to the certain fact skyline which is a contradiction to the initial assumption that there is a clustering in $\overline{cCF(\Gamma)}$ that contains the cluster C.
Consequently, CSSel(Γ) contains each cluster of $\bigcup\overline{cCF(\Gamma)}$, i.e. CSSel($\Gamma$) $\supseteq \bigcup\overline{cCF(\Gamma)}$.

Assumption 2: CSSel(Γ) contains a cluster C that is not in $\bigcup\overline{cCF(\Gamma)}$.
C belongs to CSSel(Γ).
$\Rightarrow$ the cluster is a subcluster of another cluster in any alternative of Γ.
$\Rightarrow$ the cluster is a certain fact of Γ.
$\Rightarrow$ the clustering $\mathcal{C} = \{C\}$ must be dominated by at least one clustering of the certain fact skyline of Γ.
$\Rightarrow$ the cluster belongs to any clustering of $\overline{cCF(\Gamma)}$ or there is a cluster C' in any clustering of $\overline{cCF(\Gamma)}$ that is a superset of C.
The first case is a contradiction to the initial assumption.
$\Rightarrow$ the cluster C' is certain and is a superset of C which is a contradiction to our assumption that C belongs to CSSel(Γ).
Consequently, CSSel(Γ) has no cluster that does not belong to $\bigcup\overline{cCF(\Gamma)}$, i.e. CSSel($\Gamma$) $\subseteq \bigcup\overline{cCF(\Gamma)}$.

From CSSel(Γ) $\supseteq \bigcup\overline{cCF(\Gamma)}$ (Assumption 1) and CSSel(Γ) $\subseteq \bigcup\overline{cCF(\Gamma)}$ (Assumption 2) follows that CSSel(Γ) is equivalent to $\bigcup\overline{cCF(\Gamma)}$.

Proof of Theorem 67

Theorem 67: *Let Γ be a cluster-disjoint incomplete clustering. The clusterings CCSel(Γ) and CSSel(Γ) are both cluster-disjoint.*

Theorem 67 can be proved as follows:

Proof 68 *The clustering CCSel(Γ) is a fact of each clustering of the certain fact skyline of Γ and hence is a certain fact of Γ. Since Γ is cluster-disjoint, its certain facts are cluster-disjoint as well and therefore the clustering CCSel(Γ) is cluster-disjoint.*

Each clustering of the certain fact skyline of Γ *is per definition cluster-disjoint if* Γ *is cluster-disjoint. Moreover, following Theorem 64 the integration of all clustering of the certain fact skyline is cluster-disjoint as well. Since the clustering CSSel*(Γ) *is the union (and hence the integration) of all clusterings of the certain fact skyline, it is a cluster-disjoint clustering.*

Proof of Theorem 68

Theorem 68: *Let* Γ *be a cluster-disjoint incomplete clustering. The clustering CCSel*(Γ) *is absolutely certainty sound with respect to* Γ*, i.e.* $CF(CCSel(\Gamma)) \subseteq cCF(\Gamma)$.

Theorem 68 can be proved as follows:

Proof 69 *The clustering CCSel*(Γ) *is a fact of every clustering of the certain fact skyline.*
The elements of the certain fact skyline are per definition absolutely certainty sound with respect to Γ *and a clustering is only absolutely certainty sound with respect to* Γ *if all its subfacts are absolutely certainty sound with respect to* Γ.
Consequently, the clustering CCSel(Γ) *is absolutely certainty sound with respect to* Γ.

Proof of Theorem 69

Theorem 69: *Let* Γ *be a cluster-disjoint incomplete clustering. The clustering CSSel*(Γ) *is absolutely certainty complete with respect to* Γ*, i.e.* $CF(CSSel(\Gamma)) \supseteq cCF(\Gamma)$.

Theorem 69 can be proved as follows:

Proof 70 *The set* $cCF(\Gamma)$ *is completely represented by the certain fact skyline, i.e. for every clustering* $\mathcal{C} \in cCF(\Gamma)$ *there is a clustering* $\mathcal{C}' \in \overline{cCF(\Gamma)}$ *that dominates* $\mathcal{C}$.
Thus, a clustering dominates all certain facts of Γ *if it dominates all clusterings of the certain fact skyline of* Γ.
The clustering CSSel(Γ) *is equal to the integration of all clusterings of the certain fact skyline of* Γ.
$\Rightarrow$ *the clustering CSSel*(Γ) *dominates all clusterings of the certain fact skyline of* Γ.
$\Rightarrow$ *the clustering CSSel*(Γ) *dominates all certain facts of* Γ.
$\Rightarrow$ *the clustering CSSel*(Γ) *is absolutely certainty complete with respect to* Γ.

Proof of Theorem 70

Theorem 70: *Let Γ be a cluster-disjoint incomplete clustering. The clustering CCSel(Γ) is only maximally certainty sound with respect to Γ, i.e. there is no other clustering that is absolutely certainty sound with respect to Γ and dominates CCSel(Γ), iff the certain fact skyline of Γ contains a single element.*

Theorem 70 can be proved as follows:

Proof 71 *($\Rightarrow$) Assumption 1: CCSel(Γ) is maximally certainty sound with respect to Γ, but $\overline{cCF(\Gamma)}$ contains more than one element.*
The set $\overline{cCF(\Gamma)}$ contains more than one element.
$\Rightarrow$ the clustering CCSel(Γ) $= \bigcap \overline{cCF(\Gamma)}$ is not equivalent to any clustering in $\overline{cCF(\Gamma)}$ but is dominated by every clustering in $\overline{cCF(\Gamma)}$.
Each clustering of the certain fact skyline is per definition absolutely certainty sound with respect to Γ.
$\Rightarrow$ every clustering in $\overline{cCF(\Gamma)}$ is absolutely certainty sound with respect to Γ and dominates CCSel(Γ).
$\Rightarrow$ the clustering CCSel(Γ) is not maximally certainty sound with respect to Γ which is a contradiction to the initial assumption.
Consequently, if the clustering CCSel(Γ) is maximally certainty sound with respect to Γ, the certain fact skyline contains a single element.

($\Leftarrow$) Assumption 2: $\overline{cCF(\Gamma)}$ contains exact one element, but CCSel(Γ) is not maximally certainty sound with respect to Γ.
The set $\overline{cCF(\Gamma)}$ contains exact one element.
$\Rightarrow$ the clustering CCSel(Γ) $= \bigcap \overline{cCF(\Gamma)}$ is the single element in $\overline{cCF(\Gamma)}$.
CCSel(Γ) is not maximally certainty sound with respect to Γ.
$\Rightarrow$ there must be a clustering $\mathcal{C}$ that dominates CCSel(Γ) and that is absolutely certainty sound with respect to Γ, i.e. it is a certain fact of Γ.
$\Rightarrow$ $\mathcal{C}$ must be dominated by any clustering in $\overline{cCF(\Gamma)}$.
$\Rightarrow$ $\mathcal{C}$ must be dominated by CCSel(Γ).
$\Rightarrow$ $\mathcal{C}$ dominates CCSel(Γ) and CCSel(Γ) dominates $\mathcal{C}$.
$\Rightarrow$ $\mathcal{C}$ is equivalent to CCSel(Γ).
$\Rightarrow$ CCSel(Γ) is maximally certainty sound with respect to Γ which is a contradiction to the initial assumption.
Consequently, if the certain fact skyline only contains a single element, the clustering CCSel(Γ) is maximally certainty sound with respect to Γ.

Proof of Theorem 71

Theorem 71 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. A clustering $\mathcal{C}$ is a certain fact of Γ, iff for each factor set $F \in \mathcal{F}^{set}$ the clustering $\pi_F(\mathcal{C})$ is a certain fact of Γ_F, i.e. $\mathcal{C} \in cCF(\Gamma) \Leftrightarrow \forall F \in \mathcal{F}^{set}: \pi_F(\mathcal{C}) \in cCF(\Gamma_F)$.*

Before proving Theorem 71 we consider the following Lemma.

Lemma 5 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ with the factor sets $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. If the entity pair $\{e_1, e_2\}$ is a* POSSIBLE MATCH *in Γ, it is a* POSSIBLE MATCH *in its corresponding factor Γ_F and vice versa, i.e. $\{e_1, e_2\} \in P(\Gamma) \Leftrightarrow \forall F \in \mathcal{F}^{set}\colon \{e_1, e_2\} \in P(\Gamma_F) \vee e_1 \notin F, e_2 \notin F$.*

Lemma 5 can be proved as follows:

Proof 72 *Assumption 1: $\{e_1, e_2\}$ is a* POSSIBLE MATCH *in Γ, but is not a* POSSIBLE MATCH *in Γ_F although $e_1, e_2 \in F$.*
$\{e_1, e_2\} \in P(\Gamma) \Rightarrow$ there are two clusterings $\mathcal{C}_1, \mathcal{C}_2 \in \Gamma$ so that $\{e_1, e_2\}$ is a MATCH *in $\mathcal{C}_1$ and is an* UNMATCH *in $\mathcal{C}_2$. Note that pairs whose entities belong to different factors sets are certain* UNMATCHES *in Γ. Consequently, e_1 and e_2 must belong to the same factor set (in our case F). $\Rightarrow \{e_1, e_2\}$ is a* MATCH *in $\pi_F(\mathcal{C}_1)$ and is an* UNMATCH *in $\pi_F(\mathcal{C}_2)$. $\pi_F(\mathcal{C}_1), \pi_F(\mathcal{C}_2) \in \Gamma_F \Rightarrow \{e_1, e_2\}$ is a* POSSIBLE MATCH *in Γ_F which is a contradiction to the initial assumption. Consequently, if an entity pair is a* POSSIBLE MATCH *in Γ it is a* POSSIBLE MATCH *in its corresponding factor Γ_F.*

Assumption 2: $\{e_1, e_2\}$ is a POSSIBLE MATCH *in Γ_F, but is not a* POSSIBLE MATCH *in Γ.*
$\{e_1, e_2\} \in P(\Gamma_F) \Rightarrow$ there are two clusterings $\mathcal{C}_1, \mathcal{C}_2 \in \Gamma_F$ so that $\{e_1, e_2\}$ is a MATCH *in $\mathcal{C}_1$ and is an* UNMATCH *in $\mathcal{C}_2$.*
$\Rightarrow$ there are two clusterings $\mathcal{C}_1^*, \mathcal{C}_2^* \in \Gamma$ so that $\pi_F(\mathcal{C}_1^*) = \mathcal{C}_1$ and $\pi_F(\mathcal{C}_2^*) = \mathcal{C}_2$.
$\Rightarrow \{e_1, e_2\}$ is a MATCH *in $\mathcal{C}_1^*$ and is an* UNMATCH *in $\mathcal{C}_2^*$.*
$\Rightarrow \{e_1, e_2\}$ is a POSSIBLE MATCH *in Γ which is a contradiction to the initial assumption. Consequently, if an entity pair is a* POSSIBLE MATCH *in its corresponding factor Γ_F it is a* POSSIBLE MATCH *in Γ.*
Therefore, $\{e_1, e_2\} \in P(\Gamma) \Leftrightarrow \forall F \in \mathcal{F}^{set}\colon \{e_1, e_2\} \in P(\Gamma_F) \vee e_1 \notin F, e_2 \notin F$.

Using Lemma 5 Theorem 71 can be proved as follows:

Proof 73 *($\Rightarrow$) Assumption 1: cCF(Γ) contains $\mathcal{C}$, but there is a factor set $F \in \mathcal{F}^{set}$ so that $\pi_F(\mathcal{C}) \notin$ cCF(Γ_F).*
$\pi_F(\mathcal{C}) \notin$ cCF(Γ_{F_j}) $\Rightarrow$ there is a pair $\{e_1, e_2\} \in$ Pairs($\pi_F(\mathcal{C})$) that is a POSSIBLE MATCH *in Γ_F (Theorem 62).*
$\Rightarrow \{e_1, e_2\} \in$ Pairs($\mathcal{C}$) and is a POSSIBLE MATCH *in Γ.*
$\Rightarrow \mathcal{C}$ cannot belong to cCF(Γ) which is a contradiction to the initial assumption. Consequently, if $\mathcal{C} \in$ cCF(Γ) it holds that $\pi_F(\mathcal{C}) \in$ cCF(Γ_F) for every $F \in \mathcal{F}^{set}$.

($\Leftarrow$) Assumption 2: cCF(Γ) does not contain $\mathcal{C}$, but for all factor sets $F \in \mathcal{F}^{set}$ it holds that $\pi_F(\mathcal{C}) \in$ cCF(Γ_F).
$\mathcal{C} \notin$ cCF(Γ) $\Rightarrow$ there is a pair $\{e_1, e_2\} \in$ Pairs($\mathcal{C}$) that is a POSSIBLE MATCH *in Γ (Theorem 62).*
all pairs whose entities belong to different factor sets are certain UNMATCHES *$\Rightarrow e_1$ and e_2 belong to the same factor set.*
$\Rightarrow$ there is a factor set F so that $e_1, e_2 \in F$ and $\{e_1, e_2\}$ is a POSSIBLE MATCH *in Γ_F.*

$\Rightarrow$ *$\mathcal{C}$ cannot belong to cCF(Γ_F) which is a contradiction to the initial assumption. Consequently, if* $\pi_F(\mathcal{C}) \in$ *cCF*(Γ_F) *for every* $F \in \mathcal{F}^{set}$ *it holds that* $\mathcal{C} \in$ *cCF*(Γ).
Therefore, it holds that $\mathcal{C} \in$ *cCF*$(\Gamma) \Leftrightarrow \forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in$ *cCF*(Γ_F).

Proof of Theorem 72

Theorem 72 *Let* Γ *be an incomplete clustering and let* $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *be a range-disjoint factorization of* Γ *with the factor sets* $\mathcal{F}^{set} = \{F_1, \ldots, F_k\}$. *A clustering* $\mathcal{C}$ *is an element of the certain fact skyline of* Γ, *iff for each factor set* $F \in \mathcal{F}^{set}$ *the clustering* $\pi_F(\mathcal{C})$ *is an element of the certain fact skyline of* Γ_F, *i.e.* $\mathcal{C} \in \overline{cCF(\Gamma)} \Leftrightarrow \forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in \overline{cCF(\Gamma_F)}$.

