Fallbasierte Unterstützung von Experten im Bereich Service und Support

Ein Ansatz auf Basis von Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen

Dissertation
zur Erlangung des Doktorgrades
am Fachbereich Informatik der Universität Hamburg

vorgelegt von

Gerd Kamp

aus Bingen

Hamburg 2003

Genehmigt vom Fachbereich Informatik der Universität Hamburg
auf Antrag von Prof. Dr. Bernd Neumann
und Prof. Dr. Michael M. Richter

Hamburg, den 18. Dezember 2003

Prof.Dr. Hans Siegfried Stiehl
(Dekan)

Für Josef (Jupp) und Nikola

Danksagungen

Zum Gelingen dieser Arbeit haben sehr viele Menschen beigetragen, denen ich hiermit Dank sagen möchte.

Zuallererst möchte ich mich bei Prof. Bernd Neumann dafür bedanken, daß er die Arbeit über den gesamten Zeitraum kritisch und zugleich wohlwollend begleitet hat. Besonderen Dank für die Freiräume die du einem immer bei der wissenschaftlichen Arbeit gelassen hast, und für das Vorbild dafür, daß es sich lohnt über den Tellerrand zu blicken.

Bei Prof. Michael M. Richter bedanke ich mich für die zahlreichen anregenden Diskussionen, nicht nur zum fallbasierten Schließen, sowie für die Bereitschaft das Zweitreferat zu übernehmen, obwohl du ja jetzt deine Zeit auch anders verbringen könntest.

Meiner Frau Vera danke ich ≫für einfach alles≪. Ohne deine Liebe und grenzenlose Unterstützung wäre diese Arbeit sicher nicht zustande gekommen.

Ansonsten möchte ich allen aus dem Familien-, Freundes- und Bekanntenkreis danken, die mich in fachlicher, moralischer oder irgendeiner anderen Form bei der Erstellung dieser Arbeit unterstützt haben.

Besonderer Dank gilt Sabine, Andreas, Manfred, Peter, Nada, meiner Mutter, Schwester und Schwiegermutter dafür, daß sie mich regelmäßig und freundlich daran erinnert haben, daß da neben meiner Arbeit noch ≫etwas≪ ist, ohne daran zu zweifeln, daß eben ≫jenes≪ nicht fertig gestellt werden würde.

Inhaltsverzeichnis

I never waste memory on things that can easily be stored and retrieved from elsewhere.

– Albert Einstein –

Einleitung

In diesem Vorhaben sollen Grundlagen zur fallbasierten Fehlerdiagnose an technischen Systemen untersucht werden. Falldaten über Fehler und ihre Ursachen sollen durch menschliche Experten rechnerunterstützt erfaßt werden. Eine Diagnose erfolgt dann durch eine Auswertung der Falldaten, z.B. Verallgemeinerung und Vergleich, und erfordert keine regelbasierte Wissensbasis. Die Untersuchungen konzentrieren sich auf Methoden zur strukturierten Repräsentation von Falldaten und auf die Grundlagen fallbasierten Schließens.
(aus dem Projektantrag zum DFG-Grundlagenprojekt »Falldaten«)

Diagnose ist eines der Kernanwendungsgebiete fallbasierter Systeme

Diagnose ist eine Aufgabe, zu deren Lösung normalerweise ein großes Maß an Erfahrung benötigt wird. Aus diesem Grund wird die Diagnose im allgemeinen als ein Anwendungsbereich angesehen, in dem fallbasiertes Schließen mit großem Erfolg eingesetzt werden kann.

So ist es nicht verwunderlich, daß eine Reihe von fallbasierten Methoden und Verfahren entwickelt wurden, die in Systemen zur Unterstützung der Diagnose technischer Systeme zum Einsatz kommen.

Service und Support

Unterschiedliche Nutzerkreise innerhalb des Anwendungsbereiches Service und Support werden durch verschiedene Systemarten unterstützt

Diese Systeme entstammen zumeist dem Anwendungsbereich Service und Support. Ein Großteil davon sind so genannte Help-Desk-Systeme (HDS), andere im Kontext dieser Arbeit relevante Systemklassen aus dem Bereich des Service und Supports sind so genannte Self-Help-Systeme (SHS) und Second-Level-Support-Systeme (SLSS).

Self-Help-Systeme, Help-Desk-Systeme und Second-Level-Support-Systeme unterstützen jeweils unterschiedlich kompetente Anwenderkreise (man spricht in diesem Zusammenhang auch von Service-Level) u.a. bei der Fehlersuche und Problembehebung. Kann mit Unterstützung des jeweiligen Systems das Problem nicht behoben resp. der Fehler nicht gefunden werden, erfolgt eine Eskalation an den nächsthöheren Service-Level.

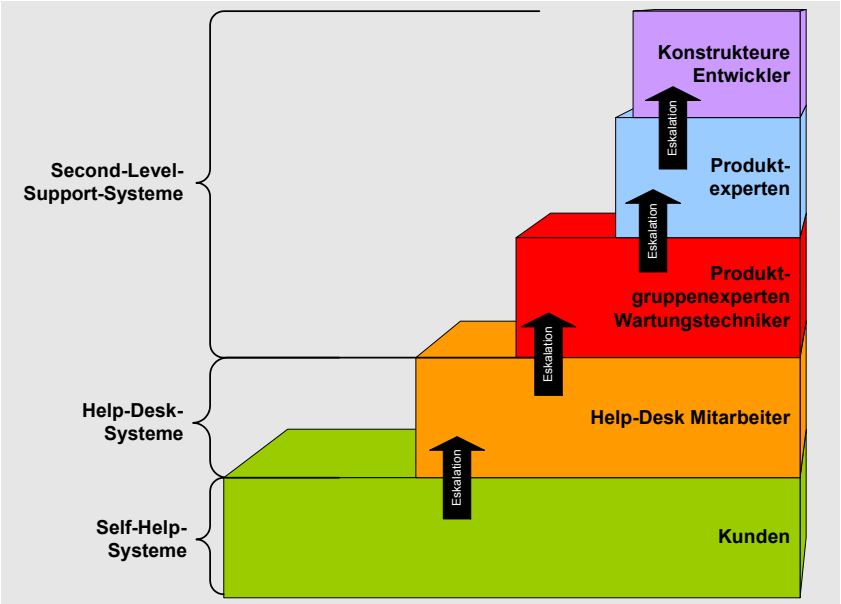


Abbildung 0.1.

Das Verhältnis zwischen Help-Desk-, Self-Help- und Second-Level-Support-Systemen

Help-Desk-Systeme

Help-Desk-Systeme = Schematische Lösung einfacher Probleme ohne Fachwissen

Help-Desks und Help-Desk-Systeme wurden zuerst eingesetzt, um Probleme zu lösen, die im Zusammenhang mit der Benutzung von Computern auftraten. Mittlerweile versteht man unter einem Help-Desk-System ein System, das Help-Desk-Mitarbeiter bei der Erteilung von Hilfestellungen und der Behebung von Problemen unterstützt. Zumeist arbeiten die Help-Desk-Mitarbeiter in Call-Centern, und die Menschen, die Hilfe suchen, sind Kunden, die die Help-Desk-Mitarbeiter telefonisch erreichen.

Help-Desk-Systeme unterstützen Help-Desk-Mitarbeiter in einer frühen Phase der Diagnose und bei der Lösung einfacher Probleme an technischen Geräten. Durch die Einrichtung von Help-Desks und Help-Desk-Systemen werden Experten wie z.B. Wartungstechniker und Produktexperten von Routineaufgaben entlastet, und die Gesamtkosten des Supports reduziert.

Aktuelle Help-Desk-Systeme erfüllen diese Zielsetzung: Help-Desk-Mitarbeiter sind häufig angelernte Kräfte, etwa Schüler und Studenten, die nur ein spezielles Training zur Benutzung des Help-Desk-Systems erhalten, aber keinerlei (bzw. kaum) Fachwissen besitzen.

Self-Help-Systeme

Self-Help-Systeme = Verlagerung der Arbeit zum Kunden

Um weitere Kosten im Bereich Service und Support zu sparen und um die Fehlerbehebungszeit weiter zu minimieren, gehen immer mehr Hersteller dazu über,

die ersten Schritte zur Fehlerbehebung dem Kunden selbst zu überlassen. So genannte Self-Help-Systeme sollen den Kunden in die Lage versetzen, die einfachen Probleme selbst zu diagnostizieren.

Möglich wurde die Realisierung von Self-Help-Systemen erst durch das World-Wide-Web. Erst dieses eröffnete den einfachen, interaktiven Zugang zu den benötigten aktuellen Informationen. Daher sind Self-Help-Systeme zumeist mit einem Web-Interface versehene Help-Desk-Systeme.

Second-Level-Support-Systeme

Second-Level-Support-Systeme = Flexible Lösung, schwieriger Probleme aufgrund umfassenden Wissens

Können die Help-Desk-Mitarbeiter keine Lösung für ein Problem finden, wird es an die zuständigen Wartungstechniker und Produktgruppenexperten weitergeleitet; die »Fälle« werden eskaliert. Wartungstechniker und Produktgruppenexperten versuchen, für die ihnen zugetragenen schwierigen Fälle eine Lösung zu finden.

Aufgrund ihrer Fachkenntnisse und ihrer umfangreichen Erfahrung gelingt dies auch in den meisten Fällen. Wartungstechniker und Produktgruppenexperten sind in der Lage, sich dem Problem von verschiedenen Seiten zu nähern und es dadurch »einzukreisen«. Dazu stellen sie häufig mehrere Hypothesen bzw. mögliche Erklärungsmodelle auf, die sie dann durch die unterschiedlichsten Techniken wie etwa:

- Simulationen,
- What-If-Analysen,
- Modellbasierte Diagnose,

etc. überprüfen. Dabei unterscheiden sich die tatsächlich eingesetzten Techniken von Fall zu Fall.

Help-Desk = First-Level-Support, Self-Help = Level-0-Support

Da Wartungstechniker und Produktgruppenexperten erst dann in den Problemlösungsprozeß eingreifen, wenn die Help-Desk-Mitarbeiter keine Lösung finden konnten, werden diese Mitarbeiter auch häufig als »second-level support« bezeichnet, während die Help-Desk Mitarbeiter den »first-level support« bilden. Unterstützungssysteme für Mitarbeiter des Second-Level-Support bezeichnet man daher als Second-Level-Support-Systeme.

Manchmal gibt es mehr als zwei Supportebenen

Oftmals sind der zweiten Supportebene weitere Eskalationsstufen nachgeschaltet. Etwa Produktexperten als dritte, sowie Konstrukteure und Produktentwickler als vierte Supportebene. Davon unabhängig werden Unterstützungssysteme auch für die dritte und höhere Supportebenen verallgemeinernd als Second-Level-Support-Systeme bezeichnet [1].

1. Entsprechend werden die Kunden, die ein Self-Help-System benutzen, auch als »Nullte-Support-Ebene« bezeichnet. Diese etwas seltsame Art der Nummerierung hat ihren Ursprung darin, daß historisch zunächst Help-Desk-Systeme verfügbar waren. Zum damaligen Zeitpunkt konnte man sich nicht vorstellen, daß Kunden selbst Diagnoseaufgaben übernehmen würden

Gegenstand der Arbeit

Gegenstand dieser Arbeit ist die fallbasierte Unterstützung von Mitarbeitern des Second-Level-Supports, wie etwa Wartungstechnikern oder Produktgruppenexperten.

Bereitstellung eines flexiblen Informationssystems

Es wird gezeigt,daß dazu ein flexibles Informationssystem benötigt wird, das in der Lage ist, aus einer gegebenen Menge von Informationen auf systematische Art und Weise neue Informationen abzuleiten. Dazu muß es aber zunächst möglich sein, das zur Ableitung neuer Informationen notwendige Wissen in einem solchen System repräsentieren zu können.

Repräsentation des Domänen- und des Hintergrundwissens

Dies betrifft zunächst Wissen über die Struktur und Art der Geräte und Geräteteile. Darüber hinaus muß es auch möglich sein, Hintergrundwissen, etwa in Form von Randbedingungen, die die einzelnen Geräte und Geräteteile erfüllen bzw. erfüllen müssen, im Repräsentationsformalismus beschreiben zu können.

Darüber hinaus existiert allgemein gültiges Hintergrundwissen, etwa in Form von physikalischen Gesetzen sowie Wissen über Normal- bzw. Fehlverhalten von Komponenten. Auch dieses Wissen muß repräsentiert werden können, da es z.B. für Simulationen unverzichtbar ist.

Erweiterbarkeit der Repräsentation

Weiterhin muß die Repräsentation muß jederzeit um zusätzliche Informationen z.B. neue Gerätetypen und um hinzugekommenes Wissen z.B. neue Fehlverhaltensmodelle erweitert werden können.

Ableitung neuer Informationen unter Verwendung des repräsentierten Wissens

Darüber hinaus muß das System Mechanismen besitzen, die es erlauben, aus einer gegebenen Menge von Informationen und dem repräsentierten Wissen neue Informationen abzuleiten und diese dem Benutzer zu präsentieren.

Ziel des Systems muß es dann sein, dem Benutzer so viele Informationen abzuleiten und bereitzustellen, daß er in die Lage versetzt wird, die aktuelle Aufgabe zu lösen. Da das Aufgabenspektrum breit gefächert ist, muß es möglich sein, die aktuelle Situation und den bestehenden Informationsbedarf einfach und flexibel beschreiben zu können. Zur Ableitung neuer Informationen muß das repräsentierte Wissen eingesetzt werden können.

Inhalte der Arbeit

Bestimmte Beschreibungslogiken erfüllen die Anforderungen

Im Laufe dieser Arbeit wird gezeigt, daß Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen sowohl die Mittel zur Repräsentation des Domänen und des Hintergrundwissens, als auch die Inferenzen zur Ableitung neuer Informationen zur Verfügung stellen.

Dieser Ansatz erweitert die Techniken des fallbasierten Schließens um Mechanismen zur erweiterbaren objektorientierten Repräsentation des Gegenstandsbereiches und zum flexiblen Retrieval auf der Grundlage von korrekten und vollständigen Inferenzalgorithmen zur semantischen Indizierung der repräsentierten Objekte.

Dabei ist die zugrundeliegende Vorgehensweise immer die gleiche: Eine Anfrage (z.B. in Form einer Menge von Instanzen mit assoziierten Attributwerten bzw. Attributwerteinschränkungen) wird im Rahmen der bereitgestellten Anfragesprache formuliert und durch eine Menge von Objekten in der Wissensbasis beantwortet, auf die diese Anfrage zutrifft. Um diese Objekte zu ermitteln, werden die Inferenzverfahren zur Ableitung neuer Fakten benutzt.

Beschreibungslogiken mit korrekten und vollständigen Algorithmen sind besonders interessant

Den Kern des Gesamtsystems bilden Beschreibungslogiken. Beschreibungslogiken [BS85, BBH+92, Baa96] sind ein objektorientierter Wissensrepräsentationsformalismus für Fragmente der Prädikatenlogik erster Stufe. Unterschieden werden Beschreibungen von Konzepten, die im wesentlichen den Klassenbeschreibungen normaler objektorientierter Repräsentationen entsprechen, und Instanzbeschreibungen die den Objekten in objektorientierten Systemen entsprechen [2].

Verschiedene Beschreibungslogiken unterscheiden sich durch das durch in ihnen realisierten Fragment der Prädikatenlogik erster Stufe. D.h. im wesentlichen in den von Ihnen zur Konstruktion der Beschreibungen zur Verfügung gestellten Operatoren. Zur Unterstützung von Wartungstechnikern werden als Operatoren sowohl Konjunktion, als auch Disjunktion und Negation zur Verknüpfung von Konzept- und Instanzbeschreibungen benötigt.

Beschreibungslogiken sind insbesondere deshalb für das fallbasierte Schließen interessant, weil es möglich ist, korrekte und vollständige Basisinferenzen über der Menge der erlaubten Beschreibungen bereitzustellen. Auf diesen Inferenzen aufsetzend können korrekte und vollständige Algorithmen sowohl zur Konzept- (dem beschreibungslogischen Äquivalent einer Klasse) als auch zur Instanzklassifikation und zum Instanzretrieval realisiert werden.

Ausdrucksstarke konkrete Gegenstandsbereiche sind das Schlüsselelement im betrachteten Anwendungsszenario

Um Beschreibungslogiken im Bereich Service und Support einsetzen zu können, müssen diese um ausdrucksstarke konkrete Gegenstandsbereiche nach Baader und Hanschke [BH91a, Han96] erweitert werden.

Konkrete Gegenstandsbereiche erlauben es, numerische und textuelle Attribute [3] von Konzepten zu definieren. Ausdrucksstark ist ein konkreter Gegenstandsbereich dann, wenn nicht nur Attributwerte von Instanzen mit den Mitteln des konkreten Gegenstandsbereiches beschrieben werden können, also ein numerischer bzw. textueller Wert eines Attributes angegeben werden kann.

In den in dieser Arbeit betrachteten konkreten Gegenstandsbereichen ist es

2. Oft auch in objektorientierten Systemen Instanzen genannt.

3. Andere Attributtypen können ebenfalls über konkrete Gegenstandsbereiche realisiert werden, werden aber im Kontext der Arbeit nicht diskutiert.

möglich, sowohl komplexe Wertebereichseinschränkungen für Attribute von Konzepten, als auch unscharfe Attributwerte für Instanzen zu beschreiben. Damit ist es z.B. möglich Normal- und Fehlverhalten von Geräten und Geräteteilen zu modellieren sowie ungenaue Angaben und Messwerte des Benutzers zu beschreiben.

Für die Einsetzbarkeit im Bereich des Service und Supports ist zudem noch die Möglichkeit zur Formulierung von Abhängigkeiten zwischen einer Menge von Attributen unverzichtbar, etwa um physikalische Gesetze wie das Drehmomentgesetz zu beschreiben.

Die in dieser Arbeit behandelten konkreten Gegenstandsbereiche erlauben die Verknüpfung einer beliebigen Anzahl von Attributen über logische Formeln, deren Terme (Un)gleichungen zwischen quadratischen Polynomen über den Attributen sind. Damit lassen sich komplexe Sachverhalte darstellen, die aber, wie das Drehmomentgesetz, von entscheidender Bedeutung für die Repräsentation des Gegenstandsbereiches sind.

Die Korrektheit und Vollständigkeit der Basisinferenzen (Subsumption, Konsistenzcheck) der Beschreibungslogik wird durch die Erweiterung um ausdrucksstarke konkrete Gegenstandsbereiche nicht verletzt. Damit ist es auch in der erweiterten Beschreibungssprache möglich, korrekte und vollständige Objekt- und Konzeptklassifikation sowie optimales Retrieval zu garantieren.

Dies bedeutet, daß alle aus den in der Anfrage gegebenen Daten und dem beschriebenen Hintergrundwissen ableitbare, Informationen bei der Klassifikation und dem Retrieval auch berücksichtigt werden, die Verfahren also mit Recht als intelligent bezeichnet werden können.

So werden z.B. physikalische Gesetze zur Ermittlung von weiteren Parameterwerten und zur Einschränkung der zulässigen Werte benutzt. So kann beispielsweise über das Drehmomentgesetz bei Vorliegen zweier der drei involvierten Parameter der dritte Parameter berechnet werden (z.B. das Moment bei vorliegen von Radius und Kraft) bzw.. bei Einschränkungen der beiden Parameter die resultierenden Einschränkungen an den dritten.

Einschränkungen können wiederum zu einer Spezialisierung der Anfrageobjekte führen, was wiederum zu weiteren Einschränkungen führen kann etc.. Auf diese Weise lassen sich, ausreichende Repräsentation von Wissen vorausgesetzt, Simulation, What-If Analysen und Diagnose realisieren. Die dynamische Erweiterbarkeit der Domänenbeschreibung erlaubt es aber auch, zunächst nur wenig Wissen zu repräsentieren, etwa wenn die Wissensakquisition zu zeitraubend ist oder das Wissen (etwa über mögliche Fehlverhalten) noch nicht vorliegt.

Beschreibungslogiken erlauben eine dynamische Erweiterung der Repräsentation

Ein weiterer Vorteil eines Ansatzes auf der Basis von Beschreibungslogiken ist die dynamische Erweiterbarkeit der Instanz- und Konzeptbeschreibungen.

So ist es möglich, existierende Instanzbeschreibungen um beliebige Attribute und Attributwerte zu erweitern. Man ist nicht auf die in der Klassendefinition vorgegebenen Attributwerte festgelegt. So lassen sich leicht weitere Meßwerte erfassen.

Ferner ist es möglich, nach der Einführung dieser Meßwerte neue Konzepte zu definieren, die die Meßwerte in ihrer Beschreibung benutzen und Einschränkungen auf den zulässigen Wertebereichen zur Kategorisierung und Diagnose verwenden.

Die Basisinferenzen sorgen dafür, daß die neuen Konzeptbeschreibungen an die richtige Stelle in der Konzepthierarchie eingeordnet werden und die abgelegten Instanzen automatisch neu unter den speziellsten Konzepten, deren Beschreibung sie erfüllen, indiziert werden.

Heuristische Algorithmen des klassischen fallbasierten Schließens sind auf Grundlage der durch die Basisinferenzen bestimmten Indizes möglich

Die Basisinferenzen der Beschreibungslogik stellen sicher, daß die Konzepthierarchie korrekt aufgebaut und die Instanzen optimal in Bezug auf die Konzepte der Konzepthierarchie indiziert sind. Damit ist diese Indizierung hervorragend dazu geeignet, auf der Bewertung von Indexmengen bzw. Indexvektoren basierende heuristische Algorithmen zur Ähnlichkeitsbestimmung zu realisieren.

Bei diesen, im fallbasierten Schließen häufig verwendeten Algorithmen wird die Ähnlichkeit eines Suchbegriffes mit einem Fall der Fallbasis als Ähnlichkeitsmaß zwischen den zugeordneten Indexmengen (Indexvektoren) bestimmt. Im einfachsten Fall ist das Ähnlichkeitsmaß die Identität. Aber auch Hammingabstand, Tverskymaß oder der Abstand zweier Indexvektoren in einem entsprechend dimensionalem Vektorraum werden sowohl im fallbasierten Schließen als auch im Information Retrieval und in multidimensionalen Datenbanken traditionell benutzt. So lassen sich diese Ansätze optimal mit der darunterliegenden Beschreibungslogik kombinieren.

Syntaktische Konzeptbeschreibungen können zur Abbildung von Indexstrukturen verwendet werden

Neben Konzeptbeschreibungen, die semantische Gegebenheiten des Gegenstandbereiches, etwa Modelle des Normal- und Fehlverhaltens von Geräteteilen beschreiben, lassen sich über Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbeschreibungen auch ≫syntaktische Konzepte≪ beschreiben, wie sie etwa als Ergebnis von Clusteralgorithmen entstehen. Von besonderem Interesse sind hier umschreibende Vielecke, insbesondere umschreibende, orthonormale Rechtecke:die so genannten Bounding Boxes, die häufig als Mittel zur Clusterung einer Menge von Fällen benutzt werden. Sowohl Bounding Boxes als auch umschreibende Vielecke sind in ausdrucksstarken konkreten Gegenstandsbereichen darstellbar. Hinzu kommen umschreibende Hyperkugeln.

Dadurch ist ebenfalls möglich, die im fallbasierten Schließen benutzten multidimensionalen Indexstrukturen wie etwa k-d-Bäume, als Hierarchie syntaktischer Konzepte zu realisieren.

Der entwickelte Ansatz bildet die Basis für die Realisierung eines breiten Spektrums an Verfahren

Durch die Kombination von korrekten und vollständigen Algorithmen zur Ableitung neuer Informationen aus dem Hintergrundwissen und auf Indexmengen basierenden heuristischen Algorithmen zur Ähnlichkeitsbestimmung läßt sich ein breites Spektrum an Retrievalverfahren realisieren.

An einem Ende dieses Spektrums ist (noch) kein Hintergrundwissen vorhanden. Das komplette Wissen liegt als Fallwissen in Form von Instanzbeschreibungen vor. Das Retrieval und die Ähnlichkeitsbestimmung erfolgt über heuristische Algorithmen wie im klassischen fallbasierten Schließen. Externe Algorithmen berechnen

syntaktische Konzepte zur Beschreibung der Indexstruktur.

Am anderen Ende dieses Spektrums ist das ganze Hintergrundwissen bekannt. Das Normal- und alle Fehlverhalten des Gerätes sind bekannt und als Menge von Konzepten definiert. Dann kann die Diagnose des Gerätes unter ausschließlicher Verwendung des Hintergrundwissens erfolgen, auf die Verwendung von Fallwissen (d.h. Instanzen) kann verzichtet werden.

Die dynamische Erweiterbarkeit der Domänenbeschreibung erlaubt es sogar, daß sich eine Anwendung im Laufe ihrer Entwicklung weiterentwickelt und mit dem Bekanntwerden von Hintergrundwissen dieses beschrieben und benutzt werden kann.

So kann zunächst vollkommen ohne Hintergrundwissen gestartet werden. Es wird eine Reihe von Erfahrungen gesammelt. Zur Unterstützung wird syntaktisches Retrieval über den Instanzen verwendet. Es existieren weder syntaktische noch semantische Konzepte.

Existieren genügend Erfahrungen, kann dann ein Clustering in syntaktische Konzepte erfolgen (z.B. durch Bestimmung der Bounding Boxes der Fälle mit einer gleichen Diagnose) und das Retrieval auf Basis dieser Konzepte realisiert werden. Die dem Clustering zugrundeliegenden Instanzen können, müssen aber nicht mehr beim Retrieval berücksichtigt werden.

Sind schließlich die genauen Zusammenhänge bekannt, können diese als semantische Konzepte in die Konzepthierarchie eingefügt werden, syntaktische Konzepte und Instanzen bleiben weiterhin verfügbar oder können aus der Wissensbasis entfernt werden.

Natürlich können syntaktische und semantische Konzeptbeschreibungen sowie Fallwissen auch von Anfang an gemischt werden, z.B. wenn bereits Hintergrundwissen vorliegt.

Zusammenfassung der Kerninhalte

Zusammenfassend lassen sich die Inhalte dieser Arbeit damit wie folgt zusammenfassen:

- Wartungstechniker und andere Mitarbeiter des Second-Level-Support benötigen ein System, das sie auf vielfältige und flexible Art und Weise, benutzen können. Zur Unterstützung ihrer Aufgaben sind die im Rahmen fallbasierter Help-Desk-Systeme entwickelten Techniken nicht ausreichend.

- Kernelemente eines Systems, das die Anforderungen von Wartungstechnikern erfüllt, sind:

  - Objektorientierte Repräsentation des Domänen- und des Hintergrundwissens,

- ◆ Flexibles, semantisch fundiertes Retrieval unter Berücksichtigung des Hintergrundwissens.

- ■ Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen erfüllen sowohl die Anforderungen an die Repräsentation sowohl des Domänen als auch des Hintergrundwissens als auch an das flexible, semantisch fundierte Retrieval.

Aufbau der Arbeit

Motivation

Beschreibung der fallbasierten Techniken von Help-Desk-Systemen

Im ersten Kapitel der Arbeit wird der in den darauf folgenden Kapiteln vorgestellte Ansatz zum intelligenten Retrieval auf der Basis von Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen motiviert. Dazu werden die in fallbasierten Help-Desk-Systemen eingesetzten Techniken kurz vorgestellt und es wird begründet, warum für die Unterstützung der zweiten Support-Ebene ein anderer Ansatz benötigt wird.

Leitbeispiel und Anforderungsdefinition

Einführung des Leitbeispiel und Ableitung von Anforderungen an einen objektorientierten Repräsentationsformalismus

An die Motivation schließt sich in Kapitel 2 die Einführung des Leitbeispiels: Unterstützung bei der Diagnose von Rad-Ketten-Getrieben, insbesondere von Fahrradantrieben, an.

Die in der Motivation noch allgemein gehaltenen Anforderungen an ein Unterstützungssystem werden anhand des Leitbeispiels präzisiert, und die Anforderungen an einen ausdrucksstarken objektorientierten Wissensrepräsentationsformalismus mit automatischen Inferenzen herausgearbeitet.

Beschreibungslogiken als Wissensrepräsentationsformalismus

Vorstellung der Ursprünge und Grundlagen von Beschreibungslogiken

In Kapitel 3 wird gezeigt, daß Beschreibungslogiken ein Wissenrepräsentationsformalismus ist, der bereits eine Vielzahl der formulierten Anforderungen erfüllt. Mit normalen Beschreibungslogiken lassen sich allerdings, da sie nur über einem abstrakten Gegenstandsbereich operieren, weder Meßwerte erfassen, noch Abhängigkeiten zwischen mehreren Parametern, wie sie etwa physikalische Gesetze darstellen, beschreiben.

Begründung der Notwendigkeit zur Erweiterung um zulässige konkrete Gegenstandsbereiche

Dies wird erst durch eine Erweiterung der Beschreibungslogiken um so genannte zulässige konkrete Gegenstandsbereiche möglich. Zulässige konkrete Gegenstandsbereiche sind konkrete Gegenstandsbereiche (z.B. numerische), die einige zusätzliche Anforderungen erfüllen müssen, um die Vollständigkeit und Korrektheit der Basisinferenzen nicht zu zerstören.

CTL – Ein beschreibungslogisches System mit ausdrucksstarken konkreten Gegenstandsbereichen

Vorstellung eines modularen und erweiterbaren beschreibungslogischen Systemes

Zunächst werden in diesem Kapitel eine Reihe beschreibungslogischer Systeme der zweiten Generation auf ihren Sprachumfang, sowohl im abstrakten als auch im konkreten Gegenstandsbereich untersucht.

Insbesondere in Bezug auf die Ausdrucksstärke der konkreten Gegenstandsbereiche, der Menge der implementierten Basisinferenzen und der Mächtigkeit der Querysprache sind Mängel gegenüber den postulierten Anforderungen und den theoretischen Ergebnissen aus dem Bereich der Beschreibungslogiken zu verzeichnen.

Daran anschließend wird mit CTL (Kapitel 4) ein beschreibungslogisches System vorgestellt, das diese Mängel überwindet. CTL realisiert vollständige Basisinferenzen über dem abstrakten Gegenstandsbereich auf Basis eines modellgenerierenden Algorithmus. An den beschreibungslogischen Kern können über eine schmale Schnittstelle verschiedene zulässige konkrete Gegenstandsbereiche angebunden werden.

Systematische Betrachtung zulässiger numerischer Gegenstandsbereiche unterschiedlicher Ausdrucksstärke

Nicht jede Anwendung stellt die gleichen Anforderungen an die Ausdrucksstärke des zulässigen konkreten Gegenstandsbereiches. Da die Ausdrucksstärke immer mit der Komplexität des Entscheidungsverfahrens korreliert, ist die Fragestellung welche konkreten Gegenstandsbereiche zulässig sind, eine äußerst interessante. Daher wird im zweiten Abschnitt dieses Kapitels ein breites Spektrum zulässiger numerischer Gegenstandsbereichen zusammen mit den ihnen zugrundeliegenden Entscheidungsverfahren vorgestellt. Zusätzlich wird eine Reihe von Unentscheidbarkeitsresultaten präsentiert, die die Zulässigkeit bestimmter numerischer Gegenstandsbereiche prinzipiell ausschliessen und damit die prinzipiellen Grenzen des in dieser Arbeit verfolgten Ansatzes demonstrieren.

Quantoreneliminationsverfahren erlauben die Berechnung von Parametereinschränkungen

Besonders interessant sind konstruktive Entscheidungsverfahren für konkrete Gegenstandsbereiche. Dies sind Entscheidungsverfahren, die, falls das Problem entscheidbar ist, zugleich eine Lösung des Problems berechnen. Es wird gezeigt, wie sich diese Verfahren, insbesondere Verfahren zur Quantorenelimination, zusammen mit der Modellgenerierung der Basisinferenzen nutzen lassen, um abgeleitetes Wissen wie etwa Parametereinschränkungen etc anzeigen zu lassen.

Vorstellung von RACER

Zum Abschluss des Kapitels wird mit RACER[4] ein beschreibungslogisches System der dritten Generation vorgestellt und auf seine Eignung für die Aufgabenstellung untersucht.

Umsetzung des Leitbeispiel

Repräsentation von Domänen- und Hintergrundwissen

In diesem Kapitel wird zunächst gezeigt, wie es auf Basis der Beschreibungssprache von CTL möglich ist, das im Leitbeispiel formulierte Wissen zu repräsentieren.

4. RACER – REnamed Abox and Concept Expression Reasoner.

Dann wird vorgestellt, wie unter Zuhilfenahme dieses Wissens bereits die Basisinferenzen ein ähnlichkeitsbasiertes Retrieval definieren, und mit Hilfe dieser Inferenzen Aufgaben des Second-Level-Supports gelöst werden können. Konkret werden die Verhaltenssimulation, die Diagnose und die Validierung von Modellbibliotheken betrachtet.

Umsetzung von Simulation, What-If-Analyse, modellbasierter Diagnose und Verwaltung von Modellbibliotheken

Zusammenhänge mit klassischen fallbasierten Systemen

Nach einigen einleitenden Bemerkungen zum generellen Verhältnis zwischen logikbasierten und approximativen Ansätzen zum fallbasierten Schließen wird im ersten Abschnitt von Kapitels 6.1 gezeigt, wie Verfahren zum ähnlichkeitsbasierten Retrieval auf Basis der in CTL entwickelten Techniken und Basisinferenzen umgesetzt werden können.

Realisierung von syntaktischen Verfahren zum ähnlichkeitsbasierten Retrieval auf Grundlage der Basisinferenzen

Im zweiten Teil des Kapitels wird dann untersucht inwieweit sich Fallrepräsentationen und Indexstrukturen logikbasierter und klassischer Ansätze zum fallbasierten Schliessen aneinander annähern.

Zusammenfassung und Ausblick

Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen sind somit ein Formalismus, der es erlaubt, die Anforderungen an ein Second-Level-Support-System zu erfüllen. Ihre Flexibilität ermöglicht ein breites Einsatzspektrum, das von dem fast völligen Verzicht auf die Repräsentation von Hintergrundwissen, der Verwendung einfacher konkreter Gegenstandsbereiche und dem Einsatz herkömmlicher indexbasierter Verfahren des ähnlichkeitsbasierten Retrievals am einen Ende des Spektrums, bis hin zu der Repräsentation von Hintergrundwissens, der Verwendung ausdrucksstarker konkreter Gegenstandsbereiche und des über die Basisinferenzen definierten deduktivem Ähnlichkeitsmaßes am anderen Ende des Spektrums reicht. All dies ist auf Basis eines modularen Systems und eines einheitlichen Repräsentationsformalismus möglich.

Die Anforderungen an Second-Level-Support-Systeme werden durch CTL erfüllt.

Der Hauptteil der Arbeit schließt mit einer Zusammenfassung der erzielten Ergebnisse. Dabei werden diese in den Entwicklungskontext an der Universität Hamburg eingebettet.

1 Motivation

In diesem Kapitel soll in Abschnitt 1.1 zunächst kurz auf Help-Desk-Systeme und die dort realisierte fallbasierte Diagnoseunterstützung eingegangen werden. In Abschnitt 1.2 wird aufgezeigt, daß für die Unterstützung von Experten durch Second-Level-Support-Systeme ein anderer Systemansatz benötigt wird. Die Basisarchitektur eines solchen Systemes wird vorgestellt und die grundlegenden Anforderungen werden formuliert.

1.1. Help-Desk-Systeme

Es wird oft vorgeschlagen, die in Help-Desk-Systemen entwickelten Techniken zur fallbasierten Diagnose auch zur Unterstützung des Second-Level-Supports zu verwenden. Um einen Vergleich mit den Anforderungen an ein Second-Level-Support-System zu ermöglichen, sollen daher zunächst die Grundlagen dieser Techniken präsentiert werden.

Einsatzszenario

Benutzer, die bei der Bedienung eines bestimmten Gerätes Hilfestellung benötigen, wenden sich an einen Help-Desk-Mitarbeiter [1]. Dieser hört sich die Problembeschreibung des Benutzers an und gibt, auf der Basis seiner Erfahrungen eine Empfehlung zur Behebung des Problems.

Läßt sich das Problem auch mit der Hilfe des Help-Desks-Mitarbeiters nicht durch den Kunden beheben, veranlasst der Help-Desk-Mitarbeiter die notwendigen Schritte, etwa den Besuch eines Wartungstechnikers oder die Einlieferung des Gerätes. Kann er die Ursache für das Problem nicht identifizieren, leitet er es an die nächste Supportebene weiter, die sich dann des Problemes annimmt.

1. Oft sind Help-Desks Teil eines umfassenderen Call-Centers. In einem Call-Center erhält ein Kunde typischerweise neben Unterstützung bei technischen Fragestellungen auch Informationen über neue Geräte, offene Rechnungen, etc. kurz gesagt: Antworten auf alle seine Fragen.

Anforderungen

Mitarbeiter, die allein aus ihrer Erfahrung heraus Probleme identifizieren und Lösungshinweise geben können, sind schwer zu finden bzw. teuer. Das Ziel eines Help-Desk-Systems ist es daher, weniger erfahrene Personen in die Lage zu versetzen, häufig vorkommende Standardfragen zu beantworten und einfache Probleme zu lösen. Dadurch werden die Experten entlastet und können sich den komplizierten, außergewöhnlichen Problemen und Fragen annehmen.

Aktuelle Help-Desk-Systeme erfüllen diese Zielsetzung: Help-Desk-Mitarbeiter sind häufig angelernte Kräfte, etwa Schüler und Studenten, die nur ein spezielles Training zur Benutzung des Help-Desk-Systems erhalten, aber keinerlei (bzw. kaum) Fachwissen besitzen.

1.1.1. Die Rolle des fallbasierten Schließens in Help-Desk-Systemen

Zur Unterstützung der Problemlösung bietet sich die Verwendung fallbasierter Techniken an. So werden Help-Desk-Systeme oft als der kommerziell erfolgreichste Anwendungsbereich fallbasierter Technologie genannt. Allerdings tragen zum Erfolg der Help-Desk-Systeme zu einem beträchtlichen Teil auch die mit der Einführung des Help-Desk-Systems verbundene Neu- und Umorganisation des Unternehmens, sowie nicht direkt mit fallbasierten Techniken assoziierte Teile des Systems, etwa das sog. ≫Call-tracking≪- und Problem-Management-Komponenten bei [Weß96].

Zudem wird in vielen Help-Desk-Systemen die fallbasierte Unterstützung der Problemlösung nur als eine einer Reihe von Methoden zur Unterstützung der Problemlösung eingesetzt. Sie ist allerdings neben der Volltextsuche die erfolgreichste der angebotenen Problemlösungsmethoden.

1.1.2. Das Prinzip der fallbasierten Diagnoseunterstützung

Bei der fallbasierten Diagnoseunterstützung werden zunächst die Beschreibungen einer Reihe von Standardproblemen zusammen mit ihren Lösungen in einer so genannten Fallbasis abgelegt. Die Diagnose eines bislang nicht erfassten Problemfalles erfolgt dann auf Grundlage der Hypothese, daß ähnliche Symptome ähnliche Lösungen implizieren.

In einer vom Benutzer gelieferten Problembeschreibung versuchen die Help-Desk-Mitarbeiter daher zunächst Muster und bestimmte Schlüsselbegriffe, etwa den Typ des Gerätes, die Art des Problemes etc. zu identifizieren und in eine initiale Problemfallbeschreibung zu übersetzen. Auf Basis dieser Fallbeschreibung bestimmt dann das fallbasierte Help-Desk-System die zum Problemfall ähnlichsten Fälle der Fallbasis.

In einfachen Help-Desk-Systemen werden die Fälle der Fallbasis bzw. deren Diagnosen in absteigender Reihenfolge des Ähnlichkeitswertes dem Help-Desk-

Mitarbeiter präsentiert. Aus den dort angezeigten Werten leitet der Mitarbeiter seine nächsten Schritte bzw. Fragen an den Benutzer ab.

Fortgeschrittene Help-Desk-Systeme, wie z.B. CBRExpress, leiten zusätzlich aus den Unterschieden zwischen dem aktuellen Problemfall und den ähnlichsten Fälle der Fallbasis Diskriminatoren ab. D.h. es werden die Symptome bestimmt, deren Erhebung einen möglichst großen Fortschritt im Diagnoseprozeß bedeutet. Dem Help-Desk-Mitarbeiter wird dann die Symptomerhebung, in entsprechende Fragen umgesetzt, ebenfalls präsentiert (siehe Abb. 1.1).

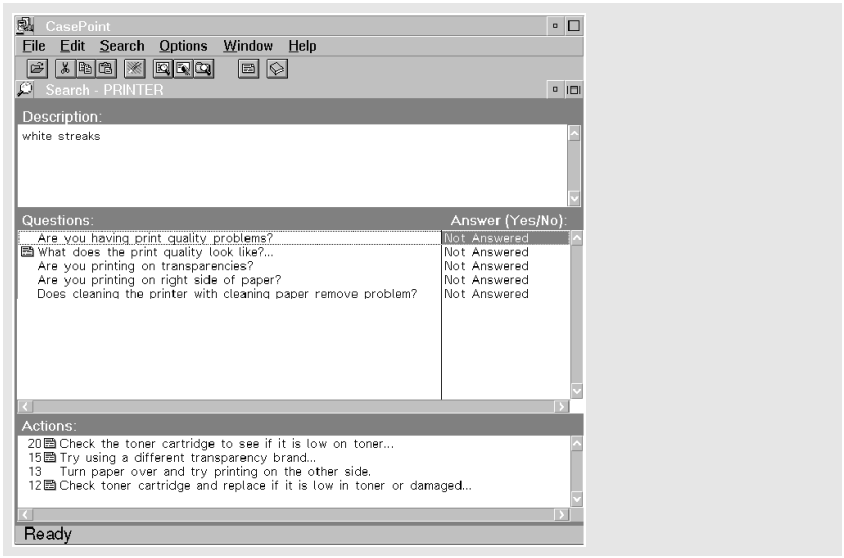


Abbildung 1.1.

Benutzeroberfläche von CBRExpress

Der Mitarbeiter muß in diesem System dann nur noch eine Frage an den Kunden weiterleiten und die erhaltene Antwort in das System eingeben. Dadurch wird die Ähnlichkeitsbestimmung erneut angestoßen.

Wird in diesen Systemen bei der Ähnlichkeitsbestimmung ein bestimmter Wert überschritten, wird die Diagnose des Falles in der Fallbasis, der den Schwellwert überschreitet, vom Help-Desk-System auch für den aktuellen Problemfall vorgeschlagen (bzw. die dort gespeicherte Aktion zur Behebung des Problems präsentiert).

Überschreiten mehrere Fälle der Fallbasis den Schwellwert, so wird eine Mehrheitsentscheidung bei der Bestimmung der Diagnose gefällt.

Kann mit dieser Vorgehensweise keine Diagnose gefunden werden, d.h. es können keine weiteren Fragen mehr gestellt werden bzw. keine weiteren alternativen Diagnosen mehr gefunden werden, dann wird das Problem »eskaliert« und an die nächste Supportebene weitergeleitet.

1.1.3. Grundlagen

Übersetzt in eine formalere Notation läßt sich die obige Vorgehensweise wie folgt beschreiben:

Fallrepräsentation

Fälle werden in Help-Desk-Systemen zumeist in Form von Attribut-Wert-Vektoren repräsentiert. Jeder einzelne Fall der Fallbasis wird dabei in der Form: Fall = Symptome + Diagnose beschrieben.

D.h. ein Fall C_i einer Fallbasis CB ist ein Tupel

$$C_i = \langle s_i, d_i \rangle = \langle \langle s_{i1}, \ldots, s_{in} \rangle, d_i \rangle \in CB = \{C_1, \ldots, C_m\}$$

bestehend aus einem endlichen Merkmalsvektor s_i der Dimensionalität n und der Diagnose d_i. sym und diag sind die Projektionen eines Falles, mit ihren natürlichen Erweiterungen auf Fallmengen, d.h.:

$$\mathrm{sym}(C) = \mathrm{sym}(\langle s, d \rangle) = s \; ; \quad \mathrm{sym}(C') = \bigcup_{C_i \in C'} \mathrm{sym}(C_i),$$
$$\mathrm{diag}(C) = \mathrm{diag}(\langle s, d \rangle) = d \; ; \quad \mathrm{diag}(C') = \bigcup_{C_i \in C'} \mathrm{diag}(C_i),$$

Weiterhin ist für eine beliebige Teilmenge $C' \subseteq CB$

$$C'|_d = \{C_i \in C' \mid \mathrm{diag}(C_i) = d\}$$

die Menge der Fälle in C' mit der Diagnose d.

Ein neuer Problemfall D wird, da die Diagnose hier noch unbekannt ist, als Vektor seiner Symptome repräsentiert. Ein Problemfall D ist also ein n-dimensionaler Vektor

$$D = \langle s_1, \ldots, s_n \rangle$$

Merkmalsarten

Bei den in den verschiedenen Fallrepräsentationen verwendeten Merkmalen kann zwischen quantitativen und qualitativen Merkmalen unterschieden werden. Quantitative Merkmale sind Merkmale über numerischen Wertebereichen. Die Wertebereiche qualitativer Merkmale sind im Gegensatz dazu Symbolmengen. In der Regel umfaßt der Wertebereich eines qualitativen Merkmals nur einige wenige Ausprägungen, während die Kardinalität eines quantitativen Wertebereichs sehr viel höher liegt und in vielen Fällen unendlich ist. Qualitative Wertebereiche mit zwei Ausprägungen heissen auch binäre Merkmale.

Merkmalsskalen

Weiterhin können Merkmale nach der Art der verwendeten Skala (bzw. des in der Skala kodierten Informationsgehaltes) unterschieden werden. Es gibt drei Gruppen von Skalen: nominale Skalen, ordinale Skalen und kardinale Skalen.

Nominale Skalen besitzen den geringsten Informationsgehalt, kardinale Skalen den höchsten. Auf einer nominalen Skala können nur die unterschiedlichen Ausprägungen des Merkmals unterschieden werden:

$$x_i = x_j, x_i \neq x_j$$

Die ordinale Skala liefert zusätzlich Informationen über den Rang der Ausprägung in Bezug auf andere Ausprägungen des Merkmals:

$$x_i > x_j, x_i < x_j$$

Auf kardinalen Skalen läßt sich zusätzlich der Abstand der beiden Merkmalsausprägungen quantifizieren:

$$d(x_i, x_j) = |x_i - x_j|$$

Häufig werden in praktischen Realisierungen ordinale und nominale Skalen in kardinale Skalen eingebettet. Im Regelfall ist die kardinale Skala ein Ausschnitt der reellen Zahlen $\mathbb{R}$. So kann etwa die ordinale Skala

$$\text{gering} < \text{mittel} < \text{hoch}$$

eines qualitativen Merkmals über

$$f(\text{gering}) = 0 < f(\text{mittel}) = 1 < f(\text{hoch}) = 2$$

in die kardinale Skala der positiven reellen Zahlen $\mathbb{R}^+$ eingebettet werden. Die Motivation für diese Einbettung ist die leichtere maschinelle Handhabbarkeit kardinaler Skalen.

Werden die resultierenden kardinalen Skalen auch zur Ähnlichkeitsbestimmung benutzt, so muß man sich bewußt sein, daß man künstlich den Informationsgehalt der Daten, die der Ähnlichkeitsbestimmung zugrunde liegen, erhöht hat. Diese künstliche Erhöhung des Informationsgehaltes kann eine Reihe von Effekten haben, die das Ergebnis der Ähnlichkeitsbestimmung verändern und damit die Ähnlichkeitsaussage verfälschen können.

Ähnlichkeit von Fällen

Zur Ähnlichkeitsbestimmung zwischen einem neuen Problemfall D und einem Fall der Fallbasis C wird eine numerische Bewertungsfunktion sim über den beiden Merkmalsvektoren verwendet:

$$\text{sim}(D, C) = f(D, \text{sym}(C))$$

Statt eines Ähnlichkeitsmaßes wird häufig auch ein Distanzmaß verwendet: Während Ähnlichkeitsmaße $\mathrm{sim}(D, C)$ den Grad der Ähnlichkeit zwischen zwei Fällen bestimmen, dienen Distanzmaße $\mathrm{dist}(D, C)$ der Bestimmung des Grades der Unähnlichkeit (des Abstandes) zweier Fälle. Ähnlichkeitsmaße und Distanzmaße sind reflexive und symmetrische Funktionen und ineinander transformierbar [2].

Oft wird als Bewertungsfunktion eine gewichtete Summe der Ähnlichkeiten (bzw. der Distanzen) der korrespondierenden Symptome verwendet [3]:

$$\mathrm{sim}(X, Y) = \sum w_i \, \mathrm{sim_i}(x_i, y_i);$$
$$\mathrm{dist}(X, Y) = \sum w_i \, \mathrm{dist_i}(x_i, y_i)$$

Aber auch andere Funktionen, wie etwa das Tversky-Maß [Tve77], kommen zum Einsatz.

Ähnlichkeit von Merkmalen

Die Ähnlichkeit korrespondierender Merkmale wird im einfachsten Fall über die Identitätsfunktion bestimmt:

$$\mathrm{sim_i}(x_i, y_i) = \mathrm{id}(x_i, y_i) = \begin{cases} 1 & \text{falls } x_i = y_i \\ 0 & \text{sonst.} \end{cases}$$

Zumeist werden jedoch komplexere Funktionen verwendet. Eine oft verwendete Funktion ist die folgende Transformation eines lokalen Distanzmaßes $d_i(x_i, y_i)$ in ein lokales Ähnlichkeitsmaß:

$$\mathrm{sim_i}(x_i, y_i) = 1 - \frac{\mathrm{dist_i}(x_i, y_i)}{\mathrm{dist_i}(x_i, y_i) + 1}$$

Dabei findet in den meisten Fällen das Distanzmaß $d_i(x_i, y_i) = |x_i - y_i|$ Verwendung.

Diagnoseermittlung

Die Diagnose $d(D)$ für den aktuellen Problemfall D wird auf Basis der Diagnosen einer Teilmenge $C_{rel}(D)$ der Fälle der Fallbasis ermittelt:

$$d(D) = \mathrm{diag}(C_{rel}(D)), \qquad C_{rel}(D) \subseteq CB$$

2. Eine Übersicht über den Zusammenhang zwischen Ähnlichkeits- und Distanzmaßen findet sich z.B. in [Weß95].

3. Im folgenden wird, der einfacheren Notation halber, bei der Bestimmung der Ähnlichkeit bzw. des Abstandes ein Fall der Fallbasis mit seinem Merkmalsvektor identifiziert, d.h.: $\mathrm{sim}(D, C) := \mathrm{sim}(D, \mathrm{sym}(C))$

Im einfachsten Fall wird die Diagnose d_k des ähnlichsten Falles der Fallbasis übernommen [4]:

$$C_{rel}(D) = \{C_k \mid \text{sim}(D, C_k) \geq \text{sim}(D, C_j), 1 \leq j \leq m = |CB|\}$$

Um zu vermeiden, daß allzu unähnliche Diagnosen vorgeschlagen werden, wird in einer Reihe von Systemen eine Diagnose erst dann vorgeschlagen, wenn eine gewisse Mindestähnlichkeit vorliegt:

$$C_{rel}(D) = \{C_k \mid \text{sim}(D, C_k) \geq \text{sim}(D, C_j) \geq sim_\delta, 1 \leq j \leq |CB|\}$$

Statt der Diagnose des ähnlichsten Falles werden häufig auch die Diagnosen der n ähnlichsten Fälle bestimmt:

$$C_{rel}(D) = \{C_{rel_1}, \ldots, C_{rel_n}\}, \text{mit}$$
$$\text{sim}(D, C_i) \geq \text{sim}(D, C_j), C_i \in C_{rel}, C_j \in CB \setminus C_{rel}$$

Eine weitere Alternative für die Diagnoseermittlung ist es, alle Diagnosen der Fälle, die eine gewisse Mindestähnlichkeit aufweisen, als Ergebnismenge zurückzuliefern:

$$C_{rel}(D) = \{C_k \mid \text{sim}(D, C_k) \geq sim_\delta\}$$

Insbesondere in den letzten beiden Fällen ist die Definition einer Ordnung auf der Menge der zurückgelieferten Diagnosen wichtig. Diese läßt sich über die Ähnlichkeit der der jeweiligen Diagnose zugrundeliegenden Fälle der Fallbasis definieren. So definiert z.B.

$$< (d_i, d_j) := sim(D, d_i) > sim(D, d_j) \text{ mit}$$
$$sim(D, d) = \min_{C_k \in C_{rel}|_d} \text{sim}(D, s_k)$$

eine Ordnungsrelation über der Menge der zurückgelieferten Diagnosen. Genauso gut läßt sich eine Ordnung auch über die mittlere Ähnlichkeit der einer Diagnose zugrundeliegenden Fälle

$$< (d_i, d_j) := sim(D, d_i) > sim(D, d_j) \text{ mit}$$
$$sim(D, d) = \frac{\sum_{C_k \in C_{rel}|_d} \text{sim}(D, s_k)}{|C_{rel}|_d|}$$

oder die Häufigkeit des Vorkommens der Diagnose d_i in C_{rel} definieren.

$$< (d_i, d_j) := |C_{rel}|_{d_i}| > |C_{rel}|_{d_j}|$$

4. Sollten zwei Fälle mit unterschiedlichen Diagnosen den gleichen Ähnlichkeitswert aufweisen, dann werden beide Diagnosen als Wert von $d(D)$ zurückgeliefert.

Geometrische Interpretation

Die Merkmalsvektoren des aktuellen Problemfalles und der Fälle der Fallbasis lassen sich als Punkte eines Raumes entsprechender Dimension geometrisch interpretieren.

Die Ähnlichkeit eines Falles der Fallbasis zum aktuellen Problemfall läßt sich nach der Transformation des Ähnlichkeitsmaßes in ein Distanzmaß, in dieser geometrischen Interpretation als der Abstand der beiden entsprechenden Punkte visualisieren. Färbt man zusätzlich die Fälle der Fallbasis entsprechend ihrer Diagnose, dann lassen sich die unterschiedlichen Strategien zur Diagnoseermittlung visualisieren (siehe Abb. 1.2).

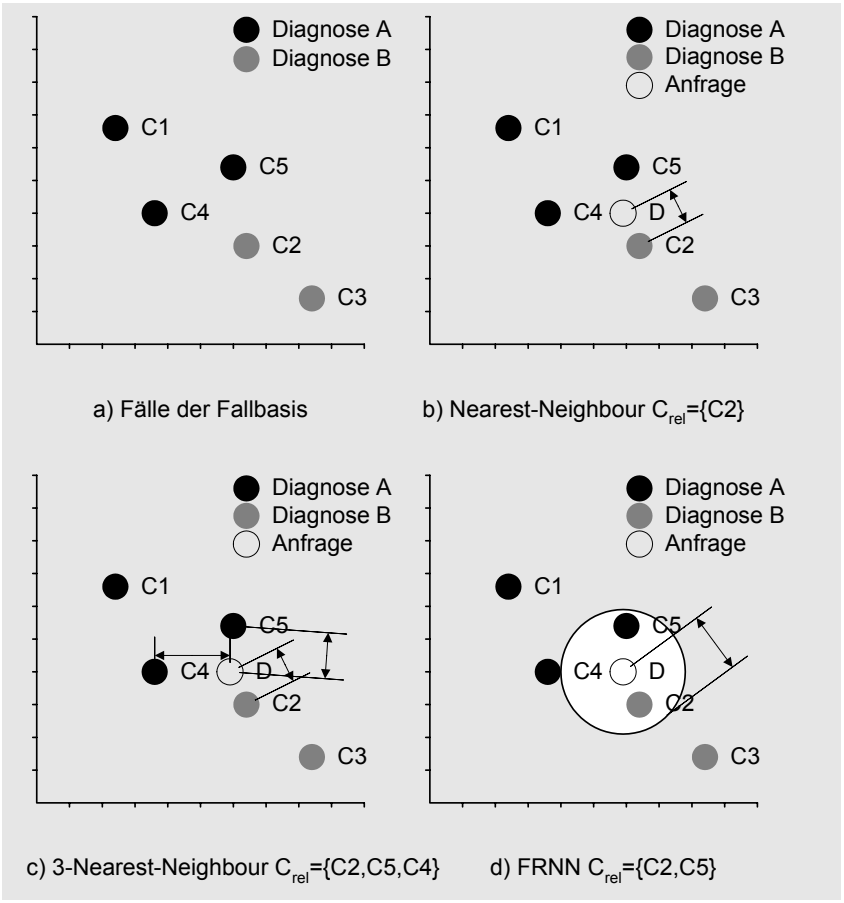


Abbildung 1.2.

Geometrische Interpretation der Ähnlichkeitsbestimmung in Help-Desk-Systemen

Gesucht ist also im einfachsten Fall der Punkt der Fallbasis, der dem Punkt des aktuellen Problemfalles am nächsten liegt, gesucht ist der »nächste Nachbar« des Anfragepunktes.

Nearest-Neighbour-Problem Dies ist eine so häufige auftauchende Fragestellung, daß das Problem in der Mathematik einen eigenen Namen bekam, es ist unter dem Namen »Nearest-Neighbour-Problem« bekannt.

k-Nearest-Neighbour-Problem Werden die k ähnlichsten Fälle gesucht, entspricht die Fragestellung dem »k-Nearest-Neighbour-Problem«.

Fixed-Radius-Nearest-Neighbour-Problem Auch die anderen angedeuteten Varianten lassen sich geometrisch interpretieren. Die Überschreitung eines gewissen Schwellwertes entspricht der Unterschreitung eines gewissen Abstandes, in der Mathematik ist diese Frage auch als »Fixed-Radius-Nearest-Neighbour-Problem« bekannt

Eine Mehrheitsentscheidung bei der Diagnose entspricht der Bestimmung aller Fälle der Fallbasis, die einen Schwellwertabstand unterschreiten und der Auszählung der Farben dieser Fälle. Die Farbe mit der größten Häufigkeit bestimmt die Diagnose.

1.1.4. Implementierungen fallbasierter Diagnoseunterstützung

Ein Schwerpunkt bei der Implementierung der unterschiedlichen Help-Desk-Systeme liegt darauf, die Bestimmung der nächsten n-Nachbarn möglichst effizient zu realisieren und für das konkrete Einsatzszenario zu optimieren. Dieses unterscheidet sich von den ursprünglichen Betrachtungen in der Mathematik unter anderem dadurch, daß der Raum i.d.R. eine deutlich höhere Dimension aufweist (Ausgangspunkt in der Mathematik waren konkrete geometrische Fragestellungen in zwei resp. dreidimensionalen Räumen).

Weiterhin kann nicht davon ausgegangen werden, daß es sich bei den Fällen der Fallbasis und insbesondere der Problemfälle tatsächlich um Punkte in einem n-dimensionalen Raum handelt. Vielmehr sind zumeist eine Reihe der Attributwerte unbekannt bzw. nur näherungsweise bekannt.

Daher werden in fallbasierten Systemen z.B. spezielle Zugriffstrukturen wie n-dimensionale Suchbäume realisiert [Weß95].

Ein anderer Schwerpunkt ist die Bereitstellung von Mechanismen für eine möglichst flexible Definition der Fallrepräsentation, der Merkmale und der Ähnlichkeitsmaße.

1.2. Second-Level-Support-Systeme

In diesem Abschnitt werden die Anforderungen an ein Second-Level-Support-System formuliert und gezeigt, daß die Techniken zur fallbasierten Diagnoseunterstützung in Help-Desk-Systemen, die im vorhergehenden Abschnitt vorgestellt wurden, zur Realisierung dieser Anforderungen ungeeignet sind [Kam94c].

1.2.1. Anforderungen

Daß gravierende Unterschiede bei den Anforderungen an Unterstützungssysteme für die erste und zweite Supportebene bestehen, wird in der folgenden, sehr treffenden, Zusammenfassung von Pepper [Pep96b] deutlich:

> ≫First-level analysts want integration with their call management system, easy to use search techniques, a single correct answer and complete instructions how to implement the solution.
> Second-level analysts, however, are more interested in using a **research tool** for solving **tough problems**, they like to search through **large databases** and **examine multiple possible solutions**.≪

Im folgenden sollen die sich aus dieser Zusammenfassung resultierenden inhaltlichen Anforderungen an ein Second-Level-Support-System (SLSS) genauer untersucht werden.

Research Tool

Während bei Laien und angelernten Kräften wie Help-Desk-Mitarbeitern eine Architektur vorzuziehen ist, bei der die Initiative beim System liegt, und der Benutzer bei der Fehlersuche angeleitet wird, stößt ein solches System bei Experten sehr schnell auf Ablehnung.

Soll ein Experte bei seiner Arbeit unterstützt werden, so muß er eine aktive Rolle bei der Systembenutzung einnehmen können. Dies ist nicht nur aufgrund der Aufgabenstellung notwendig, sondern entspricht auch der Grundhaltung von Experten. Sie fühlen sich bevormundet, wenn sie in eine passive Rolle gedrängt werden. Die Gründe dafür sind in ihrer Kompetenz und ihrer traditionell unabhängigen Arbeitsweise zu sehen.

Diese Grundhaltung ist unabhängig von der tatsächlichen Leistungsfähigkeit des Systems. Auch ein ideales, fehlerfreies System, das den Experten in eine passive, eher handlangerartige Rolle drängt, würde nicht akzeptiert werden. Daher muß ein System für Experten die Initiative dem Experten überlassen.

Multiple Possible Solutions

Zur optimalen Umsetzung ihrer Aufgaben benötigen Experten ein System, das ihnen den flexiblen, inhaltsorientierten Zugriff auf situationsrelevante Daten ermöglicht. Nur so können sie die ≫multiple possible solutions≪, die sie identifiziert haben, begutachten und überprüfen.

Experten sind, im Gegensatz zu Laien und angelernten Kräften, in der Lage, die in einer bestimmten Situation aktuellen Relevanzkriterien selbst zu spezifizieren [5],

5. Es gilt auch hier das oben gesagte. Wichtiger als die tatsächliche Fähigkeit ist die Tatsache, daß die Experten *meinen*, sie könnten die Relevanzkriterien spezifizieren.

und können daher ein solch flexibles System sinnvoll nutzen.

Large Databases

Im Vergleich zu angelernten Mitarbeitern verfügen Experten über ein erheblich größeres Fachwissen und einen viel größeren Erfahrungsschatz. Im Gegensatz zu Help-Desk-Mitarbeitern, die entweder nur für ein relativ enges Produktspektrum zuständig sind und dort etwas tiefergehendes Wissen aufweisen oder sehr oberflächliches Wissen für ein breiteres Produktspektrum besitzen, decken z.B. Wartungstechniker in ihrer Tätigkeit häufig das ganze Produktspektrum eines Herstellers ab und besitzen umfangreiches und ≫tiefes≪ Wissen über diese Geräte und ihre Funktionsweise.

Dazu sind sie aufgrund ihrer Ausbildung und ihres tiefgehenden technischen Verständnis in der Lage. Sie haben jedoch dann Probleme, wenn sie überprüfen sollen, welche der gespeicherten Daten die aktuellen Relevanzkriterien erfüllen oder wenn sie versuchen sich an alle Informationen zu erinnern.

Tough Problems

Experten sind im Gegensatz zu Help-Desk-Mitarbeitern nicht auf eine Aufgabe festgelegt. Sie sind in alle Phasen des Produktlebenszyklus eingebunden. Nicht nur die Problemlösung spielt bei ihnen eine Rolle, sondern auch die regelmäßige Wartung, die vorbeugende Kontrolle, und die Kooperation mit der Produktentwicklung und Konstruktion etc. [Pep96a]:

≫Suddenly the Help-Desk[a] is no longer a „downstream“ activity, solving problems with systems already selected, but it is intimately involved in the whole technology life cycle, from requirements definition to ongoing support.≪

a. Pepper gebraucht hier den Begriff Help-Desk im Kontext der Unterstützung der zweiten Support-Ebene

Zur Lösung ihrer Aufgaben greifen Experten, oftmals selbst Ingenieure, auf ein breites Spektrum von Ingenieurstechniken zurück, die Sie im Rahmen Ihrer Ausbildung erlernt haben.

Für die Produktauswahl und die Anforderungsdefinition sowie für die Diagnose Normal- und Fehlverhalten werden Geräte simuliert, ebenso werden What-If Analysen gemacht. Für die Reparatur und die Produktauswahl wird in Katalogen nach Bauteilen gesucht, die eine bestimmte Spezifikation erfüllen usw. Ein Unterstützungsystem für Experten muß diese Techniken und Verfahren anbieten und integrieren.

1.2.2. Resultierende Anforderungen

Wartungstechniker und andere Mitarbeiter des Second-Level-Support benötigen somit ein System, das sie auf vielfältige Art und Weise benutzen können. Zur Unterstützung ihrer Aufgaben sind die im Rahmen fallbasierter Help-Desk-Systeme entwickelten Techniken nicht ausreichend.

Wartungstechniker benötigen ein flexibles Informationssystem, das in der Lage ist, aus einer gegebenen Menge von Informationen neue Informationen abzuleiten. In einem solchen System muß das zur Ableitung neuer Informationen notwendige Wissen repräsentierbar sein. Die Repräsentation muß jederzeit um zusätzliche Informationen z.B. neue Gerätetypen und um hinzugekommenes Wissen z.B. neue Fehlverhaltensmodelle erweitert werden können.

Darüber hinaus muß das System Mechanismen besitzen, die es erlauben, aus einer gegebenen Menge von Informationen und dem repräsentierten Wissen neue Informationen abzuleiten und diese dem Benutzer zu präsentieren.