Theorem 72 can be proved as follows:

Proof 74 *(*$\Rightarrow$*) Assumption 1:* $\overline{cCF(\Gamma)}$ *contains* $\mathcal{C}$ *but there is a factor set* $F_j \in \mathcal{F}^{set}$ *so that* $\pi_{F_j}(\mathcal{C}) \notin \overline{cCF(\Gamma_{F_j})}$.
$\mathcal{C} \in \overline{cCF(\Gamma)} \Rightarrow \mathcal{C} \in$ *cCF*(Γ).
$\pi_{F_j}(\mathcal{C}) \in$ *cCF*(Γ_{F_j}) *(Theorem 71) and* $\pi_{F_j}(\mathcal{C}) \notin \overline{cCF(\Gamma_{F_j})}$
$\Rightarrow$ *there is a clustering* $\mathcal{C}' \in \overline{cCF(\Gamma_{F_j})}$ *that strongly dominates* $\pi_{F_j}(\mathcal{C})$.
$\Rightarrow \bigboxtimes_{i=1, i\neq j}^{k} \pi_{F_i}(\mathcal{C}) \bigboxtimes \mathcal{C}'$ *strongly dominates* $\mathcal{C}$ *and is a certain fact of* Γ.
$\Rightarrow$ *$\mathcal{C}$ cannot belong to* $\overline{cCF(\Gamma)}$ *which is a contradiction to out initial assumption. Consequently, if* $\mathcal{C}$ *belongs to* $\overline{cCF(\Gamma)}$ *it follows that for every factor set* $F_j \in \mathcal{F}^{set}$ $\pi_{F_j}(\mathcal{C}) \in \overline{cCF(\Gamma_{F_j})}$.

($\Leftarrow$*) Assumption 2:* $\overline{cCF(\Gamma)}$ *does not contain* $\mathcal{C}$ *but or all factor sets* $F \in \mathcal{F}^{set}$ *it holds that* $\pi_F(\mathcal{C}) \in \overline{cCF(\Gamma_F)}$.
$\mathcal{C} \notin \overline{cCF(\Gamma)} \Rightarrow$ *there is a clustering* $\mathcal{C}' \in \overline{cCF(\Gamma)}$ *that strongly dominates* $\mathcal{C}$ *(i.e.* $rng(\mathcal{C}') \supsetneq rng(\mathcal{C})$*).*
$\Rightarrow$ *there is a factor set* $F \in \mathcal{F}^{set}$ *so that* $(F \cap rng(\mathcal{C}')) \supsetneq (F \cap rng(\mathcal{C}))$.
$\Rightarrow$ $\pi_F(\mathcal{C}')$ *strongly dominates* $\pi_F(\mathcal{C})$.
$\Rightarrow$ $\pi_F(\mathcal{C})$ *cannot belong to* $\overline{cCF(\Gamma_F)}$ *which is a contradiction to the initial assumption. Consequently, if for every factor set* $F_j \in \mathcal{F}^{set}$ $\pi_{F_j}(\mathcal{C}) \in \overline{cCF(\Gamma_{F_j})}$ *it follows that* $\mathcal{C}$ *belongs to* $\overline{cCF(\Gamma)}$. *Therefore, it holds that* $\mathcal{C} \in \overline{cCF(\Gamma)} \Leftrightarrow \forall F \in \mathcal{F}^{set}\colon \pi_F(\mathcal{C}) \in \overline{cCF(\Gamma_F)}$.

Proof of Theorem 73

Theorem 73 *Let* Γ *be an incomplete clustering and let* $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ *be a range-disjoint factorization of* Γ, *the set* $\{CCSel(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ *is a range-disjoint factorization of the certain clustering* $CCSel(\Gamma)$.

Theorem 73 can be proved as follows:

Proof 75 *Following Definition 12, the set* $\{CCSel(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ *is a range-disjoint factorization of the certain clustering* $CCSel(\Gamma)$ *if we can reconstruct* $CCSel(\Gamma)$ *by integrating the clusterings in* $\{CCSel(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$, *and hence if* $CCSel(\Gamma) = \bigboxtimes_{i=1}^{k} CCSel(\Gamma_{F_i})$.

This in turn can be formally proved as follows:

$$\begin{aligned} \mathit{CCSel}(\Gamma) &= \bigcap \Gamma \\ &= \{C \mid \forall \mathcal{C} \in \Gamma \colon C \in \mathcal{C}\} \\ &= \{C \mid \forall \mathcal{C} \in \boxtimes \mathcal{F} \colon C \in \mathcal{C}\} \end{aligned}$$

due to the factorization is range-disjoint, clusters are restricted to single factors, it follows:

$$\begin{aligned} &= \{C \mid \exists \Gamma_F \in \mathcal{F} \colon \forall \mathcal{C} \in \Gamma_F \colon C \in \mathcal{C}\} \\ &= \bigcup_{i=1}^{k} (\bigcap \Gamma_{F_i}) \end{aligned}$$

due to the definition of the crisp version of the integration operator, it follows:

$$\begin{aligned} &= \boxtimes_{i=1}^{k} (\bigcap \Gamma_{F_i}) \\ &= \boxtimes_{i=1}^{k} \ \mathit{CCSel}(\Gamma_{F_i}) \end{aligned}$$

Proof of Theorem 74

Theorem 74 *Let Γ be an incomplete clustering and let $\mathcal{F} = \{\Gamma_{F_1}, \ldots, \Gamma_{F_k}\}$ be a range-disjoint factorization of Γ, the set $\{\mathit{CSSel}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ is a range-disjoint factorization of the certain clustering $\mathit{CSSel}(\Gamma)$.*

Theorem 74 can be proved as follows:

Proof 76 *Following Definition 12, the set $\{\mathit{CSSel}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$ is a range-disjoint factorization of the certain clustering $\mathit{CSSel}(\Gamma)$ if we can reconstruct $\mathit{CSSel}(\Gamma)$ by integrating the clusterings in $\{\mathit{CSSel}(\Gamma_F) \mid \Gamma_F \in \mathcal{F}\}$, and hence if $\mathit{CSSel}(\Gamma) = \boxtimes_{i=1}^{k} \mathit{CSSel}(\Gamma_{F_i})$.*

This in turn can be formally proved in two steps. First we prove that the non-cluster-disjoint clustering $\mathit{CSC}(\Gamma)$ can be reconstructed from the factors by computing $\mathit{CSC}(\Gamma) = \boxtimes_{i=1}^{k} \mathit{CSC}(\Gamma_{F_i})$. Then we prove that the cluster-disjoint clustering $\mathit{CSSel}(\Gamma) = \overline{\mathit{CSC}(\Gamma)}$ can be reconstructed from the factors by computing $\overline{\mathit{CSC}(\Gamma)} = \boxtimes_{i=1}^{k} \ \overline{\mathit{CSC}(\Gamma_{F_i})}$.

We start with the non-cluster-disjoint clustering $\mathit{CSC}(\Gamma)$ that contains each cluster that is a certain subcluster in Γ.

$$\begin{aligned} \mathit{CSC}(\Gamma) &= \{C_p \mid \forall \mathcal{C} \in \Gamma \colon \exists C_q \in \mathcal{C} \colon C_p \subseteq C_q\} \\ &= \{C_p \mid \forall \mathcal{C} \in \boxtimes \mathcal{F} \colon \exists C_q \in \mathcal{C} \colon C_p \subseteq C_q\} \end{aligned}$$

due to the factorization is range-disjoint, clusters are restricted to single factors, it follows:

$$\begin{aligned} &= \{C_p \mid \exists \Gamma_F \in \mathcal{F} \colon \forall \mathcal{C} \in \Gamma_F \colon \exists C_q \in \mathcal{C} \colon C_p \subseteq C_q\} \\ &= \bigcup_{i=1}^{k} \{C_p \mid \forall \mathcal{C} \in \Gamma_{F_i} \colon \exists C_q \in \mathcal{C} \colon C_p \subseteq C_q\} \\ &= \bigcup_{i=1}^{k} \mathit{CSC}(\Gamma_{F_i}) \\ &= \boxtimes_{i=1}^{k} \mathit{CSC}(\Gamma_{F_i}) \end{aligned}$$

Using the above proved fact that $\mathit{CSC}(\Gamma) = \boxtimes_{i=1}^{k} \mathit{CSC}(\Gamma_{F_i})$ *we proceed with the cluster-disjoint clustering* $\overline{\mathit{CSC}(\Gamma)}$ *that contains each cluster that is a certain subcluster in* Γ *and that is not dominated by any other cluster in* $\overline{\mathit{CSC}(\Gamma)}$.

$$\begin{aligned}
\overline{\mathit{CSC}(\Gamma)} &= \{C_p \mid C_p \in \mathit{CSC}(\Gamma),\, \nexists C_q \in \mathit{CSC}(\Gamma) \colon C_p \prec C_q, C_p \neq C_q\} \\
&= \{C_p \mid C_p \in \bigcup_{i=1}^{k} \mathit{CSC}(\Gamma_{F_i}),\, \nexists C_q \in \bigcup_{i=1}^{k} \mathit{CSC}(\Gamma_{F_i}) \colon C_p \prec C_q, C_p \neq C_q\}
\end{aligned}$$

due to clusters can only be dominated by a cluster of the same factor, it follows:

$$\begin{aligned}
&= \bigcup_{i=1}^{k} \{C_p \mid C_p \in \mathit{CSC}(\Gamma_{F_i}),\, \nexists C_q \in \mathit{CSC}(\Gamma_{F_i}) \colon C_p \prec C_q, C_p \neq C_q\} \\
&= \bigcup_{i=1}^{k} \overline{\mathit{CSC}(\Gamma_{F_i})}
\end{aligned}$$

due to the definition of integration of certain clusterings, it follows:

$$= \boxtimes_{i=1}^{k} \overline{\mathit{CSC}(\Gamma_{F_i})}$$

List of Own Publications

The work presented in this thesis has produced the following publications:

F. PANSE, Datenunvollständigkeit aufgrund der mangelnden Modellierungsmächtigkeit aktuell dominierender Datenmodelle, In *Proceedings of the 21st GI-Workshop on Foundations of Databases (GvD 2009), Rostock-Warnemünde, Germany, 2009, pp. 123-127*

F. PANSE, Completeness of Attribute Values Representing Partial Information, In *Proceedings of the 7th International Workshop on Quality in Databases at VLDB 2009 (QDB 2009), Lyon, France, 2009*

F. PANSE und N. RITTER, Completeness in Databases with Maybe-Tuples, In *Proceedings of the 4th Workshop on Quality of Information Systems at ER 2009 (QoIS 2009), Gramado, Brazil, Springer, Berlin, 2009, pp. 202-211*

F. PANSE und N. RITTER, Towards Duplicate Detection and Data Fusion in Fuzzy Relational Databases, In *Berichte des Departments Informatik der Universität Hamburg, Band 10, Bericht 292, Hamburg, Germany, 2010, pp. 1-13*

F. PANSE, M. VAN KEULEN, A. DE KEIJZER und N. RITTER, Duplicate Detection in Probabilistic Data, In *Proceedings of the 2nd Workshop on New Trends in Information Integration at ICDE 2010 (NTII 2010), Long Beach, United States of America, 2010, pp. 179-182*

F. PANSE und N. RITTER, Relational Data Completeness in the Presence of Maybe-Tuples, In *Ingénierie des Systèmes d'Information, Volume 15, Number 6, 2010, pp. 85-104*

F. PANSE und N. RITTER, Tuple Merging in Probabilistic Databases, In *Proceedings of the 4th Workshop on Management of Uncertain Data at VLDB 2010 (MUD 2010), Singapore, Singapore, 2010, pp. 113-127*

F. PANSE und N. RITTER, Incorporating Domain Knowledge and User Expertise in Probabilistic Tuple Merging, In *Proceedings of the 5th International Conference on Scalable Uncertainty Management (SUM 2011), Dayton, United States of America, Springer, Berlin, 2011, pp. 289-302*

F. PANSE und N. RITTER, Evaluating Indeterministic Duplicate Detection Results, In *Proceedings of the 6th International Conference on Scalable Uncertainty Management (SUM 2012), Marburg, Germany, Springer, Berlin, 2012, pp. 433-446*

F. PANSE, W. WINGERATH, S. FRIEDRICH und N. RITTER, Key-based Blocking of Duplicates in Entity-Independent Probabilistic Data, In *Proceedings of the 17th International Conference on Information Quality (ICIQ 2012), Paris, France, 2012, pp. 278-296*

F. PANSE, M. VAN KEULEN und N. RITTER, Indeterministic Handling of Uncertain Decisions in Deduplication, In *Journal on Data and Information Quality, Volume 4, Number 2, 2013*

List of Figures

List of Tables

Bibliography

[AA03] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[AAAV12] Naser Ayat, Hamideh Afsarmanesh, Reza Akbarinia, and Patrick Valduriez. An uncertain data integration system. In *OTM Conferences (2)*, pages 825–842, 2012.

[AAAV13] Naser Ayat, Reza Akbarinia, Hamideh Afsarmanesh, and Patrick Valduriez. Entity resolution for distributed probabilistic data. *Distributed and Parallel Databases*, 31(4):509–542, 2013.

[AB13] Nikolaus Augsten and Michael H. Böhlen. *Similarity Joins in Relational Database Systems*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.

[ABC99] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent Query Answers in Inconsistent Databases. In *PODS*, pages 68–79, 1999.

[ABS+06] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha U. Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, pages 1151–1154, 2006.

[ACGK08] Arvind Arasu, Surajit Chaudhuri, Kris Ganjam, and Raghav Kaushik. Incorporating string transformations in record matching. In *SIGMOD Conference*, pages 1231–1234, 2008.

[ACK+11] Serge Abiteboul, T.-H. Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart. Capturing continuous data and answering aggregate queries in probabilistic xml. *ACM Trans. Database Syst.*, 36(4):25, 2011.

[ACN08] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.

[AFM06] Periklis Andritsos, Ariel Fuxman, and Renée J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE*, page 30, 2006.

[Agg09] Charu C. Aggarwal. On Clustering Algorithms for Uncertain Data. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 13, pages 389–406. Springer, 2009.

[AJP+10] Subi Arumugam, Ravi Jampani, Luis Leopoldo Perez, Fei Xu, Christopher M. Jermaine, and Peter J. Haas. Mcdb-r: Risk analysis in the database. *PVLDB*, 3(1):782–793, 2010.

[AKE04] Reema Al-Kamha and David W. Embley. Grouping search-engine returned citations for person-name queries. In *WIDM*, pages 96–103, 2004.

[AKG87] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. In *SIGMOD Conference*, pages 34–48, 1987.

[AKO09] Lyublena Antova, Christoph Koch, and Dan Olteanu. $10^{(10^6)}$ worlds and beyond: efficient representation and processing of incomplete information. *VLDB J.*, 18(5):1021–1040, 2009.