Ziel des Systems muß es sein, dem Benutzer so viele Informationen abzuleiten und bereitzustellen, daß er in die Lage versetzt wird, die aktuelle Aufgabe zu lösen[6]. Da das Aufgabenspektrum breit gefächert ist, muß es möglich sein, die aktuelle Situation und den bestehenden Informationsbedarf einfach und flexibel beschreiben zu können.

Aus diesen inhaltlichen Anforderungen resultieren die beiden folgenden methodischen Anforderungen an die zugrundeliegende Systemarchitektur:

- Objektorientierte Repräsentation des Domänen- und des Hintergrundwissens,
- Flexibles, semantisch fundiertes Retrieval unter Berücksichtigung des Hintergrundwissens.

Diese Anforderungen sollen in den nächsten Abschnitten detaillierter untersucht werden.

1.2.3. Objektorientierte Repräsentation des Domänen- und Hintergrundwissens

Ein Experte benötigt für die Unterstützung seiner vielfältigen Aufgaben Domänenwissen, Hintergrundwissen und Informationen über Verwaltungsdaten [Kam93b, KNPB94] (siehe Abb. 1.3):

6. Dies ist in gewissem Sinn auch der prinzipielle Ansatz des ≫Information Retrieval≪, bei dem der Benutzer einen bestimmten Informationsbedarf hat und ein System benutzt, um zusätzliche Informationen zu gewinnen. Im vorliegenden Fall liegt jedoch eindeutig der Schwerpunkt auf der Repräsentation und dem Einsatz von Wissen, ohne den eine Unterstützung der Simulation nicht möglich wäre.

Abbildung 1.3.

Im Second-Level-Support benötigte Informationen

Domänenwissen

Der Zugriff auf die Gerätedaten eines breiten Produktspektrums ist für alle Aufgaben eines Servicetechnikers unverzichtbar. Es muß möglich sein, Wissen über die unterschiedlichen zu unterstützenden Gerätearten abzulegen und zu benutzen.

Dies bedeutet, daß Spezialisierungsbeziehungen zwischen den verschiedenen Gerätetypen und Teil-Ganzes-Beziehungen zwischen Geräteteilen und Geräten repräsentierbar sein müssen. Darüber hinaus müssen die verschiedenen, die Geräte- und deren Teile beschreibenden Attribute und die zulässigen Attributwerte definiert werden können.

Neben den Beschreibungen der konkreten Gerätetypen enthält die Modellierung des Domänenwissens zumeist an der Wurzel der jeweiligen Hierarchie abstrakte Beschreibungen von Geräteklassen. Diese enthalten allgemeingültiges, gerätetypübergreifendes Wissen.

Hintergrundwissen

Neben dem reinen Gerätewissen gibt es in technischen Bereichen verschiedene Formen von Hintergrundwissen. Im Kontext der Arbeit relevante Beispiele für Hintergrundwissen sind:

- Randbedingungen, die die einzelnen Geräte und Geräteteile erfüllen bzw. erfüllen müssen. Ein Beispiel für eine Randbedingung ist die Tatsache, daß Radius eines Zahnrades grösser 0 sein muß.

- Allgemeingültige physikalische Gesetze, wie zum Beispiel das Drehmomentgesetz[7].
- Wissen über Normal- bzw. Fehlverhalten von Komponenten.

Für Experten ist die Verfügbarkeit und Einsetzbarkeit dieses Hintergrundwissens essentiell. Sie benutzen es in Simulationen, What-If-Analysen, Diagnosen etc. ein zur Lösung der jeweiligen Aufgabe. Sie setzen die Verfügbarkeit des Hintergrundwissens in gewissem Sinne sogar voraus, etwa indem sie nur die notwendigsten Angaben zur Beschreibung einer konkreten Situation machen. In ihren Fehlerreports erfassen Sie nur die in der jeweiligen Situation wichtigen Meßwerte. Andere, aus den Basiswerten ableitbare Werte werden nicht erfaßt, da die Ableitung bei Bedarf mit Hilfe des Hintergrundwissens erfolgen kann. Die erfaßten Basiswerte wechseln von Situation zu Situation; es kann nicht kategorisch zwischen Ausgangs- und abgeleiteten Werten[8] unterschieden werden.

Für den Vergleich und die Ähnlichkeitsbestimmung mit anderen Fällen in denen andere Parameter erfaßt wurden, ist daher die Ableitung der abhängigen Parameter notwendig. Dazu müssen aber zunächst die Zusammenhänge zwischen den Parametern erfaßt werden können.

Die Möglichkeit zur deklarativen Repräsentation des Hintergrundwissens ist somit für ein Second-Level-Support-System absolut unabdingbar. Ein solches System muß die Repräsentation des Wissens erlauben und das repräsentierte Wissen zur Ableitung bzw. zur Einschränkung von Attributwerten benutzen.

Verwaltungsdaten

Zusätzlich zu Domänen- und Hintergrundwissen benötigt der Experte Verwaltungsdaten, d.h. Informationen über Mitarbeiter, Kunden etc.. Diese werden vom Wartungstechniker z.B. dazu benötigt, um sich vor einem Besuch über die Wartungshistorie zu informieren. Auch für die Abrechnung etc. sind sie wichtig.

Die Integration dieser Daten in das Gesamtmodell ist aber auch notwendig, um immer wieder auftauchende Fragen, wie etwa ≫Servicetechniker Meyer hatte doch letzten Herbst ein ähnliches Problem≪, behandeln zu können.

Fazit

All diese Anforderungen an den Repräsentationsformalismus können nur durch eine objektorientierte Repräsentation erfüllt werden. Die in aktuellen Help-Desk-Systemen als Repräsentationsformalismus eingesetzten Attribut-Wert-Vektoren sind aus den im folgenden genannten Gründen ungeeignet:

7. Sowohl Randbedingungen als auch physikalische Gesetze lassen sich als Abhängigkeiten zwischen Attributen beschreiben.
8. In der Terminologie von Richter in [Ric95] primäre und sekundäre Attribute.

Komplexe Gerätestrukturen mit einer variablen Anzahl von Teilen unterschiedlichen Typs lassen sich nicht sinnvoll durch einen fixen Vektor beschreiben. Es müßte bei einer solchen Repräsentation Vorsorge für alle Attribute aller möglichen Konfigurationen innerhalb des Gerätespektrums getroffen werden. Durch die daraus folgende kombinatorische Explosion könnte der weitaus größte Teil der Attribute nicht sinnhaft mit Werten belegt werden; die Werte dieser Dimensionen des Vektors wären undefiniert (siehe Beispiel 1.1).

Beispiel 1.1 (Attribut-Wert-Vektor basierte Gerätерepräsentationen (I))
Kühlschmierstoffumlaufanlagen[9] bestehen unter anderem aus Pumpen, Filtern und Tanks. Pumpen, Filter und Tanks können mehrfach in einer Kühlschmierstoffumlaufanlage vorkommen; ihre Anzahl variiert sehr stark.

So gibt es einerseits Anlagen mit nur einem Tank, einem Filter und einer Pumpe. Andere Anlagen haben 20 Tanks, 10 Filter und 15 Pumpen. Neben vielen anderen Pumpenarten gibt es in diesen Anlagen u.a. Zentrifugalpumpen, Membranpumpen, Kolbenpumpen und Zahnradpumpen. Die verschiedenen Typen von Anlageteilen (z.B. die unterschiedlichen Pumpentypen) werden durch unterschiedliche Attribute beschrieben. Während für alle Pumpen der Pumpendruck wichtig ist, gibt es das Attribut Kolbenanzahl nur für Kolbenpumpen. Die Art der Membran spielt nur bei Membranpumpen eine Rolle.

Selbst wenn man vereinfachend annimmt, daß eine Kühlschmierstoffumlaufanlage nur aus Pumpen, Tanks und Filtern besteht und Tanks und Filter nur in jeweils einer Ausprägung existieren und diese mit einem einzigen Attribut (Volumen bzw. Kapazität) beschrieben werden können, muß zur Repräsentation einer Kühlschmierstoffumlaufanlage ein 120 dimensionaler Vektor verwandt werden (siehe Abb. 1.4). Von den 120 Dimensionen sind aber in der überwiegenden Anzahl der Fälle nur wenige besetzt.

Die Verschwendung von Ressourcen (insbesondere von Speicherplatz) ließe sich evtl. durch die Verwendung von so genannten »sparse vectors« vermeiden. Eine große Anzahl unbekannter Werte hat bei fallbasierten Systemen allerdings die fatale Folge, daß sich die Leistungsfähigkeit der Verfahren zur Ähnlichkeitsbestimmung drastisch verringert.

Die Tatsache, daß Geräte gleichartige (und gleichberechtigte) Komponenten mehrfach enthalten können, erschwert die Ähnlichkeitsbestimmung zusätzlich, da jede Sortierung auf der Menge der Komponenten willkürlich sein muß. Eine Sortierung muß aber bei der Abbildung auf einen Attribut-Wert-Vektor zwangsläufig erfolgen.

Somit läßt sich kein Ähnlichkeitsmaß definieren, das nur durch den Vergleich der Werte korrespondierender Attribute befriedigend funktioniert. Für eine aussagekräftige Ähnlichkeitsbestimmung müssen grundsätzlich alle Permutationen der gleichberechtigten Komponenten miteinander verglichen werden (siehe Beispiel 1.2).

9. Dieses Beispiel stammt aus AMS, einem vom Autor am Labor für Künstliche Intelligenz der Uni Hamburg durchgeführten Projekt zur Unterstützung der Wartungschemiker der Castrol Industrie.

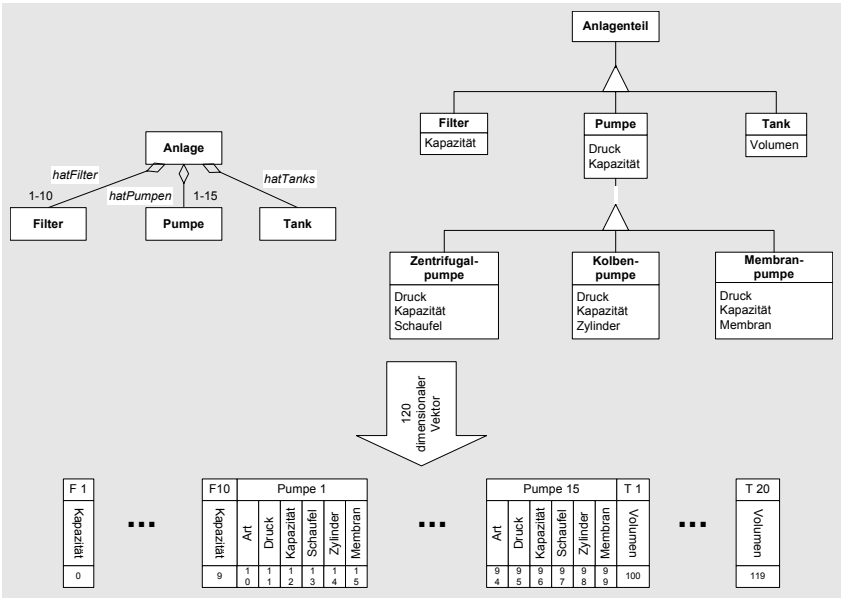


Abbildung 1.4.

Probleme mit Repräsentationen auf Basis von Attribut-Wert Vektoren

Beispiel 1.2 (Attribut-Wert-Vektor basierte Gerätерepräsentationen (II))
Vereinfachend soll in diesem Beispiel eine Kühlschmierstoffumlaufanlage aus maximal 2 Tanks bestehen, die durch ihr jeweiliges Volumen beschrieben werden. Damit läßt sich eine Anlage als zweidimensionaler Vektor $\langle v_1, v_2 \rangle$ beschreiben. Zur Bestimmung der Ähnlichkeit $\mathrm{sim_i}(x_i, y_i)$ korrespondierender Attributwerte zweier Anlagen X und Y soll die Identitätsfunktion id eingesetzt werden. Die Gesamtähnlichkeit zweier Anlagen ergibt sich aus der Summe der Attributähnlichkeiten: $\mathrm{sim}(X, Y) = \sum \mathrm{sim_i}(x_i, y_i)$.

Damit ergibt sich als Gesamtähnlichkeit $\mathrm{sim}(A_1, A_2)$ zweier Ein-Tank-Anlagen $A_1 = \langle x, \perp \rangle$[10] und $A_2 = \langle \perp, y \rangle$ immer 0, unabhängig von den Werten x und y der Tankvolumina.

Auch eine zwangsweise Sortierung, etwa derart, daß die Eintragung von Attributwerten zunächst in den niedrigen Indizes erfolgen muß, ist wenig hilfreich. Sie verhindert z.B. nicht, daß für $A_3 = \langle 0, 5x, x \rangle$ die Gesamtähnlichkeit $\mathrm{sim}(A_1, A_3) = 0$ ergibt, obwohl zwei gleichgroße Tanks in A_1 und A_3 verwendet werden.

Weitere Faktoren

Der Forderung nach einer objektorientierten Repräsentation wird noch durch folgende Faktoren bestärkt:

Einheitlichkeit Da die Situation bestimmt, welche Informationen für den Experten von Interesse sind, muß die Repräsentation aller Daten auf einheitliche Art

10. $\perp$ notiert einen undefinierten Wert

und Weise erfolgen. Eine zweigeteilte Architektur, die zwischen Call-Management-Daten und Daten zur Problemlösung unterscheidet, wie sie in einer Reihe von Help-Desk-Systemen eingesetzt wird, ist für Second-Level-Support-Systeme nicht akzeptabel. Dies bedeutet insbesondere, daß es kein dediziertes Objekt »Fall« geben kann, auf das das System besonders zugeschnitten ist. Alle Daten sind gleichberechtigt.

Erweiterbarkeit Die leichte Erweiterbarkeit und damit die leichte Wissensakquisition wird oft als Vorteil des fallbasierten Schließens (und auch der Volltextsuche) genannt. Gemeint ist damit die Erweiterung der Wissensbasis durch Hinzufügen neuer Fälle. Dies geschieht allerdings nur im Rahmen einer bereits bestehenden Domänenmodellierung.

Das Hinzufügen neuer Fragen und damit die Definition neuer Attribute ist im Rahmen von CasePoint[11] nicht möglich. Dazu muß auf die Entwicklungsumgebung CBRExpress zurückgegriffen werden.

Oftmals kommen jedoch in Help-Desk-Systemen vorgefertigte KnowledgePaks zum Einsatz und es steht vor Ort gar keine Entwicklungsumgebung zur Verfügung. Damit kann die Fallstruktur nicht geändert werden.

Wissensakquisition Auch das häufig angebrachte Argument, daß die Wissensakquisition für komplexe objektorientierte Repräsentation zu aufwendig ist, gilt im Bereich des Second-Level-Supports nur bedingt.

Im Bereich der Wartungsunterstützung ist das Wissen über die unterschiedlichen Gerätetypen und die sie beschreibenden Attribute im Normalfall vorhanden. Daher trifft in diesem Fall das oft benutzte Argument, die Wissensakquisition sei zu aufwendig, nicht zu. Zudem ist durch die zunehmende elektronische Bereitstellung von Produktdaten und die Standardisierung von Produktdatenformaten (etwa im Rahmen von STEP[12]) in Zukunft mit einer weiteren Vereinfachung der Wissensakquisition zu rechnen.

1.2.4. Flexibles, semantisch fundiertes Retrieval

Um flexibel auf die gewünschten Informationen zugreifen zu können, wird ein Werkzeug benötigt, in dem der Benutzer seine Informationsbedürfnisse adäquat ausdrücken kann, und das dafür sorgt, daß die relevanten Einträge in der Daten- und Wissensbasis zurückgeliefert werden.

Diese Anforderung kann nicht, wie in Help-Desk-Systemen, über die Verwendung eines einzigen, festen Ähnlichkeitsmaßes zur Ähnlichkeitsbestimmung rea-

11. CasePoint – CasePoint ist das meist genutzte Werkzeug zum fallbasierten Schließen in existierenden Help-Desk-Systemen..

12. STEP – Standard for the Exchange of Product Data.

lisiert werden. Stattdessen muß es eine flexible Anfragesprache erlauben, bestehende Einträge in der Daten- und Wissensbasis über komplexe Beziehungen zu verknüpfen und das Relevanz- bzw. Ähnlichkeitskriterium bei jeder Anfrage neu definieren zu können.

Zur Beschreibung der komplexen Beziehungen gilt das im letzten Abschnitt zur Beschreibung des Hintergrundwissens Gesagte. Der Experte muß die Möglichkeit haben, Randbedingungen an Attributwerte zu stellen und Abhängigkeiten zwischen Attributen zu definieren.

Die Anfrageverarbeitung muß die an die unterschiedlichen Attribute gestellten Randbedingungen und die Abhängigkeiten zwischen Attributen berücksichtigen. Als Ergebnis der Anfrage werden die Einträge der Daten- und Wissensbasis zurückgeliefert, die der Anfrage nach Ableitung aller Einschränkungen und Abhängigkeiten genügen.

Klare Retrievalsemantik

Ähnlichkeits- und Relevanzmaße in Form von mehr oder weniger komplizierten Berechnungsfunktionen, wie sie in Help-Desk-Systemen benutzt werden, haben nur einen geringen Erklärungswert. Ihre Semantik ist nicht transparent für den Benutzer.

Eine klare Semantik ist aber gerade bei Experten wichtig. Während Laien für jeden Hinweis dankbar sind, hinterfragen Experten das Ergebnis und sind in der Lage, es zu beurteilen. Eine auf den numerischen Wert eines Ähnlichkeitsmaßes reduzierte Begründung des Retrievalergebnisses ist für diesen Personenkreis unbefriedigend.

Ein System für Experten muß im Kern eine Anfragesprache mit einer sauber definierten Semantik besitzen. Die Mechanismen zur Bearbeitung der formulierten Anfragen müssen gewährleisten, daß die in der Anfragesprache formulierbaren Anfragen entsprechend der definierten Semantik beantwortet werden. Dies bedeutet insbesondere, daß syntaktisch verschiedene Varianten einer semantisch identischen Anfrage das gleiche Retrievalergebnis liefern müssen. Aufbauend auf dem semantisch sauberen Kern des Retrievals können dann weitere Retrievalmechanismen realisiert werden.

Ähnlichkeitsbasiertes Retrieval durch Berücksichtigung des Hintergrundwissens

Um sinnvolle Ergebnisse zu liefern, muß bei allen vom System ausgeführten Operationen das repräsentierte Domänen- und Hintergrundwissen berücksichtigt werden. Denn durch die Integration des Domänen- und des Hintergrundwissens wird automatisch jede Anfrage als eine ähnlichkeitsbasierte Anfrage interpretiert. Der

Ähnlichkeitsbegriff definiert sich durch das repräsentierte Wissen und ist »deduktiv« implementiert.

So können z.B. in einem Fall bestimmte in der Anfrage fehlende Attributwerte aufgrund des Hintergrundwissens berechnet bzw. eingeschränkt werden. Ein einfaches Beispiel hierfür ist die Verwendung des Drehmomentgesetzes.

Über das Drehmomentgesetz kann aus zwei beliebigen Parameterwerten der Wert des dritten Parameters berechnet werden. Das wiederum kann dazu führen, daß das betroffene kinematische Glied zu einem spezielleren Konzept klassifiziert werden kann. Dies schränkt die Menge der Instanzen ein, die die Anforderungen der Anfrage erfüllen, es erfolgt ein ähnlichkeitsbasiertes Retrieval, wobei das Ähnlichkeitsmaß durch das repräsentierte Hintergrundwissen vorgegeben wird.

1.3. Zusammenfassung

Tabelle 1.1 fasst die Anforderungen an ein Second-Level-Support-System und die Unterschiede zu Help-Desk-Systemen noch einmal zusammen.

	Help-Desk-Systeme	Second-Level-Support-Systeme
Repräsentation	einfach (Attribut-Wert-Vektoren, Texte), fix	objektorientiert und erweiterbar
Retrieval	feste Ähnlichkeitsmaße	flexibel, durch Benutzer gesteuert
Retrievalsemantik	Numerische Ähnlichkeitswerte	klar, nachvollziehbar
Hintergrundwissen	nicht repräsentierbar	repräsentierbar, Berücksichtigung beim Retrieval

Tabelle 1.1. Methodische Anforderungen an Second-Level-Support- und Help-Desk-Systeme

Bevor im weiteren Verlauf der Arbeit gezeigt wird, wie die Anforderungen an ein Second-Level-Support-System mit Hilfe von Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen erfüllt werden können, soll im nächsten Kapitel das Leitbeispiel eingeführt werden, das im weiteren zur Illustration verwandt wird.

2 Leitbeispiel: Fahrradantriebe

In diesem Kapitel wird das Leitbeispiel eingeführt, das im Rest dieser Arbeit zur Verdeutlichung des in der Motivation skizzierten Ansatzes zum intelligenten Retrieval auf Basis von Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen verwendet wird.

Im Leitbeispiel soll ein Wartungstechniker bei der Wartung und Diagnose von Getrieben unterstützt werden. Ein Beispiel für diese Getriebeart sind Fahrradantriebe. Fahrradantriebe, z.B. der in Abbildung 2.1 dargestellte Antrieb, haben den Vorteil, daß sie allgemein bekannt sind und eine beträchtlicher Umfang an öffentlicher Dokumentation und Literatur zur Verfügung steht [1].

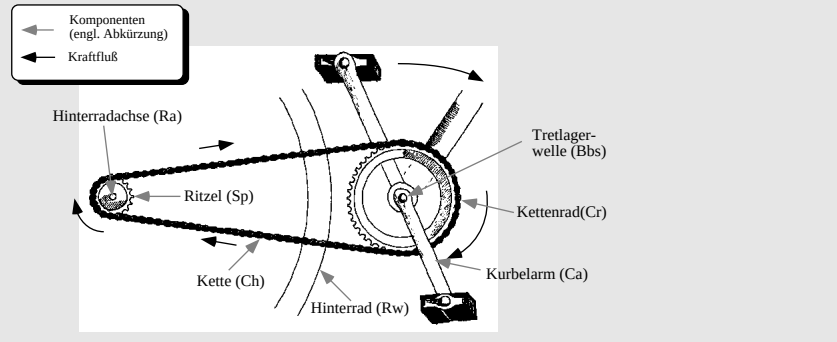


Abbildung 2.1.

Ein einfacher Fahrradantrieb

Im letzten Kapitel wurden die drei Bereiche

- Domänenwissen
- Hintergrundwissen

1. Fahrräder scheinen sich als Beispiele zur Verdeutlichung komplexer Zusammenhänge großer Beliebtheit zu erfreuen. So werden sie z.B. auch im Bereich der Ontologien und elektronischen technische Manuals verwendet.

- Verwaltungsinformationen

als die Bereiche identifiziert, deren adäquate objektorientierte Repräsentation für die Realisierung von Second-Level-Support-Systemen essentiell ist.

Bevor in den Abschnitten 2.2 – 2.4 die im letzten Kapitel noch allgemein gehaltenen Anforderungen an die objektorientierte Repräsentation für das Szenario Fahrradantriebe konkretisiert werden, soll zunächst kurz (in Abschnitt 2.1) auf objektorientierte Repräsentation und Entwurf im allgemeinen, und auf die im weiteren verwendete Notation der OMT[2] eingegangen werden.

Es wird klar, daß nicht alle Anforderungen mit einem gewöhnlichen objektorientierten Repräsentationsformalismus erfüllt werden können, sondern daß vielmehr ein sehr ausdrucksstarker Wissensrepräsentationsformalismus benötigt wird. Hierzu wird in Abschnitt 2.5 herausgearbeitet, welche Erweiterungen benötigt werden, und die graphische Notation der OMT geeignet ergänzt.

2.1. Objektorientierte Repräsentation und Entwurf

Die Verwendung einer objektorientierten Repräsentation zur adäquaten Modellierung des Anwendungsbereiches ist eine der Schlüsselanforderungen an ein System zur Unterstützung der zweiten Supportebene, die in der Einleitung formuliert wurde.

Neben den bereits in der Einleitung vorgestellten Gründen für die Notwendigkeit einer objektorientierten Repräsentation ist es insbesondere die leichte Verständlichkeit einer solchen Repräsentation für Endanwender, die sie in dem in dieser Arbeit diskutierten Szenario interessant macht.

Dieser entscheidende Vorteil objektorientierter Ansätze wurde schon 1981 von Robson in [Rob81] treffend formuliert[3]:

> ≫Many people who have no idea how a computer works find the idea of object-oriented systems quite natural. In contrast, many people who have experience with computers initially think there is something strange about object-oriented systems.≪

2. OMT – Object Modeling Technique.

3. Der zweite Teil dieser Aussage muß mittlerweile relativiert werden, da zumindest Einführungen in die objektorientierte Programmiersprachen im Curriculum praktisch jeder Hochschule zu finden sind. Obwohl es sich sicherlich auch trefflich darüber streiten läßt, ob ein Großteil der in objektorientierten Programmiersprachen (insbesondere in C++) geschriebenen Programme wirklich objektorientiert ist. Die Situation wird sich aber mit der weiteren Verbreitung von Java, einer auch vom ≫Mainstream≪ akzeptierten, und aus Sicht eines ≫Puristen≪ saubereren Programmiersprache, verbessern. Dies ändert jedoch nichts an der unveränderten Gültigkeit des ersten Teiles.

Nicht zuletzt deshalb gibt es im Bereich der KI[4] eine große Tradition der konsequenten Entwicklung und Verwendung objektorientierter Methoden. Viele der ersten, aber immer noch fortschrittlichsten objektorientierten Programmiersprachen, etwa CLOS – mit LOOPS und FLAVORS als Vorgängern – und Smalltalk, stammen aus diesem Bereich und finden dort breite Verwendung. Defizite existieren im Bereich der KI allerdings in der konsequenten Entwicklung einer einheitlichen, allgemeinverständlichen Notation für objektorientierte Repräsentationen.

Deshalb wird im folgenden zur Beschreibung des vorhandenen Wissens die Notation des OMT Objektmodelles verwendet.

2.1.1. Die OMT Notation

Zentrale Elemente der OMT sind, wie in jeder objektorientierten Repräsentation die Begriffe *Objekt*, *Klasse* und *Instanz*. Zur detaillierten Beschreibung von Objekten werden *Attribute* und *Assoziationen* benutzt. *Attribute* beschreiben Beziehungen zwischen Objekten und Datenwerten, d.h. Werten von Basisdatentypen wie Zahlen und Strings. *Assoziationen* beschreiben hingegen Beziehungen zwischen Objekten.

Aggregationen sind eine häufig vorkommende Form der Assoziation mit einer speziellen Semantik: die so genannte »Teil-Ganzes-« oder »ist-Teil-von« Relation. *Generalisierungen* beschreiben Relationen zwischen Klassen und sind ein mächtiger Abstraktionsmechanismus, um Gemeinsamkeiten von Klassen zu teilen und gleichzeitig ihre Unterschiede zu erhalten. *Instantiierungen* beschreiben Beziehungen zwischen Instanzen und Klassen. *Einschränkungen* sind funktionale Beziehungen zwischen Elementen des Objektmodelles.

Abbildung 2.2 gibt einen Überblick über die OMT Notation. Weitere Informationen z.B. die Definitionen der verwendeten Begriffe sind im Anhang A überblicksartig zusammengefaßt. Weitere Details können [RBP+91] entnommen werden.

2.2. Repräsentation der Gerätedaten

Bei den Gerätedaten kann zwischen anwendungsspezifischem und anwendungsunabhängigem aber domänenspezifischem Wissen unterschieden werden. Im Leitbeispiel ist das anwendungsspezifische Wissen Wissen über Fahrradantriebe während das anwendungsunabhängige Wissen allgemeines Getriebewissen ist.

2.2.1. Fahrradspezifisches Wissen

In einem typischen Fahrradreparaturbuch ([Pla86]) wird ein Fahrradantrieb wie folgt beschrieben:

4. KI – Künstlichen Intelligenz.

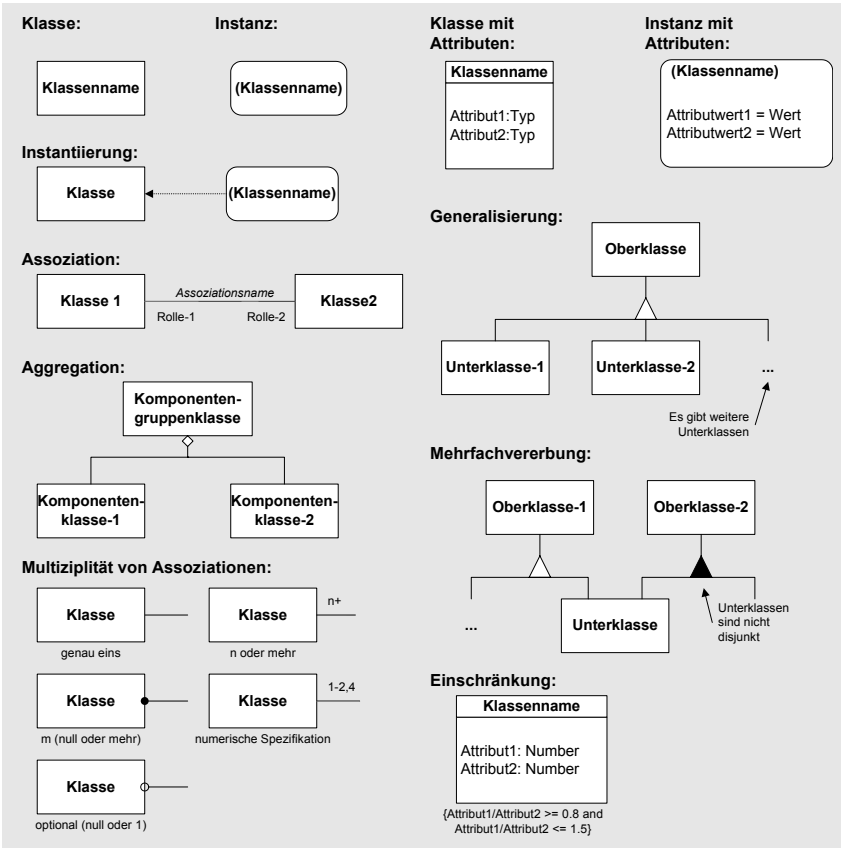


Abbildung 2.2.

Die Notation des OMT Objektmodells

≫Der Antrieb des Fahrrades besteht aus all jenen Teilen, mit denen die Kräfte des Fahrers auf das Hinterrad übertragen werden. Im Tretlagergehäuse ist das Tretlager angebracht, auf dessen Achse die Tretkurbeln befestigt sind. Die Pedale sind an den Enden der Tretkurbel eingeschraubt. Ebenfalls mit der rechten Tretkurbel verbunden ist das Kettenblatt (bei einem Rad mit Kettenschaltung mehrere Kettenblätter). Die Kette verbindet das Kettenblatt mit dem Ritzel des Hinterrades.≪

Ähnlich sind die Beschreibungen in anderen Fahrradreparaturbüchern [Her93]:

≫Der Antrieb besteht aus Tretlager, Kettenrad, Kurbelarmen, Pedalen, Kette und Zahnkranz. Diese Teile wandeln die Strampelbewegung der Beine in Vorwärtsbewegung um.≪

Diese Kurzbeschreibungen sind insoweit typisch für die Beschreibung eines Ge-

triebes [5], als daß sie sich im wesentlichen auf die Beschreibung der Dekompositions- und Verbindungsstruktur der Baugruppe beschränken. Darüber hinaus spielt noch die Typhierarchie der einzelnen Elemente der Baugruppe eine Rolle. Zur Funktionsweise der Baugruppen werden nur relativ allgemeine Bemerkungen gemacht, die insbesondere Kenntnisse der zugrundeliegenden physikalischen Gesetzmäßigkeiten voraussetzen.

Generalisierung und Aggregation

Gerätewissen ist also im wesentlichen Wissen über Gerätetypen und deren Teile. Die Beschreibung des Gerätewissens umfaßt die Beschreibung der Generalisierungshierarchie der unterschiedlichen Gerätetypen und eine Beschreibung der Aggregationsstruktur des Gerätes.

Eine direkte Umsetzung der obigen Beschreibungen [6] resultiert somit in einer Modellierung der Fahrradteiltypen und der Fahrradstruktur wie sie in den Abbildungen 2.3 und 2.4 wiedergegeben ist.

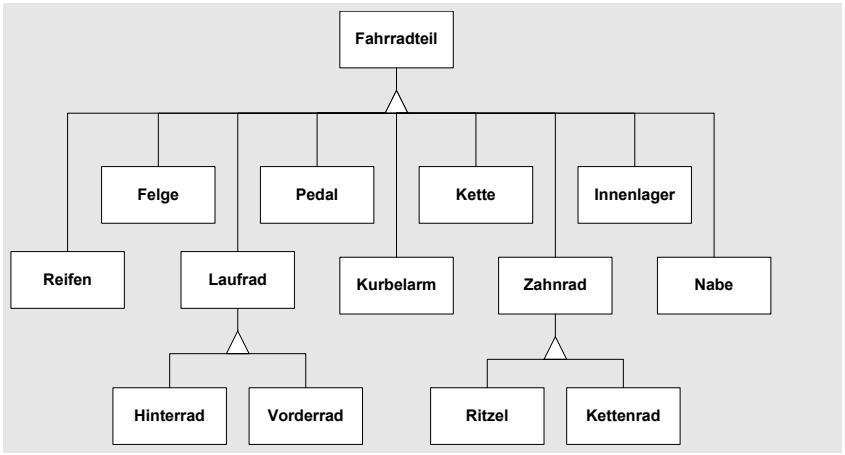


Abbildung 2.3.