[ALWW09] Charu C. Aggarwal, Yan Li, Jianyong Wang, and Jing Wang. Frequent Pattern Mining Algorithms with Uncertain Data. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 15, pages 427–460. Springer, 2009.

[AM74] Duane Boes Alexander Mood, Franklin Graybill. *Introduction to the Theory of Statistics*. McGraw-Hill., 3rd edition, 1974.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[AO05] Akiko N. Aizawa and Keizo Oyama. A fast linkage detection scheme for multi-source information integration. In *WIRI*, pages 30–39, 2005.

[APR04] Javed A. Aslam, Ekaterina Pelekhov, and Daniela Rus. The star clustering algorithm for static and dynamic information organization. *J. Graph Algorithms Appl.*, 8:95–129, 2004.

[ARS09] Arvind Arasu, Christopher Ré, and Dan Suciu. Large-scale deduplication with constraints using dedupalog. In *ICDE*, pages 952–963, 2009.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[AS13] Antoine Amarilli and Pierre Senellart. On the connections between relational and xml probabilistic data models. In *BNCOD*, pages 121–134, 2013.

[ASUW10] Parag Agrawal, Anish Das Sarma, Jeffrey D. Ullman, and Jennifer Widom. Foundations of uncertain-data integration. *PVLDB*, 3(1):1080–1090, 2010.

[AW10] Parag Agrawal and Jennifer Widom. Generalized uncertain databases: First steps. In *MUD*, pages 99–111, 2010.

[AY08a] Charu C. Aggarwal and Philip S. Yu. A framework for clustering uncertain data streams. In *ICDE*, pages 150–159, 2008.

[AY08b] Charu C. Aggarwal and Philip S. Yu. Outlier detection with uncertain data. In *SDM*, pages 483–493, 2008.

[BB98] Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566, 1998.

[BB10] Emilio Corchado Bruno Baruque. *Fusion Methods for Unsupervised Learning Ensembles*. Springer Science & Business Media, 2010.

[BBB+05] Alexander Bilke, Jens Bleiholder, Christoph Böhm, Karsten Draba, Felix Naumann, and Melanie Weis. Automatic data fusion with hummer. In *VLDB*, pages 1251–1254, 2005.

[BBC04] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[BBC+11] Lorenzo Blanco, Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Automatically building probabilistic databases from the web. In *WWW (Companion Volume)*, pages 185–188, 2011.

[BBKL12] Zeinab Bahmani, Leopoldo E. Bertossi, Solmaz Kolahi, and Laks V. S. Lakshmanan. Declarative entity resolution via matching dependencies and answer set programs. In

KR, 2012.

[BBR11] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm. *Schema Matching and Mapping*. Springer, 2011.

[BCC03] Rohan Baxter, Peter Christen, and Tim Churches. A comparison of fast blocking methods for record linkage. In *ACM SIGKDD '03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 25–27, 2003.

[BCC+13] Thomas Bernecker, Reynold Cheng, David W. Cheung, Hans-Peter Kriegel, Sau Dan Lee, Matthias Renz, Florian Verhein, Liang Wang, and Andreas Züfle. Model-based probabilistic frequent itemset mining. *Knowl. Inf. Syst.*, 37(1):181–217, 2013.

[BCKT07] Nilesh Bansal, Fei Chiang, Nick Koudas, and Frank Wm. Tompa. Seeking stable clusters in the blogosphere. In *VLDB*, pages 806–817, 2007.

[BCMP10] Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Probabilistic Models to Reconcile Complex Data from Inaccurate Data Sources. In *CAiSE*, pages 83–97, 2010.

[BDJ+06] Douglas Burdick, Prasad M. Deshpande, T. S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Efficient allocation algorithms for olap over imprecise data. In *VLDB*, pages 391–402, 2006.

[BDJ+07] Douglas Burdick, Prasad M. Deshpande, T. S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Olap over uncertain and imprecise data. *VLDB J.*, 16(1):123–144, 2007.

[BDM+05] Jihad Boulos, Nilesh N. Dalvi, Bhushan Mandhani, Shobhit Mathur, Christopher Ré, and Dan Suciu. Mystiq: a system for finding more answers by using probabilities. In *SIGMOD Conference*, pages 891–893, 2005.

[Ber11] Leopoldo E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[Bes12] George Beskales. *Modeling and Querying Uncertainty in Data Cleaning*. PhD thesis, University of Waterloo, 2012.

[BFO+09] Christian Böhm, Frank Fiedler, Annahita Oswald, Claudia Plant, and Bianca Wackersreuther. Probabilistic skyline queries. In *CIKM*, pages 651–660, 2009.

[BFOS84] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[BG04] Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *DMKD*, pages 11–18, 2004.

[BG06] Indrajit Bhattacharya and Lise Getoor. A latent dirichlet model for unsupervised entity resolution. In *SDM*, 2006.

[BG07a] Irad Ben-Gal. Bayesian networks. In *Encyclopedia of Statistics in Quality and Reliability*. John Wiley & Sons, 2007.

[BG07b] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *TKDD*, 1(1), 2007.

[BG07c] Indrajit Bhattacharya and Lise Getoor. Online collective entity resolution. In *AAAI*, pages 1606–1609, 2007.

[BGMG+07] Omar Benjelloun, Hector Garcia-Molina, Heng Gong, Hideki Kawai, Tait Eliott Larson, David Menestrina, and Sutthipong Thavisomboon. D-swoosh: A family of algorithms for generic, distributed entity resolution. In *ICDCS*, page 37, 2007.

[BGMK+06] Omar Benjelloun, Hector Garcia-Molina, Hideki Kawai, Tait Eliott Larson, David Menestrina, Qi Su, Sutthipong Thavisomboon, and Jennifer Widom. Generic entity resolution in the serf project. *IEEE Data Eng. Bull.*, 29(2):13–20, 2006.

[BGMM+09] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. Swoosh: a generic approach to entity resolution. *VLDB J.*, 18(1):255–276, 2009.

[BGMP90] Daniel Barbará, Hector Garcia-Molina, and Daryl Porter. A probalilistic relational data model. In *EDBT*, pages 60–74, 1990.

[BGMP92] Daniel Barbará, Hector Garcia-Molina, and Daryl Porter. The Management of Probabilistic Data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.

[Bha43] Anil Kumar Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1943.

[BI98] Aaron F. Bobick and Yuri A. Ivanov. Action recognition using probabilistic parsing. In *CVPR*, pages 196–202, 1998.

[Bil06] Alexander Bilke. *Duplicate based schema matching*. PhD thesis, 2006.

[BJ07] Olga Brazhnik and John F. Jones. Anatomy of data integration. *Journal of Biomedical Informatics*, 40(3):252–269, 2007.

[BKD+08] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter N. Belhumeur, and Shree K. Nayar. Face swapping: automatically replacing faces in photographs. *ACM Trans. Graph.*, 27(3), 2008.

[BKL11] Leopoldo E. Bertossi, Solmaz Kolahi, and Laks V. S. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. In *ICDT*, pages 268–279, 2011.

[BKM06] Mikhail Bilenko, Beena Kamath, and Raymond J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM*, pages 87–96, 2006.

[BKOS10] Michael Benedikt, Evgeny Kharlamov, Dan Olteanu, and Pierre Senellart. Probabilistic xml via markov chains. *PVLDB*, 3(1):770–781, 2010.

[BKOZ13] Nurzhan Bakibayev, Tomás Kociský, Dan Olteanu, and Jakub Zavodny. Aggregation and ordering in factorised databases. *PVLDB*, 6(14):1990–2001, 2013.

[BL13] Victoria Meyer Bertrand Lisbach. *Linguistic Identity Matching*. Springer Science & Business Media, 2013.

[BM02] Mikhail Bilenko and Raymond J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical report, University of Texas, 2002.

[BM03] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, pages 39–48, 2003.

[BN05] Alexander Bilke and Felix Naumann. Schema matching using duplicates. In *ICDE*, pages 69–80, 2005.

[BN06] Jens Bleiholder and Felix Naumann. Conflict handling strategies in an integrated information system. Technical report, Humboldt-Universität Berlin, 2006.

[BN08] Jens Bleiholder and Felix Naumann. Data fusion. *ACM Comput. Surv.*, 41(1), 2008.

[BS06] Carlo Batini and Monica Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, 2006.

[BS13] Amir Dayyan Borhanian and Fereidoon Sadri. A compact representation for efficient uncertain-information integration. In *IDEAS*, pages 122–131, 2013.

[BSHW06] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[BSI08] George Beskales, Mohamed A. Soliman, and Ihab F. Ilyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. *PVLDB*, 1(1):326–339, 2008.

[BSI+10] George Beskales, Mohamed A. Soliman, Ihab F. Ilyas, Shai Ben-David, and Yubin Kim. Probclean: A probabilistic duplicate detection system. In *ICDE*, pages 1193–1196, 2010.

[BSIBD09] George Beskales, Mohamed A. Soliman, Ihab F. Ilyas, and Shai Ben-David. Modeling and querying possible repairs in duplicate detection. *PVLDB*, 2(1):598–609, 2009.

[BSK09] Christian Borgelt, Matthias Steinbrecher, and Rudolf R Kruse. *Graphical Models: Representations for Learning, Reasoning and Data Mining*. John Wiley & Sons, 2009.

[BT06] David Guy Brizan and Abdullah Uz Tansel. A survey of entity resolution and record linkage methodologies. *Communications of the IIMA 6*, 6:41–50, 2006.

[BT07] Peter Buneman and Wang Chiew Tan. Provenance in databases. In *SIGMOD Conference*, pages 1171–1173, 2007.

[BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[CA03] Kai Lai Chung and Farid AitSahlia. *Elementary Probability Theory: With Stochastic Processes and an Introduction to Mathematical Finance*. Springer Science & Business Media, 2003.

[Car50] Rudolf Carnap. *Logical Foundations of Probability*. University of Chicago Press, 1950.

[CCT96] Arbee L. P. Chen, Jui-Shang Chiu, and Frank Shou-Cheng Tseng. Evaluating aggregate operations over imprecise data. *IEEE Trans. Knowl. Data Eng.*, 8(2):273–284, 1996.

[CCT09] James Cheney, Laura Chiticariu, and Wang Chiew Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[CD08] Lixia Chen and Alin Dobra. Efficient processing of aggregates in probabilistic databases. Technical report, University of Florida, 2008.

[CDLS99] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhater. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[CG07a] Peter Christen and Karl Goiser. Quality and complexity measures for data linkage and deduplication. In *Quality Measures in Data Mining*, pages 127–151. 2007.

[CG07b] Graham Cormode and Minos N. Garofalakis. Sketching probabilistic data streams. In *SIGMOD Conference*, pages 281–292, 2007.

[CG08] Peter Christen and Ross W. Gayler. Towards scalable real-time entity resolution using a similarity-aware inverted index approach. In *AusDM*, pages 51–60, 2008.

[CGH97] Enrique Castillo, Jose M. Gutierrez, and Ali S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, 1997.

[CGL07] Ricardo G. Cota, Marcos André Gonçalves, and Alberto H. F. Laender. A heuristic-based hierarchical clustering method for author name disambiguation in digital libraries. In *SBBD*, pages 20–34, 2007.

[CGM05] Surajit Chaudhuri, Venkatesh Ganti, and Rajeev Motwani. Robust Identification of Fuzzy Duplicates. In *ICDE*, pages 865–876, 2005.

[Cha07] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Science*, 4:300–307, 2007.

[Cha08] Sung-Hyuk Cha. Distance between histograms with shuffled cost matrix. In *SSPR/SPR*, pages 633–643, 2008.

[Che76] Peter P. Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[CHL10] Pável Calado, Melanie Herschel, and Luís Leitão. An overview of xml duplicate detection algorithms. In *Soft Computing in XML Data Management*, pages 193–224. 2010.

[Chr07] Peter Christen. Towards parameter-free blocking for scalable record linkage. Technical report, 2007.

[Chr08a] Peter Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *KDD*, pages 151–159, 2008.

[Chr08b] Peter Christen. Automatic training example selection for scalable unsupervised record linkage. In *PAKDD*, pages 511–518, 2008.

[Chr11] Peter Christen. A survey of indexing techniques for scalable record linkage and deduplication. *TKDE*, 2011.

[Chr12] Peter Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, 2012.

[CK05] Marek Capinski and Ekkehard Kopp. *Measure, Integral and Probability*. Springer Verlag, 2005.

[CKH07] Chun Kit Chui, Ben Kao, and Edward Hung. Mining frequent itemsets from uncertain data. In *PAKDD*, pages 47–58, 2007.

[CKLS01] Munir Cochinwala, Verghese Kurien, Gail Lalk, and Dennis Shasha. Efficient data reconciliation. *Inf. Sci.*, 137(1-4):1–15, 2001.

[CKP03] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD Conference*, pages 551–562, 2003.

[CKP04] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Trans. Knowl. Data Eng.*, 16(9):1112–1127, 2004.

[CKZ94] Mou-Yen Chen, Amlan Kundu, and Jian Zhou. Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE Trans. Pattern*

Anal. Mach. Intell., 16(5):481–496, 1994.

[CL08] Andrea Calì and Thomas Lukasiewicz. An approach to probabilistic data integration for the semantic web. In *URSW (LNCS Vol.)*, pages 52–65, 2008.

[CLY09] Graham Cormode, Feifei Li, and Ke Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, pages 305–316, 2009.

[Cod83] E. F. Codd. A relational model of data for large shared data banks (reprint). *Commun. ACM*, 26(1):64–69, 1983.

[Cod86] E. F. Codd. Missing information (applicable and inapplicable) in relational databases. *SIGMOD Record*, 15(4):53–78, 1986.

[Cod87] E. F. Codd. More Commentary on Missing Information in Relational Databases (Applicable and Inapplicable Information). *SIGMOD Record*, 16(1):42–50, 1987.

[CP87] Roger Cavallo and Michael Pittarelli. The theory of probabilistic databases. In *VLDB*, pages 71–81, 1987.

[CP03] Reynold Cheng and Sunil Prabhakar. Managing uncertainty in sensor database. *SIGMOD Record*, 32(4):41–46, 2003.

[CR02] William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD*, pages 475–480, 2002.

[CS02] Sung-Hyuk Cha and Sargur N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6):1355–1370, 2002.

[CSP05] Reynold Cheng, Sarvjeet Singh, and Sunil Prabhakar. U-dbms: A database system for managing constantly-evolving data. In *VLDB*, pages 1271–1274, 2005.

[Cus07] James Cussens. Logic-based formalisms for statistical relational learning. In Ben Taskar Lise Getoor, editor, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[CWW00] Yingwei Cui, Jennifer Widom, and Janet L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25(2):179–227, 2000.