Typen von Fahrradteilen

2.2.2. Allgemeines Getriebewissen

Neben der fahrradspezifischen und damit auch domänenabhängigen Modellierung existiert Wissen über Getriebe im allgemeinen.

Getriebe können aus einer beliebig großen Anzahl von Getriebegliedern unterschiedlichsten Typs (z.B. Rädern, Achsen, Zahnräder und Ketten) in nahezu beliebigen Konstellationen bestehen.

5. Aber auch anderer Baugruppen.

6. Und das Studium weiterer Fahrradliteratur.

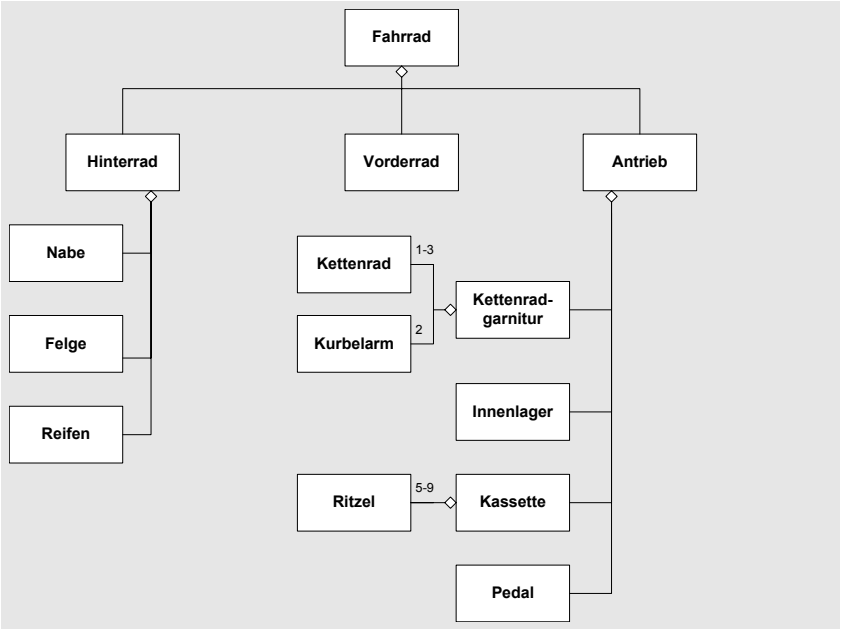


Abbildung 2.4.

Fahrradstruktur

Daher wird ein Repräsentationsformalismus benötigt, der es erlaubt, die Struktur von Getrieben flexibel auf Basis ihrer Komponenten zu beschreiben. Zur Beschreibung von Getrieben hat sich eine Methode auf der Basis von Gliedern und kinematischen Paaren [ISO94, Reu76] etabliert.

Glieder beschreiben die Generalisierungshierarchie der verschiedenen Getriebeteile während kinematische Paare zur Beschreibung der Getriebestruktur benutzt werden. Hierzu werden die Glieder eines Getriebes paarweise über kinematische Paare verbunden.

Glieder

Zur Beschreibung von Rad-Ketten-Getrieben und damit des Fahrradantriebs reicht es aus, Rotations- und Zugmittelglieder als spezielle Getriebeglieder einzuführen. Räder, Achsen und Zahnräder sind spezielle Rotationsglieder, Ketten spezielle Zugmittelglieder.

Die verschiedenen Gliedertypen stehen damit in einer Generalisierungs- bzw. Spezialisierungsbeziehung. Räder sind einerseits spezieller als Rotationsglieder, andererseits genereller als Zahnräder.

Als erste generelle Anforderung an einen Repräsentationsformalismus für den Bereich Service und Support läßt sich somit festhalten:

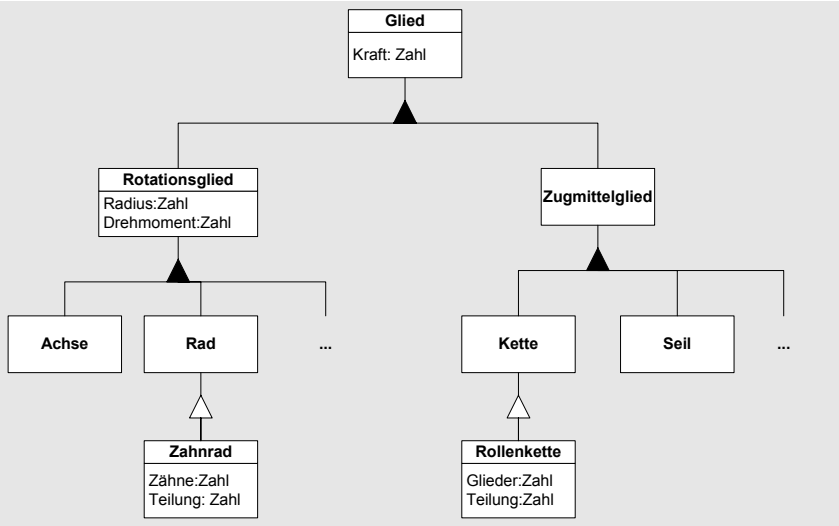


Abbildung 2.5.

Typen kinematischer Glieder

Anforderung 2.1 (Spezialisierungshierarchie)
Der Repräsentationsformalismus muß die Anordnung der zu repräsentierenden Begriffe in einer Spezialisierungshierarchie erlauben.

Die unterschiedlichen Gliederarten werden durch verschiedene Attributmengen beschrieben. Attribute von Gliedern sind im Normalfall physikalische Größen.

Notiert man diese konsequent jeweils in ihrer Basiseinheit [7], so können sie auch durch Zahlen beschrieben werden. Dies ist in praktisch allen Umsetzungen technischer Problemstellungen in Rechensystemen der Fall.

Somit läßt sich die folgende allgemeine Anforderung an den Repräsentationsformalismus ableiten:

Anforderung 2.2 (Numerische Attribute)
Im Rahmen des Repräsentationsformalismus muß es möglich sein, die unterschiedlichen Klassen durch numerische Attribute zu beschreiben.

In dem hier diskutierten Szenario bedeutet dies: Ein Attribut aller Gliedertypen ist die auf sie einwirkende Kraft. Rotationsglieder benötigen zusätzlich noch Attribute für den Radius und das übertragene Drehmoment. Zahnräder weisen neben einem Radius noch Attribute für die Zahnanzahl und evtl. noch die Teilung auf. Abbildung 2.5 gibt diese Zusammenhänge wieder.

7. Zu den Definitionen der Begriffe physikalische Größe, Dimension, Einheit etc. und diesbezügliche Normen siehe z.B. [FV93].

Kinematische Paare

Ein kinematisches Paar beschreibt die Verbindung zweier Glieder. Je nach Art der Verbindung und den sich ergebenden Freiheitsgraden der Bewegung können Typen von kinematischen Paaren unterschieden werden [Reu76, ISO94].

Zur Beschreibung planarer Rad-Ketten-Getriebe wie dem obigen Fahrradantrieb genügt die Beschreibung von Paaren, die entweder nur aus Rotationsgliedern bestehen oder Paaren zwischen einem Rotations- und einem Zugmittelglied.

Diese lassen sich je nach Typ der beteiligten Glieder weiter in Räderpaare, Zahnradpaare etc. unterscheiden (siehe Abb. 2.6).

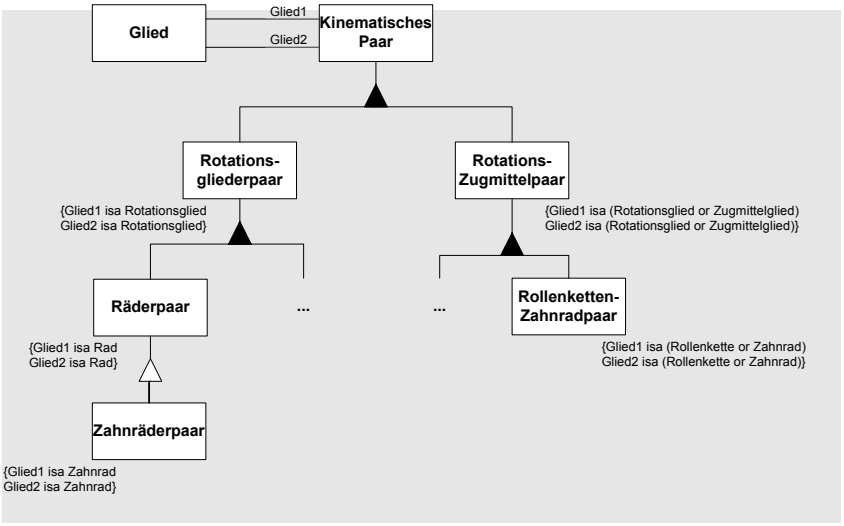


Abbildung 2.6.

Typen kinematischer Paare

Ein kinematisches Paar verweist also im wesentlichen auf die beiden kinematischen Glieder, die ihm angehören. Bei der Modellierung eines kinematischen Paars wurden zwei Assoziationen, jeweils eine für jedes Glied, verwandt. Dies erleichtert später die Identifikation der Glieder und entspricht der Modellierung in [ISO94] [8].

Aus dieser Beschreibung läßt sich die dritte grundlegende Anforderung an den Repräsentationsformalismus ableiten:

Anforderung 2.3 (Objekte als Attributwerte)
Im Rahmen des Repräsentationsformalismus müssen Objekte als Werte von Attributen zugelassen werden.

Eine Alternative zu dieser Modellierung ist die in Abb. 2.7) Dargestellte. Dort erfolgt die Modellierung über ein Attribut, das mehrere Werte aufnehmen kann.

8. An dieser Stelle könnten natürlich auch Aggregationen statt Assoziationen benutzt werden. Interpretiert man Aggregationen als physikalische Komponenten und Komponentengruppen, dann liegt eine Modellierung über Assoziationen nahe.

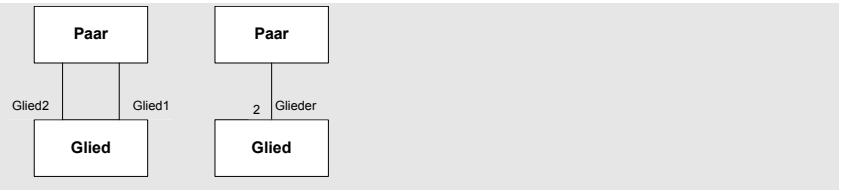


Abbildung 2.7.

Unterschiedliche Repräsentationen kinematischer Paare

Beispiel 2.1 (Ein einfacher Fahrradantrieb)
Auf Grundlage der bisherigen Repräsentation des Domänenwissens läßt sich ein Fahrradantrieb wie folgt beschreiben:

- Zunächst werden die Glieder des Fahrradantriebes identifiziert. Den Kern eines Fahrradantriebes bilden das Kettenrad CR, die Kette CH und das Ritzel SP. Zusätzlich werden zur vollständigen Beschreibung eines Hinterradantriebes noch die Glieder: Kurbelarm CR, Tretlagerwelle BBS, Hinterradachse RA und Hinterrad RW benötigt [9].
- Im nächsten Schritt werden die Typen der einzelnen Glieder identifiziert. Bei Kettenrad und Ritzel handelt es sich um Zahnräder, bei der Kette um eine Kette, bei Kurbelarm, Tretlagerwelle, Hinterradachse und Hinterrad um Rotationsglieder.
- Abschließend werden die einzelnen Glieder über kinematische Paare miteinander in Beziehung gesetzt. Dazu folgt man am zweckmäßigsten dem Weg der Kraftübertragung: der kinematischen Kette des Getriebes. Über den Kurbelarm wird die Kraft auf die Tretlagerwelle gebracht. Von dort gelangt sie über das Kettenrad, die Kette, das Ritzel und die Hinterradachse auf das Hinterrad. Damit werden zur Modellierung des Hinterradantriebes 6 kinematische Paare KP_1 bis KP_6 benötigt, in denen die jeweils 2 Glieder eines kinematischen Paares miteinander verbunden werden.

Abbildung 2.8 zeigt den Aufbau dieses einfachen Fahrradantriebes.

Am Beispiel wird deutlich, daß das OMT Modell über keinen Mechanismus zur Benennung der unterschiedlichen Instanzen verfügt.

2.3. Repräsentation des Hintergrundwissens

Zusätzlich zum Gerätewissen existiert im diskutierten Gegenstandsbereich Hintergrundwissen. Dieses Wissen, z.B. über das Funktionieren und Nichtfunktionieren der Geräte, muß im Rahmen eines Unterstützungssystems für Experten zum Einsatz kommen. Etwa zur Berechnung fehlender Parameterwerte. Eine Vorbedingung dazu ist die deklarative Beschreibung dieses Wissens.

9. Die Namen der einzelnen Glieder leiten sich aus den englischen Bezeichnungen für Kettenrad, Kette, Ritzel, Tretlagerwelle, Hinterradachse und Hinterrad ab. Diese lauten: Chainring, Chain, Sprocket, Bottom-Bracket-Spindle, Rear-Axle und Rear-Wheel.

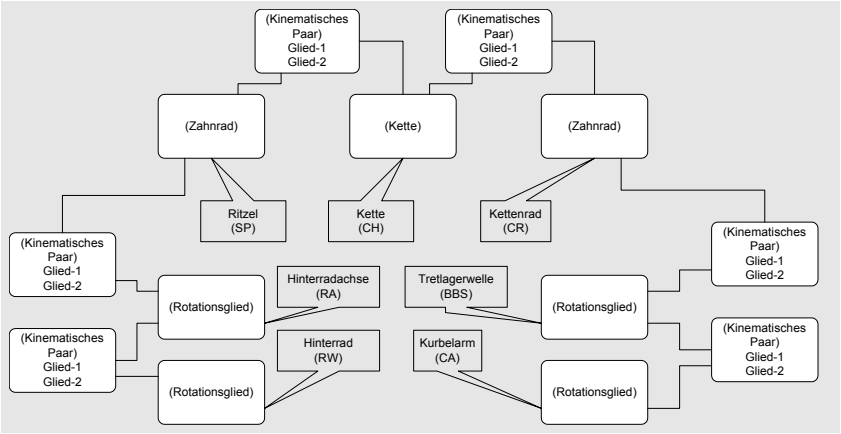


Abbildung 2.8.

Modellierung eines Fahrradantriebes

2.3.1. Physikalische Gesetze und Randbedingungen

Das Hintergrundwissen besteht im wesentlichen aus allgemeingültigen physikalischen Gesetzen und aus Randbedingungen an Attribute. Im Rahmen der OMT Notation wird diese Art von Hintergrundwissen mit Hilfe von Einschränkungen modelliert.

Die OMT stellt dabei nur einen leeren Container zur Beschreibung von Einschränkungen bereit. Es ist den Anwendungen vorbehalten, die genaue Syntax und Semantik der zulässigen Einschränkungen herauszuarbeiten. Im Beispiel existiert Hintergrundwissen in Form der folgenden physikalischen Gesetze und Randbedingungen:

> ≫Das Drehmoment eines Rotationsgliedes ergibt sich aus dem Produkt der angreifenden Kraft und des Radius.
> Die an einem Getriebe angreifende Kraft ist positiv. Der Radius eines Rotationsgliedes ist strikt positiv.≪

Diese Anforderungen lassen sich durch Gleichungen bzw. Ungleichungen über einem (Randbedingungen) bzw. mehreren (physikalische Gesetze) numerischen Parametern beschreiben:

$$M_{Rotglied} = F_{Rotglied} \cdot r_{Rotglied} \tag{2.1}$$

$$F_{Glied} \geq 0 \tag{2.2}$$

$$r_{Rotglied} > 0 \tag{2.3}$$

Wichtig für die Repräsentation ist die Tatsache, daß Gleichung (2.1) eine nichtlineare Gleichung ist. Da keiner der Werte festgelegt wurde, ist die rechte Seite ein quadratisches Polynom.

Aus diesen Anforderungen des Leitbeispiels lassen sich die folgende generellen Anforderungen an den Repräsentationsformalismus ableiten:

Anforderung 2.4 (Einschränkungen über numerischen Attributen)
Im Rahmen des Repräsentationsformalismus müssen Einschränkungen für die zulässigen Werte von Attributen definiert werden können. Die Einschränkungen können auch in Form von Abhängigkeiten zwischen mehreren Attributen formuliert sein.

Abbildung 2.9 gibt die entsprechend erweiterte Notation für Getriebeglieder wieder.

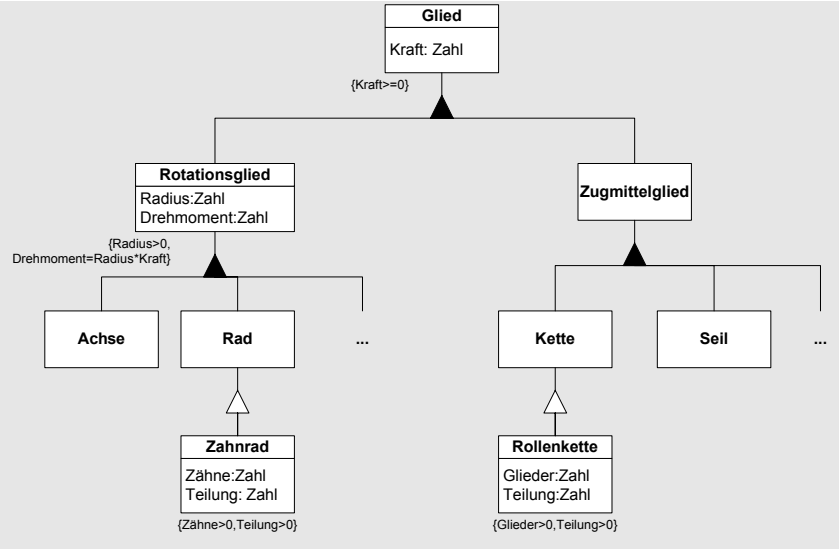


Abbildung 2.9.

Typen kinematischer Glieder

2.3.2. Verhaltensmodelle

Zusätzlich zu allgemeingültigen physikalischen Gesetzen und Randbedingungen kennt man im Bereich der Diagnose technischer Geräte häufig auch das potentielle Verhalten von Komponenten. Zumindest das Normalverhalten von Komponenten ist oft bekannt, da diese ja gerade so entworfen wurden, um dieses Verhalten zu zeigen.

Aber auch einige Fehlverhalten sind bekannt. So ist das Durchrutschen eines Rotationspaares ein bekannter Fehler im Bereich der Rädergetriebe. Ein anderer typischer Fehler ist ein gebrochenes Rotationspaar, bei dem gar keine Übertragung des Drehmomentes mehr stattfindet.

Es ist aber nicht nur möglich, Normal- und verschiedene Fehlverhalten zu identifizieren, das Verhalten kann sogar in einigen Fällen genau beschrieben werden.

Normalverhalten

So läßt sich das Normalverhalten von Rotationsgliederpaaren und Rotations-Zugmittelgliederpaaren informell wie folgt beschreiben:

> ≫In einem Rotationsglied wird das Drehmoment übertragen.
> In einem Rotations-Zugmittel-Paar wird die Kraft in Zugrichtung übertragen, d.h. an zwei von der Kette angetriebenen Zahnrädern greift die gleiche Kraft an.≪

Zur formalen Repräsentation werden wieder Ungleichungen zwischen Polynomen benötigt:

$$M_{Rotglied_1} = M_{Rotglied_2} \tag{2.4}$$
$$F_{Zugglied} = F_{Rotglied} \tag{2.5}$$

Damit wird das Normalverhalten des Fahrradantriebes aus Abbildung 2.1 durch das Ungleichungssystem (2.6) beschrieben [10].

$$\begin{array}{lll} M_{Ca} = F_{Ca} \cdot r_{Ca} & F_{Sp} = F_{Ch} & \\ M_{Ca} = M_{Bbs} & M_{Sp} = F_{Sp} \cdot r_{Sp} & F_{Ca}, F_{Cr}, F_{Bbs}, F_{Ch}, \\ M_{Bbs} = F_{Bbs} \cdot r_{Bbs} & M_{Ra} = M_{Sp} & F_{Sp}, F_{Ra}, F_{Rw} \geq 0 \\ M_{Cr} = M_{Bbs} & M_{Ra} = F_{Ra} \cdot r_{Ra} & r_{Ca}, r_{Cr}, r_{Bbs}, \\ M_{Cr} = F_{Cr} \cdot r_{Cr} & M_{Rw} = M_{Ra} & r_{Sp}, r_{Ra}, r_{Rw} > 0 \\ F_{Ch} = F_{Cr} & M_{Rw} = F_{Rw} \cdot r_{Rw} & \end{array} \tag{2.6}$$

Damit ist das Gleichungssystem (2.6) ein nichtlineares System. Es wird erst dann linear, wenn konkrete Werte für die Radien resp. die Momente der Rotationsglieder festgelegt werden.

Fehlverhalten

Zumindest ein Teil der Fehlverhalten kinematischer Paare ist bekannt. So lassen sich durchrutschende und gebrochene Rotationspaare wie folgt beschreiben:

> ≫In einem durchrutschenden Rotationspaar sind die Drehmomente der beteiligten Glieder unterschiedlich groß, aber beide verschieden von 0.
> In einem gebrochenem Rotationspaar ist das Drehmoment eines Gliedes ungleich Null, das des anderen gleich Null.≪

10. Die Indizes der Gleichungen referieren auf die in Abb. 2.1 verwendeten Abkürzungen der entsprechenden Komponenten.

Formalisiert man diese Beschreibungen, so wird eine Repräsentationssprache benötigt, die logische Operatoren zur Verknüpfung der einzelnen Ungleichungen zuläßt. Dann läßt sich ein durchrutschendes Rotationspaar durch die folgenden logischen Formeln beschreiben:

$$\begin{aligned}&((M_{Rotglied_1} > M_{Rotglied_2}) \vee (M_{Rotglied_1} > M_{Rotglied_2}))\wedge \\ &(M_{Rotglied_1} > 0) \wedge (M_{Rotglied_2} > 0)\end{aligned} \tag{2.7}$$

Zur Beschreibung eines gebrochenen Paares bieten sich die beiden folgenden alternativen Varianten an:

$$\begin{aligned}&((M_{Rotglied_1} = 0 \wedge M_{Rotglied_2} > 0)) \vee (M_{Rotglied_1} > 0 \wedge M_{Rotglied_2} = 0)) \text{ bzw.} \\ &M_{Rotglied_1} * M_{Rotglied_2} = 0 \wedge (M_{Rotglied_1} > 0 \vee M_{Rotglied_2} > 0)\end{aligned} \tag{2.8}$$

Dies resultiert in der folgenden Erweiterung von Anforderung 2.4:

Anforderung 2.5 (Logische Formeln über nichtlinearen Ungleichungen)
In der Repräsentationssprache für Einschränkungen muß es möglich sein, logische Formeln über (nicht)linearen Ungleichungen zwischen numerischen Attributen zu beschreiben.

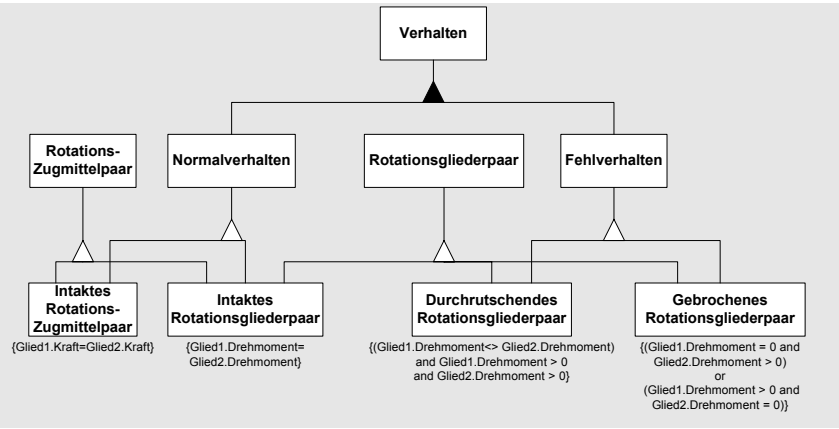


Abbildung 2.10.
Typen des Verhaltens kinematischer Paare

Abbildung 2.10 zeigt diese Zusammenhänge. Weiterhin verdeutlicht sie, daß zur adäquaten Modellierung des Normal- und Fehlverhaltens multiple Vererbung benötigt wird: Ein durchrutschendes Paar von Rotationsgliedern ist sowohl ein Paar von Rotationsgliedern als auch eine Baugruppe die Fehlverhalten zeigt. Generell ist es zur Beschreibung des Verhaltens von kinematischen Paaren also notwendig, festzulegen, um welche Art von kinematischem Paar und um welche Art

des Verhaltens es sich handelt. Dadurch ergibt sich folgende Anforderung an den Repräsentationsformalismus:

Anforderung 2.6 (Multiple Vererbung)
Der Repräsentationsformalismus muß multiple Vererbung unterstützen.

2.4. Repräsentation der Verwaltungsinformationen

Obwohl der Schwerpunkt im Bereich des Second-Level Supports und damit auch der Schwerpunkt der Betrachtungen eindeutig auf den Gerätedaten bzw. dem Hintergrundwissen liegt, wird auch von einem Kundendiensttechniker der Zugriff auf bestimmte Verwaltungsinformationen benötigt.

Kundendaten sind hier genauso wichtig wie Informationen über andere Servicetechniker. Aber auch Geräteinformationen, die nicht direkt der Beschreibung der Funktion der Geräte dienen, sind wichtig. Etwa Teilenummern in Katalogen sowie Angaben über Hersteller und Distributoren der Teile usw. Abbildung 2.11 gibt einen Vorschlag für eine Modellierung der minimal benötigten Verwaltungsinformationen wieder.

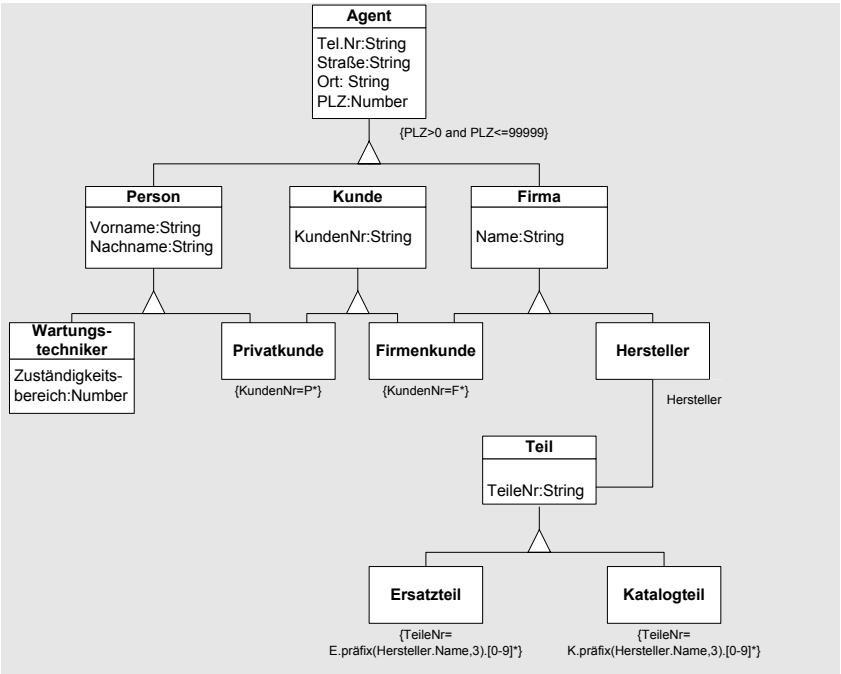


Abbildung 2.11.
Im Second-Level benötigte Verwaltungsinformationen

Zur Repräsentation des Geräte- und Hintergrundwissens wurden bislang nur numerische Attribute benötigt. Zur Beschreibung der Attribute der Verwaltungsin-

formationen, z.B. zur Ablage von Adressen, Name etc., werden zusätzlich Zeichenketten benötigt. Numerische Attribute sind in diesem Zusammenhang von eher untergeordneter Bedeutung, etwa zur Repräsentation von Postleitzahlen. Damit ergibt sich die folgende Anforderung an den Repräsentationsformalismus.

Anforderung 2.7 (Strings als Attributwerte)
Der Repräsentationsformalismus muß neben numerischen Attributen und Objektreferenzen auch Strings als Werte von Attributen erlauben.

Im Bereich der Verwaltungsinformationen existiert im allgemeinen weniger Hintergrundwissen in Form von Einschränkungen. Es ist allerdings vorhanden, insbesondere bei Teile- und Kundennummern. Diese sind oft – wie Identifikationsnummern im allgemeinen – nach einem bestimmten Schema aufgebaut, das eine Menge an Informationen kodiert.

Dieses Schema stellt eine Einschränkung an den zulässigen Wertebereich des entsprechenden Attributes dar. Im Beispiel soll vereinfachend angenommen werden, daß bei den Kunden zwischen Privat- und Firmenkunden unterschieden werden soll. Die Kundennummern der Privatkunden beginnen mit einem »P«, während die Kundennummern von Firmenkunden mit »F« starten. Tatsächlich sind Identifikationsnummern viel komplexer aufgebaut und stellen oft eine Kodierung der Attributwerte einer ganzen Reihe von Attributen des Objektes dar [11].

Aus Sicht der Wissensrepräsentation ist diese nichtdeklarative Repräsentation von Wissen und Informationen nicht unbedingt anzustreben. In der Praxis ist sie jedoch so weit verbreitet, daß es möglich sein muß, diese impliziten Informationen zu beschreiben, auszuwerten und zur Definition von sinntragenden Einheiten einsetzen zu können.

Wie bei numerischen Attributen muß es möglich sein, für string-wertige Attribute die Syntax und Semantik der zulässigen Einschränkungen zu definieren. Die oben genannten Identifikationsnummern können im allgemeinen durch reguläre Ausdrücke beschrieben werden. Daraus ergibt sich die folgende Anforderung an den Repräsentationsformalismus:

Anforderung 2.8 (Reguläre Ausdrücke als Einschränkungen über Strings)
Zur Beschreibung der Einschränkungen über Strings muß der Repräsentationsformalismus reguläre Ausdrücke zulassen.

11. Gerade im industriellen Bereich sind solche Identifikationsnummern weit verbreitet, ja oft sogar Teil internationaler Normen und Standards. Ein prominentes Beispiel dafür sind die Codes, die für Schrauben verwendet werden. So dürfte der Code M6 jedermann bekannt sein, während die Werte, für die er steht, etwa die Steigung, weitgehend unbekannt sind.

2.5. Notwendige Erweiterungen: Von objektorientierten Repräsentationen zur Wissensrepräsentation

Das OMT Objektmodell erlaubt es, einen großen Teil des vorhandenen Wissens adäquat zu repräsentieren. Die Notwendigkeit zur Spezifikation der Syntax und Semantik für Einschränkungen über numerischen Attributen und Strings zeigt allerdings ebenso wie fehlende Möglichkeit zur Benennung von Instanzen, daß das Objektmodell der OMT nicht alle Anforderungen erfüllen kann.

Um die Beschreibung des vorhandenen Wissens zu ermöglichen, muß die Notation der OMT erweitert werden. Mindestens die folgenden Erweiterungen sind notwendig:

- Benennung von Instanzen,
- Flexiblere Beschreibungen von Attributwerten bei Instanzen,
- Flexiblere Beschreibung von Instanzen,
- Flexiblere Spezifikation von Instantiierungen,
- Spezialisierung über Wertebereichseinschränkungen von Assoziationen,
- Unterscheidung zwischen unvollständigen und vollständigen Klassenbeschreibungen,
- Möglichkeit zur Beschreibung von Generalisierungen.

2.5.1. Benennung von Instanzen

Im Beispiel 2.1 wurde deutlich, daß innerhalb der OMT Notation keine Mittel zur Benennung von Instanzen vorgesehen sind. Dies erschwert die übersichtliche Repräsentation des Antriebes enorm, eine nichtgraphische Repräsentation wird aufgrund von möglichen Zyklen in der graphischen Notation ohne die Einführung eines Referenzmechanismus unmöglich. Daraus ergibt sich die folgende Anforderung an den Repräsentationsformalismus:

Anforderung 2.9
Der Repräsentationsformalismus muß die Benennung von Instanzen erlauben.

Abbildung 2.12 zeigt die um die Möglichkeit der Vergabe von Instanznamen erweiterte Notation.

Beispiel 2.2 (Einfacher Fahrradantrieb mit Instanznamen)
Abbildung 2.13 zeigt die Modellierung des Fahrradantriebes aus Beispiel 2.1 mit der Vergabe von Instanznamen.

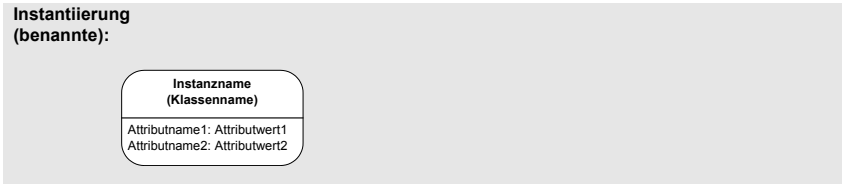


Abbildung 2.12.

Erweiterte Notation zur Vergabe von Instanznamen

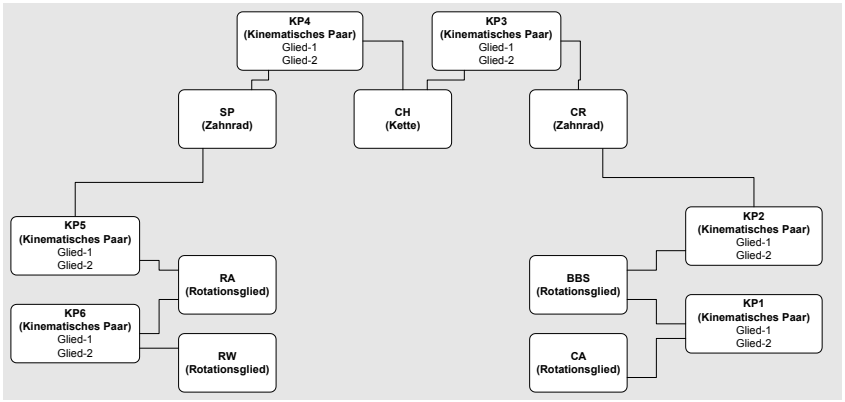


Abbildung 2.13.

Modellierung eines Fahrradantriebes mit Instanznamen

2.5.2. Flexiblere Attributwertbeschreibungen für Instanzen

Im OMT Objektmodell sind nur einfache Werte des jeweiligen Basisdatentyps als Attributwerte der Instanzen vorgesehen. Im Bereich des Service und Supports sind viele Attributwerte aber ungenau oder unsicher.

So führen z.B. Unsicherheiten und Ungenauigkeiten bei der Meßwerterfassung genauso wie Toleranzen, Schätzungen und qualitative Bewertungen von Attributen zu Intervallen von zulässigen Attributwerten für ein Attribut. Diese Intervalle müssen z.B. bei der Diagnose verarbeitet werden können. Für Simulationen ist es sogar unerläßlich, daß Intervalle der Eingangsparameter verarbeitet werden können.

Beispiel 2.3 (Intervalle als zulässige Attributwerte)
So kann bei der Erfassung der Attributwerte der unterschiedlichen Attribute eines Fahrradantriebes die von einem Fahrer auf das Pedal gebrachte Kraft im Normalfall nicht exakt ermittelt werden. Daher ist es notwendig die Kraft als Intervall angeben zu können.

In einem anderen Szenario wurde die durchschnittliche Kraft, die eine bestimmte Fahrergruppe (etwa untrainierte bzw. leicht trainierte Fahrer) aufwendet, empirisch ermittelt und kann nur in Form eines Intervalles beschrieben werden.

Ein möglicher Ansatz zur Integration von Intervallen ist es, einen zusätzlichen Basisdatentyp »Intervall« für Attributwerte zuzulassen. Erfahrungen aus AMS[12]

zeigen allerdings das auch ein zusätzlicher Basisdatentyp Intervall im allgemeinen nicht ausreicht. So werden z.B. in bestimmten Situationen Disjunktionen von Werten bzw.. Disjunktionen von Intervallen benötigt. Ein Beispiel ist die folgende Modellierung von Mehrfachzahnkränzen.

Beispiel 2.4 (Fahrradantriebe mit Mehrfachzahnkränzen)
Bislang wurden nur Fahrradantriebe mit einem Kettenrad und einem Ritzel betrachtet. Üblich sind mittlerweile jedoch Kettenschaltungen, die bis zu drei Kettenräder und Mehrfachzahnkränze mit bis zu 9 Ritzeln enthalten.

Ein Mehrfachzahnkranz läßt sich am besten als ein Zahnkranz beschreiben, der als Wert des Attributes Zahnanzahl eine Disjunktion von Werten, den Zahnanzahlen der jeweiligen Einzelzahnkränze aufweist [13]

Auch diese Modellierung ließe sich theoretisch durch einen separaten Basisdatentyp ≫Menge von Zahlen≪ realisieren. Der Ansatz, über zusätzliche Basisdatentypen weitere Attributwerte zu realisieren, ist aus einer Reihe von Gründen nicht optimal [14]:

Erstens ist nicht klar, welche Menge von Basisdatentypen über den Zahlen prinzipiell benötigt wird. So könnte an anderer Stelle der Basisdatentyp ≫Menge von Intervallen≪ als zugrundeliegender Basisdatentyp benötigt werden usw..

Zweitens müssen, läßt man unterschiedliche Basisdatentypen für numerische Attribute zu, auch Disjunktionen und Kombinationen all dieser Basisdatentypen zugelassen werden bzw. behandelt werden können. Dies hat seinen Grund in der Modellierung des Hintergrundwissens. Über dieses werden Attributwerte bzw. Einschänkungen über diesen auf andere Attribute propagiert, wie das folgende Beispiel zeigt:

Beispiel 2.5 (Kombination numerischer Basisdatentypen)
Das Kettenrad CR besteht aus 2 Zahnkränzen mit den Radien 100 und 150 mm. Daran greift eine Kraft F von 200 - 250 N an. Dies läßt sich über die Verwendung des Basisdatentyps Intervall für die Kraft und des Basisdatentyps Disjunktion für den Radius bei der Modellierung des Kettenrades erreichen.

Aufgrund des Drehmomentgesetzes ergibt sich daraus die folgende Einschränkung an den Wert des Attributes Drehmoment M: $20 \leq M \leq 25 \vee 30 \leq M \leq 37,5$. Diese Einschränkung läßt sich nur noch als Disjunktion von Intervallen beschreiben.

Durch eine andere Wahl der Parameter (etwa einer Kraft F von 200 - 300 N) können die beiden Intervalle allerdings wieder zu einem Intervall zusammenfallen.

12. AMS – Anwendungsmanagementsystem.

13. Eine alternative Modellierung wäre die Beschreibung eines jeden Zahnkranzes (bis zu 9 des Mehrfachzahnkranzes). Dadurch würde das Modell jedoch unnötig aufgebläht. Zudem würde die Modellierung der Disjunktion dadurch nicht überflüssig.

14. Dieser Ansatz wird z.B. in Fresko und daraus abgeleiteten Systemen, also auch AMS, verwendet.

Je nach Art und Weise der Ausgangsdatentypen können die unterschiedlichsten abgeleiteten Datentypen entstehen. Je größer die Anzahl der Basisdatentypen ist und um so komplexer das Hintergrundwissen ist, um so schwieriger wird es die Menge der abgeleiteten Datentypen zu ermitteln und zu verwalten.

Daher ist der folgende Ansatz zur Repräsentation numerischer Attributwerte sinnvoller. Ihm liegt die Feststellung zugrunde, daß alle oben aufgeführten komplexen Basisdatentypen Teilmengen des Basisdatentyps Zahlen sind. Diese Teilmengen können zudem durch Einschränkungen, wie sie bei der Definition von Klassen zugelassen wurden, beschrieben werden.

Als zulässige Attributwerte werden daher nicht einzelne Werte aus dem Basisdatentyp, sondern all die Teilmengen, die über zulässige Einschränkungen definierbar sind, zugelassen.

Da als zulässige Einschränkungen über Zahlen logische Formeln über nichtlinearen Ungleichungen erlaubt sind, sind alle oben aufgeführten erweiterten Basisdatentypen, von exakten Werten bis zu den Mengen von Intervallen, gültige Attributwerte. Dies gilt bereits bei der Verwendung von linearen Ungleichungen.

Abbildung 2.14 zeigt die Notation für erweiterte Attributwerte.

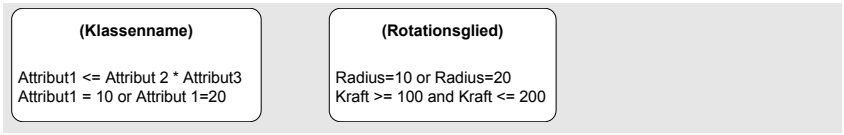


Abbildung 2.14.

Erweiterte Notation zur Beschreibung von Attributwerten von Instanzen

2.5.3. Flexiblere Beschreibung von Instanzen

Im Bereich des Service und Supports werden Informationen zumeist inkrementell verfügbar. Dies bedeutet insbesondere, daß:

- Instanzen dynamisch Klassen zugeordnet werden,
- Attributwerte von Unterklassen bzw. Werte neuer Attribute erfaßt werden.

Innerhalb der meisten objektorientierten Repräsentationen ist dies nicht möglich.

Dynamische Zuordnung der Klassenzugehörigkeit

In objektorientierten Repräsentationen wird die Zugehörigkeit einer Instanz zu einer bestimmten Klasse zumeist zum Erzeugungszeitpunkt festgelegt und kann nicht einfach verändert werden. Zumindest ist es in solchen Repräsentationen eher die Ausnahme als die Regel, daß sich die Klassenzugehörigkeit einer Instanz permanent verändert.

Da sich im Service und Support viele Fakten erst nach einer Messung, einem Test, oder nach dem Öffnen einer Baugruppe erschließen ist die leichte Veränderung der Klassenzugehörigkeit einer Instanz eine essentielle Anforderung an den Repräsentationsformalismus. Beispiel 2.6 soll diesen Zusammenhang verdeutlichen:

Beispiel 2.6
Bei einem Hollandfahrrad ist der Antrieb komplett gekapselt und von außen nicht einsehbar. Erst nach dem Abbau der Verkleidung kann man sehen, ob es sich um einen Kettenantrieb, bestehend aus zwei Zahnrädern und einer Kette, oder um einen Riemenantrieb (bestehend aus einem Zahnriemen und zwei Zahnscheiben) handelt.

Es ist zum Zeitpunkt t_1 also erst bekannt, daß es sich bei dem Antrieb um einen Antrieb aus zwei Rotationsgliedern (g_1, g_2) und einem Zugmittelglied (z_1) handelt. Erst nach dem Öffnen der Verkleidung kann man erkennen, daß es ein Kettenantrieb ist. Dann ist eine Spezialisierung der Instantiierungsbeziehung von Rotationsglied zu Zahnrad möglich.

Neben dieser expliziten Veränderung der Instantiierungsbeziehungen, die zumindest noch von einigen objektorientierten Systemen angeboten wird, gibt es allerdings implizite Veränderungen, die von keiner objektorientierten Repräsentation angeboten, ja aufgrund der fehlenden formalen Semantik der Klassendefinitionen nicht angeboten werden kann.

Zusätzliche Attribute und Attributwerte von Unterklassen

Wird z.B. bekannt, daß ein Rotationsglied eine Zähneanzahl von 52 hat, dann muß es sich um ein Zahnrad handeln, da nur die Subklasse Zahnrad das Attribut ≫Anzahl Zähne≪ definiert. In objektorientierten Repräsentationen ist es jedoch nicht möglich, ein Rotationsglied mit einer Zahnanzahl von 52 zu definieren, da die Zahnanzahl erst als Attribut eines Zahnrades, d.h. einer Unterklasse definiert wird.

Zulässig ist in objektorientierten Beschreibungen aber nur die Erfassung von Werten für Attribute, die in der aktuellen Klasse und deren Oberklassen definiert sind.

Anforderung 2.10
Der Repräsentationsformalismus muß es erlauben, über die Definition der aktuellen Klasse hinaus zusätzliches Wissen über eine Instanz zu erfassen.

Objektorientierte Repräsentationen können solche impliziten Veränderungen der Instantiierungsbeziehungen nicht entdecken.

Die Bereitstellung einer formalen Semantik, die die Definition automatischer Inferenzen zur Bestimmung der semantisch korrekten Instantiierungsbeziehungen erlaubt, ist, wie in der Einleitung dargestellt, ein wesentlichen Ziel dieser Arbeit.

2.5.4. Flexiblere Spezifikation von Instantiierungen

In der OMT und allgemein in objektorientierten Repräsentationen, ist eine Instanz fest *einer* Klasse zugeordnet. Selbst bei multipler Vererbung ist es nicht möglich, eine Instantiierungsbeziehung zwischen einer Instanz und mehreren Klassen aufzubauen. Um also ausdrücken zu können, daß a eine Instanz der Klassen A und B ist, muß eine Unterklasse C von A und B existieren. Dann kann man eine Instantiierungsbeziehung zwischen a und C aufbauen und damit ausdrücken, daß a eine Instanz von A und B ist.

Dies ist kein gangbarer Weg. Denn erstens ist die Existenz einer gemeinsamen Unterklasse nicht immer gesichert. Im Gegenteil, im allgemeinen wird sie nicht vorhanden sein. Aber auch wenn gemeinsame Unterklassen existieren, so ist es nicht korrekt, diese zu verwenden, da sie nicht der intendierten Semantik entsprechen. Das folgende Beispiel verdeutlicht diese Tatsache:

Beispiel 2.7 (Flexible Spezifikation von Instantiierungen)
Am Anfang der Diagnose ist nur bekannt daß es sich bei einem betrachteten kinematischen Paar um ein Rotationsgliederpaar handelt, das nicht sein Normalverhalten zeigt. Es soll also zum Ausdruck gebracht werden, daß es sich sowohl um ein Rotationsgliederpaar als auch um ein Fehlverhalten handelt.

Gemeinsame Unterklassen dieser beiden Klassen sind sowohl ≫Durchrutschendes Rotationsgliederpaar≪ als auch ≫Gebrochenes Rotationsgliederpaar≪. Es kann nun nicht einfach eine der beiden Unterklassen ausgewählt werden, da noch nicht klar ist, um welche Art von Fehlverhalten es sich handelt. Herauszufinden um welches Fehlverhalten es sich handelt, ist ja gerade das Ziel der Diagnose.

Eine proaktive Erzeugung von Klassen für alle möglichen Klassenkombinationen macht wenig Sinn. Auch die dynamische Erzeugung entsprechender Kombinationsklassen ist nur ein Workaround und keine eigentliche Lösung des Problems.

Gleichzeitig zeigt das Beispiel die Bedeutung der Beschreibung multipler Instantiierungsbeziehungen für das angestrebte flexible Informationssystem. Diese Möglichkeit wird benötigt, um partielles Wissen über Instanzen zum Ausdruck bringen zu können, und um dieses Wissen sukzessive zu vervollständigen. Dies ist zum Beispiel bei der Diagnose der Fall, wenn sukzessiv neue Meßwerte erhoben werden. Aber auch bei der Modifikation von Anfragen wird diese Funktionalität benötigt.

Im erweiterten Objektmodell sind daher multiple Instantiierungsbeziehungen für einzelne Instanzen erlaubt. Diese werden durch multiple Instantiierungspfeile und durch die Angabe einer Liste von Klassennamen im Kopf der Instanzenbeschreibung notiert.

Die soeben geschilderte Notwendigkeit zur Beschreibung multipler Instantiierungsbeziehungen ist im Kern nur ein Spezialfall eines allgemeineren Flexibilisierungsbedarfes. Eine mehrfache Instantiierung ist logisch eine Instantiierungsbeziehung zwischen einer Instanz und einer implizit durch die Konjunktion einer

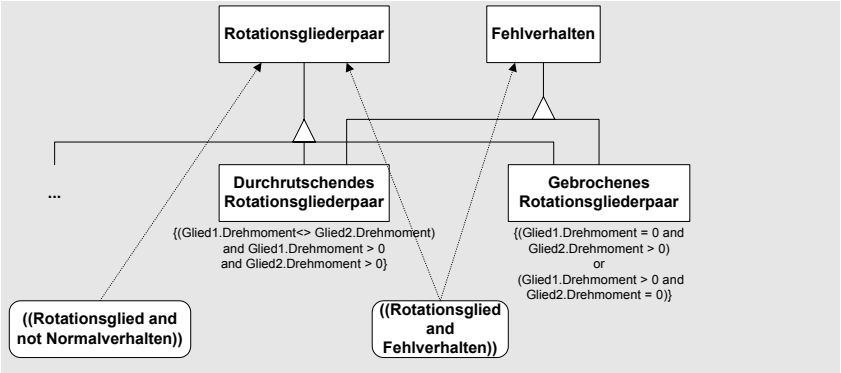


Abbildung 2.15.

Erweiterte Notation zur Beschreibung von Instantiierungsbeziehungen

Menge von Klassen beschriebenen virtuellen Klasse.

Dieses Prinzip läßt sich verallgemeinern, indem in der Instantiierungsbeziehung von Konjunktionen zu beliebigen logischen Operationen bzw. zu beliebigen logischen Formeln übergegangen wird. Diese Art erweiterter Instanzbeschreibungen ist insbesondere im Bereich der What-If-Analyse interessant, etwa um die Auswirkungen alternativ einsetzbarer Komponenten zu beschreiben. Dazu wird eine Disjunktion der Form ´»entweder eine Komponente der Klasse A oder eine Komponente der Klasse B« benötigt.

Auch Negationen sind in der What-If Analyse und in der Simulation interessant. Etwa um zu beschreiben, daß im Prinzip ein beliebiges Teil der Klasse A aber nicht der Unterklasse B zum Einsatz kommen soll, z.B. in einer Aussage wie »ein beliebiges Zugmittelglied aber kein Seil« oder »eine Rollenkette aber nicht vom Hersteller X« etc.. Beschreibungen dieser Art sind insgesamt im Bereich vom Service und Support weit verbreitet.

2.5.5. Spezialisierung über Wertebereichseinschränkungen von Assoziationen

Nicht nur zur Beschreibung von Instantiierungsbeziehungen ist eine Erweiterung des Objektmodelles notwendig. Gleiches gilt für die Beschreibung der Spezialisierungs- bzw. Generalisierungsbeziehungen zwischen Klassen.

Die Erweiterung wird benötigt, um bekannte Zusammenhänge zwischen Klassen beschreiben zu können. Erst dadurch wird es möglich, die Semantik der Beziehung zu formalisieren. Eine formale Definition ist aber die Grundvoraussetzung für die Realisierung von Inferenzmechanismen zur automatischen Berechnung der Spezialisierungsbeziehung und damit zu der angestrebten automatischen semantischen Indexierung.

Im OMT Objektmodell wird die Spezialisierungs-/Generalisierungsbeziehung umgangssprachlich über den Begriff der Verfeinerung definiert. In einer etwas

genaueren, aber immer noch informellen Charakterisierung werden zwei Arten von Verfeinerungen, *Erweiterung* und *Begrenzung* unterschieden ([RBP+91], Seite 77):

> »Eine Instanz einer Klasse ist eine Instanz aller Vorfahren dieser Klasse. Dies ist ein Teil der Definition für Generalisierung. Deshalb müssen alle Merkmale einer Vorfahrenklasse für die Instanzen einer Unterklasse gelten. Eine Nachkommenklasse kann ein Attribut einer Vorfahrenklasse nicht weglassen oder unterdrücken, da es sich sonst nicht um eine echte Vorfahrensinstanz handeln würde.
> ...
> Eine Unterklasse kann neue Merkmale hinzufügen. Dies heißt *Erweiterung*. Eine Unterklasse kann darüber hinaus Vorfahrenattribute auch einschränken. Dies heißt *Begrenzung*, weil die Werte, die Instanzen annehmen können, eingeschränkt werden.«

Im Bereich der Wissensrepräsentation ist die Spezialisierung über die Einschränkung der zulässigen Wertebereiche für Attribute und Assoziationen [15] weit verbreitet. Eine erste Erweiterung zur exakteren Beschreibung von Klassen ist daher die Spezialisierung des Wertebereiches von Assoziationen.

Ähnlich wie bei Instantiierungen sind auch hier nicht nur einzelne Klassen zur Spezifikation von Wertebereichen zugelassen, sondern logische Formeln über Mengen von Klassen. So ist etwa ein Räderpaar ein Rotationsgliederpaar, dessen beide Glieder Räder sind. Ein Rotations-Zugmittelgliederpaar ist ein kinematisches Paar, das aus einem Rotationsglied und einem Zugmittelglied besteht. Diese Beziehung kann nicht über die Wertebereiche einzelner Assoziationen definiert werden, vielmehr müssen die Wertebereiche unterschiedlicher Assoziationen miteinander in Beziehung gesetzt werden.

Während die Einschränkung des Wertebereichs von Attributen mit Hilfe der Einschränkungen des OMT Modelles zu beschreiben ist, gibt es für die Restriktion von Wertebereichen keinen eindeutigen Mechanismus. Im Folgenden wird die Beschreibung über Einschränkungen, wie in Abbildung 2.16 dargestellt, verwendet.

2.5.6. Vollständige und unvollständige Klassenbeschreibungen

Abbildung 2.16 zeigt außerdem den Bedarf an einer weiteren Erweiterung des Objektmodelles auf. Er betrifft die Vollständigkeit der Beschreibung der Spezialisierungsbeziehung. So ist ein Paar dann und nur dann ein Zahnradpaar, wenn beide Glieder Zahnräder sind; die Beschreibung ist vollständig.

15. Und damit auch Aggregationen.

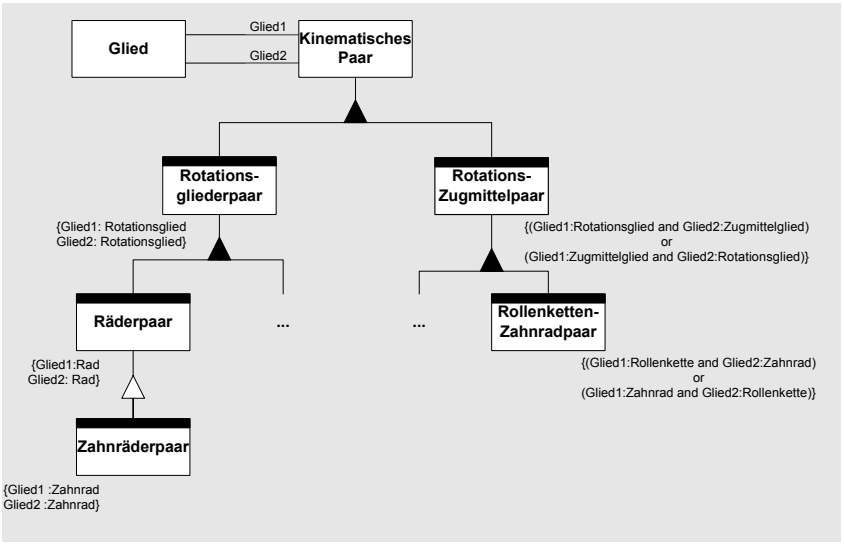


Abbildung 2.16.

Erweiterte Beschreibung kinematischer Paare

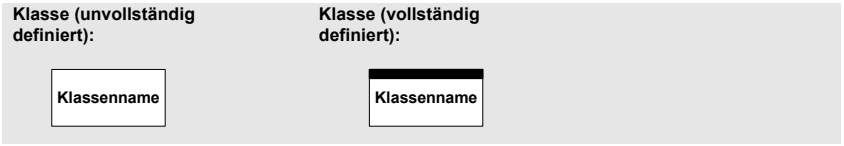
Logisch gesehen können bei einem Zahnradpaar also nicht nur notwendige Bedingungen sondern auch hinreichende Bedingungen zur Charakterisierung angegeben werden. Das Repräsentationsschema sollte diese Unterscheidung ermöglichen. Sie ist für den Entwurf eines flexiblen Informationssystemes essentiell.

So können für vollständige Beschreibungen automatische Verfahren zur Erkennung der Instantiierungsbeziehung entworfen werden. Diese Verfahren erkennen, daß eine Instanz eines Paares, dessen Glieder beide Zahnräder sind, eine Instanz eines Zahnradpaares ist. Damit kann die Instanz automatisch dieser Klasse zugeordnet werden. Die Instanz kann also semantisch korrekt indexiert werden.

Vollständige Beschreibungen werden in der erweiterten Notation durch einen schwarzen Balken am oberen Rand des Klassenrechtecks gekennzeichnet (siehe Abb.2.17).

Abbildung 2.17.

Notation für unvollständige und vollständige Klassenbeschreibungen



2.5.7. Möglichkeit zur Beschreibung von Generalisierungen

In objektorientierten Repräsentationen ist es nur möglich, Spezialisierungen von Klassen über Verfeinerungen zu definieren. Damit ist es möglich, Unterklassen als

Konjunktion einer Menge von Oberklassen und einer Menge lokaler Einschränkungen (zusätzliche Attribute und Wertebereichseinschränkungen von Attributen und Relationen) zu beschreiben.

Eine Möglichkeit zur Definition von Oberklassen als Disjunktion einer Menge von Unterklassen fehlt jedoch. Die Möglichkeit zur Beschreibung von Generalisierungen ist aber sowohl für die Definition von Klassenbeschreibungen als auch für die Spezifikation von Anfragen wichtig. So läßt sich etwa das Normalverhalten von kinematischen Paaren vollständig nur über eine Disjunktion der Normalverhalten der verschiedenen Paartypen definieren.

Neben der Disjunktion zur Spezifikation von Generalisierungen ist auch die Negation zur Beschreibung von Spezialisierungs- und Generalisierungbeziehungen nützlich. So ist jedes Verhalten, das kein Normalverhalten ist, ein Fehlverhalten. Abbildung 2.18 zeigt diese Zusammenhänge nochmals auf.

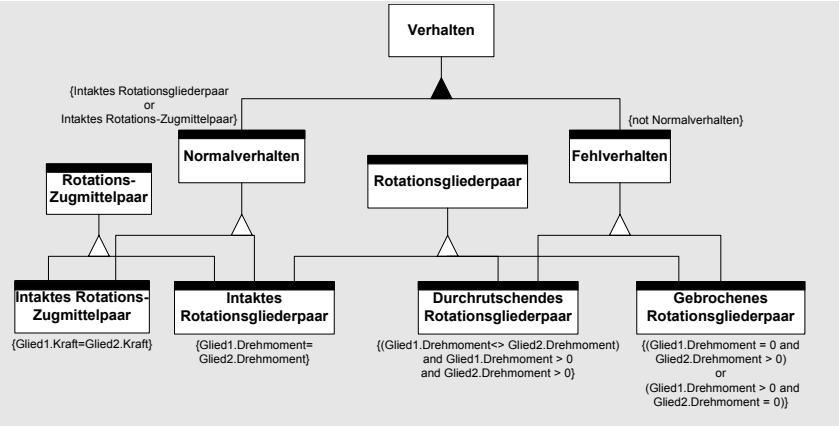


Abbildung 2.18.

Erweiterte Beschreibung der Verhaltensmodelle

2.6. Zusammenfassung

Am Beispiel der Rad-Ketten-Getriebe wird deutlich, daß objektorientierte Repräsentationen, wie z.B. OMT, nicht ausreichen, um alle zur Unterstützung der zweiten und dritten Supportebene benötigten Informationen über Klassen, Instanzen und Beziehungen zwischen diesen zu beschreiben. Dazu ist eine erweiterte, flexiblere Repräsentation, die Mechanismen aus dem Bereich der Wissensrepräsentation einbezieht, notwendig.

Im nächsten Kapitel sollen daher Beschreibungslogiken als Wissensrepräsentationsformalismus vorgestellt und auf ihre Eignung für die Repräsentation des Leitbeispiels und damit ihre generelle Eignung als Basis für ein Second-Level-Support-System untersucht werden.

3 Beschreibungslogiken

Motivation

Mitarbeiter der zweiten Support-Ebene benötigen zu ihrer Unterstützung ein System, das es ihnen ermöglicht, flexibel auf unterschiedlichste Informationen zuzugreifen. Beim Zugriff auf die Informationen muß das System vorhandenes Domänen- und Hintergrundwissen nutzen, um einen intelligenten Zugang zu diesen Informationen zu erlauben.

Dazu wird im Kern des Systems ein Repräsentationsformalismus benötigt, der es erlaubt, das vorhandene Wissen adäquat darzustellen. Eine Erweiterung bestehender objektorientierter Repräsentationsformalismen, die dies erlaubt, wurde am Ende des letzten Kapitels vorgestellt.

Ohne eine formale Semantik für diese Erweiterung ist es allerdings nicht möglich, Inferenzmechanismen zu definieren, die das repräsentierte Wissen nutzen, etwa um automatisch die Daten- und Wissensbankinhalte zu indizieren, oder um die vom Benutzer gestellten Anfragen zu beantworten.

In diesem Kapitel wird gezeigt, daß Beschreibungslogiken ein Wissensrepräsentationsformalismus sind, der die formale Semantik und die benötigten Inferenzen zur Verfügung stellt, und es möglich ist, ein Second-Level-Support-System auf Basis von Beschreibungslogiken zu realisieren.

Allerdings sind dazu Erweiterungen der ≫klassischen≪ Beschreibungslogiken um ausdrucksstarke konkrete Gegenstandsbereiche notwendig. Erst diese erlauben es, das Hintergrundwissen in Form von Randbedingungen, Physikalischen Gesetzen sowie Normal-und Fehlverhalten zu beschreiben.

Ziel dieses Kapitels ist es also *nicht*, Beschreibungslogiken in ihrer ganzen Breite und Tiefe darzustellen. Vielmehr liegt der Schwerpunkt ganz klar auf der Untersuchung ihrer Eignung für den Gegenstand dieser Arbeit: der Unterstützung von Experten im Second-Level-Support.

Für Experten im Bereich der Beschreibungslogiken ist im vorliegenden Kapitel daher im wesentlichen der Abschnitt 3.4 interessant. Leser, die weitere Informationen zum Thema Beschreibungslogiken erhalten möchten seien auf die entsprechen-

de Literatur, insbesondere auf »The Description Logic Handbook« [BCM+03] verwiesen.

Aufbau des Kapitels

Motivation, Entwicklungsgeschichte

In Abschnitt 3.1 werden zunächst die Ursprünge und Grundideen von Beschreibungslogiken abrißartig dargestellt. Es wird klar, daß es nicht *eine* Beschreibungslogik, sondern daß es vielmehr ein breites Spektrum unterschiedlich ausdrucksstarker Beschreibungslogiken gibt, die sich in Hinsicht auf die Entscheidbarkeit bzw. die Komplexität der unterstützten Inferenzen sehr stark unterscheiden.

Syntax und Semantik von Beschreibungslogiken (am Beispiel von $\mathcal{ALCF}$)

In Abschnitt 3.2 wird die Syntax und die Semantik von Beschreibungslogiken anhand der Sprache $\mathcal{ALCF}$ vorgestellt, einer Beschreibungssprache, die von ihrer Ausdrucksstärke in etwa in der Mitte des Spektrums der bislang vorgeschlagenen Beschreibungssprachen liegt.

Inferenzen und Algorithmen

Anschliessend werden in Abschnitt 3.3 eine Reihe von Basisinferenzen von Beschreibungslogiken, wie die Konzept- und die Instanzklassifikation, die Instantiierung und das Retrieval, vorgestellt und auf verschiedenartige Ansätze für Algorithmen zur Lösung dieser Inferenzprobleme diskutiert.

Korrespondenz zu objekt-orientierten Beschreibungen

Nachdem Beschreibungslogiken in den vorhergehenden Abschnitten vorgestellt wurden, gilt es in Abschnitt 3.4, zu prüfen, inwieweit sie die im letzten Kapitel postulierten Anforderungen an eine erweiterte objekt-orientierte Repräsentation erfüllen können.

Die $\mathcal{AL}$-Sprachfamilie

Unverzichtbar für die Realisierung eines Second-Level-Support-Systems ist die Bereitstellung von Attributen über Basisdatentypen. Diese fehlen allerdings in $\mathcal{ALCF}$. Bevor diese Fragestellung detailliert behandelt wird, soll vorher in Abschnitt 3.5 die $\mathcal{AL}$-Sprachfamilie vorgestellt werden, die eine Methodik zur Einordnung der verschiedenen, bislang untersuchten beschreibungslogischen Sprachen bereitstellt.

Bereitstellung von Basisdatentypen durch konkrete Gegenstandsbereiche

In Abschnitt 3.6 werden die von Baader und Hanschke entwickelten Kriterien für eine Integration von Basisdatentypen in Beschreibungslogiken, die sogenannten zulässigen konkreten Gegenstandsbereiche vorgestellt und die Syntax und Semantik der erweiterten Beschreibungssprache $\mathcal{ALCF}(\mathcal{D})$ präsentiert.

Mit $\mathcal{ALCF}(\mathcal{D})$ steht damit eine Sprache zur Verfügung, die geeignete zulässige konkrete Gegenstandsbereiche vorausgesetzt, die die Anforderungen aus Kapitel 1 und 2 an eine erweiterte objekt-orientierte Repräsentation mit einer wohldefinierten Semantik und flexiblen Inferenzen erfüllt.