[CXP+04] Reynold Cheng, Yuni Xia, Sunil Prabhakar, Rahul Shah, and Jeffrey Scott Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In

VLDB, pages 876–887, 2004.

[Dar09] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[Das10] Anirban DasGupta. *Fundamentals of Probability: A First Course*. Springer Science & Business Media, 2010.

[DB07] Jay L. Devore and Kenneth N. Berk. *Modern Mathematical Statistics with Applications*. Cengage Learning, 2007.

[DBS09] Nilesh N. Dalvi, Philip Bohannon, and Fei Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *SIGMOD Conference*, pages 335–348, 2009.

[DD06] Elena Deza and Michel-Marie Deza. *Dictionary of Distances*. Elsevier Books, 2006.

[DD09] Elena Deza and Michel Marie Deza. *Encyclopedia of Distances*. Springer, 2009.

[Dec96] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *UAI*, pages 211–219, 1996.

[DEFI06] Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theor. Comput. Sci.*, 361(2-3):172–187, 2006.

[DeM89] Linda G. DeMichiel. Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains. *IEEE Trans. Knowl. Data Eng.*, 1(4):485–493, 1989.

[Dey08] Debabrata Dey. Entity matching in heterogeneous databases: A logistic regression approach. *Decis. Support Syst.*, 44(3):740–747, February 2008.

[DG04] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[DGL+09] Landon Detwiler, Wolfgang Gatterbauer, Brenton Louie, Dan Suciu, and Peter Tarczy-Hornoch. Integrating and ranking uncertain scientific data. In *ICDE*, pages 1235–1238, 2009.

[DGS09] Amol Deshpande, Lise Getoor, and Prithviraj Sen. Graphical models for uncertain data. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, vol-

ume 35 of *Advances in Database Systems*, chapter 4, pages 77–112. Springer, 1 edition, 2009.

[DHA01] Richard O. Duda, Peter E. Hart, and David G. Stork (Author). *Pattern Classification*. Wiley-Interscience, 2 edition, 2001.

[DHM05] Xin Dong, Alon Y. Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD Conference*, pages 85–96, 2005.

[DHY09] Xin Luna Dong, Alon Y. Halevy, and Cong Yu. Data integration with uncertainty. *VLDB J.*, 18(2):469–500, 2009.

[dK08] Ander de Keijzer. *Management of Uncertain Data - Towards unattended integration*. PhD thesis, University of Twente, 2008.

[DK10a] Sushovan De and Subbarao Kambhampati. Defining and mining functional dependencies in probabilistic databases. *CoRR*, abs/1005.4714, 2010.

[dK10b] Ander de Keijzer. Data integration using uncertain xml. In *Soft Computing in XML Data Management*, pages 79–103. 2010.

[DKNS01] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

[dKvK07a] Ander de Keijzer and Maurice van Keulen. Quality measures in uncertain data management. In *SUM*, pages 104–115, 2007.

[dKvK07b] Ander de Keijzer and Maurice van Keulen. User feedback in probabilistic integration. In *DEXA Workshops*, pages 377–381, 2007.

[dKvK08] Ander de Keijzer and Maurice van Keulen. Imprecise: Good-is-good-enough data integration. In *ICDE*, pages 1548–1551, 2008.

[dKvKL06] A. de Keijzer, M. van Keulen, and Y. Li. Taming data explosion in probabilistic information integration. Technical Report TR-CTIT-06-05, Enschede, February 2006.

[DLKS10] Mario Döller, Sebastian Lehrack, Harald Kosch, and Ingo Schmitt. Quantum logic based mpeg query format algebra. In *Adaptive Multimedia Retrieval*, pages 204–219, 2010.

[DLLH03] AnHai Doan, Ying Lu, Yoonkyong Lee, and Jiawei Han. Profile-based object matching for information integration. *IEEE Intelligent Systems*, 18(5):54–59, 2003.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977.

[DMP+04] George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. The automatic content extraction (ace) program - tasks, data, and evaluation. In *LREC*, 2004.

[DN09] Xin Luna Dong and Felix Naumann. Data fusion - resolving data conflicts for integration. volume 2, pages 1654–1655, 2009.

[DN11] Uwe Draisbach and Felix Naumann. A generalization of blocking and windowing algorithms for duplicate detection. In *ICDKE*, pages 18–24, 2011.

[DPW12] Alexandre Decan, Fabian Pijcke, and Jef Wijsen. Certain conjunctive query answering in sql. In *SUM*, pages 154–167, 2012.

[DRS01] Alex Dekhtyar, Robert B. Ross, and V. S. Subrahmanian. Probabilistic temporal databases, i: algebra. *ACM Trans. Database Syst.*, 26(1):41–95, 2001.

[DRS09] Nilesh N. Dalvi, Christopher Ré, and Dan Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

[DRS11] Nilesh N. Dalvi, Christopher Re, and Dan Suciu. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci.*, 77(3):473–490, 2011.

[DS96] Debabrata Dey and Sumit Sarkar. A probabilistic relational model and algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.

[DS98] Curtis E. Dyreson and Richard T. Snodgrass. Supporting valid-time indeterminacy. *ACM Trans. Database Syst.*, 23(1):1–57, 1998.

[DS07a] Nilesh N. Dalvi and Dan Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, pages 293–302, 2007.

[DS07b] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[DS07c] Nilesh N. Dalvi and Dan Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, pages 1–12, 2007.

[DS12] Nilesh N. Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30, 2012.

[DSD98] Debabrata Dey, Sumit Sarkar, and Prabuddha De. Entity matching in heterogeneous databases: a distance-based decision model. In *System Sciences, Proceedings of the Thirty-First Hawaii International Conference on Bd. 7*, 1998.

[dVKCC09] Timothy de Vries, Hui Ke, Sanjay Chawla, and Peter Christen. Robust record linkage blocking using suffix arrays. In *CIKM*, pages 305–314, 2009.

[dVKCC11] Timothy de Vries, Hui Ke, Sanjay Chawla, and Peter Christen. Robust record linkage blocking using suffix arrays and bloom filters. *TKDD*, 5(2):9, 2011.

[EEV02] Mohamed G. Elfeky, Ahmed K. Elmagarmid, and Vassilios S. Verykios. Tailor: A record linkage tool box. In *ICDE*, pages 17–28, 2002.

[Eis69] Martin Eisen. *Introduction to Mathematical Probability Theory*. Prentice Hall, 1969.

[EIV07] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[EKM+12] Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Züfle. Querying uncertain spatio-temporal data. In *ICDE*, pages 354–365, 2012.

[ELLS01] Thomas Eiter, James J. Lu, Thomas Lukasiewicz, and V. S. Subrahmanian. Probabilistic object bases. *ACM Trans. Database Syst.*, 26(3):264–312, 2001.

[ELW01] Thomas Eiter, Thomas Lukasiewicz, and Michael Walter. A data model and algebra for probabilistic complex values. *Ann. Math. Artif. Intell.*, 33(2-4):205–252, 2001.

[FC00] Amy Feekin and Zhengxin Chen. Duplicate detection using k-way sorting method. In *SAC (1)*, pages 323–327, 2000.

[FFM05] Ariel Fuxman, Elham Fazli, and Renée J. Miller. Conquer: Efficient management of inconsistent databases. In *SIGMOD Conference*, pages 155–166, 2005.

[FG96] Bert Fristedt and Lawrence Gray. *A modern approach to probability theory*. Birkhäuser, 1996.

[FGL12] Anderson A. Ferreira, Marcos André Gonçalves, and Alberto H. F. Laender. A brief survey of automatic methods for author name disambiguation. *SIGMOD Record*, 41(2):15–26, 2012.

[FGPP10] Alfredo Ferro, Rosalba Giugno, Piera Laura Puglisi, and Alfredo Pulvirenti. An efficient duplicate record detection using q-grams array inverted index. In *DaWak*, pages 309–323, 2010.

[FHO12] Robert Fink, Larisa Han, and Dan Olteanu. Aggregation in probabilistic databases via knowledge compilation. *PVLDB*, 5(5):490–501, 2012.

[FHOR11] Robert Fink, Andrew Hogue, Dan Olteanu, and Swaroop Rath. Sprout^2: a squared query engine for uncertain web data. In *SIGMOD Conference*, pages 1299–1302, 2011.

[FLNS01] Marco Fortini, Brunero Liseo, Alessandra Nuccitelli, and Mauro Scanu. On bayesian record linkage. *Research in Official Statistics*, 4(1):185–198, 2001.

[FR97] Norbert Fuhr and Thomas Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[FRC11] Tobias Farrell, Kurt Rothermel, and Reynold Cheng. Processing continuous range queries with spatiotemporal tolerance. *IEEE Trans. Mob. Comput.*, 10(3):320–334, 2011.

[FS69] Ivan Fellegi and Alan Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[FS03] Vladimir Filkov and Steven Skiena. Integrating microarray data by consensus clustering. In *ICTAI*, pages 418–425, 2003.

[FTT03] Gary William Flake, Robert Endre Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2003.

[FWLG13] Huabin Feng, Hongzhi Wang, Jianzhong Li, and Hong Gao. Entity resolution on uncertain relations. In *WAIM*, pages 77–86, 2013.

[Gad98] T.N. Gadd. 'fisching fore werds': Phonetic retrieval of written text in information systems. *Program 22*, pages 222–237, 1998.

[Gal06a] Avigdor Gal. Managing uncertainty in schema matching with top-k schema mappings. pages 90–114, 2006.

[Gal06b] Avigdor Gal. Why is schema matching tough and what can we do about it? *SIGMOD Record*, 35(4):2–5, 2006.

[Gal11a] Avigdor Gal. *Uncertain Schema Matching*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[Gal11b] Avigdor Gal. Uncertain schema matching: the power of not knowing. In *CIKM*, pages 2615–2616, 2011.

[GB04] Lifang Gu and Rohan A. Baxter. Adaptive filtering for efficient record linkage. In *SDM*, 2004.

[GB06] Lifang Gu and Rohan A. Baxter. Decision models for record linkage. In *Selected Papers from AusDM*, pages 146–160, 2006.

[GBBH95] J.C. Gee, L. Le Briquer, C. Barillot, and D.R. Haynor. Probabilistic matching of brain images. In *Information Processing in Medical Imaging*, page 113–125. Kluwer Academic Publishers, 1995.

[GBF+06] Minos N. Garofalakis, Kurt P. Brown, Michael J. Franklin, Joseph M. Hellerstein, Daisy Zhe Wang, Eirinaios Michelakis, Liviu Tancau, Eugene Wu, Shawn R. Jeffery, and Ryan Aipperspach. Probabilistic data management for pervasive computing: The *data furnace* project. *IEEE Data Eng. Bull.*, 29(1):57–63, 2006.

[GBKS09] Wolfgang Gatterbauer, Magdalena Balazinska, Nodira Khoussainova, and Dan Suciu. Believe it or not: Adding belief annotations to databases. *PVLDB*, 2(1):1–12, 2009.

[GC06] Ruibin Gong and Tony Kai Yun Chan. Syllable alignment: A novel model for phonetic string search. *IEICE Transactions*, 89-D(1):332–339, 2006.

[GCC12] Jian Gong, Reynold Cheng, and David W. Cheung. Efficient management of uncertainty in xml schema matching. *VLDB J.*, 21(3):385–409, 2012.

[GD04] Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *CVPR (1)*, pages 220–227, 2004.

[GD05] Kristen Grauman and Trevor Darrell. Efficient image matching with distributions of local invariant features. In *CVPR (2)*, pages 627–634, 2005.

[GFS+01] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-Augustin Saita. Declarative data cleaning: Language, model, and algorithms. In *VLDB*, pages 371–380, 2001.

[GFSS00] Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. Ajax: An extensible data cleaning tool. In *SIGMOD Conference*, page 590, 2000.

[GI89] Daniel M. Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. The MIT Press, 1989.

[Gil97] Leicester E. Gill. OX-LINK: The Oxford medical record linkage system. In *Proceedings of the International Record Linkage Workshop and Exposition*, pages 15–33, 1997.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman and Company, 1979.

[GJS10] Wolfgang Gatterbauer, Abhay Kumar Jha, and Dan Suciu. Dissociation and propagation for efficient query evaluation over probabilistic databases. In *MUD*, pages 83–97, 2010.

[GKMS04] Sudipto Guha, Nick Koudas, Amit Marathe, and Divesh Srivastava. Merging the results of approximate match operations. In *VLDB*, pages 636–647, 2004.

[GM12] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: Theory, practice & open challenges. *PVLDB*, 5(12):2018–2019, 2012.

[GM13] Lise Getoor and Ashwin Machanavajjhala. Entity resolution for big data. In *KDD*, page 1527, 2013.

[GMSS09] Avigdor Gal, Maria Vanina Martinez, Gerardo I. Simari, and V. S. Subrahmanian. Aggregate query answering under uncertain schema mappings. In *ICDE*, pages 940–951, 2009.

[GMT07] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[GOT13a] Gösta Grahne, Adrian Onet, and Nihat Tartal. Conditional tables in practice. *CoRR*, abs/1304.0959, 2013.

[GOT13b] Gösta Grahne, Adrian Onet, and Nihat Tartal. Possdb: An uncertainty database management system. In *SUM*, pages 247–254, 2013.

[GPW14] Sergio Greco, Fabian Pijcke, and Jef Wijsen. Certain query answering in partially consistent databases. *PVLDB*, 7(5):353–364, 2014.

[Gra79] John Grant. Partial values in a tabular database model. *Inf. Process. Lett.*, 9(2):97–99, 1979.

[Gra84] Gösta Grahne. Dependency satisfaction in databases with incomplete information. In *VLDB*, pages 37–45, 1984.

[Gra91] Gösta Grahne. *The Problem of Incomplete Information in Relational Databases*, volume 554 of *Lecture Notes in Computer Science*. Springer, 1991.

[Gra09] Gösta Grahne. Conditional tables. In *Encyclopedia of Database Systems*, pages 446–447. 2009.

[Gre09] Todd J. Green. Models for incomplete and probabilistic information. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*. Springer, 2009.

[GS06] Rahul Gupta and Sunita Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, pages 965–976, 2006.

[GS13] Venkatesh Ganti and Anish Das Sarma. *Data Cleaning: A Practical Perspective*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.

[GT96] Michael Greiner and Gottfried Tinhofer. *Stochastik für Studienanfänger der Informatik*. Carl Hanser Verlag, 1996.

[GUP06] Jose Galindo, Angelica Urrutia, and Mario Piattini. *Fuzzy Databases - Modeling, Design and Implementation*. Idea Group Publishing, 2006.