3.1. Beschreibungslogiken als Wissensrepräsentationsformalismus

Wissensrepräsentationsformalismen zeichnen sich gegenüber anderen Repräsentationsformalismen durch folgende Charakteristika und Ziele aus (siehe z.B. [Baa96]):

- Deklarative Semantik

  Die Bedeutung der Einträge in die Wissensbasis ist unabhängig von den darauf operierenden Programmen definiert.

- Strukturierte Darstellung

  Die Zusammenhänge zwischen verschiedenen Einträgen in der Wissensbasis sind wichtig und werden expliziert.

- »Intelligente« Zugriffsmechanismen

  - Der Zugriff auf nur implizit gespeichertes Wissen ist möglich.
  - Die zum Zugriff benutzten Schlußfolgerungsverfahren sind in ihrem Verhalten nur von der Semantik und nicht von der syntaktischen Form der Einträge und der Anfrage abhängig.

Diese Charakteristika decken sich weitestgehend mit den Kernanforderungen an ein flexibles Informationssystem zur Unterstützung der zweiten Support-Ebene. Repräsentationsformalismen, die diese Charakteristika und Ziele umsetzen, sind also auch für die in dieser Arbeit verfolgten Zwecke optimal geeignet. Beschreibungslogiken sind ein Wissensrepräsentationsformalismus, der den formulierten Zielen sehr nahe kommt.

3.1.1. Ausgangspunkte und Vorläufer

Beschreibungslogiken in ihrer aktuellen Form sind das Ergebnis eines langen Entwicklungsprozesses. Insbesondere bauen sie auf den Ergebnissen der folgenden drei Bereiche der Wissensrepräsentation auf:

- Prädikatenlogik,
- Semantische Netze,
- Frames.

Prädikatenlogik Semantische Netze und Frames können somit als Vorläufer von Beschreibungslogiken betrachtet werden. Um ihre Einflüsse auf Beschreibungslogiken zu verstehen, soll kurz auf die unterschiedlichen Formalismen eingegangen werden.

Prädikatenlogik

Einer der ersten verwendeten Formalismen zur Beschreibung von Wissen war die Prädikatenlogik erster Stufe. Die Tarski-Semantik erfüllt zwar die Anforderung nach einer deklarativen Semantik für »sicheres« Wissen, kann aber unsicheres, unvollständiges und subjektives Wissen nicht adäquat behandeln. Dafür müssten andere Logiken eingesetzt werden, etwa probabilistische Logiken, nichtmonotone Logiken und Modallogiken.

Wichtiger für die verfolgte Aufgabenstellung ist allerdings die Tatsache, daß die Repräsentation des Wissens durch nicht strukturierte Formelmengen erfolgt und relevante Schlußfolgerungsprobleme wie das Gültigkeits- und das Erfüllbarkeitsproblem unentscheidbar sind. Diese Schwächen der Prädikatenlogik führten dazu, daß vielfach nach anderen Wissensrepräsentationsformalismen gesucht wurde.

Semantische Netze

Semantische Netze [Qui66] sind ein Ergebnis dieser Bemühungen. Ursprünglich waren sie ein aus dem Bereich der Psychologie stammender Formalismus zur strukturierten Beschreibung von Begriffen und Beziehungen auf Basis eines gerichteten, gefärbten Graphen. Dieser Formalismus wurde schnell in der KI aufgegriffen und in lauffähige Programme umgesetzt. Die Gründe hierfür waren im wesentlichen die folgenden [LOK91]:

- »Semantische Netze lassen sich graphisch darstellen und eignen sich daher in dieser Darstellung sehr gut als Kommunikationsmedium.
- Sie entsprechen den intuitiven und durch psychologische Untersuchungen mit einer gewissen Relevanz versehenen Annahmen über die Nähe bzw. den Abstand zwischen Begriffen durch verschieden lange Kantenpfade zwischen den jeweiligen Begriffen.
- Sie können relativ einfach durch Abbildung in innerhalb der Informatik bekannte Strukturen wie Records und Pointer realisiert werden.«

Semantische Netze können auch als Visualisierungen prädikatenlogischer Formeln angesehen werden. Knoten stehen für Elemente eines Modells, Kanten für binäre Relationen zwischen den Elementen. Ohne weitere Beschreibungsmittel lassen sich allerdings nur Konjunktionen und die existentielle Quantifizierung ausdrücken.

Semantische Netze bestehen einerseits aus speziellen nur in wenigen Netzen benötigten Relationen, andererseits gibt es eine Reihe von allgemeinen Relationen, die immer wieder in semantischen Netzen vorkommen. Die wichtigsten allgemeinen Relationen sind wie bei objektorientierten Repräsentationen die Generalisierung, die Individualisierung (Instantiierung) und die Aggregierung.

Die unterschiedlichen Implementierungen semantischen Netze unterschieden sich in der Menge der allgemeinen Relationen und in den Mechanismen, mit denen sie in den semantischen Netzen für Inferenzleistungen eingesetzt wurden, also der prozeduralen Semantik der Relationen. Dies führte zu einer heftigen Diskussion zur Frage der Semantik in semantischen Netzen auf. Insbesondere Woods [Woo75] kritisierte die mangelnde semantische Fundierung der verschiedenen Vorschläge. Brachman fasst die Entwicklung wie folgt zusammen [Bra77]:

≫Each implementation of a semantic net has adapted the basic node-plus-link idea to its own immediate purpose, creating virtually as many stylized 'formalisms' as implementations. Procedures to operate on the structure – which are really what make it meaningful – have usually been idiosyncratic.≪

Das Fehlen einer formalen Semantik hatte zur Folge, daß selbst identische Wissensbasen in verschiedenen Systemen zu sehr unterschiedlichen Ergebnissen führten.

Frames

Aus der Unzufriedenheit mit den unstrukturierten logikbasierten Repräsentationen entstanden in den 70er Jahren nach einem Vorschlag von Minsky [Min75] Wissensrepräsentationsverfahren auf Basis von Frames. strebte keinen logikbasierten Ansatz an, er hielt diesen für ungeeignet:

≫such attempts will continue to fail, because of the character of logistic in general . . . ≪.

Minsky suchte nach einem im Sinne der Kognitionswissenschaften adäquaten Weg, einen Ausschnitt der Umwelt zu beschreiben. Wenig später stellte Hayes [Hay77] den Zusammenhang zur Logik wieder her. Er bemerkt:

≫most of 'frames' is just a new syntax for predicate logic≪.

Allerdings muß bemerkt werden, daß der Frame-Begriff in seiner vollen Allgemeinheit im wesentlichen äquivalent zur Prädikatenlogik höherer Stufe ist, und daher keine vernünftige Theorie existiert.

Frames stellen neben dem Klassenkonzept von Simula und den Actorsprachen einen der Vorläufer objektorientierter Repräsentationen dar[1], wie die folgende kurze Zusammenfassung zeigt:

- Ein Frame besteht aus einer Reihe von Slots.

1. So handelt es sich bei der objektorientierten Repräsentationssprache von AMS tatsächlich um eine framebasierte Sprache.

- Slots nehmen als Werte Elemente eines bestimmten Wertebereichs auf. Die Slots werden mit entsprechenden Werten »gefüllt«.
- Teilweise gefüllte Frames sind Instanzen des Frames.
- Wertebereiche können sowohl Basisdatentypen als auch andere Frames sein.
- Eine Reihe spezieller Slots ist vordefiniert. Der wichtigste ist der isa-Slot der zur Definition von Spezialisierungsbeziehungen dient.
- Für diese speziellen Slots sind Mechanismen definiert, die die prozedurale Semantik der Slots definieren.
- Im Falle des isa-Slots ist dies die Vererbung von Slots und deren Werte.

Der Zusammenhang mit semantischen Netzen ist klar: Slots entsprechen den Kanten eines semantischen Netzes, Frames den Knoten. Ein Slot selbst ist im allgemeinen wiederum strukturiert Seine Komponenten heißen Facetten. Unterschiedliche Framesysteme unterscheiden sich in der Menge und den Aufgaben der bereitgestellten Facetten. Bereitgestellt werden z.B. Defaultwerte und Slotbedingungen. Sie entsprechen den Einschränkungen der OMT. Weiterhin gibt es eine Reihe von Facetten, die prozedurale Zusätze, sogenannte Dämonen, aufnehmen. Die beiden wichtigsten sind »if-needed« und »if-added« Dämonen.

Scripts

Scripts und Memory Organization Packets basieren auf Frames

In dem Kontext des fallbasierten Schließens ist weiterhin interessant, daß Frames auch als Vorlage für die von Roger Schank [Sch77] entwickelten *Scripts* dienten, die wiederum von ihm zu der Basis des von ihm und seinen Schülern entwickelten Ansatz zum fallbasierten Schließen [RS89], den sogenannten *Memory Organization Packages* weiterentwickelt wurden.

3.1.2. Beschreibungslogiken – der prinzipielle Ansatz

Aufbauend auf den Ausgangspunkten: Prädikatenlogik Semantischen Netze und Frames entstand der Wissensrepräsentationsformalismus der Beschreibungslogiken, der die Vorteile der objektorientierten Repräsentation der Frames mit der formalen Semantik der Prädikatenlogik vereint und damit auch die Semantik semantischer Netze festlegt.

Beschreibungslogiken entstanden aus dem Vorschlag Brachmans für strukturierte Vererbungsnetze. Brachmans Ziel war es, einen Formalismus zur Definition der relevanten Begriffe eines Anwendungsbereiches zu entwickeln, der:

- eine strukturierte Darstellung wie in Frames erlaubt,

- die Mehrdeutigkeiten semantischer Netze vermeidet,
- die Definition neuer Begriffe auf Basis einer kleinen Anzahl epistemologisch adäquater Konstruktoren erlaubt,
- die verschiedenen Begriffe miteinander in Beziehung setzt, und
- diese sogenannten Konzeptbeschreibungen im Rahmen von Inferenzprozessen einsetzt.

Entstanden ist daraus ein Repräsentationsformalismus, der sich – nach einer langen Entwicklungsgeschichte – durch folgende Charakteristika auszeichnet (aus [LOK91]):

- Objektorientierte und deklarative Wissensrepräsentation,
- Repräsentation impliziten Wissens möglich,
- Formale Semantik für alle Sprachkonstrukte,
- Mächtige Inferenzprozesse.

Die strukturierte Beschreibung des Wissens in Form von Konzeptbeschreibungen ist objektorientiert und unterscheidet sich nur wenig von Strukturbeschreibungen wie sie etwa im Rahmen von Frames erstellt werden können. Konzepte entsprechen den Klassen objektorientierter Repräsentationen. Instanzen beschreibungslogischer Repräsentationen den Instanzen in objektorientierten Repräsentationen.

Inferenzen erlauben den ≫intelligenten≪ Zugriff

Im Vergleich zu semantischen Netzen und Frames ist es bei Beschreibungslogiken allerdings möglich, eine modelltheoretische Semantik für *alle* zur Begriffsdefinition verwendeten Konstruktoren anzugeben. Diese Semantik erlaubt es, eine Reihe von Inferenzen zu definieren, die auf der Menge der beschreibbaren Begriffe operieren. Die Inferenzen erlauben den ≫intelligenten≪, von der Syntax der Fragestellung unabhängigen, Zugriff auf die in der Wissensbasis enthaltenen Informationen. Besonders wichtig und bekannt sind die folgenden Basisinferenzen:

- Automatische Klassifikation von Instanzbeschreibungen,
- Automatische Klassifikation von Klassenbeschreibungen.

Die automatische Klassifizierung der Instanzen ermöglicht eine automatische und inkrementelle Indexierung. Alle Instanzen werden unter den jeweils speziellsten Konzepten zu denen sie aufgrund der modelltheoretischen Semantik gehören indiziert. Damit ist eine optimale Indexierung aller Instanzen in Bezug auf das repräsentierte Wissen sichergestellt. Auf Basis dieser Indexierung lassen sich dann eine Reihe von Verfahren zum intelligenten ähnlichkeitsbasierten Retrieval definieren.

Die automatische Klassifikation nicht nur der Instanzbeschreibungen sondern auch der Konzeptbeschreibungen stellt sicher, daß die Spezialisierungshierarchie der Konzeptbeschreibungen korrekt bzgl. der modelltheoretischen Semantik ist. Diese automatische Berechnung der Subsumptionshierarchie stellt nicht nur die Basis für die Indexierung der Instanzen in Bezug auf die Subsumptionshierarchie dar, sie kann ebenfalls für das intelligente ähnlichkeitsbasierte Retrieval genutzt werden.

Auf ähnliche Weise können die Basisinferenzen genutzt werden, um all die Instanzen zu finden, die eine bestimmte Konzeptbeschreibung erfüllen. Konzeptbeschreibungen können somit dazu genutzt werden, Anfragen an die Wissensbasis zu formulieren, stellen also eine flexible Anfragesprache dar.

3.1.3. Zusammenfassung

Beschreibungslogiken sind Fragmente der Prädikatenlogik erster Stufe

Aus logischer Sicht stellen Beschreibungslogiken Fragmente der Prädikatenlogik erster Stufe dar. Die definierten Begriffe sind einstellige Relationen, die Beziehungen zwischen Begriffen zweistellige Relationen. Die von der Beschreibungslogik bereitgestellten Konstruktoren stellen somit bestimmte Muster für logische Formeln zur Verfügung.

Unterschiedliche Beschreibungslogiken unterscheiden sich durch die Menge der zur Verfügung gestellten Konstruktoren

Damit bestimmen Menge und Art der in der Beschreibungslogik verfügbaren Konstruktoren die Teilklasse der Prädikatenlogik, die durch die Beschreibungen darstellbar ist. Unterschiedliche Konstruktorenmengen resultieren in unterschiedlichen Beschreibungslogiken. Damit kann nicht von *der* Beschreibungslogik an sich gesprochen werden. Der Begriff Beschreibungslogik charakterisiert vielmehr ein Rahmenwerk, das durch die Wahl der Konstruktoren (und andere Faktoren) parametrisiert werden kann.

Innerhalb dieses Rahmenwerkes wurden eine Vielzahl von Beschreibungslogiken entwickelt und untersucht, so daß sich das Feld der Beschreibungslogiken von außen als sehr divers darstellt.

Interessant sind die Konstruktorenmengen mit algorithmisch handhabbaren Inferenzproblemen

Aus der gewählten Konstruktorenmenge, und damit aus der entsprechenden Teilklasse der Prädikatenlogik, ergibt sich die Komplexität der Inferenzprobleme. Im Kontext des Rahmenwerkes beschreibungslogischer Systeme ist man an Mengen von Konstruktoren interessiert, die in Teilklassen der Prädikatenlogik mit algorithmisch handhabbaren Inferenzproblemen resultieren.

Basis für moderne Algorithmen ist das Tableaux-Kalkül

Zur Realisierung der Inferenzen werden zwei prinzipielle Ansätze verfolgt, die sich in Vollständigkeit und Komplexität unterscheiden. Einerseits werden strukturorientierte Algorithmen verwandt, um korrekte, aber unvollständige Inferenzen über relativ ausdrucksstarken Sprachen zu realisieren, andererseits werden Techniken des Tableaux-Kalküls eingesetzt, um korrekte und vollständige Inferenzen zu realisieren.

Während in der Anfangszeit strukturorientierte Ansätze im Vordergrund standen haben sich in den letzten 10 Jahren die Tableaux-basierten Ansätze durchge-

setzt.

Die zur Verfügung gestellten Inferenzen zur automatischen Klassifikation von Konzepten und Instanzen, erlauben den im Anwendungsbereich benötigten flexiblen Zugriff auf Informationen.

Die bereitgestellten Inferenzen haben ein weites Anwendungsspektrum

Ziel ist es nun, in dem Spektrum von Beschreibungssprachen und beschreibungslogischen Systemen, die Sprache bzw. das System zu finden, das die beste Basis für die Realisierung eines Second-Level-Support-Systems bildet, es soll keine vollständig neue Beschreibungslogik entwickelt werden.

Es gilt eine für den Second-Level-Support geeignete Beschreibungslogik zu finden

3.2. Syntax und Semantik

Beschreibungslogiken bestehen im wesentlichen aus zwei Teilen:

- der Terminologischen Box (TBox), und
- der Assertionalen Box (ABox).

Während die TBox die Begriffsdefinitionen, das terminologische Wissen, enthält, dient die ABox zur Ablage der konkreten Situationsbeschreibungen, dem assertionalem Wissen. Übertragen auf objektorientierte Repräsentationen dient die TBox somit der Definition der Klassen, während die ABox die Beschreibungen der einzelnen Instanzen enthält.

Begriffe werden auf Basis einer festen Menge von Konstruktoren definiert. Für jeden der Konstruktoren kann eine modelltheoretische Semantik angegeben werden. Somit unterscheiden sich verschiedene Beschreibungslogiken im wesentlichen durch die Menge und Art der von ihnen verwendeten Konstruktoren.

Eine Vorstellung der grundlegenden Ideen, die hinter dem Konzept Beschreibungslogiken stehen, erfolgt am besten auf Basis einer konkreten Beschreibungssprache. In diesem und den nächsten Abschnitten sollen die grundlegenden Ideen mit Hilfe der Beschreibungssprache $\mathcal{ALCF}$ [BH91a, Han92, Han96] präsentiert werden.

$\mathcal{ALCF}$ wurde gewählt, da sie, wie sich herausstellen wird, eine Beschreibungssprache ist, die den im letzten Kapitel formulierten Anforderungen an eine Wissensrepräsentationssprache bereits sehr nahe kommt.

3.2.1. Syntax

Zur Notation der Beschreibungen wird im folgenden die abstrakte Syntax für Beschreibungslogiken, die im Rahmen der KRSS[2] Standardisierungsbemühungen definiert wurde [PSS93], verwendet [3].

2. KRSS – Knowledge Representation System Specification.

TBox

Zentral für Beschreibungslogiken sind Möglichkeiten zur Definition von Begriffen mit Hilfe von *Konzepttermen*. Ausgehend von einer Menge primitiver Konzept- und Rollenterme werden neue Konzeptterme mit Hilfe von konzeptbildenden Operatoren gebildet.

Definition 3.1 (Feature-, Rollen- und Konzeptnamen)

- Im weiteren seien C,R und F, paarweise disjunkte (d.h. $\mathsf{C} \cap \mathsf{R} = \mathsf{C} \cap \mathsf{F} = \mathsf{F} \cap \mathsf{R} = \emptyset$) Alphabete von Symbolen: den **Konzept-**, **Rollen-** und **Featurenamen**.
- Weiterhin sind $\top \in \mathbf{C}$ und $\bot \in \mathbf{C}$ zwei spezielle Konzeptnamen genannt **Top** und **Bottom**.

Definition 3.2 (Featureterme)
Die Menge **A** der **Featureterme** ist definiert durch:

- Alle $\mathsf{f} \in \mathsf{F}$ sind Featureterme.
- Sind f und g Featureterme, dann ist auch die Komposition $(f \circ g)$ ein Featureterm.

Definition 3.3 (Rollenterme)
Die Menge **R** der **Rollenterme** ist definiert durch:

- Alle $\mathsf{R} \in \mathsf{R}$ sind Rollenterme.
- Alle Featureterme $a \in \mathbf{A}$ sind Rollenterme.
- Sind R und S Rollenterme, dann ist auch die Komposition $(\mathrm{R} \circ \mathrm{S})$ ein Rollenterm.

Definition 3.4 (Konzeptterme)
Die Menge der **Konzeptterme** ist induktiv über die folgende Regeln definiert:

- Jedes Element $\mathsf{C} \in \mathsf{C}$ ist ein (atomarer) Konzeptterm.
- Sind C und D Konzeptterme und R ein Rollenterm, und f und g Featureterme, dann sind auch
  - $(C \sqcap D)$ (Konjunktion)
  - $(C \sqcup D)$ (Disjunktion)

3. Die abstrakte Syntax wurde entwickelt, da alle realisierten beschreibungslogischen Systeme eine jeweils andere Syntax aufwiesen bzw. sich die Semantik syntaktisch gleicher Konstrukte unterschied. Sie stellt somit eine Interlingua dar, auf die man sich im Bereich der Beschreibungslogiken verständigt hat.

- ◆ $\neg C$ (Negation)
- ◆ $\exists$ R . C (Existenzbeschränkung)
- ◆ $\forall$ R . C (Wertebereichsbeschränkung)
- ◆ f = g (Attributagreement)
- ◆ f $\neq$ g (Attributdisagreement)

Konzeptterme.

Bemerkung 3.1

- Eine alternative Schreibweise für die Syntax der erlaubten Konzeptterme, die oft in beschreibungslogischen Publikationen benutzt wird, ist:
  - ◆ $C, D \rightarrow \mathsf{C} \mid \top \mid \bot \mid (C \sqcap D) \mid (C \sqcup D) \mid \neg C \mid \exists R.C \mid \forall R.C \mid f = g \mid f \neq g$
- Featureterme und Rollenterme werden in Beschreibungslogiken auch als **Attributterme** bzw. Attribute und **Rollen** bezeichnet.
- $(f_1 \circ \ldots \circ f_n), f_1, \ldots, f_n \in \mathbf{A}$ heißt auch **Featurekette**, $(f_1 \circ \ldots \circ f_n), f_1, \ldots, f_n \in \mathbf{A}$ **Rollenkette**.

In der beschreibungslogischen Literatur werden sogar überwiegend die Begriffe Attribut, Attributnamen etc. benutzt. Wegen der bestehenden Verwechslungsgefahr mit den Attributen der OMT werden in dieser Arbeit die Begriffe Feature, Featureterm und Featurename benutzt.

Der Begriff Feature wurde aus dem Bereich der Featurelogiken [AK84] in den Bereich der Beschreibungslogiken übernommen. Features fanden Eingang in den Bereich der Beschreibungslogiken als klar wurde, daß bestimmte Konstrukte in sogenannten »Role Value Maps« dazu führen, daß die Beschreibungslogik unentscheidbar wird. Eingeschränkt auf Features entspricht eine Role Value Map aber gerade dem Agreementoperator, was dazu führt, daß $\mathcal{ALCF}$ entscheidbar ist.

Auch der Begriff der Rolle ist im Bereich der Beschreibungslogiken und der objektorientierten Repräsentationen unterschiedlich besetzt. Rollen aus dem Bereich der Beschreibungslogiken entsprechen den Assoziationen objektorientierter Repräsentationen.

Aufbauend auf Konzepttermen können terminologische Axiome und die TBox definiert werden. *Terminologische Axiome* assoziieren einen Konzeptnamen C mit einem Konzeptterm C und werden zur Definition der relevanten Konzepte einer Anwendung benutzt. Die *TBox* ist die Menge der terminologischen Axiome einer Anwendungsdomäne.

Definition 3.5 (Terminologische Axiome)
Seien $\mathsf{C} \in \mathbf{C}$ ein Konzeptname, C ein Konzeptterm, $\mathsf{R} \in \mathbf{R}$ ein Rollenname und $\mathsf{f} \in \mathbf{F}$ ein Featurename. **Terminologische Axiome** sind dann definiert als:

- $\mathsf{C} \doteq C$ (Konzeptdefinition)
- $\mathsf{C} \dot{\sqsubseteq} C$ (Konzeptspezialisierung)
- $\mathsf{R} \dot{\sqsubseteq} \top \times \top$ (Rolleneinführung)
- $\mathsf{f} \dot{\sqsubseteq} \top \mapsto \top$ (Featureeinführung)

Definition 3.6 (TBox)
Eine endliche Menge $\mathbf{T} = \{T_1, \ldots, T_n\}$ von terminologischen Axiomen heißt **TBox** (bzw. terminologische Box), falls:

- jedes $\mathsf{C} \in \mathbf{C}$ maximal einmal auf einer linken Seite eines terminologischen Axioms in $\mathbf{T}$ auftaucht, und
- $\mathbf{T}$ keine Zyklen enthält.

Definierte und primitive Konzepte Terminologische Axiome können auch als Abkürzungen für Konzeptterme angesehen werden. Axiome der Form $\mathsf{C} \doteq C$ definieren notwendige und hinreichende Bedingungen an einen Begriff. C heißt deshalb auch **definiertes Konzept**. Axiome der Form $\mathsf{C} \dot{\sqsubseteq} C$ dienen zur Beschreibung von Begriffen, für die nur notwendige aber keine hinreichenden Bedingungen bekannt sind. Damit sind sie zu Klassenbeschreibungen objektorientierter Repräsentationen äquivalent [CLN94]. C wird in diesem Falle auch **primitives Konzept** genannt.

Konzeptnamen $\mathsf{C} \in \mathbf{C}$, die nicht auf einer linken Seite in $\mathbf{T}$ auftauchen, sind ebenfalls primitive Konzepte. Im allgemeinen werden solche Konzepte durch Axiome der Form $\mathsf{C} \dot{\sqsubseteq} \top$ eingeführt. Im folgenden wird davon ausgegangen, daß alle benutzen Konzeptnamen vorher durch ein terminologisches Axiom eingeführt wurden.

Auf die gleiche Weise werden auch die verwendeten Features und Rollen durch die Axiome der Form $\mathsf{R} \dot{\sqsubseteq} \top \times \top$ und $\mathsf{A} \dot{\sqsubseteq} \top \mapsto \top$ eingeführt.

Sonstiges Es wurden auch Beschreibungslogiken untersucht, die zyklische Definitionen in $\mathbf{T}$ enthalten können [Neb90, Neb91, Baa90, Baa91, Sch94a]. Diese Untersuchungen sind aber bislang noch nicht in Implementierungen umgesetzt worden.

Ebenso gibt es auch Beschreibungslogiken, die über zusätzliche Operatoren für Rollen- und Featureterme, etwa die Konjunktion oder Restriktion von Rollen verfügen. In diesen Sprachen macht es Sinn, nicht nur Axiome zur Einführung von Rollen und Features zu definieren, sondern auch solche zur Definition von Rollen und Attributen.

Beispiel 3.1
Gesucht ist eine Umsetzung des folgenden Wissens über Verwandtschaftsbeziehungen[4]:

- Eine Frau ist ein weiblicher Mensch.
- Ein Mann ist ein männlicher Mensch.
- Eine Ehefrau ist eine Frau, die mit einem Mann verheiratet ist.
- Ein Elternteil ist ein Mensch, der einen Menschen als Kind hat.
- Eine Mutter ist ein weibliches Elternteil.
- Eine Frau ohne Töchter ist eine Frau, deren Kinder alle nicht weiblich sind.
- Eine Großmutter ist eine Frau, die ein Kind hat, das Elternteil ist.

Die Begriffe Mensch, $\mathrm{Weiblich}$ und $\mathrm{Maennlich}$ primitiv, da sie in diesem Zusammenhang nicht weiter spezifiziert werden. $\mathrm{hatKind}$ und $\mathrm{verheiratetMit}$ sind primitive Beziehungen. Auf Basis dieser primitiven Begriffe können die anderen Begriffe definiert werden.

Eine entsprechende TBox ist etwa $\mathbf{T} = \{\mathsf{S}_1, \ldots, \mathsf{S}_5, \mathsf{C}_1, \ldots, \mathsf{C}_7\}$:

- $\mathsf{S}_1 =$ Mensch $\dot{\sqsubseteq}\ \top$
- $\mathsf{S}_2 =$ Weiblich $\dot{\sqsubseteq}\ \top$
- $\mathsf{S}_3 =$ Maennlich $\dot{\sqsubseteq}\ \top$
- $\mathsf{S}_4 =$ hatKind $\dot{\sqsubseteq}\ \top \times \top$
- $\mathsf{S}_5 =$ verheiratetMit $\dot{\sqsubseteq}\ \top \mapsto \top$

- $\mathsf{C}_1 =$ Frau $\doteq$ (Mensch $\sqcap$ Weiblich)
- $\mathsf{C}_2 =$ Mann $\doteq$ (Mensch $\sqcap$ Maennlich)
- $\mathsf{C}_3 =$ Ehefrau $\doteq$ (Frau $\sqcap \exists$ verheiratetMit . Mann)
- $\mathsf{C}_4 =$ Elternteil $\doteq$ (Mensch $\sqcap \exists$ hatKind . Mensch)
- $\mathsf{C}_5 =$ Mutter $\doteq$ (Elternteil $\sqcap$ Frau)
- $\mathsf{C}_6 =$ FrauOhneToechter $\doteq$ (Frau $\sqcap \forall$ hatKind . $\neg$Weiblich)
- $\mathsf{C}_7 =$ Grossmutter $\doteq$ (Frau $\sqcap \exists$ hatKind . Elternteil)

Die definierten Konzepte stehen dabei für die folgenden prädikatenlogischen Formeln:

- $\forall x(\mathrm{Frau}(x) \leftrightarrow \mathrm{Mensch}(x) \wedge \mathrm{Weiblich}(x))$
- $\forall x(\mathrm{Mann}(x) \leftrightarrow \mathrm{Mensch}(x) \wedge \mathrm{Maennlich}(x))$
- $\forall x(\mathrm{Ehefrau}(x) \leftrightarrow \mathrm{Frau}(x) \wedge \exists y(\mathrm{verheiratetMit}(x, y) \wedge \mathrm{Mann(y)}))$
- $\forall x(\mathrm{Elternteil}(x) \leftrightarrow \mathrm{Mensch}(x) \wedge \exists y(\mathrm{hatKind}(x, y) \wedge \mathrm{Mensch}(y)))$
- $\forall x(\mathrm{Mutter}(x) \leftrightarrow \mathrm{Elternteil}(x) \wedge \mathrm{Frau}(x))$
- $\forall x(\mathrm{FrauOhneToechter}(x) \leftrightarrow \mathrm{Frau}(x) \wedge \forall y(\mathrm{hatKind}(x, y) \wedge \neg\, \mathrm{Weiblich}(y)))$
- $\forall x(\mathrm{Grossmutter}(x) \leftrightarrow \mathrm{Frau}(x) \wedge \exists y(\mathrm{hatKind}(x, y) \wedge \mathrm{Elternteil}(y)))$

Obwohl die zu beschreibenden Begriffe sehr einfach sind, ist die Repräsentation durch prädikatenlogische Formeln schon unübersichtlich. Für komplexe Begriffe sind die entsprechenden Formeln nicht mehr handhabbar. Beschreibungslogiken stellen im Vergleich dazu eine strukturierte, objektorientierte und handhabbare Repräsentation zur Verfügung.

4. Verwandtschaftsbeziehungen sind im Bereich der Beschreibungslogiken die klassische Beispielsdomäne (siehe z.B. [BBH+92]).

ABox

In der TBox werden die Begriffe definiert, mit denen mögliche Welten beschrieben werden können. Die ABox dient der Beschreibung konkreter Welten auf Basis der in der TBox definierten Begriffe.

Konkrete Welten werden durch eine Menge von Instanzen und Assertionen über diesen Instanzen beschrieben. Durch Assertionen können einerseits die Zugehörigkeit einer Instanz zu einem bestimmten Konzept, andererseits Beziehungen zwischen zwei Objekten beschrieben werden.

Definition 3.7 (Assertionen)
Sei O ein Alphabet von **Objektnamen** mit $\mathsf{O} \cap (\mathsf{C} \cup \mathsf{R} \cup \mathsf{F}) = \emptyset$. Seien weiterhin $\mathsf{o}, \mathsf{p} \in \mathsf{O}$, C ein Konzeptterm und R ein Rollenterm. Dann sind

- $\mathsf{o} : C$ (Zugehörigkeit)
- $\langle \mathsf{o}, \mathsf{p} \rangle : R$ (Beziehung)
- $\mathsf{o} = \mathsf{p}$ (Gleichheit)
- $\mathsf{o} \neq \mathsf{p}$ (Ungleichheit)

Assertionen.

Eine ABox besteht dann aus einer endlichen Zahl von Assertionen.

Definition 3.8 (ABox)
Eine **ABox** ist eine endliche Menge $\mathbf{A} = \{A_1, \ldots, A_n\}$ von Assertionen.

In praktischen Realisierungen werden im Normalfall die benutzten Objektnamen vor ihrer Verwendung in Assertionen explizit eingeführt.

3.2.2. Semantik

Beschreibungslogiken unterscheiden sich von anderen Wissensrepräsentationsformalismen dadurch, daß es möglich ist, eine modelltheoretische Semantik für alle Konstrukte anzugeben. Dazu definiert man zunächst die Semantik für die unterschiedlichen Konzept-, Rollen und Featureterme.

Definition 3.9 (Semantik der TBox Konstrukte)
Eine Interpretation $\mathcal{I}$ für $\mathcal{ALCF}$ besteht aus einer Menge $\Delta_a$und einer Interpretationsfunktion $(\cdot)^{\mathcal{I}}$. Diese Interpretationsfunktion assoziiert mit jedem Konzeptnamen $\mathsf{C} \in \mathsf{C}$ eine Teilmenge $\mathsf{C}^{\mathcal{I}} \subset \Delta_a$, mit jedem Rollennamen $\mathsf{R} \in \mathsf{R}$ eine binäre Relation $\mathsf{R}^{\mathcal{I}} \subseteq \Delta_a \times \Delta_a$ und mit jedem Feature $\mathsf{f} \in \mathsf{F}$ eine partielle Funktion $\mathsf{f}^{\mathcal{I}} : \Delta_a \mapsto \Delta_a$.

Die Interpretationsfunktion über den atomaren Termen wird wie folgt auf beliebige Konzept-, Rollen- und Featureterme fortgesetzt:

- $(R \circ S)^{\mathcal{I}} = \{(d,e) \in \Delta_a \times \Delta_a \mid \exists f.(d,f) \in R \wedge (f,e) \in S\}$
- $\top^{\mathcal{I}} = \Delta_a$
- $\bot^{\mathcal{I}} = \emptyset$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} = \Delta_a \setminus C^{\mathcal{I}}$
- $(\exists R.C)^{\mathcal{I}} = \{d \in \Delta_a \mid \exists e.(d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
- $(\forall R.C)^{\mathcal{I}} = \{d \in \Delta_a \mid \forall e.(d,e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}$
- $(f = g)^{\mathcal{I}} = \{d \in \Delta_a \mid f(d)^{\mathcal{I}} = g(d)^{\mathcal{I}}\}$
- $(f \neq g)^{\mathcal{I}} = \{d \in \Delta_a \mid f(d)^{\mathcal{I}} \neq g(d)^{\mathcal{I}}\}$

Bemerkung 3.2

- Aufgrund ihrer Semantik werden Features oft auch **funktionale Rollen** genannt.
- Zur Abkürzung werden oft auch n-stellige Varianten von Konjunktion und Disjunktion und n-fache Komposition verwendet. Die n-fache Komposition wird auch **Pfad** genannt.
  - $(C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I}} = {C_1}^{\mathcal{I}} \cap \ldots \cap {C_n}^{\mathcal{I}}$
  - $(C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}} = {C_1}^{\mathcal{I}} \cup \ldots \cup {C_n}^{\mathcal{I}}$
  - $(R_1 \circ \ldots \circ R_n)^{\mathcal{I}} = \{(d,e) \in \Delta_a \times \Delta_a \mid \exists d_1, \ldots, d_{n-1}.(d,d_1) \in R_1 \wedge \ldots \wedge (d_{n-1},e) \in R_n\}$
- Für $f(o)^{\mathcal{I}} = p$ schreibt man oft auch $(o,p) \in f^{\mathcal{I}}$.

Die Interpretation wird wie folgt auf terminologische Axiome erweitert.

Definition 3.10 (Semantik terminologischer Axiome)
Seien $\mathsf{C} \in \mathbf{C}$ ein Konzeptname, C ein Konzeptterm, $\mathsf{R} \in \mathbf{R}$ ein Rollenname und $\mathsf{f} \in \mathbf{F}$ ein Featurename. Dann werden terminologische Axiome wie folgt interpretiert:

- $(\mathsf{C} \doteq C)^{\mathcal{I}} = (\mathsf{C}^{\mathcal{I}} = C^{\mathcal{I}})$
- $(\mathsf{C} \dot{\sqsubseteq} C)^{\mathcal{I}} = (\mathsf{C}^{\mathcal{I}} \subseteq C^{\mathcal{I}})$
- $(\mathsf{R} \dot{\sqsubseteq} \top \times \top)^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}} \subseteq \Delta_a \times \Delta_a)$
- $(\mathsf{f} \dot{\sqsubseteq} \top \mapsto \top)^{\mathcal{I}} = (\mathsf{f}^{\mathcal{I}} \subseteq \Delta_a \times \Delta_a)$

Bemerkung 3.3 (Primitive und definierte Konzepte)
Die unterschiedliche Definition der Semantik primitiver und definierter Konzepte macht deutlich, daß primitive Konzepte notwendige Bedingungen an die Begriffsdefinition darstellen, während definierte Konzepte notwendige und hinreichende Bedingungen an einen Begriff formulieren.

Es wird weiterhin klar, daß primitive Konzeptdefinitionen nur eine abkürzende Schreibweise sind. Eine primitive Konzeptdefinition $\mathsf{C} \dot{\sqsubseteq} C$ kann durch die Einführung eines zusätzlichen primitiven Konzeptnamens $\mathsf{C}^* \in \mathbf{C}$ in ein äquivalentes definiertes Konzept umgewandelt werden:

$$\mathsf{C} \dot{\sqsubseteq} C \Leftrightarrow \mathsf{C} \doteq (C \sqcap \mathsf{C}^*)$$

Daher wurde in der ursprünglichen Definition von $\mathcal{ALCF}$ [Han92] auf terminologische Axiome der Form $\mathsf{C} \dot{\sqsubseteq} C$ verzichtet. Begreift man Beschreibungslogiken als Erweiterungen objektorientierter Sprachen ist diese Art terminologischer Axiome wegen der Äquivalenz zu Klassendefinitionen objektorientierter Repräsentationen jedoch unverzichtbar.

Auf Basis der Semantik der Konzeptterme und der terminologischen Axiome kann dann der zentrale Begriff des Modelles einer TBox definiert werden.

Definition 3.11 (Modell einer TBox)
Eine Interpretation $\mathcal{I}$ ist ein **Modell einer TBox** $\mathbf{T} = T_1, \ldots, T_n$, gdw. die Interpretation alle terminologischen Axiome $T_1, \ldots, T_n$ erfüllt.

Nachdem die Semantik der TBox Konstrukte definiert ist, kann darauf aufbauend die Semantik der ABox Konstrukte definiert werden.

Definition 3.12 (Semantik der ABox Konstrukte)
Eine Interpretation für Assertionen ist eine Fortsetzung der Interpretation über den TBox Konstrukten, in der zusätzlich jeder Objektname $\mathsf{o} \in \mathsf{O}$ mit einem Objekt $\mathsf{o}^{\mathcal{I}} \in \Delta_a$ identifiziert wird. Eine solche Interpretation erfüllt eine Assertion falls:

- $(\mathsf{o} : C)^{\mathcal{I}} = \mathsf{o}^{\mathcal{I}} \in C^{\mathcal{I}}$
- $(\langle \mathsf{o}, \mathsf{p} \rangle : R)^{\mathcal{I}} = \langle o^{\mathcal{I}}, p^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$
- $(\mathsf{o} = \mathsf{p}^{\mathcal{I}}) = (\mathsf{o}^{\mathcal{I}} = \mathsf{p}^{\mathcal{I}})$
- $(\mathsf{o} \neq \mathsf{p}^{\mathcal{I}}) = (\mathsf{o}^{\mathcal{I}} \neq \mathsf{p}^{\mathcal{I}})$

Definition 3.13 (Modell einer ABox (bzgl. einer TBox))
Eine Interpretation $\mathcal{I}$ ist ein **Modell der ABox A** gdw. falls sie alle Assertionen $A_1, \ldots, A_n$ von $\mathbf{A}$ erfüllt.

Sie ist ein **Modell der ABox A bzgl. der TBox T**, falls sie ein Modell von $\mathbf{T}$ und ein Modell von $\mathbf{A}$ ist.

3.2.3. Keine Unique-Name-Assumption

In der ABox von $\mathcal{ALCF}$ wird nicht angenommen, daß unterschiedliche Objektnamen o und b auch unterschiedliche Objekte $\mathsf{o}^{\mathcal{I}}$und $\mathsf{p}^{\mathcal{I}}$denotieren, d.h. für unterschiedliche Objekte stehen. Damit unterscheidet sich $\mathcal{ALCF}$ von den meisten anderen Beschreibungslogiken, in denen diese **Unique Name Assumption** gemacht wird.

Gerade im Bereich des Service und Supports ist es allerdings von großem Vorteil, die Unique Name Assumption nicht zu machen. Dadurch wird die inkrementelle und verteilte Erhebung von Wissen erheblich unterstützt bzw. erst möglich. So kommt es relativ häufig vor, daß zwei Servicetechniker eine Anlage beschreiben, und erst später feststellen, daß sie über die gleiche Anlage reden.

Mit Hilfe des Ungleichheitsoperators der ABox kann die Unique Name Assumption leicht simuliert werden, man hat also die zusätzliche Möglichkeit die Unique Name Assumption selektiv zu verwenden.

3.3. Inferenzen und Algorithmen

Die modelltheoretische Semantik erlaubt die formale Definition einer Reihe von mächtigen Inferenzleistungen, die zur Ableitung neuer Informationen eingesetzt werden können. Dies betrifft sowohl die TBox als auch die ABox. Im ersten Teil dieses Abschnittes wird eine Reihe von Basisinferenzen vorgestellt.

Die formale Definition der Inferenzen muß, soll sie gewinnbringend eingesetzt werden, in Algorithmen umgesetzt werden. Im zweiten Teil dieses Abschnittes werden daher die unterschiedlichen Ansätze zur Umsetzung der Inferenzen in Algorithmen präsentiert und ihre Vor- und Nachteile diskutiert.

3.3.1. TBox Inferenzen

Subsumption

Ausgangspunkt für alle TBox Inferenzen ist die Ermittlung der sogenannten Subsumptionsbeziehung zwischen zwei Konzepten. Die Subsumptionsbeziehung ist die Teilmengenbeziehung zwischen den Mengen der Modelle der beiden untersuchten Konzepte.

Damit definiert die Subsumptionsbeziehung eine Halbordnung auf einer Menge von Konzepten bzw. einer TBox. Diese Halbordnung beschreibt die bestehenden Spezialisierungsbeziehungen zwischen den unterschiedlichen Konzepten der TBox. Und zwar die Menge der Spezialisierungsbeziehungen die aufgrund der modelltheoretischen Semantik bestehen, nicht nur die Menge der vorgegebenen Spezialisierungsbeziehungen, wie sie etwa in einer objektorientierten Modellierung formuliert werden.

Läßt sich die Subsumptionsbeziehung zwischen zwei Konzepten effektiv ermitteln, dann läßt sich durch die sukzessive Ermittlung der Subsumptionsbeziehung zwischen je zwei Konzepttermen die gesamte Spezialisierungshierarchie der in der TBox enthaltenen Konzepte ermitteln. Insbesondere läßt sich so die Menge der (direkten) oberen Nachbarn und der (direkten) unteren Nachbarn jedes Konzeptes bestimmen. Diesen Prozeß nennt man auch die Klassifikation der TBox.

Durch das gleiche Verfahren läßt sich auch die Position eines neuen Konzeptes in einer bestehenden Spezialisierungshierarchie bestimmen und damit das neue Konzept klassifizieren, d.h. die Menge seiner (direkten) Nachbarn bestimmen.

Äquivalenz und Erfüllbarkeit Neben der Berechnung der Subsumptionsrelation sind die Bestimmung der Äquivalenz zweier Konzepte bzw. die Bestimmung der Erfüllbarkeit eines Konzeptes weitere häufig benutzte TBox Inferenzen. Sind zwei Konzepte äquivalent, dann stellen sie im wesentlichen zwei syntaktisch unterschiedliche Formulierungen für den gleichen Begriff dar. Ist ein Konzept erfüllbar, so existiert ein Modell. Interessanter ist allerdings der Fall, wenn ein Konzept nicht erfüllbar ist. Dann kann prinzipiell kein Modell für dieses Konzept existieren. Dies ist oft ein Hinweis darauf, daß die Konzeptbeschreibung fehlerhaft ist.

Realisiert werden diese Inferenzen zunächst für Konzeptterme:

Definition 3.14 (Inferenzen für Konzeptterme)

- Seien C und D zwei Konzeptterme. D **subsumiert** C gdw. für alle Interpretationen $\mathcal{I}$ gilt $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

  Wird C von D subsumiert, so schreibt man auch $C \sqsubseteq D$.

- C und D heißen **äquivalent**, falls C D subsumiert und umgekehrt, also $C^{\mathcal{I}} = D^{\mathcal{I}}$; sie heißen **disjunkt**, falls $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$.

  Für äquivalente Konzeptterme schreibt man auch $C = D$.

- Ein Konzeptterm C heißt **erfüllbar**, wenn eine Interpretation $\mathcal{I}$ mit $C^{\mathcal{I}} \neq \emptyset$ existiert.

Reduzierbarkeit der Subsumption Ein Konzept D subsumiert ein zweites Konzept C genau dann, wenn das Konzept $(\neg D \sqcap C)$ nicht erfüllbar ist. Damit läßt sich die Subsumption und damit auch die Äquivalenz und die Disjunktheit zwischen zwei Konzepttermen auf die Erfüllbarkeit eines Konzepttermes reduzieren. Es genügt also, einen Algorithmus zur Überprüfung der Erfüllbarkeit eines Konzeptes zu finden, um auch die anderen Inferenzleistungen zu erbringen. Andererseits ist ein Konzept C nicht erfüllbar, wenn es von $\bot$ subsumiert wird, d.h.: $C \sqsubseteq \bot$ gilt.

Terminologische Axiome Terminologische Axiome sind im wesentlichen Abkürzungen für Konzeptterme. Daher ist es nicht erstaunlich, daß die Subsumption, Äquivalenz etc. von Konzepten einer TBox sich auf die entsprechenden Inferenzen über Konzepttermen reduzieren lassen. Betrachtet werden dann nicht alle Interpretationen, sondern nur die Modelle einer TBox.

Definition 3.15 (Inferenzen für Konzepte einer TBox)
Sei $\mathbf{T}$ eine TBox und seien $\mathsf{C} \doteq C$ und $\mathsf{D} \doteq D$ terminologische Axiome aus $\mathbf{T}$.

- D **subsumiert** C gdw. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ für alle Modelle von $\mathbf{T}$.
- **Erfüllbarkeit**, **Äquivalenz** und **Disjunktheit** von Konzepten werden analog definiert.

Expansion terminologischer Axiome Seien C' und D' die Konzeptterme, die aus den rechten Seiten der terminologischen Axiome $\mathsf{C} \doteq C$ und $\mathsf{D} \doteq D$ durch rekursive Ersetzung von Vorkommen linker Seiten durch die entsprechenden rechten Seiten entstehen. C' heißt dann **Expansion** von C.

Da die TBox per Definition azyklisch ist, terminiert die Expansion der Konzepte. Wegen der Eindeutigkeit der terminologischen Axiome ist die Expansion auch eindeutig, die Semantik der TBox wird nicht verändert. Damit reduzieren sich die Subsumption und die weiteren Inferenzen für Konzepte auf die entsprechenden Inferenzen für Konzeptterme.

Beispiel 3.2
Die Expansion von Grossmutter bzgl. der weiteren in Beispiel 3.1 vorgestellten Konzeptdefinitionen ist $(\mathsf{Frau} \sqcap \exists\ \mathsf{hatKind}\ .\ \mathsf{Elternteil}) = (\mathsf{Frau} \sqcap \exists\ \mathsf{hatKind}\ .\ (\mathsf{Mensch} \sqcap \exists\ \mathsf{hatKind}\ .\ \mathsf{Mensch}))$

Die in Beschreibungslogiken realisierten Subsumptionsalgorithmen sind aufgrund dieser Beschreibungen in der Lage zu erkennen, daß eine Grossmutter ein Elternteil, ist denn es gilt $\forall x(\text{Grossmutter}(x) \rightarrow \text{Elternteil}(x))$ da:

$$\begin{aligned}
\text{Grossmutter}(x) &\Leftrightarrow \text{Frau}(x) \wedge \exists y(\text{hatKind}(x,y) \wedge \text{Elternteil}(y) \\
&\Leftrightarrow \text{Mensch}(x) \wedge \text{Weiblich}(x) \wedge \exists y(\text{hatKind}(x,y) \wedge \text{Mensch}(y) \\
&\quad \wedge \exists z(\text{hatKind}(y,z) \wedge \text{Mensch}(z))) \\
\text{Elternteil}(x) &\Leftrightarrow \text{Mensch}(x) \wedge \exists y(\text{hatKind}(x,y) \wedge \text{Mensch}(y))
\end{aligned}$$

und

$$\begin{aligned}
&\forall x((\text{Mensch}(x) \wedge \text{Weiblich}(x) \wedge \exists y(\text{hatKind}(x,y) \wedge \text{Mensch}(y) \\
&\quad \wedge \exists z(\text{hatKind}(y,z) \wedge \text{Mensch}(z)))) \\
&\rightarrow \\
&(\text{Mensch}(x) \wedge \exists y(\text{hatKind}(x,y) \wedge \text{Mensch}(y))))
\end{aligned}$$

Ausgehend von den obigen Begriffsbeschreibungen können konkrete Situationen beschrieben werden, z.B:

- Diana ist eine Frau
- William ist ein Kind von Diana

Die Beschreibungen für diese Situationen sind aus prädikatenlogischer Sicht Grundatome:

- $\mathrm{Frau}(\mathrm{Diana})$
- $\mathrm{hatKind}(\mathrm{Diana}, \mathrm{William})$

Der Instanztest ist dann in der Lage, automatisch Instantiierungsbeziehungen abzuleiten. So wird zum Beispiel durch den Instanztest detektiert, daß Diana eine Mutter sein muß.

Konzeptklassifikation

Auf Basis der Subsumption läßt sich auch die Konzeptklassifikation definieren. Die Subsumptionsrelation definiert eine Halbordnung auf der gesamten Menge der Konzepte. Diese kann durch einen gerichteten Graphen beschrieben werden. Kanten in diesem Graphen beschreiben Spezialisierungsbeziehungen, die zwischen Konzepten aufgrund der modelltheoretischen Semantik bestehen.

Ein Konzept ist spezieller als ein anderes, falls die Menge seiner Modelle vollständig in der Menge der Modelle des allgemeineren Konzeptes enthalten ist. Allgemeinstes Konzept ist damit $\top$, speziellstes Konzept ist $\bot$.

Reduktion Läßt sich die Subsumptionsbeziehung zwischen zwei Konzepten effektiv ermitteln, erhält man ein Verfahren zur automatischen Bestimmung der Spezialisierungshierarchie der in der TBox definierten Konzepte. Ausgangspunkt ist der initiale, aus $\top$ und $\bot$ bestehende Konzeptgraph. In diesen werden nacheinander die Konzepte der TBox eingefügt und die Spezialisierungsbeziehungen des jeweils neuen Konzeptes mit den schon in den Konzeptgraph integrierten Konzepten ermittelt.

Bildlich gesprochen läßt man das in den Konzeptgraphen zu integrierende Konzept in den Konzeptgraph einsinken. Die Tiefe des Einsinkens ist durch die modelltheoretische Semantik bestimmt. Oberhalb des zu klassifizierenden Konzept befinden sich nur noch allgemeinere Konzepte, d.h. Konzepte die das zu klassifizierende Konzept subsumieren, darunter speziellere d.h. Konzepte die von dem zu klassifizierenden Konzept subsumiert werden..

Die Mengen der allgemeineren und spezielleren Konzepte im Konzeptgraphen lassen sich dann über den im Bereich der Graphentheorie verbreiteten Begriff des (oberen bzw. unteren) Nachbarn definieren.

Definition 3.16 (Konzeptklassifikation, Obere und untere Nachbarn)
Die **Konzeptklassifikation** ist eine TBox Inferenz, die, gegeben ein Konzept C_q und eine TBox $\mathbf{T}$, die **direkten oberen Nachbarn** $\overline{\mathsf{C}}_\mathsf{q} = \{\overline{C}_{q_1}, \ldots, \overline{C}_{q_n}\}$ in dem durch die Subsumptionsrelation auf der Menge der TBox Konzepte induzierten gerichteten Graphen bestimmt.

$$\overline{C}_{q_i} \in \overline{\mathsf{C}}_\mathsf{q} \Leftrightarrow (C_q \sqsubseteq C_{q_i} \wedge (\nexists C_{q_j}.C_{q_j} \sqsubseteq C_{q_i} \wedge C_q \sqsubseteq C_{q_j}))$$

Analog kann die Menge der **direkten unteren Nachbarn** als $\underline{\mathsf{C}}_\mathsf{q} = \{\underline{C}_{q_1}, \ldots, \underline{C}_{q_m}\}$ und die Menge der **direkten Nachbarn** als $\overline{\mathsf{C}}_\mathsf{q} \cup \underline{\mathsf{C}}_\mathsf{q}$ bestimmt werden.

Aufbauend auf der Definition der direkten Nachbarn können, wie allgemein üblich die (oberen, unteren) **Nachbarn n-ter Stufe** $\overline{\mathsf{C}}^\mathsf{n}_\mathsf{q}, \underline{\mathsf{C}}^\mathsf{n}_\mathsf{q}$ als n-fache Applikation der jeweiligen Basisrelation und die Menge der (oberen, unteren) **Nachbarn** $\overline{\mathsf{C}}^*_\mathsf{q}, \underline{\mathsf{C}}^*_\mathsf{q}$ als transitive Hülle der jeweiligen Basisrelation definiert werden.

Beispiel 3.3
Abbildung 3.1 zeigt ein Beispiel für die sukzessive Klassifikation einer TBox. In den initialen nur aus $\top$ und $\bot$ bestehenden Spezialisierungsgraphen werden nacheinander die Konzepte der TBox eingefügt bis alle Konzepte der TBox klassifiziert sind.

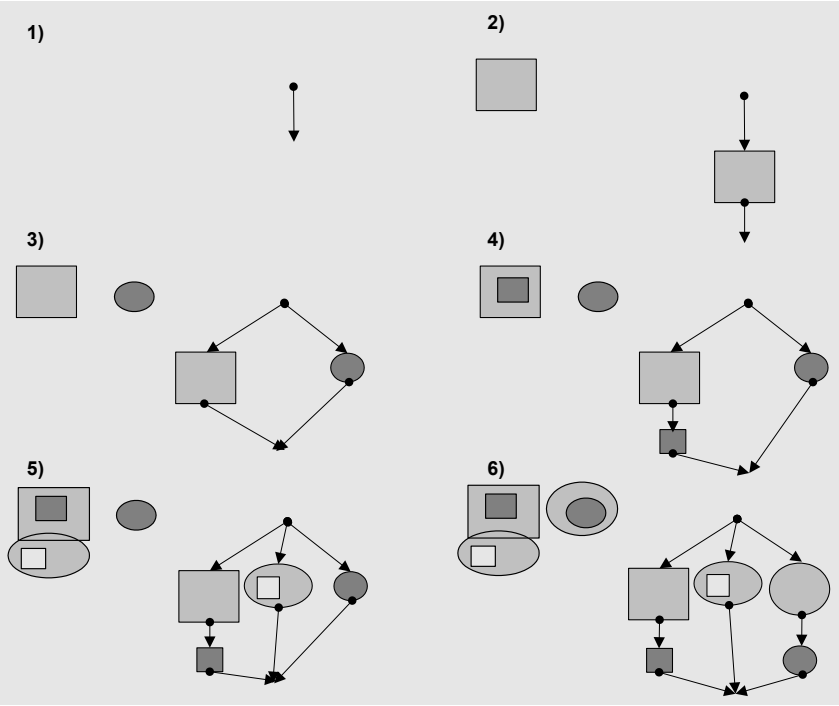


Abbildung 3.1.
Konzeptklassifikation

Konzeptklassifikation als Basis für ähnlichkeitsbasiertes Retrieval Das oben geschilderte Verfahren zur Konzeptklassifikation ist nicht auf die durch terminologische Axiome festgelegten Konzepte einer TBox beschränkt. Vielmehr kann es für beliebige, als Konzeptterme ausdrückbare Begriffe angewandt werden. Damit kann die Position beliebiger Konzeptterme in dem durch die TBox Begriffe induzierten Konzeptgraphen bestimmt werden.

Interpretiert man den neuen Konzeptterm als Anfrage, so liefert die Konzeptklassifikation eine Menge von Indizes zurück z.B. die Menge der direkten oberen und unteren Nachbarn. Diese Indizes dienen dann als Ausgangspunkt für Verfahren des ähnlichkeitsbasierten Retrievals.

Damit bietet die Konzeptklassifikation den Ausgangspunkt für ein Verfahren zur Beantwortung von ad hoc Anfragen an die Daten- und Wissensbasis und eine Reihe von Verfahren zum ähnlichkeitsbasierten Retrieval auf Basis von Beschreibungslogiken. Auf diese wird im Kapitel 6 detaillierter eingegangen.

3.3.2. ABox Inferenzen

Sind alle Konzepte der TBox erfüllbar, können Aussagen über Instanzen in der ABox gemacht und Inferenzen über diesen Aussagen gezogen werden. Voraussetzung dafür ist, daß das repräsentierte Wissen konsistent ist, da in der Logik jede beliebige Inferenz aus einer inkonsistenten Wissensbasis gezogen werden kann.

Konsistenz

Die Frage, ob eine ABox bzgl. einer TBox konsistent ist, ist damit ein grundlegendes und interessantes Problem.

Definition 3.17 (Konsistenz einer ABox (bzgl. einer TBox))
Sei **A** eine ABox und **T** eine TBox. **A** ist genau dann bzgl. **T konsistent**, wenn sie ein Modell bzgl. **T** hat. Ist **A** nicht konsistent bzgl. T, dann heißt **A inkonsistent**.

Der Test auf die Konsistenz einer ABox bzgl. einer TBox wird von manchen Autoren auch Erfüllbarkeitstest für eine Wissensbasis genannt.

Die Konsistenz einer ABox **A** bzgl. einer TBox **T** läßt sich durch Expansion der TBox auf die Konsistenz einer ABox $\mathbf{A}'$ bzgl. einer leeren TBox reduzieren. $\mathbf{A}'$ entsteht aus **A**, indem alle definierten Konzepte in Konzepttermen durch ihre entsprechenden Expansionen ersetzt werden.

Auch das Erfüllbarkeitsproblem für Konzepte und damit das Subsumptionsproblem ist auf das Konsistenzproblem für ABoxen reduzierbar. Ein Konzept C ist genau dann erfüllbar, wenn die ABox $\mathsf{o} : C$ konsistent ist, also $(\mathsf{o} : C)^{\mathcal{I}} = \emptyset$.

Damit lassen sich alle wichtigen Basisinferenzen auf das Konsistenzproblem für ABoxen reduzieren. Ist ein Algorithmus für diese Aufgabenstellung gefunden, dann können auch alle anderen Probleme gelöst werden.

Instantiierung

Ist das Wissen konsistent, dann können Anfragen an die Wissensbasis gestellt werden. Eine wichtige Fragestellung ist die Überprüfung, ob ein Objekt der ABox eine Instanz eines bestimmten Konzeptes ist. Die Inferenz, die diese Fragestellung beantworten kann, heißt Instantiierung.

Definition 3.18 (Instantiierung)
Sei $\mathbf{A}$ eine ABox und $\mathbf{T}$ eine TBox, $\mathsf{o} \in \mathsf{O}$ ein Objekt der ABox und C ein Konzeptterm. Dann ist o eine **Instanz** von C bzgl. $\mathbf{A}$ und $\mathbf{T}$ gdw. $\mathsf{o}^{\mathcal{I}} \in C^{\mathcal{I}}$ für alle Modelle von $\mathbf{A}$ bzgl. $\mathbf{T}$.

Wichtig ist in diesem Zusammenhang, daß es sich bei dem Konzeptterm nicht notwendigerweise um ein Konzept der TBox handeln muß, sondern es ein beliebiger Konzeptterm sein kann. Auf dieser Basis lassen sich leicht ad hoc Anfragen realisieren, die untersuchen, ob eine Objekt Instanz eines in der aktuellen Situation interessanten Konzeptes ist. Bei der Formulierung des Konzepttermes ist man vollkommen frei, so können z.B. zu einem in der in der TBox definierten Konzept beliebige zusätzliche Restriktionen formuliert werden oder Konjunktionen resp. Disjunktionen von TBox Konzepten als Anfrageterme realisiert werden.

Das Instantiierungsproblem kann auf das Konsistenzproblem reduziert werden, denn o ist Instanz von C bzgl. $\mathbf{A}$ und $\mathbf{T}$ gdw. die expandierte ABox $\mathbf{A}' \cup \{\mathsf{o} : \neg C\}$ inkonsistent ist.

Objektklassifikation und Retrieval

Besonders interessant sind die beiden folgenden komplexen Inferenzen, die auf die oben vorgestellten Basisinferenzen zurückgeführt werden können.

Definition 3.19 (Objektklassifikation)
Objektklassifikation[5] ist eine ABox Inferenz, die, gegeben ein Objekt o der ABox, die Menge $\overline{C}(o) = \{C_1, \ldots, C_n\}$ der speziellsten Konzepte in der Konzepthierarchie (Formal: $\overline{C}(o) = \{C_i \in \mathcal{T} \mid \{\mathcal{A} \cup \neg(o : C_i)\}^{\mathcal{I}} = \emptyset \ \wedge \ C_i \text{ minimal in } \mathcal{T}\}$) zurückliefert, für die gilt, daß o Instanz von C_i ist.

Die schwache Objektklassifikation liefert im Gegensatz dazu die Menge $\underline{C}(o)$ der speziellsten Konzepte, zu denen o noch potentiell realisiert werden kann (Formal: $\underline{C}(o) = \{C_i \in \mathcal{T} \mid \{\mathcal{A} \cup (o : C_i)\}^{\mathcal{I}} \neq \emptyset \ \wedge \ C_i \text{ maximal in } \mathcal{T}\}$).

Beispiel 3.4
Abbildung 3.2 zeigt ein Beispiel für die sukzessive Objektklassifikation. In die klassifizierte TBox aus Beispiel 3.3 werden sukzessive eine Reihe von Instanzen eingefügt und realisiert (Konzepte sind grau, Instanzen rot dargestellt).

5. Auch Realisierung genannt.

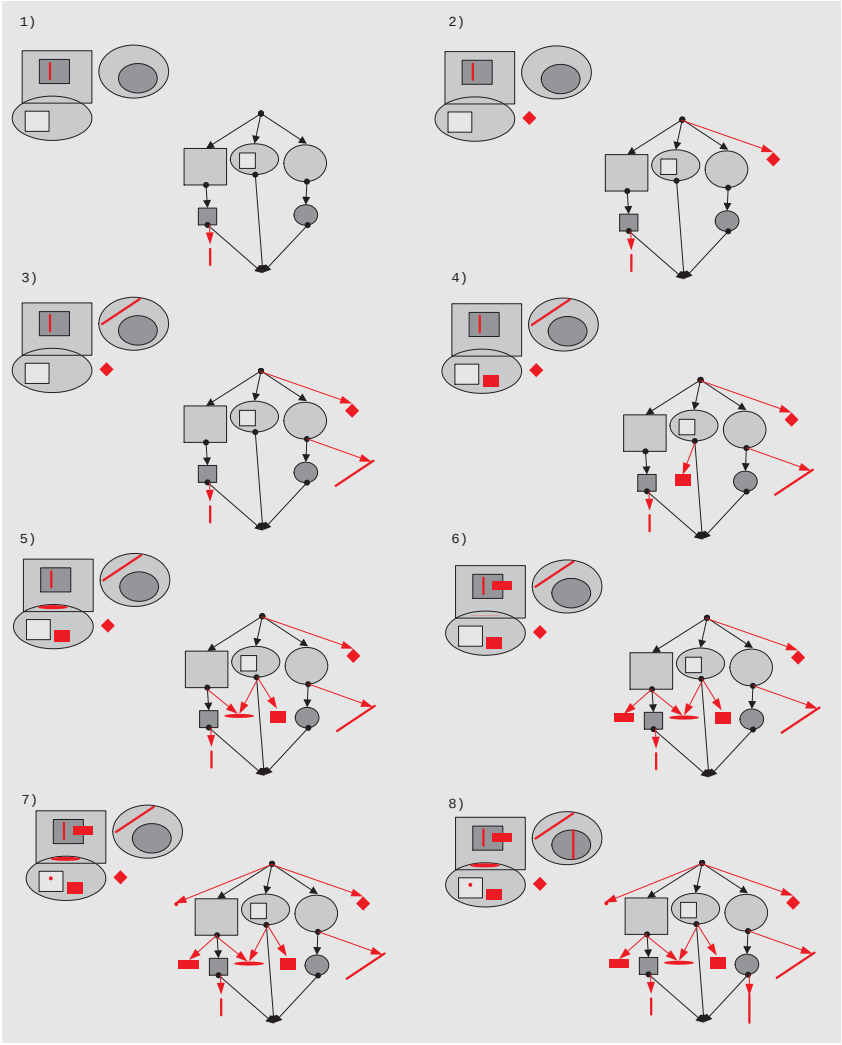


Abbildung 3.2.

Objektklassifikation

Definition 3.20 (Retrieval)
Das Retrieval-Problem ist in gewissem Sinne dual zur Objektklassifikation. Hier wird zu einem gegebenen Konzept C die Menge der ABox-Objekte $\overline{O}(C) = \{o_1, \ldots, o_m\}$ gesucht, die die Konzeptbeschreibung C erfüllen, bzw. ABox-Objekte $\underline{O}(C) = \{o_1, \ldots, o_m\}$, die noch zu diesem Objekt gehören können (schwaches Retrieval).