[GZM09] Tingjian Ge, Stanley B. Zdonik, and Samuel Madden. Top-*k* queries on uncertain data: on score distribution and typical answers. In *SIGMOD Conference*, pages 375–388, 2009.

[HA85] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, pages 193–218, 1985.

[HAKO09] Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. MayBMS: a probabilistic database management system. In *SIGMOD Conference*, pages 1071–1074, 2009.

[Hal90] Joseph Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[Har12] Thorben Harms. Evaluation von duplikatenclustering methoden an realen und generierten datensätzen. Master's thesis, University of Hamburg, 2012.

[Haz01] Michiel Hazewinkel. *Composite function*. Encyclopedia of Mathematics. Springer, 2001.

[HB11] Tom Heath and Christian Bizer. *Linked Data - Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2011.

[HCML09] Oktie Hassanzadeh, Fei Chiang, Renée J. Miller, and Hyun Chul Lee. Framework for evaluating clustering algorithms in duplicate detection. *PVLDB*, 2(1):1282–1293, 2009.

[HD96] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *Int. J. Approx. Reasoning*, 15(3):225–263, 1996.

[HDI12] Alon Halevy, AnHai Doan, and Zachary G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[HG93] Lee Harvey and Diana Green. Defining quality. *Assessment and Evaluation in Higher Education*, 18:9–34, 1993.

[HGI00] Taher H. Haveliwala, Aristides Gionis, and Piotr Indyk. Scalable techniques for clustering the web. In *WebDB (Informal Proceedings)*, pages 129–134, 2000.

[HGS03] Edward Hung, Lise Getoor, and V. S. Subrahmanian. Pxml: A probabilistic semistructured data model and algebra. In *ICDE*, pages 467–478, 2003.

[HGS07] Edward Hung, Lise Getoor, and V. S. Subrahmanian. Probabilistic interval xml. *ACM Trans. Comput. Log.*, 8(4), 2007.

[HHMO+06] H.Kawai, H.Garcia-Molina, O.Benjelloun, D. Menestrina, E. Whang, and H.Gong. P-swoosh: Parallel algorithm for generic entity resolution. Technical report, Department of Computer Science, Stanford University, 2006.

[HHvK13] Hans Henseler, Jop Hofste, and Maurice van Keulen. Digital-forensics based pattern recognition for discovering identities in electronic evidence. In *EISIC*, pages 112–116, 2013.

[Hin62] Jaakko Hintikka. *Knowledge and Belief - An Introduction to the Logic of the Two Notions*. Cornell University Press: Ithaca, NY., 1962.

[Hit41] F. L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematical Physics*, 20:224–230, 1941.

[HKK07] Bernd Heinrich, Marcus Kaiser, and Mathias Klier. Metrics for Measuring Data Quality - Foundations for an Economic Data Quality Management. In *ICSOFT (ISDM/EHST/DC)*, pages 87–94, 2007.

[HKP06] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan Kaufmann, second edition, 2006.

[HL90] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Mathematical Programming*. Mcgraw-Hill, 1990.

[HM09] Oktie Hassanzadeh and Renée J. Miller. Creating probabilistic databases from duplicated data. *VLDB J.*, 18(5):1141–1166, 2009.

[HMH01] Mauricio A. Hernández, Renée J. Miller, and Laura M. Haas. Clio: A Semi-Automatic Tool For Schema Mapping. In *SIGMOD Conference*, page 607, 2001.

[HNST12] Melanie Herschel, Felix Naumann, Sascha Szott, and Maik Taubert. Scalable iterative graph duplicate detection. *IEEE Trans. Knowl. Data Eng.*, 24(11):2094–2108, 2012.

[HPZL08] Ming Hua, Jian Pei, Wenjie Zhang, and Xuemin Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *SIGMOD Conference*, pages 673–686, 2008.

[HRO06] Alon Y. Halevy, Anand Rajaraman, and Joann J. Ordille. Data Integration: The Teenage Years. In *VLDB*, pages 9–16, 2006.

[HS95] Mauricio A. Hernández and Salvatore J. Stolfo. The Merge/Purge Problem for Large Databases. In *SIGMOD Conference*, pages 127–138, 1995.

[HS98] Mauricio A. Hernández and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.

[HSM08] Robert Hall, Charles A. Sutton, and Andrew McCallum. Unsupervised deduplication using cross-field dependencies. In *KDD*, pages 310–317, 2008.

[HSW07] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. *Data quality and record linkage techniques*. Springer, 2007.

[HTF11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*. Springer, 2 edition, 2011.

[Hub06] Franz Huber. Belief and degrees of belief. Technical report, California Institute of Technology, 2006.

[HvK10] Emiel S. Hollander and Maurice van Keulen. Storing and querying probabilistic xml using a probabilistic relational dbms. In *MUD*, pages 35–49, 2010.

[HvK12] Mena B. Habib and Maurice van Keulen. Improving toponym disambiguation by iteratively enhancing certainty of extraction. In *KDIR*, pages 399–410, 2012.

[HW79] J. A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society*, 28:100–108, 1979.

[IL84] Tomasz Imielinski and Witold Lipski. Incomplete Information in Relational Databases. *J. ACM*, 31(4):761–791, 1984.

[Imi89] Tomasz Imielinski. Incomplete Information in Logical Databases. *IEEE Data Eng. Bull.*, 12(2):29–40, 1989.

[INNV10] Ekaterini Ioannou, Wolfgang Nejdl, Claudia Niederée, and Yannis Velegrakis. On-the-fly entity-aware query processing in the presence of linkage. *PVLDB*, 3(1):429–438, 2010.

[INNV11] Ekaterini Ioannou, Wolfgang Nejdl, Claudia Niederée, and Yannis Velegrakis. Linkdb: a probabilistic linkage database system. In *SIGMOD Conference*, pages 1307–1310, 2011.

[INV91] Tomasz Imielinski, Shamim A. Naqvi, and Kumar V. Vadaparty. Incomplete objects - a data model for design and planning applications. In James Clifford and Roger King, editors, *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data, Denver, Colorado, May 29-31, 1991*, pages 288–297. ACM Press, 1991.

[Ioa11] Ekaterini Ioannou. *Entity linkage for heterogeneous, uncertain, and volatile data*. PhD thesis, 2011.

[IS11] Ihab F. Ilyas and Mohamed A. Soliman. *Probabilistic Ranking Techniques in Relational Databases*. Morgan & Claypool Publishers, 2011.

[ISO11] Accuracy (trueness and precision) of measurement methods and results — part 1: Introduction and basic principles, 2011.

[Jae13] Manfred Jaeger. Relational bayesian networks. *CoRR*, abs/1302.1550, 2013.

[Jar85] M.A. Jaro. Advances in Record Linkage Methodologies as Applied to Matching the 1985 Census of Tampa Bay, Florida. *Journal of American Statistical Society 84*, 406:414–420, 1985.

[Jar89] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[JH11] Micheline Kamber Jiawei Han, editor. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[JIB10] Anja Jentzsch, Robert Isele, and Christian Bizer. Silk - generating rdf links while publishing or consuming linked data. In *ISWC Posters&Demos*, 2010.

[JKV07] T. S. Jayram, Satyen Kale, and Erik Vee. Efficient aggregation algorithms for probabilistic data. In *SODA*, pages 346–355, 2007.

[JLM03] Liang Jin, Chen Li, and Sharad Mehrotra. Efficient record linkage in large data sets. In *DASFAA*, pages 137–, 2003.

[JMMV08] T. S. Jayram, Andrew McGregor, S. Muthukrishnan, and Erik Vee. Estimating statistical aggregates on probabilistic data streams. *ACM Trans. Database Syst.*, 33(4), 2008.

[JP11] Bin Jiang and Jian Pei. Outlier detection on uncertain data: Objects, instances, and inferences. In *ICDE*, pages 422–433, 2011.

[JPTL13] Bin Jiang, Jian Pei, Yufei Tao, and Xuemin Lin. Clustering uncertain data based on probability distribution similarity. *IEEE Trans. Knowl. Data Eng.*, 25(4):751–763, 2013.

[JXW+08] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Christopher M. Jermaine, and Peter J. Haas. Mcdb: a monte carlo approach to managing uncertain data. In *SIGMOD Conference*, pages 687–700, 2008.

[JXW+11] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Chris Jermaine, and Peter J. Haas. The monte carlo database system: Stochastic analysis close to the data. *ACM Trans. Database Syst.*, 36(3):18, 2011.

[Kan11] Mehmed Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. Wiley-IEEE Press, 2011.

[KBRZ09] Hans-Peter Kriegel, Thomas Bernecker, Matthias Renz, and Andreas Zuefle. Probabilistic Join Queries in Uncertain Databases. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 9, pages 257–298. Springer, 2009.

[KBS08] Nodira Khoussainova, Magdalena Balazinska, and Dan Suciu. Probabilistic event extraction from rfid data. In *ICDE*, pages 1480–1482, 2008.

[KE99] Alfons Kemper and Andre Eickler. *Datenbanksysteme. Eine Einführung*. Oldenbourg Wissenschaftsverlag, 1999.

[KG09] Batya Kenig and Avigdor Gal. Efficient entity resolution with mfiblocks. In *VLDB*, 2009.

[KK10] Oliver Kennedy and Christoph Koch. Pip: A database system for great and small expectations. In *ICDE*, pages 157–168, 2010.

[KKPR06] Hans-Peter Kriegel, Peter Kunath, Martin Pfeifle, and Matthias Renz. Probabilistic similarity join on uncertain data. In *DASFAA*, pages 295–309, 2006.

[KL51] Solomon Kullback and Richard Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

[Kle94] Hans-Joachim Klein. How to modify sql queries in order to guarantee sure answers. *SIGMOD Record*, 23(3):14–20, 1994.

[Kle01] Hans-Joachim Klein. Null values in relational databases and sure information answers. In *Semantics in Databases*, pages 119–138, 2001.

[KLM89] Richard M. Karp, Michael Luby, and Neal Madras. Monte-carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.

[KML10] Mohamed E. Khalefa, Mohamed F. Mokbel, and Justin J. Levandoski. Skyline query processing for uncertain data. In *CIKM*, pages 1293–1296, 2010.

[KMS07] Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *STOC*, pages 95–103, 2007.

[KMZ10] Hideaki Kimura, Samuel Madden, and Stanley B. Zdonik. Upi: A primary index for uncertain databases. *PVLDB*, 3(1):630–637, 2010.

[KNS10] Evgeny Kharlamov, Werner Nutt, and Pierre Senellart. Updating probabilistic xml. In *EDBT/ICDT Workshops*, 2010.

[KO08] Christoph Koch and Dan Olteanu. Conditioning Probabilistic Databases. *CoRR*, abs/0803.2212, 2008.

[Koc09] Christoph Koch. MayBMS: A System for Managing Large Uncertain and Probabilistic Databases. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*. Springer, 2009.

[Kol60] Andrei Nikolajewitsch Kolmogorow. *Foundations of the theory of probability*. Chelsea Pub Co, 1960.

[KP05] Hans-Peter Kriegel and Martin Pfeifle. Density-based clustering of uncertain data. In *KDD*, pages 672–677, 2005.

[KR01] Kristian Kersting and Luc De Raedt. Bayesian logic programs. *CoRR*, cs.AI/0111058, 2001.

[KR11] Arun Kumar and Christopher Ré. Probabilistic management of ocr data using an rdbms. *PVLDB*, 5(4):322–333, 2011.

[Kri63] Saul A. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[KS80] Ross Kindermann and Laurie J. Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980.

[KS13] Benny Kimelfeld and Pierre Senellart. Probabilistic xml: Models and complexity. In *Advances in Probabilistic Databases for Uncertain Information Management*, pages 39–66. 2013.

[KTG92] Werner Kießling, Helmut Thöne, and Ulrich Güntzer. Database support for problematic knowledge. In *EDBT*, pages 421–436, 1992.

[KTR10] Lars Kolb, Andreas Thor, and Erhard Rahm. Parallel sorted neighborhood blocking with mapreduce. *CoRR*, abs/1010.3053, 2010.

[KTR12] Lars Kolb, Andreas Thor, and Erhard Rahm. Multi-pass sorted neighborhood blocking with mapreduce. *Computer Science - R&D*, 27(1):45–63, 2012.

[Kuh55] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[LB01] Elizaveta Levina and Peter J. Bickel. The earth mover's distance is the mallows distance: Some insights from statistics. In *ICCV*, pages 251–256, 2001.

[LC10] Xiang Lian and Lei Chen. Set similarity join on probabilistic data. *PVLDB*, 3(1):650–659, 2010.

[LC11a] Luís Leitão and Pável Calado. Duplicate detection through structure optimization. In *CIKM*, pages 443–452, 2011.

[LC11b] Xiang Lian and Lei Chen. Similarity join processing on uncertain data streams. *IEEE Trans. Knowl. Data Eng.*, 23(11):1718–1734, 2011.

[LC13] Luís Leitão and Pável Calado. Efficient xml duplicate detection using an adaptive two-level optimization. In *SAC*, pages 832–837, 2013.

[LCH13a] Trieu Minh Nhut Le, Jinli Cao, and Zhen He. Top-k best probability queries and semantics ranking properties on probabilistic databases. *Data Knowl. Eng.*, 88:248–266, 2013.

[LCH13b] Luís Leitão, Pável Calado, and Melanie Herschel. Efficient and effective duplicate detection in hierarchical data. *IEEE Trans. Knowl. Data Eng.*, 25(5):1028–1041, 2013.

[LCW07] Luís Leitão, Pável Calado, and Melanie Weis. Structure-based inference of xml similarity for fuzzy duplicate detection. In *CIKM*, pages 293–302, 2007.

[LD09] Jian Li and Amol Deshpande. Consensus answers for queries over probabilistic databases. In *PODS*, pages 259–268, 2009.

[LeB04] David C. LeBlanc. *Statistics: Concepts and Applications for Science*, volume 2. Jones & Bartlett Learning, 2004.

[Leh03] Wolfgang Lehner. *Datenbanktechnologie für Data-Warehouse-Systeme. Konzepte und Methoden*. Dpunkt Verlag, 2003.

[Leh06] Patrick Lehti. *Unsupervised Duplicate Detection Using Sample Non-Duplicates*. PhD thesis, TU Darmstadt, 2006.

[Leh09] Sebastian Lehrack. A quantum logic-based query processing approach for extending relational query languages. In *Grundlagen von Datenbanken*, pages 63–67, 2009.

[Len02] Maurizio Lenzerini. Data Integration: A Theoretical Perspective. In *PODS*, pages 233–246, 2002.

[Lev66] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[Lew86] David Lewis. *On the Plurality of Worlds*. Oxford & New York: Basil Blackwell, 1986.

[LF05a] Patrick Lehti and Peter Fankhauser. A precise blocking method for record linkage. In *DaWaK*, pages 210–220, 2005.

[LF05b] Patrick Lehti and Peter Fankhauser. Probabilistic iterative duplicate detection. In *OTM Conferences (2)*, pages 1225–1242, 2005.

[LLL01] Wai Lup Low, Mong-Li Lee, and Tok Wang Ling. A knowledge-based approach for duplicate elimination in data cleaning. *Inf. Syst.*, 26(8):585–606, 2001.

[LLRS97] Laks V. S. Lakshmanan, Nicola Leone, Robert B. Ross, and V. S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Trans. Database Syst.*, 22(3):419–469, 1997.

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[LN06] Ulf Leser and Felix Naumann. *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt, 2006.

[Log04] Beth Logan. Music recommendation from song sets. In *ISMIR*, 2004.

[LS94a] Laks V. S. Lakshmanan and Fereidoon Sadri. Modeling uncertainty in deductive databases. In *DEXA*, pages 724–733, 1994.

[LS94b] Laks V. S. Lakshmanan and Fereidoon Sadri. Probabilistic deductive databases. In *ILPS*, pages 254–268, 1994.

[LS97] Laks V. S. Lakshmanan and Fereidoon Sadri. Uncertain deductive databases: A hybrid approach. *Inf. Syst.*, 22(8):483–508, 1997.

[LS01] Laks V. S. Lakshmanan and Fereidoon Sadri. On a theory of probabilistic deductive databases. *TPLP*, 1(1):5–42, 2001.

[LS07] Vebjorn Ljosa and Ambuj K. Singh. Apla: Indexing arbitrary probability distributions. In *ICDE*, pages 946–955, 2007.

[LS09] Vebjorn Ljosa and Ambuj K. Singh. Probabilistic Querying and Mining of Biological Images. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 16, pages 461–488. Springer, 2009.

[LSD09] Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *CoRR*, abs/0904.1366, 2009.

[LSPR93] Ee-Peng Lim, Jaideep Srivastava, Satya Prabhakar, and James Richardson. Entity identification in database integration. In *ICDE*, pages 294–301, 1993.

[LSS96a] Laks V. S. Lakshmanan, Fereidoon Sadri, and Iyer N. Subramanian. Schemasql - a language for interoperability in relational multi-database systems. In *VLDB*, pages 239–250, 1996.

[LSS96b] Ee-Peng Lim, Jaideep Srivastava, and Shashi Shekhar. An evidential reasoning approach to attribute value conflict resolution in database integration. *IEEE Trans. Knowl. Data Eng.*, 8(5):707–723, 1996.

[LSS99] Laks V. S. Lakshmanan, Fereidoon Sadri, and Subbu N. Subramanian. On efficiently implementing schemasql on an sql database system. In *VLDB*, pages 471–482, 1999.

[LSS01] Laks V. S. Lakshmanan, Fereidoon Sadri, and Subbu N. Subramanian. Schemasql: An extension to sql for multidatabase interoperability. *ACM Trans. Database Syst.*, 26(4):476–519, 2001.

[LSS12] Sebastian Lehrack, Sascha Saretz, and Ingo Schmitt. Qsql2: Query language support for logic-based similarity conditions on probabilistic databases. In *RCIS*, pages 1–12, 2012.

[LSV02] Jens Lechtenbörger, Hua Shu, and Gottfried Vossen. Aggregate queries over conditional tables. *J. Intell. Inf. Syst.*, 19(3):343–362, 2002.

[LSW13] Sebastian Lehrack, Sascha Saretz, and Christian Winkel. Proqua: a system for evaluating logic-based scoring functions on uncertain relational data. In *EDBT*, pages 761–764, 2013.

[Luo05] Xiaoqiang Luo. On coreference resolution performance metrics. In *HLT/EMNLP*, 2005.

[LW93] Leonid Libkin and Limsoon Wong. Semantic representations and query languages for or-sets. In Catriel Beeri, editor, *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC, USA*, pages 37–48. ACM Press, 1993.

[MA06] Amihai Motro and Philipp Anokhin. Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources. *Information Fusion*, 7(2):176–196, 2006.

[Mag08] Federico Maggi. A survey of probabilistic record matching models, techniques and tools. Technical report, Politecnico di Milano, 2008.

[Mal72] C. L. Mallows. A note on asymptotic joint normality. *The Annals of Mathematical Statistics*, 43(2):508–515, 4 1972.

[Maz10] David R. Mazur. *Combinatorics: A Guided Tour*. MAA, 2010.

[MBGM06] David Menestrina, Omar Benjelloun, and Hector Garcia-Molina. Generic entity resolution with data confidences. In *CleanDB*, 2006.

[ME96] Alvaro E. Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *KDD*, pages 267–270, 1996.

[ME97] Alvaro E. Monge and Charles Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *DMKD*, pages 0–, 1997.

[Mei03] Marina Meila. Comparing clusterings by the variation of information. In *COLT*, pages 173–187, 2003.

[Mei07] Marina Meila. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98:873 – 895, 2007.

[MG88] Zbigniew Michalewicz and Lindsay Groves. Sets and Uncertainty in Relational Databases. In *IPMU*, pages 127–137, 1988.

[MG07] Anan Marie and Avigdor Gal. Managing uncertainty in schema matcher ensembles. In *SUM*, pages 60–73, 2007.

[MIW11] Raghotham Murthy, Robert Ikeda, and Jennifer Widom. Making aggregation work in uncertain and probabilistic databases. *IEEE Trans. Knowl. Data Eng.*, 23(8):1261–1273, 2011.

[MK77] G B. Moore and J.L. Kuhns. Accessing individual records from personal data files using nonunique identifiers. Technical report, 1977.

[MK06] Matthew Michelson and Craig A. Knoblock. Learning blocking schemes for record linkage. In *AAAI*, 2006.

[MKM98] Robert Mandelbaum, G. Kamberova, and Max Mintz. Stereo depth estimation: A confidence interval approach. In *ICCV*, pages 503–509, 1998.

[MM07] Matteo Magnani and Danilo Montesi. Uncertainty in data integration: current approaches and open problems. In *MUD*, pages 18–32, 2007.

[MM08a] Matteo Magnani and Danilo Montesi. Management of interval probabilistic data. *Acta Inf.*, 45(2):93–130, 2008.

[MM08b] Matteo Magnani and Danilo Montesi. Preference-based uncertain data integration. In *EKAW*, pages 136–145, 2008.

[MM09a] Matteo Magnani and Danilo Montesi. Probabilistic data integration. Technical report, University of Bologna, 2009.

[MM09b] Matteo Magnani and Danilo Montesi. Towards relational schema uncertainty. In *SUM*, pages 150–164, 2009.

[MM10a] Matteo Magnani and Danilo Montesi. A survey on uncertainty management in data integration. *J. Data and Information Quality*, 2(1), 2010.

[MM10b] Matteo Magnani and Danilo Montesi. Us-sql: managing uncertain schemata. In *SIGMOD Conference*, pages 1195–1198, 2010.

[MNHG12] Hitesh Maidasani, Galileo Namata, Bert Huang, and Lise Getoor. Entity resolution evaluation measures. Technical report, University of Maryland, 2012.

[MNU00] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, pages 169–178, 2000.

[Mok11] Hoda M. O. Mokhtar. Data warehouses for uncertain data. *JOURNAL OF COMPUTING,*, 3(6), 2011.

[MPM94] Juan Miguel Medina, Olga Pons, and María Amparo Vila Miranda. GEFRED: A Generalized Model of Fuzzy Relational Databases. *Inf. Sci.*, 76(1-2):87–109, 1994.

[MR06] Matteo Magnani and Nikos Rizopoulos. Uncertain schema matching. In *SEBD*, pages 281–288, 2006.

[MRMM05] Matteo Magnani, Nikos Rizopoulos, Peter McBrien, and Danilo Montesi. Schema integration based on uncertain semantic mappings. In *ER*, pages 31–46, 2005.

[MSS01] Sally I. McClean, Bryan W. Scotney, and Mary Shapcott. Aggregation of imprecise and uncertain information in databases. *IEEE Trans. Knowl. Data Eng.*, 13(6):902–912, 2001.

[MTdK+07] Michi Mutsuzaki, Martin Theobald, Ander de Keijzer, Jennifer Widom, Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Raghotham Murthy, and Tomoe Sugihara. Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS (Demo). In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pages 269–274, 2007.

[Mug00] Stephen Muggleton. Learning stochastic logic programs. *Electron. Trans. Artif. Intell.*, 4(B):141–153, 2000.

[MW04] Andrew McCallum and Ben Wellner. Conditional Models of Identity Uncertainty with Application to Noun Coreference. In *NIPS*, 2004.

[MW07] Raghotham Murthy and Jennifer Widom. Making aggregation work in uncertain and probabilistic databases. In *MUD*, pages 76–90, 2007.

[MWGM10] David Menestrina, Steven Whang, and Hector Garcia-Molina. Evaluating entity resolution results. *PVLDB*, 3(1):208–219, 2010.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[NAV12] Hamideh Afsarmanesh Naser Ayat, Reza Akbarinia and Patrick Valduriez. Entity resolution for uncertain data. In *28e journées Bases de Données Avancées*, 2012.

[NBE+93] Wayne Niblack, Ron Barber, William Equitz, Myron Flickner, Eduardo H. Glasman, Dragutin Petkovic, Peter Yanker, Christos Faloutsos, and Gabriel Taubin. The qbic project: Querying images by content, using color, texture, and shape. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 173–187, 1993.

[NDB69] Jr. N.S.D'Andrea Du Bois. A solution to the problem of linking multivariate documents. *Journal of the American Statistical Association*, 64(325):163–174, March 1969.

[NH97] Liem Ngo and Peter Haddaway. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[NH10] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[NJ02] Andrew Nierman and H. V. Jagadish. Protdb: Probabilistic data in xml. In *VLDB*, pages 646–657, 2002.

[NJ09] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian Networks and Decision Graphs*. Springer Science & Business Media, 2009.

[NK62] Howard B. Newcombe and James M. Kennedy. Record linkage: making maximum use of the discriminating power of identifying information. *Commun. ACM*, 5(11):563–566, November 1962.

[NKAJ59] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic Linkage of Vital Records. *Science*, 130:954–959, October 1959.

[NKC+06] Wang Kay Ngai, Ben Kao, Chun Kit Chui, Reynold Cheng, Michael Chau, and Kevin Y. Yip. Efficient clustering of uncertain data. In *ICDM*, pages 436–445, 2006.

[NMMMBLP07a] Jordi Nin, Victor Muntés-Mulero, Norbert Martínez-Bazan, and Josep-Lluis Larriba-Pey. On the use of semantic blocking techniques for data cleansing and integration. In *IDEAS*, pages 190–198, 2007.

[NMMMBLP07b] Jordi Nin, Victor Muntés-Mulero, Norbert Martínez-Bazan, and Josep-Lluis Larriba-Pey. Semantic blocking for record linkage. In *CCIA*, pages 141–149, 2007.

[NS92] Raymond T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Inf. Comput.*, 101(2):150–201, 1992.

[NT07] Jordi Nin and Vicenc Torra. Blocking anonymized data. In *Proc. of the AGOP*, pages 83–87, 2007.

[NW70] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[NZE+13] Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamoulis, Lei Chen, and Hans-Peter Kriegel. Probabilistic nearest neighbor queries on uncertain moving object trajectories. *PVLDB*, 7(3):205–216, 2013.

[OHK09] Dan Olteanu, Jiewen Huang, and Christoph Koch. Sprout: Lazy vs. eager query plans for tuple-independent probabilistic databases. In *ICDE*, pages 640–651, 2009.

[OHK10] Dan Olteanu, Jiewen Huang, and Christoph Koch. Approximate confidence computation in probabilistic databases. In *ICDE*, pages 145–156, 2010.

[ORP98] Nuria Oliver, Barbara Rosario, and Alex Pentland. Statistical modeling of human interactions. In *Symposium on Computer Vision and Pattern Recognition*, page 39–46, 1998.

[Pan09a] Fabian Panse. Completeness of attribute values representing partial information. In *Proceedings of the 7th International Workshop on Quality in Databases at VLDB 2009*, 8 2009. Workshop-Beiträge nur als PDFs auf Konferenz eigenem USB-Stick.

[Pan09b] Fabian Panse. Datenunvollständigkeit aufgrund der mangelnden modellierungsmächtigkeit aktuell dominierender datenmodelle. In *Grundlagen von Datenbanken*, pages 123–127, 2009.

[PBGK10] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. k-nearest neighbors in uncertain graphs. *PVLDB*, 3(1):997–1008, 2010.

[PD04] Parag and Pedro Domingos. Multi-relational record linkage. In *KDD Workshop on Multi-Relational Data Mining*, pages 31–48, 2004.

[Pea00] Karl Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonbly supposed to have arisen from random sampling. *Philosophical Magazine*, L:157–172, 1900.

[Pea88] Judaea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Pet96] Frederick E. Petry. *Fuzzy Databases - Principles and Applications*. Kluwer Academic Publishers, 1996.

[PFW05] Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of audio-based music similarity and genre classificaton. In *ISMIR*, pages 628–633, 2005.

[PG98] Witold Pedrycz and Fernando Gomide. *An Introduction to Fuzzy Sets - Analysis and Design*. The MIT Press, 1998.

[Phi90] Lawrence Philips. Hanging on the metaphone. *Computer Language Magazine*, 7(12):39–44, December 1990.

[Phi00] Lawrence Philips. The double metaphone search algorithm. *C/C++ Users J.*, 18(6):38–43, June 2000.

[PIN+11] George Papadakis, Ekaterini Ioannou, Claudia Niederée, Themis Palpanas, and Wolfgang Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *JCDL*, pages 85–94, 2011.

[PINF11] George Papadakis, Ekaterini Ioannou, Claudia Niederée, and Peter Fankhauser. Efficient entity resolution for large heterogeneous information spaces. In *WSDM*, pages 535–544, 2011.

[Pit94] Michael Pittarelli. An algebra for probabilistic databases. *IEEE Trans. Knowl. Data Eng.*, 6(2):293–303, 1994.

[PJLY07] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. Probabilistic skylines on uncertain data. In *VLDB*, pages 15–26, 2007.

[PLSM09] Clifton Phua, Vincent C. S. Lee, and Kate Smith-Miles. The personal name problem and a data mining solution. In *Encyclopedia of Data Warehousing and Mining*, pages 1524–1531. 2009.

[PMM+02] Hanna Pasula, Bhaskara Marthi, Brian Milch, Stuart J. Russell, and Ilya Shpitser. Identity uncertainty and citation matching. In *NIPS*, pages 1401–1408, 2002.

[Poo93] David Poole. Probabilistic horn abduction and bayesian networks. *Artif. Intell.*, 64(1):81–129, 1993.

[Poo97] David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.*, 94(1-2):7–56, 1997.

[Por94] Sidney C. Port. *Theoretical Probability for Applications*. Series in Probability & Mathematical Statistics. John Wiley & Sons Inc, 1994.

[Pos69] Hans Joachim Postel. Die kölner phonetik. ein verfahren zur identifizierung von personennamen auf der grundlage der gestaltanalyse. *IBM-Nachrichten*, pages 925–931, 1969.

[PR09] Fabian Panse and Norbert Ritter. Completeness in databases with maybe-tuples. In *ER Workshops*, pages 202–211, 2009.

[PR10a] Fabian Panse and Norbert Ritter. Relational data completeness in the presence of maybe-tuples. *Ingénierie des Systèmes d'Information*, 15(6):85–104, 2010.

[PR10b] Fabian Panse and Norbert Ritter. Towards duplicate detection and data fusion in fuzzy relational databases. *Berichte des Departments Informatik der Universität Hamburg*, 10(Bericht 292):1–13, 4 2010.

[PR10c] Fabian Panse and Norbert Ritter. Tuple merging in probabilistic databases. In *Proceedings of Workshop on Management of Uncertain Data*, 2010.

[PR11] Fabian Panse and Norbert Ritter. Incorporating domain knowledge and user expertise in probabilistic tuple merging. In *SUM*, pages 289–302, 2011.

[PR12] Fabian Panse and Norbert Ritter. Evaluating indeterministic duplicate detection results. In *SUM*, pages 433–446, 2012.

[PSS09] Sunil Prabhakar, Rahul Shah, and Sarvjeet Singh. Indexing uncertain data. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 10, pages 299–326. Springer, 2009.

[PT13] Ofir Pele and Ben Taskar. The tangent earth mover's distance. In *GSI*, pages 397–404, 2013.

[PvKdKR10] Fabian Panse, Maurice van Keulen, Ander de Keijzer, and Norbert Ritter. Duplicate Detection in Probabilistic Data. In *Proceedings of the 2nd Workshop on New Trends in Information Integration (NTII 2010) co-located with ICDE 2010*, pages 179–182, 2010.

[PvKR13] Fabian Panse, Maurice van Keulen, and Norbert Ritter. Indeterministic Handling of Uncertain Decisions in Deduplication. *J. Data and Information Quality*, 4(2):9, 2013.

[PW08] Ofir Pele and Michael Werman. A linear time histogram metric for improved sift matching. In *ECCV (3)*, pages 495–508, 2008.

[PW09] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *ICCV*, pages 460–467, 2009.

[PWCC97] Edward H. Porter, William E. Winkler, Bureau Of The Census, and Bureau Of The Census. Approximate string comparison and its effect on an advanced record linkage system. In *Advanced Record Linkage System. U.S. Bureau of the Census, Research Report*, pages 190–199, 1997.

[PWFR12] Fabian Panse, Wolfram Wingerath, Steffen Friedrich, and Norbert Ritter. Key-based blocking of duplicates in entity-independent probabilistic data. In *The 17th International Conference on Information Quality (ICIQ)*, 2012.

[PWN06] Sven Puhlmann, Melanie Weis, and Felix Naumann. Xml duplicate detection using sorted neighborhoods. In *EDBT*, pages 773–791, 2006.

[PWR89] Shmuel Peleg, Michael Werman, and Hillel Rom. A unified approach to the change of resolution: Space and gray-level. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):739–742, 1989.

[QPB13] Disheng Qiu, Paolo Papotti, and Lorenzo Blanco. Future locations prediction with uncertain data. In *ECML/PKDD (1)*, pages 417–432, 2013.

[Qui83] John Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine Learning. An Artificial Intelligence Approach*, pages 463–482, 1983.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[QXPT09] Biao Qin, Yuni Xia, Sunil Prabhakar, and Yi-Cheng Tu. A rule-based classification algorithm for uncertain data. In *ICDE*, pages 1633–1640, 2009.

[Rac84] S. T. Rachev. The monge–kantorovich mass transference problem and its stochastic applications. *Theory of Probability and its Applications*, 29(4):647–676, 1984.

[Ran71] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[RCK+10] Matthias Renz, Reynold Cheng, Hans-Peter Kriegel, Andreas Züfle, and Thomas Bernecker. Similarity search and mining in uncertain databases. *PVLDB*, 3(2):1653–1654, 2010.

[RDG11] Vibhor Rastogi, Nilesh N. Dalvi, and Minos N. Garofalakis. Large-scale collective entity matching. *PVLDB*, 4(4):208–218, 2011.

[RDS07] Christopher Re, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, pages 886–895, 2007.

[Rei77] Raymond Reiter. On closed world data bases. In *Logic and Data Bases*, pages 55–76, 1977.

[Rem12] Steffen Remus. Automatically identifying lexical chains by means of statistical methods — a knowledge-free approach. Master's thesis, Technical University of Darmstadt, 2012.

[RH07] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, pages 410–420, 2007.

[RLBS08] Christopher Ré, Julie Letchner, Magdalena Balazinska, and Dan Suciu. Event queries on correlated probabilistic streams. In *SIGMOD Conference*, pages 715–728, 2008.

[Ros11] Walter A. Rosenkrantz. *Introduction to Probability and Statistics for Science, Engineering, and Finance*. CRC Press, 2011.

[Rot64] Gian Rota. The Number of Partitions of a Set. *The American Mathematical Monthly*, 71(5):498–504, 1964.

[Rot96] Dan Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2):273–302, 1996.

[RR09] Roi Reichart and Ari Rappoport. The nvi clustering evaluation measure. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*, 2009.

[RS07] Christopher Re and Dan Suciu. Materialized views in probabilistic databases for information exchange and query optimization. In *VLDB*, pages 51–62, 2007.

[RS08] Christopher Ré and Dan Suciu. Managing probabilistic data with mystiq: The can-do, the could-do, and the can't-do. In *SUM*, pages 5–18, 2008.

[RS09] Christopher Ré and Dan Suciu. The trichotomy of having queries on a probabilistic database. *VLDB J.*, 18(5):1091–1116, 2009.

[RSG02] Robert B. Ross, V. S. Subrahmanian, and John Grant. Probabilistic aggregates. In *ISMIS*, pages 553–564, 2002.

[RSG05] Robert B. Ross, V. S. Subrahmanian, and John Grant. Aggregate operators in probabilistic databases. *J. ACM*, 52(1):54–101, 2005.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[Rus22] R.C. Russell. a method of phonetic indexing, 1922.

[Sad12] Fereidoon Sadri. On the foundations of probabilistic information integration. In *CIKM*, pages 882–891, 2012.

[SB02] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *KDD*, pages 269–278, 2002.

[SB04] Hamid Haidarian Shahri and Ahmad Abdollahzadeh Barforoush. A Flexible Fuzzy Expert System for Fuzzy Duplicate Elimination in Data Cleaning. In *DEXA*, pages 161–170, 2004.

[SB12] Sriparna Saha Sanghamitra Bandyopadhyay. *Unsupervised Classification*. Springer Science & Business Media, 2012.

[SBH+09] Anish Das Sarma, Omar Benjelloun, Alon Y. Halevy, Shubha U. Nabar, and Jennifer Widom. Representing uncertain data: models, properties, and algorithms. *VLDB J.*, 18(5):989–1019, 2009.

[SBKM02] Sunita Sarawagi, Anuradha Bhamidipaty, Alok Kirpal, and Chandra Mouli. Alias: An active learning led interactive deduplication system. In *VLDB*, pages 1103–1106, 2002.

[SBY10] Jie Song, Yubin Bao, and Ge Yu. A multilevel and domain-independent duplicate detection model for scientific database. In *WAIM*, pages 729–741, 2010.

[Sch03] Alexander Schrijver. *Combinatorial Optimization*. Springer, 2003.

[SCH09] Dan Suciu, Andrew Connolly, and Bill Howe. Embracing Uncertainty in Large-Scale Computational Astrophysics. In *MUD*, pages 63–77, 2009.

[SCXM11] Liangcai Shu, Aiyou Chen, Ming Xiong, and Weiyi Meng. Efficient spectral neighborhood blocking for entity resolution. In *ICDE*, pages 1067–1078, 2011.

[SD06] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *ICDM*, pages 572–582, 2006.

[SD07] Prithviraj Sen and Amol Deshpande. Representing and Querying Correlated Tuples in Probabilistic Databases. In *ICDE*, pages 596–605, 2007.

[SD09] Sumit Sarkar and Debabrata Dey. Relational models and algebra for uncertain data. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 3, pages 45–76. Springer, 1 edition, 2009.

[SDH09] Anish Das Sarma, Xin Dong, and Alon Halevy. Uncertainty in Data Integration. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 7, pages 185–222. Springer, 2009.

[SF06] Thilo Stadelmann and Bernd Freisleben. Fast and robust speaker clustering using the earth mover's distance and mixmax models. In *ICASSP (1)*, pages 989–992, 2006.

[SG03] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, March 2003.

[Sha76] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[SI09] Mohamed A. Soliman and Ihab F. Ilyas. Ranking with uncertain scores. In *ICDE*, pages 317–328, 2009.

[SIC07] Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. Top-k query processing in uncertain databases. In *ICDE*, pages 896–905, 2007.

[Sin13] Vijay P. Singh. *Entropy Theory and its Application in Environmental and Water Engineering*. John Wiley & Sons, 2013.

[SJ08] Sameer Shirdhonkar and David W. Jacobs. Approximate earth mover's distance in linear time. In *CVPR*, 2008.

[SJMB11] Anish Das Sarma, Ankur Jain, Ashwin Machanavajjhala, and Philip Bohannon. Cblock: An automatic blocking mechanism for large-scale de-duplication tasks. *CoRR*, abs/1111.3689, 2011.

[sKL10] Hung sik Kim and Dongwon Lee. Harra: fast iterative hashed record linkage for large-scale data collections. In *EDBT*, pages 525–536, 2010.

[SKSM08] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun K. Majumdar. Credit card fraud detection using hidden markov model. *IEEE Trans. Dependable Sec. Comput.*, 5(1):37–48, 2008.

[SLD05] Warren Shen, Xin Li, and AnHai Doan. Constraint-based entity matching. In *AAAI*, pages 862–867, 2005.

[SLG13] Chunyao Song, Zheng Li, and Tingjian Ge. Top-k oracle: A new way to present top-k tuples for uncertain data. In *ICDE*, pages 146–157, 2013.

[SLP02] Sam Yuan Sung, Zhao Li, and Sun Peng. A fast filtering scheme for large database cleansing. In *CIKM*, pages 76–83, 2002.

[SM90] Sujeet Shenoi and Austin Melton. An extended version of the fuzzy relational database model. *Inf. Sci.*, 52(1):35–52, 1990.

[SMM+08a] Sarvjeet Singh, Chris Mayfield, Sagar Mittal, Sunil Prabhakar, Susanne E. Hambrusch, and Rahul Shah. Orion 2.0: native support for uncertain data. In *SIGMOD Conference*, pages 1239–1242, 2008.

[SMM+08b] Sarvjeet Singh, Chris Mayfield, Sagar Mittal, Sunil Prabhakar, Susanne E. Hambrusch, and Rahul Shah. The orion uncertain data management system. In *COMAD*, pages 273–276, 2008.

[SORK11] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[SQL03] SQL Standard 2003, SQL-Foundation ISO/IEC 9075-2:2003 (E), 8 2003.

[SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

[SS13] Asma Souihli and Pierre Senellart. Optimizing approximations of dnf query lineage in probabilistic xml. In *ICDE*, pages 721–732, 2013.

[SSRS10] G.V. Suresh, Shabbeer Shaik, E.V. Reddy, and Usman Ali Shaik. Gaussian process model for uncertain data classification. *International Journal of Computer Science*

and Information Security, 8(9), December 2010.

[Str02] Alexander Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, 2002. AAI3088578.

[Sub09] V. S. Subrahmanian. Probabilistic temporal databases. In *Encyclopedia of Database Systems*, pages 2165–2168. 2009.

[SUW09] Anish Das Sarma, Jeffrey D. Ullman, and Jennifer Widom. Schema design for uncertain databases. In *AMW*, 2009.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[SW83] Helen C. Shen and Andrew K. C. Wong. Generalized texture representation and metric. *Computer Vision, Graphics, and Image Processing*, 23(2):187–206, 1983.

[SW12] Klaus-Dieter Schewe and Qing Wang. On the decidability and complexity of identity knowledge representation. In *DASFAA (1)*, pages 288–302, 2012.

[SWF+09] Yannis Sismanis, Ling Wang, Ariel Fuxman, Peter J. Haas, and Berthold Reinwald. Resolution-aware query answering for business intelligence. In *ICDE*, pages 976–987, 2009.

[T+92] Frank Shou-Cheng Tseng et al. A Probabilistic Approach to Query Processing in Heterogeneous Database Systems. In *RIDE-TQP*, pages 176–183, 1992.

[Taf70] Robert L. Taft. *Name Search Techniques*. Bureau of Systems Development, New York State Identification and Intelligence System,, 1970.

[Tal11] John R. Talburt. *Entity Resolution and Information Quality*. Morgan Kaufman Publ. Inc., 2011.

[Tan01] Inder Jeet Taneja. *Generalized Information Measures and Their Applications*. Universidade Federal de Santa Catarina, 2001.

[Tao09] Yufei Tao. Range and Nearest Neighbor Queries on Uncertain Spatiotemporal Data. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 11. Springer, 2009.

[TB98] Giri Kumar Tayi and Donald P. Ballou. Examining data quality - introduction. *Commun. ACM*, 41(2):54–57, 1998.

[TCX+05] Yufei Tao, Reynold Cheng, Xiaokui Xiao, Wang Kay Ngai, Ben Kao, and Sunil Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *VLDB*, pages 922–933, 2005.

[TCY93] Frank Shou-Cheng Tseng, Arbee L. P. Chen, and Wei-Pang Yang. Answering Heterogeneous Database Queries with Degrees of Uncertainty. *Distributed and Parallel Databases*, 1(3):281–302, 1993.

[Tep68] Benjamin J. Tepping. A model for optimum linkage of records. *Journal of the American Statistical Association*, Volume 63(Issue 324):1321–1332, 1968.

[THCS90] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson and Clifford Stein. *Introduction to Algorithms*. McGraw Hill and MIT Press, 1990.

[TSC+09] Thanh T. L. Tran, Charles A. Sutton, Richard Cocci, Yanming Nie, Yanlei Diao, and Prashant J. Shenoy. Probabilistic inference over rfid streams in mobile environments. In *ICDE*, pages 1096–1107, 2009.

[Tve77] Amos Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[TWPH07] John R. Talburt, Ningning Wu, Elizabeth M. Pierce, and Ray R. Hashemi. Entity identification using indexed entity catalogs. In *IKE*, pages 338–342, 2007.

[UF94] Motohide Umano and Satoru Fukami. Fuzzy relational algebra for possibility-distribution-fuzzy-relational model of fuzzy data. *J. Intell. Inf. Syst.*, 3(1):7–27, 1994.

[UGMW02] Jeffrey D. Ullman, Hector Garcia-Molina, and Jennifer D. Widom. *Database Systems: The Complete Book*. Prentice-Hall, 2002.

[Var82] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, pages 137–146, 1982.

[VBA+95] Marc B. Vilain, John D. Burger, John S. Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *MUC*, pages 45–52, 1995.

[VBGK09a] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In *International Semantic Web Conference*,

pages 650–665, 2009.

[VBGK09b] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - a link discovery framework for the web of data. In *LDOW*, 2009.

[vD09] Stijn Marinus van Dongen. *Graph clustering by flow Simulation*. PhD thesis, University of Utrecht, The Netherlands, 2009.

[Vee09] Erik Vee. Sketching Aggregates over Probabilistic Streams. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 8, pages 223–256. Springer, 2009.

[vK12] Maurice van Keulen. Managing uncertainty: The road towards better data interoperability. *it - Information Technology*, 54(3):138–146, 2012.

[vKdK09] Maurice van Keulen and Ander de Keijzer. Qualitative effects of knowledge rules and user feedback in probabilistic data integration. *The VLDB Journal*, 18(5):1191–1217, 2009.

[vKdKA05] Maurice van Keulen, Ander de Keijzer, and Wouter Alink. A Probabilistic XML Approach to Data Integration. In *ICDE*, pages 459–470, 2005.

[vKH11] Maurice van Keulen and Mena B. Habib. Handling uncertainty in information extraction. In *URSW*, pages 109–112, 2011.

[VM04] Vassilios S. Verykios and George V. Moustakides. A generalized cost optimal decision model for record matching. In *IQIS*, pages 20–26, 2004.

[VME03] Vassilios S. Verykios, George V. Moustakides, and Mohamed G. Elfeky. A bayesian decision model for cost optimal record matching. *VLDB J.*, 12(1):28–40, 2003.

[Vos08] Gottfried Vossen. *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*. Oldenbourg Wissenschaftsverlag, 2008.

[Was04] Larry A. Wasserman. *All of Statistics: a Concise Course in Statistical Inference*. Springer, 2004.

[WB09] Derry Tanti Wijaya and Stéphane Bressan. Ricochet: A family of unconstrained algorithms for graph clustering. In *DASFAA*, pages 153–167, 2009.

[WBGM09] Steven Euijong Whang, Omar Benjelloun, and Hector Garcia-Molina. Generic entity resolution with negative rules. *VLDB J.*, 18(6):1261–1277, 2009.

[WCC+12] Liang Wang, David Wai-Lok Cheung, Reynold Cheng, Sau Dan Lee, and Xuan S. Yang. Efficient mining of frequent item sets on large uncertain databases. *IEEE Trans. Knowl. Data Eng.*, 24(12):2170–2183, 2012.

[Wei08] Melanie Weis. *Duplicate detection in XML data*. PhD thesis, 2008.

[WF10] Wolfram Wingerath and Steffen Friedrich. Search-space reduction techniques for duplicate detection in probabilistic data. Bachelor thesis, University of Hamburg, 2010.

[Wid09] Jennifer Widom. Trio: A System for Data, Uncertainty, and Lineage. In Charu C. Aggarwal, editor, *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*, chapter 5, pages 113–148. Springer, 2009.

[Wij09] Jef Wijsen. Consistent query answering under primary keys: a characterization of tractable queries. In *ICDT*, pages 42–52, 2009.

[Wij10] Jef Wijsen. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In *PODS*, pages 179–190, 2010.

[Wij12] Jef Wijsen. Certain conjunctive query answering in first-order logic. *ACM Trans. Database Syst.*, 37(2):9, 2012.

[Wij14] Jef Wijsen. A survey of the data complexity of consistent query answering under key constraints. In *FoIKS*, pages 62–78, 2014.

[Win90] William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, 1990.

[Win93] William E. Winkler. Improved decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods, American Statistical Association*, pages 274–279, 1993.

[Win02] William E. Winkler. Methods for record linkage and bayesian networks. Technical Report Statistical Research Report Series RRS2002/05, U.S. Bureau of the Census, Washington, D.C., 2002.

[WM89] Y. Richard Wang and Stuart E. Madnick. The Inter-Database Instance Identification Problem in Integrating Autonomous Systems. In *Proceedings of the Fifth International Conference on Data Engineering, February 6-10, 1989, Los Angeles, California, USA*, pages 46–55. IEEE Computer Society, 1989.

[WMF+10] Daisy Zhe Wang, Eirinaios Michelakis, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. Probabilistic declarative information extraction. In *ICDE*, pages 173–176, 2010.

[WMGH08] Daisy Zhe Wang, Eirinaios Michelakis, Minos N. Garofalakis, and Joseph M. Hellerstein. Bayesstore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1):340–351, 2008.

[WMK+09] Steven Euijong Whang, David Menestrina, Georgia Koutrika, Martin Theobald, and Hector Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD Conference*, pages 219–232, 2009.

[WMM10] Michael L. Wick, Andrew McCallum, and Gerome Miklau. Scalable probabilistic databases with factor graphs and mcmc. *CoRR*, abs/1005.1934, 2010.

[WN05] Melanie Weis and Felix Naumann. Dogmatix tracks down duplicates in xml. In *SIGMOD Conference*, pages 431–442, 2005.

[WN06a] Melanie Weis and Felix Naumann. Detecting Duplicates in Complex XML Data. In *ICDE*, page 109, 2006.

[WN06b] Melanie Weis and Felix Naumann. Relationship-based duplicate detection. Technical report, Humboldt-Universität zu Berlin, 2006.

[WNJ+08] Melanie Weis, Felix Naumann, Ulrich Jehle, Jens Lufter, and Holger Schuster. Industry-scale duplicate detection. *PVLDB*, 1(2):1253–1264, 2008.

[Wol98] Catherine D. Wolfram. Strategic bidding in a multiunit auction: An empirical analysis of bids to supply electricity in england and wales. *The RAND Journal of Economics*, 29(4):703–725, 1998.

[WPR85] Michael Werman, Shmuel Peleg, and Azriel Rosenfeld. A distance metric for multidimensional histograms. *Computer Vision, Graphics, and Image Processing*, 32(3):328–336, 1985.

[WSB76] M. S. Waterman, T. F. Smith, and W. A. Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20:367–387, 1976.

[WT91] William E. Winkler and Yves Thibaudeau. An application of the fellegi-sunter model of record linkage to the 1990 u.s. decennial census. Technical report, U.S. Bureau of the Census, 1991.

[WWL09] Fei Wang, Xin Wang, and Tao Li. Generalized cluster aggregation. In *IJCAI*, pages 1279–1284, 2009.

[WZL01] Richard Y. Wang, Mostapha Ziad, and Yang W. Lee. *Data Quality*, volume 23 of *Advances in Database Systems*. Kluwer, 2001.

[XBE+09] Fei Xu, Kevin S. Beyer, Vuk Ercegovac, Peter J. Haas, and Eugene J. Shekita. E = mc^3: managing uncertain enterprise data in a cluster-computing environment. In *SIGMOD Conference*, pages 441–454, 2009.

[XC08] Dong Xu and Shih-Fu Chang. Video event recognition using kernel methods with multilevel temporal alignment. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1985–1997, 2008.

[XZTY12] Jia Xu, Zhenjie Zhang, Anthony K. H. Tung, and Ge Yu. Efficient and effective similarity search over probabilistic data based on earth mover's distance. *VLDB J.*, 21(4):535–559, 2012.

[YLKG07] Su Yan, Dongwon Lee, Min-Yen Kan, and C. Lee Giles. Adaptive sorted neighborhood methods for efficient record linkage. In *JCDL*, pages 185–194, 2007.

[YTX+10] Sze Man Yuen, Yufei Tao, Xiaokui Xiao, Jian Pei, and Donghui Zhang. Superseding nearest neighbor search on uncertain spatial databases. *IEEE Trans. Knowl. Data Eng.*, 22(7):1041–1055, 2010.

[YWCK11] Mohan Yang, Haixun Wang, Haiquan Chen, and Wei-Shinn Ku. Querying uncertain data with aggregate constraints. In *SIGMOD Conference*, pages 817–828, 2011.

[Zad65] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[Zad78] Lotfi A. Zadeh. Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[ZC13] Yinuo Zhang and Reynold Cheng. Probabilistic filters: A stream protocol for continuous probabilistic queries. *Inf. Syst.*, 38(1):132–154, 2013.

[ZCJC13] Chen Jason Zhang, Lei Chen, H. V. Jagadish, and Caleb Chen Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 6(9):757–768, 2013.

[ZD96] Justin Zobel and Philip W. Dart. Phonetic string matching: Lessons from information retrieval. In *SIGIR*, pages 166–172, 1996.

[ZH06] Rong Zhou and Eric A. Hansen. Domain-independent structured duplicate detection. In *AAAI*, pages 1082–1088, 2006.

[ZH11] Vladimir Zadorozhny and Ying-Feng Hsu. Conflict-aware historical data fusion. In *SUM*, pages 331–345, 2011.

[ZK84] Maria Zemankova and Abraham Kandel. *Fuzzy relational data bases. A key to expert systems*. Verl. TUV Rheinland, 1984.

[ZKF05] Ying Zhao, George Karypis, and Usama M. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Min. Knowl. Discov.*, 10(2):141–168, 2005.

[ZKT12] Yinle Zhou, Ali Kooshesh, and John R. Talburt. Optimizing the accuracy of entity-based data integration of multiple data sources using genetic programming methods. *IJBIR*, 3(1):72–82, 2012.

[ZLGZ10] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Finding top-k maximal cliques in an uncertain graph. In *ICDE*, pages 649–652, 2010.

[ZSC10] Diego Zardetto, Monica Scannapieco, and Tiziana Catarci. Effective automated Object Matching. In *ICDE*, pages 757–768, 2010.

List of Symbols and Abbreviations

In this thesis, we use the following symbols and abbreviations: Note that if clear from context we sometimes use one of these symbols with a different meaning.

Symbol	**Description**
dom	domain
db	certain database / conventional database
idb	incomplete database
pdb	probabilistic database
udb	uncertain database
t	database tuple
T	database table
E	entity type
f_{ED}	function that extracts an entity description from a database
$\mathfrak{d}_{ED}$	function that refers to the current descriptions of database entities
Inst	instance of an entity or an entity type (entity table)
Ext	extension of an entity type (entity table)
Q	database query
$\mathfrak{L}$	language of a database system
e	database entity or real-world entity
$\mathfrak{E}$	set of database entities
$\mathfrak{W}$	set of all real-world entities
ω	mapping from database entities to real-world entities
$=_{\text{id}}$	real-world equivalence, i.e. $e_1 =_{\text{id}} e_2 \Leftrightarrow \omega(e_1) = \omega(e_2)$
$\circledcirc$	relation of set overlapping
$\circledotimes$	relation of set disjointness

Symbol	Description
W	possible world
$\mathbf{W}$	set of possible worlds
$pws()$	mapping to the possible world space
$pwr()$	mapping to the possible worlds representation
TI-database	tuple-independent database
AOR-database	attribute-OR database
AOR?-database	attribute-OR database with maybe tuples
BID-database	block-independent-disjoint database
pc-database	probabilistic conditional database
Pr	probability distribution of a probabilistic value (e.g. prob. database)
p	probability mass function that describes the marginal probability of tuples
θ	particular assignment of a set of variables
Θ	set of all possible assignments of a set of variables
Φ	tuple conidition in a pc-database
$\varkappa$	the set of assignments that are satisfied by a given tuple condition
$\mathfrak{R}$	probabilistic or incomplete representation system
ϵ	event
M	set of MATCHES
U	set of UNMATCHES
P	set of POSSIBLE MATCHES
$\theta_{P/M}$	threshold that demarcates P and M
$\theta_{U/P}$	threshold that demarcates U and P
$\theta_{U/M}$	threshold that demarcates U and M
bk	blocking key
bkv	blocking key value
θ	threshold
fs	feature score
APS	alternative description pair space
d	certain/crisp entity description
$\mathfrak{d}$	incomplete entity description
$\mathfrak{d}$	probabilistic entity description
QM	quality measure

Symbol	Description
C	cluster
$\mathcal{C}$	certain/crisp clustering
$\mathcal{C} \prec \mathcal{C}'$	clustering $\mathcal{C}$ is dominated by clustering $\mathcal{C}'$
$\mathcal{C}_{\text{gold}}$	gold standard of a duplicate detection problem
$\mathcal{F}$	factorization (set of factors) of an uncertain clustering
$\mathcal{F}^{set}$	set of all factor sets of the factorization of an uncertain clustering
CCC	certain cluster component of an uncertain clustering
rng	range of a (certain or uncertain) clustering
Γ	incomplete clustering
$\mathfrak{C}$	probabilistic clustering
$\boxtimes$	operator that integrates two clusterings
$\overline{CS}$	skyline of a clustering set CS
$\widehat{\mathfrak{C}}$	topl-level of a probabilistic clustering $\mathfrak{C}$
$CF(\mathcal{C})$	set of all clustering facts of a clustering $\mathcal{C}$
M-graph	a *matching-graph*
W-graph	a *world-graph*
${}^{G}\boxtimes$	operator that integrates two sets of *W-graphs*
f_{dec}	mapping that decomposes an *M-graph* into its independent components
$f^{\text{HC}}_{MG \mapsto WG}$	*W-graph* computation mapping using HC-restriction
$f^{\text{Naive}}_{MG \mapsto WG}$	naive *W-graph* computation mapping
$f^{\text{Naive}}_{MG \mapsto Cl}$	mapping from an *M-graph* to a probabilistic clustering (naive approach)
$\triangle$-*conflict*	trianlge-conflict in *M-graphs*

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den August 18, 2015

Fabian Panse