3.3.3. Algorithmen und Entscheidbarkeitsresultate

Die oben vorgestellten Reduktionen erlauben es, alle vorgestellten Inferenzen auf den Konsistenztest für eine ABox zu reduzieren. Es genügt also, einen Algorith-

mus für diese Inferenz zu finden, um einen Algorithmus für alle hier vorgestellten Inferenzen zu besitzen. Ein solcher Algorithmus wurde von Schmidt-Schauss und Smolka erstmalig gefunden. Er basiert auf einer an den Tableauxkalkül angelehnten Technik.

Historisch wurden allerdings zuerst andere Algorithmen entwickelt, die das Subsumptionsproblem auf Basis von Strukturtransformationen entscheiden. Grundsätzlich existieren damit zwei Ansätze zur Realisierung beschreibungslogischer Systeme:

- Struktur-orientierte Algorithmen,
- Tableaux-basierte Algorithmen.

Struktur-orientierte Algorithmen

Struktur-orientierte Algorithmen sind noch aus der Sichtweise semantischer Netze motiviert. In diesen wurden abgeleitete Informationen entlang der Kanten des Netzes propagiert. Struktur-orientierte beschreibungslogische Systeme reduzieren die verschiedenen Inferenzen auf das Subsumptionsproblem. So wird zur Berechnung des Instantiierungsproblems für ein Objekt o zuerst dessen Abstraktion, ein aus den verschiedenen Einträgen für o in der ABox berechneter Konzeptterm, ermittelt. Anschließend wird durch eine Folge von Subsumptionstests dessen Position in der Konzepthierarchie ermittelt [Neb90].

Der Subsumptionsalgorithmus selbst verläuft ebenfalls in zwei Phasen:

- Berechnung der Normalformen der beiden Konzeptterme,
- Syntaktischer Vergleich der Normalformen der beiden Konzeptterme.

Wie in Reduktionssystemen werden zur Berechnung der Normalform solange wie möglich Transformationsregeln, etwa der Form:
$((C((D \sqcup E)))) \Rightarrow (C(D \sqcup E))$ oder $(\forall R.C \sqcap \forall R.D) \Rightarrow (\forall R.(C \sqcap D))$
angewandt. Beim Vergleich der syntaktischen Strukturen werden die einzelnen Teile der beiden Normalformen verglichen und aufgrund des Ergebnisses dieser Vergleiche das Gesamtergebnis für die Normalformen ermittelt.

Sind z.B. $\forall$ R . C' und $\forall$ S . D' die Normalformen für C und D, dann wird D von C subsumiert, wenn $R = S$ ist, und C' subsumiert D' (siehe [Neb90]).

Auf dieser Basis ist es leicht möglich, effiziente Algorithmen zu entwickeln, die korrekt, aber im allgemeinen für ausdrucksstarke Beschreibungssprachen nicht vollständig sind. Dies bedeutet, daß alle erkannten Subsumptionsbeziehungen tatsächlich existieren, aber nicht alle bestehenden Subsumptionsbeziehungen wirklich erkannt werden. Es ist auch nicht klar, wie dieser Ansatz zu vollständigen Algorithmen erweitert werden kann.

Da auch alle anderen Inferenzen auf dem Subsumptionstest basieren, stimmt das Verhalten der implementierten Algorithmen mit der definierten Semantik nur mehr oder weniger überein und wird für den Benutzer unvorhersehbar und fehlerhaft.

Da die korrekte Klassifikation von Instanzen und Konzepten die Basis für das intelligente ähnlichkeitsbasierte Retrieval bildet, sind daher unvollständige Algorithmen für die Unterstützung der zweiten Supportebene wenig geeignet.

Lange Zeit waren allerdings nur struktur-basierte Algorithmen für die verschiedenen Inferenzen bekannt. Ein großer Teil der verfügbaren beschreibungslogischen Systeme benutzt weiterhin strukturorientierte Algorithmen. Ein anderer Grund für den Einsatz dieser Algorithmen lag und liegt in der Komplexität vollständiger Verfahren für ausdrucksstarke Sprachen. Diese sind zumeist NP-hart, oft sind die von aktuellen beschreibungslogischen Systemen benutzten Sprachen sogar unentscheidbar.

Tableaux-basierte Algorithmen

1988 beschreiben Schmidt-Schauss und Smolka [SSS91] den ersten vollständigen Subsumptionsalgorithmus für eine nichttriviale, propositional abgeschlossene Sprache (genannt $\mathcal{ALC}$ [6]), d.h. eine Sprache, die Disjunktion und Negation beinhaltet. Dazu benutzen sie einen gänzlich anderen Ansatz. Sie entwickeln ein Modellgenerierungsverfahren, das dem Tableaux-Kalkül für die Prädikatenlogik erster Stufe ähnelt, und benutzen dieses Verfahren, um das Erfüllbarkeitsproblem für Konzepte und damit auch das Subsumptionsproblem zu lösen.

Die ersten Algorithmen für hybride (d.h. T- und ABox enthaltende) Sprachen, die auf Basis der Ideen von Schmidt-Schauß und Smolka für hybride Sprachen entstanden [Hol90, BH91a], reduzierten zunächst das Erfüllbarkeitsproblem für Konzepte in der oben geschilderten Weise auf die Konsistenz der ABox. Weiterhin wurde das Modellgenerierungsverfahren zu einem Verfahren weiterentwickelt, das versucht ein endliches Modell für die ABox zu finden.

Das grundlegende Verfahren ist für die verschiedenen untersuchten Konzeptsprachen gleich:

- Expansion der ABox

  Ausgehend von einer initialen ABox $\mathbf{A}_0$ werden bestimmte Transformationsregeln angewandt, die das durch die Assertionen dargestellte Wissen explizieren. Dadurch wird die ABox $\mathbf{A}_i$ in eine ABox $\mathbf{A}_{i+1}$ expandiert.

Die ABox wird solange expandiert, bis eine der beiden folgenden Situationen auftritt:

6. $\mathcal{ALC}$ ist ausdrucksschwächer als die oben vorgestellte Sprache $\mathcal{ALCF}$. Mehr dazu in Abschnitt 3.5

- Die ABox $\mathbf{A}_i$ ist offensichtlich widersprüchlich.

  Eine ABox ist offensichtlich widersprüchlich, wenn sie einen aus einer Reihe von einfach zu überprüfenden Widersprüchen, sogenannten Clashes enthält. Dies ist z.B. der Fall, falls die ABox gleichzeitig $\mathsf{o} : \mathrm{C}$ und $\mathsf{o} : \neg\mathrm{C}$ enthält.

- Die ABox ist vollständig.

  Eine ABox ist vollständig, wenn keine Transformationsregel mehr angewandt werden kann. Dabei sind die Transformationsregeln selbst so definiert, daß eine vervollständigte ABox ohne offensichtlichen Widerspruch ein Modell für $\mathbf{A}$ ist.

Einige der Transformationsregeln für $\mathcal{ALCF}$ sind nichtdeterministisch, etwa die Transformationsregel für Disjunktionen. Daher kann eine ABox $\mathbf{A}_i$ mehrere potentielle Nachfolger $\mathbf{A}^1_{i+1},\ldots,\mathbf{A}^n_{i+1}$ haben. Im allgemeinen ist deshalb eine ABox genau dann konsistent, wenn eines der Blätter des ABox Expansionsbaumes keinen offensichtlichen Widerspruch enthält. Abbildung 3.3 verdeutlicht diesen Zusammenhang.

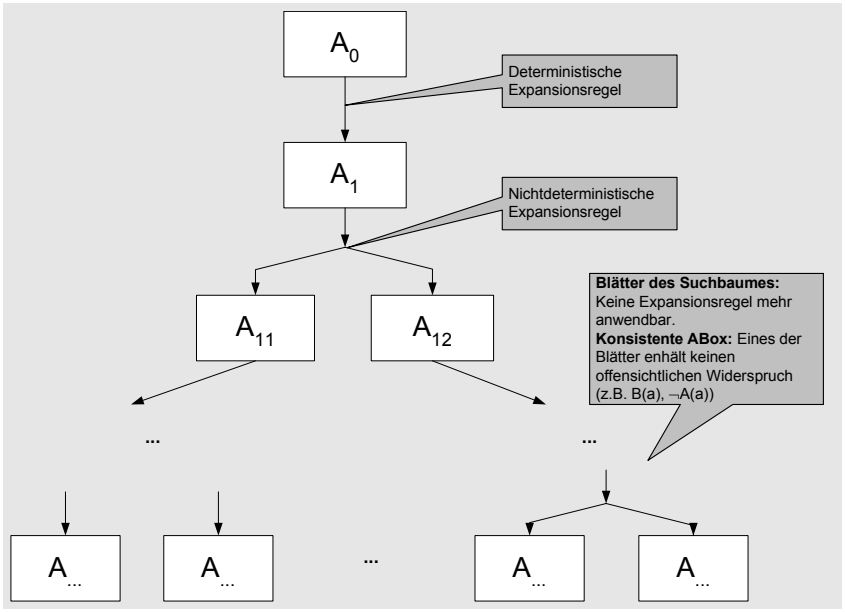


Abbildung 3.3.

Modellgenerierung durch ABox Expansion

Dieses Verfahren hat den Vorteil daß pro termerzeugenden Operator genau eine Vervollständigungsregel definiert werden muß. Eine existierende Konzeptsprache kann somit durch Hinzufügen einer geeigneten Vervollständigungsregel um einen neuen Operator erweitert werden, der Rest des Algorithmus bleibt unverändert, der Ansatz ist somit sehr modular.

Sowohl für den initialen Satz von Vervollständigungsregel als auch für jede Veränderung muß gezeigt werden, daß der Algorithmus terminiert und korrekt und vollständig ist.

Zum Beweis der Korrektheit und Vollständigkeit des Algorithmus für eine bestimmte Beschreibungssprache sind damit im wesentlichen die folgenden, im allgemeinen nichttrivialen, Schritte notwendig [Baa96]:

- Entwicklung von Transformationsregeln für die Operatoren.
- Beweis der lokalen Korrektheit der einzelnen Regeln.
- Beweis der Vollständigkeit des Regelsystems.
- Beweis der Terminierung des Regelsystems.

All dies wurde in [Han92, Han96] für $\mathcal{ALCF}$ bewiesen [7]. Daher gilt der folgende Satz:

Satz 3.1 (Entscheidbarkeit von $\mathcal{ALCF}$)
Es existiert ein korrekter und vollständiger Algorithmus für die Beschreibungssprache $\mathcal{ALCF}$, der das Konsistenzproblem für eine ABox $\mathbf{A}$ *(bzgl. einer TBox* $\mathbf{T}$*) entscheidet.*

Damit gibt es auch korrekte und vollständige Algorithmen für die Instantiierung, die Erfüllbarkeit eines Konzepttermes und der Subsumption, der Aquivalenz und der Disjunktheit von zwei Konzepttermen.

Satz 3.2 (Entscheidbarkeit der T- und ABox Inferenzen für $\mathcal{ALCF}$)
Es existieren korrekte und vollständige Algorithmen für die Beschreibungssprache $\mathcal{ALCF}$ und die Probleme der Instantiierung, der Erfüllbarkeit eines Konzeptterms und der Subsumption, Äquivalenz und Disjunktheit zweier Konzept

Komplexität, Direkte und indirekte Algorithmen für die ABox Konsistenz

Schmidt-Schauß und Smolka haben bewiesen, daß das Erfüllbarkeitsproblem für $\mathcal{ALC}$ PSPACE-vollständig ist und der von ihnen entwickelte Algorithmus daher in diesem Sinne in Bezug auf die Worst-Case-Komplexität optimal ist. Im Prinzip verschlechtert der Übergang zu $\mathcal{ALCF}$ dieses Ergebnis nicht [HN90]. Dazu benutzt Hollunder allerdings nicht einen direkten Algorithmus auf Basis des Konsistenztests, sondern zeigt stattdessen in [Hol94], daß sich auch umgekehrt der Konsistenztest für ABoxen auf die Erfüllbarkeit von Konzepten reduzieren läßt.

7. In [Han92] wird der Beweis für eine Erweiterung von $\mathcal{ALCF}$ geführt. Dies impliziert natürlich die Entscheidbarkeit von $\mathcal{ALCF}$.

Hollunder benutzt diese Reduktion, um einen funktionalen Algorithmus für die Konsistenz einer ABox zu entwickeln, der das Problem in polynomialem Platz löst. Dazu wird in einem Vorverarbeitungsschritt die ABox so erweitert, daß die Beziehungsassertionen zwischen Objekten redundant werden. Dann muß in einem zweiten Schritt nur noch für jedes Objekt der ABox die Erfüllbarkeit eines bestimmten Konzeptes überprüft werden.

In Bezug auf die direkten Verfahren zur Lösung des Konsistenzproblems bemerkt Hollunder:

≫Thus in contrast to the preprocessing rules, the rules defined in [Hol90] do not only operate on individuals explicitly mentioned in an ABox, but also introduces names for individuals which are implicitly mentioned in, for instance, exists-restictions. A complete ABox thus obtained may contain an exponential number of assertions (in the size of the input ABox), which means that the consistency algorithm described in [Hol90] has an exponential space complexity.≪

Als wesentlicher Nachteil wird von Hollunder also die Einführung zusätzlicher, impliziter Instanzen herausgestellt, die zusätzlichen und im worst case exponentiellen statt polynomialem Platz benötigt. Daher wird versucht die Einführung dieser Instanzen gänzlich zu vermeiden. Trotz der gravierenden Komplexitätsunterschiedes sind die funktionalen, in polynomialen Platz ablaufenden ≫tracebasierten≪ Algorithmen aber nicht per se besser als direkte, exponentiellen Platz benötigende Implementierungen des Tableaux-Kalküls.

So geben Bresciani et.al. in [BFT95] ein Beispiel für ein einfaches Konzept in der Beschreibungssprache $\mathcal{ALCO}$, dessen Unerfüllbarkeit mit Hilfe tracebasierter Techniken nicht entdeckt werden kann. Es ist also nicht klar, für welche ausdrucksstärkeren Beschreibungssprachen sich die dem funktionalen Algorithmus zugrundeliegende Technik der ≫Traces≪ nutzen lassen.

Weiterhin erlauben die direkten Algorithmen modularere Implementierungen, und der exponentielle Platzbedarf tritt nur im Worst-Case auf [BFT95]:

≫However, differently from other implementations, rules of tableaux are directly implemented, allowing for a high grade of modularity by just adding/removing rules. As a further consequence of this choice, it turns out that the conceptual and the individual level are treated in a uniform manner.
...
Thus, our general satisfiability algorithm does not assume any kind of independency between traces, since it has to be used for very expressive description logics. ...

However there is a price to pay for this choice: sub-languages where the complexity of satisfiability is PSPACE-complete are not handled properly with such a non-trace algorithm. In fact, the plain non-deterministic application of the propagation rules may generate an exponential number of variables. In general, such a straightforward implementation does not assure that the obtained procedure is optimal with respect to worst case complexity of the satisfiability problem for the implemented language. We believe that such worst cases are unlikely to happen in practice, just as exponential in time worst cases do not happen in real and average knowledge bases.≪

Der ≫Nachteil≪ der Erzeugung implizit notwendiger Instanzen verwandelt sich sogar immer dann in einen Vorteil, wenn man nicht nur an der Existenz eines Modelles, sondern am Aussehen dieses Modelles interessiert ist. Dies ist im Bereich des Second-Level-Supports definitiv der Fall.

So geben implizit eingeführte Instanzen dem Benutzer z.B. Hinweise darauf, daß er bei der Beschreibung eines technischen Gerätes bestimmte Komponenten vergessen oder andere Fehler gemacht hat. Besonders im Bereich der Konfigurationsunterstützung bieten sie dem Benutzer eine große Hilfe, indem etwa für eine korrekte Konfiguration fehlende Komponenten angezeigt werden, die in einem nächsten Schritt vom Benutzer genauer spezifiziert werden können.

Somit sind für in dem in dieser Arbeit diskutierten Kontext direkte Implementierungen besser geeignet als tracebasierte Algorithmen.

3.4. Korrespondenz zu objektorientierten Beschreibungen

Inwieweit ist es nun möglich, mit den Ausdrucksmitteln von $\mathcal{ALCF}$ die Konstrukte der im letzten Kapitel vorgestellten, erweiterten objektorientierten Wissensrepräsentation zu realisieren?

Zur Herstellung der Korrespondenz werden in diesem Abschnitt zuerst die Standardausdrucksmittel der OMT untersucht bevor auf die in Abschnitt 2.5 vorgestellten Erweiterungen der OMT eingegangen wird.

Es stellt sich heraus, daß $\mathcal{ALCF}$ alle geforderten Erweiterungen der OMT Notation zur Verfügung stellt. Dies ist nicht sehr überraschend, da Beschreibungslogiken ja als Wissenrepräsentationsformalismen entwickelt wurden. Dennoch ist die Bereitstellung *aller* Mechanismen keine Selbstverständlichkeit. So stellen etwa viele Beschreibungslogiken und beschreibungslogischen Systeme keine Disjunktion zur Verfügung. Damit bieten sie aber auch keine Möglichkeit zur Definition von Generalisierungen.

In Bezug auf die Standardausdrucksmittel der OMT sind allerdings einige Lükken festzustellen. Sind die fehlenden Möglichkeiten zur Repräsentation beliebiger Multiziplitäten und die fehlende Integration der speziellen Semantik von Aggregationsrelationen noch tolerierbar, so stellt das Fehlen jeglicher Möglichkeit zur Definition von Attributen, und damit auch zur Definition von Einschränkungen

über Attributen, die Einsetzbarkeit von Beschreibungslogiken in dem hier diskutierten Anwendungsbereich grundsätzlich in Frage.

3.4.1. Standardkonstrukte der OMT

Unterstützte Konstrukte

- Klassen

  Klassendefinitionen in der OMT entsprechen terminologischen Axiomen $\mathsf{C} \dot{\sqsubseteq} C$ der TBox.

  Einfachste terminologische Axiome der Form

  - $\mathsf{Glied} \dot{\sqsubseteq} \top$

  entsprechen einer Klassendefinition der Klasse Glied mit der allgemeinsten Klasse als Superklasse. Das terminologische Axiom

  - $\mathsf{Zugmittelglied} \dot{\sqsubseteq} \mathsf{Glied}$

  definiert entsprechend ein Zugmittelglied als eine Spezialisierung (Unterklasse) eines Gliedes. Das Axiom

  - $\mathsf{Privatkunde} \dot{\sqsubseteq} (\mathsf{Kunde} \sqcap \mathsf{Person})$

  definiert einen Privatkunden als Spezialisierung (Unterklasse) einer Person und eines Kunden. Mehrfachvererbung ist also über den Konjunktionsoperator darstellbar.

- Instanzen

  Instanzen objektorientierter Repräsentationen entsprechen der Menge der Assertionen, die über ein Objekt o in der ABox gemacht werden.

  So beschreibt die Menge der Assertionen

  - Glied1 : Zugmittelglied
  - Glied2 : Zugmittelglied
  - Paar1 : KinematischesPaar

  drei Instanzen: zwei Zugmittelglieder und ein kinematisches Paar.

  Die zusätzliche Assertionen

  - $\langle$Paar1, Glied1$\rangle$: glied1
  - $\langle$Paar1, Glied2$\rangle$: glied2

zeichnen die beiden Glieder als die zu dem kinematischen Paar gehörenden Glieder aus, setzen also die verschiedenen Instanzen miteinander in Beziehung.

- Assoziationen

  Assoziationen werden durch Rollen und funktionale Rollen beschrieben. Sowohl Rollen als auch funktionale Rollen beschreiben Beziehungen zwischen Klassen bzw. deren Objekten.

  Die im obigen Beispiel verwendeten Assoziationen glied1 und glied2 werden durch die folgenden funktionalen Rollen definiert:

  - $\mathsf{glied1} \;\dot{\sqsubseteq}\; \top \mapsto \top$
  - $\mathsf{glied2} \;\dot{\sqsubseteq}\; \top \mapsto \top$

- Aggregationen (eingeschränkt)

  Aggregationen werden insofern unterstützt, als daß sie Assoziationen sind, und somit als Assoziationen repräsentiert werden können. Die in der OMT angesprochene aber nicht exakt spezifizierte spezielle Semantik von Aggregationen wird nicht realisiert.

- Multiziplitäten (eingeschränkt)

  Die Unterscheidung zwischen funktionalen Rollen und Rollen in Verbindung mit Existenz- und Wertebereichsbeschränkung erlaubt die eingeschränkte Darstellung von Multiziplitäten:

  - $\exists\ \mathsf{f}\ .\ C$ definiert eine Multiziplität von genau eins.
  - $\forall\ \mathsf{f}\ .\ C$ definiert eine Multiziplität von null oder eins.
  - $\exists\ \mathsf{R}\ .\ C$ definiert eine Multiziplität von 1 oder mehr.

  Ein kinematisches Paar läßt sich auf Basis dieser Multiziplitäten durch das folgende terminologische Axiom definieren:

  - $\mathsf{KinematischesPaar} \;\dot{\sqsubseteq}\; (\top \sqcap \exists\ \mathsf{glied1}\ .\ \mathsf{Glied} \sqcap \exists\ \mathsf{glied2}\ .\ \mathsf{Glied})$

  Es beschreibt ein kinematisches Paar als Klasse mit 2 Assoziationen der Multiziplität 1.

3.4.2. Unterstützte Konstrukte der erweiterten Notation

Im Gegensatz zu den Standardkonstrukten der OMT lassen sich alle in Abschnitt 2.5 vorgestellten und geforderten Erweiterungen der OMT mit $\mathcal{ALCF}$ realisieren:

■ Flexiblere Spezifikation von Instantiierungen

Die flexiblere Beschreibung von Instantiierungen, d.h. die Beschreibung der Zugehörigkeit einer Instanz zu mehreren Klassen, ist durch multiple Zugehörigkeitsassertionen in der Form $\mathsf{o} : C, \quad \mathsf{o} : D$ möglich.

Damit wird es z.B. im Leitbeispiel möglich, ein defektes Rotationsgliederpaar zu beschreiben, obwohl die Klasse »defektes Rotationsgliederpaar« nicht definiert ist.

Die genaue Art des Defektes ist noch nicht bekannt. Herauszufinden welcher Fehler vorliegt, ist gerade Inhalt des Diagnoseprozesses. Erst durch spätere Messungen bzw. Untersuchungen wird sich herausstellen, ob es sich um ein durchrutschendes oder ein gebrochenes Paar handelt bzw. ob ein neuartiger Fehler aufgetreten ist.

Die Beschreibung eines defekten Rotationsgliederpaares ist einfach durch die beiden folgenden Assertionen möglich:

- Paar1 : Rotationsgliederpaar
- Paar1 : Fehlverhalten

■ Spezialisierung über Wertebereichseinschränkungen von Assoziationen

Die Spezialisierung von Klassen über Wertebereichseinschränkungen von Assoziationen ist offensichtlich über die Wertebereichseinschränkung möglich.

Sei z.B. $\mathsf{D} \doteq \forall\, \mathrm{R} \,.\, \mathrm{A}$ und $\mathsf{B} \sqsubseteq \mathsf{A}$, dann definiert $\mathsf{C} \doteq (\mathsf{D} \sqcap \forall\, \mathrm{R} \,.\, \mathrm{B})$ eine Spezialisierung über eine Wertebereichseinschränkung über der Rolle R.

So definiert man z.B. ein Räderpaar als ein spezielles Rotationsgliederpaar über die Einschränkung der möglichen Werte für die funktionalen Rollen glied1 und glied2:

- $\mathsf{Raederpaar} \doteq (\mathsf{Rotationsgliederpaar} \sqcap \forall\, \mathsf{glied1} \,.\, \mathsf{Rad} \sqcap \forall\, \mathsf{glied2} \,.\, \mathsf{Rad})$

Wird später bekannt, daß beide Glieder des Paares Paar1 Räder sind:

- Glied1 : Rad
- Glied2 : Rad

so kann aufgrund der Definition des Begriffes Räderpaar und der zugrundeliegenden modelltheoretischen Semantik geschlossen werden, daß Paar1 ein spezielles Rotationsgliederpaar, ein Räderpaar, sein muß.

Ableitungen dieser Art sollen und müssen im Idealfalle automatisch und vollständig erfolgen, d.h. alle möglichen Ableitungen dieser Art werden automatisch gezogen.

Genau dies ist das Ziel, das von Beschreibungslogiken verfolgt wird, und das Beschreibungslogiken für Second-Level-Support-Systeme so interessant macht. Kleinste Informationsschnipsel können in einer beliebige Reihenfolge zusammengetragen werden. Durch die vollständigen Algorithmen werden immer die maximal möglichen Schlüsse aus der verfügbaren Informationsmenge gezogen.

- Vollständige und unvollständige Klassenbeschreibungen

  Klassenbeschreibungen werden durch terminologische Axiome realisiert. Unvollständige Klassenbeschreibungen entsprechen primitiven Konzepten, vollständige Klassenbeschreibungen definierten Konzepten.

- Beschreibung von Generalisierungen

  Die Beschreibung von Generalisierungsbeziehungen ist über die Disjunktion $\mathsf{C} \doteq (\mathsf{D} \sqcup \mathsf{E})$ möglich.

  Die Negation $\mathsf{C} \doteq \neg\mathsf{D}$ erlaubt die implizite Definition von Disjunktionen bzw. die Definition von »Geschwistern«.

  So ist es z.B. möglich, das Normalverhalten eines kinematischen Paares als die Disjunktion der Normalverhalten der verschiedenen Paartypen zu definieren. Im Leitbeispiel also als Normalverhalten eines Rotationsgliederpaares oder als Normalverhalten eines Rotations-Zugmittelpaares.

  Darauf aufbauend kann dann Fehlverhalten als all das definiert werden, was nicht Normalverhalten ist. Spezielle, schon bekannte Fehlverhalten können als Spezialisierungen dieses allgemeinen Fehlverhaltensbegriffes definiert werden. Aber auch andere noch nicht bekannte Fehlverhalten sind durch diese allgemeine auf Negation beruhende Definition von Fehlverhalten definiert.

  - $\mathsf{Normalverhalten} \doteq (\mathsf{IntaktesRotationsgliederpaar} \sqcup \mathsf{IntaktesRotationsZuggmittelpaar})$
  - $\mathsf{Fehlverhalten} \doteq \neg\mathsf{Normalverhalten}$

3.4.3. Fehlende Ausdrucksmittel

$\mathcal{ALCF}$ besitzt die folgenden Einschränkungen in Bezug auf die Ausdrucksmittel der OMT:

- Keine Attribute

  Es gibt in $\mathcal{ALCF}$ keine Möglichkeit, Attribute zu definieren. Dies bedeutet, daß keine Konzepte definiert werden können, die auf Basisdatentypen Bezug nehmen. Im Anwendungsbereich Service und Support sind aber gerade Attribute, insbesondere Attribute über Zahlen, unerläßlich.

- **Keine Möglichkeit zur Repräsentation von Einschränkungen über Attributen**

  Da es schon nicht möglich ist, Attribute über Basisdatentypen zu beschreiben, können natürlich auch keine Einschränkungen über Attributen formuliert werden. Gerade diese stellen aber einen Großteil des vorhandenen und zur Unterstützung von Aufgaben, wie der Simulation und der Diagnose, unerläßlichen Wissens dar.

  So sind im Leitbeispiel alle physikalischen Gesetze und Randbedingungen Einschränkungen über numerischen Attributen. Auch Normal- und Fehlverhalten sind als (Un)gleichungen über numerischen Attributen beschrieben.

- **Aggregationen nur als Assoziationen**

  Aggregationen können im Rahmen von $\mathcal{ALCF}$ nur als Assoziationen behandelt werden. Ihre spezielle Semantik, z.B. die Transitivität, wird nicht realisiert. Dies ist allerdings auch in objektorientierten Repräsentationen der Fall, da in objektorientierten Repräsentationen die Semantik der einzelnen Konstrukte insgesamt nur operativ gegeben ist.

- **Keine beliebigen Multiziplitätsangaben**

  Es ist nicht möglich, beliebige Multiziplitätsangaben zu machen. Im Leitbeispiel spielen solche Angaben eine eher untergeordnete Rolle, sie sind nicht von essentieller Wichtigkeit. Allgemein wäre es allerdings wünschenswert, wenn diese Funktionalität zur Verfügung stünde.

3.4.4. Zusammenfassung

Insgesamt läßt sich feststellen, daß $\mathcal{ALCF}$ einen Großteil der Anforderungen an eine Wissensrepräsentation für die zweite Support Ebene erfüllt.

Basisdatentypen sind essentiell

Von den fehlenden Ausdrucksmitteln ist die Möglichkeit zur Repräsentation von Attributen und Einschränkungen über Attributen im hier betrachteten Kontext essentiell. Ohne die Möglichkeit, Attribute und Einschränkungen über Attributen repräsentieren zu können, sind Beschreibungslogiken in einem technischen Anwendungsbereich uninteressant.

Weitere Erweiterungen: die $\mathcal{AL}$-Sprachfamilie

Die Integration von Basisdatentypen war in den ursprünglich von Beschreibungslogiken unterstützten Anwendungsbereichen, z.B. natürlichsprachlicher Systeme, nicht so wichtig. In diesen Anwendungsbereichen waren eher beliebige Multiziplitätsangaben und andere Erweiterungen von Vorrang.

Insgesamt gibt es mittlerweile zu viele Erweiterungen von Beschreibungslogiken, als daß sie alle in dieser Arbeit vorstellt werden können. Daher soll an dieser Stelle (im übernächsten Abschnitt) detailliert nur auf die Erweiterung um Basisdatentypen eingegangen werden.

Weitere, relevante Erweiterungen werden im nächsten Abschnitt überblicksartig vorgestellt.

3.5. Die $\mathcal{AL}$-Sprachfamilie

Die Ideen von Schmidt-Schauß und Smolka wurden dazu genutzt, um Algorithmen für eine Vielzahl von Beschreibungssprachen und Konstruktoren zu entwickeln und in vielen Fällen ihre Optimalität in Bezug auf die Worst-Case Komplexität zu zeigen.

Auch die Sprache $\mathcal{ALCF}$ stellt bereits eine Erweiterung von $\mathcal{ALC}$ um Elemente aus der Featurelogik [AK84] wie funktionale Rollen (Features) und Agreements bzw. Disagreements über funktionalen Rollen dar.

Im Laufe der Zeit hat sich ein Basiselement für die Inklusionshierarchie der Beschreibungssprachen herauskristallisiert. Diese Basissprache, $\mathcal{AL}$, stellt sozusagen eine minimale Beschreibungssprache dar, die bereitgestellt werden muß, um sinnvolle Beschreibungen zu formulieren.

Diese Sprache besitzt im Vergleich mit $\mathcal{ALCF}$ keine Konstrukte, die im Zusammenhang mit Features stehen. Weiterhin ist die Ausdrucksmächtigkeit der Negation und der Existenzbeschränkung eingeschränkt, und die Disjunktion fehlt völlig.

Die in Bezug auf $\mathcal{AL}$ neu eingeführten Konstruktoren bzw. Konstruktorengruppen werden oft mit einem Namen (einem Buchstaben aus dem Alphabet) belegt, so daß man abkürzend von der Sprache $\mathcal{AL}$XXX sprechen kann [8].

Ein wichtiges Element der Sprachfamilie ist die Sprache $\mathcal{ALC}$. Dabei steht das $\mathcal{C}$ in $\mathcal{ALC}$ für die volle Negation, d.h. die Negation von beliebigen Konzepttermen [9]. Die Syntaxregeln für $\mathcal{ALC}$ und $\mathcal{AL}$ sind damit:

- $\mathcal{AL}$: $C, D \rightarrow A \mid \top \mid \bot \mid \neg A \mid (C \sqcap D) \mid \exists R.\top \mid \forall R.C$
- $\mathcal{ALC}$: $C, D \rightarrow \top \mid \bot \mid (C \sqcap D) \mid (C \sqcup D) \mid \neg C \mid \exists R.\top \mid \forall R.C$

Weitere untersuchte Erweiterungen umfassen die sogenannten »Number Restrictions« also Anzahlbeschränkungen ($\mathcal{N}$) [Nut94] und qualifizierte Anzahlbeschränkungen ($\mathcal{Q}$) [Hol94], mit der folgenden Syntax und Semantik [10]:

8. Dies erleichtert Insidern die Kommunikation, erschwert allerdings Neueinsteigern, die diesen »Jargon« noch nicht beherrschen, den Einstieg erheblich. So ist es den meisten Lesern nicht auf Anhieb ersichtlich welche Ausdrucksstärke sich hinter $\mathcal{ALCQHI}_{\mathcal{R}^+}$ verbirgt.

9. Die Erweiterungen von $\mathcal{AL}$ um die Disjunktion wird mit dem Buchstaben $\mathcal{U}$, die Erweiterung um die volle Existenzbeschränkung mit dem Buchstaben $\mathcal{E}$ bezeichnet. Da sich das volle Komplement durch volle Existenzbeschränkung und Disjunktion beschreiben läßt (und umgekehrt), wird $\mathcal{UE}$ im allgemeinen durch $\mathcal{C}$ ersetzt. Verkürzend wird von einigen Autoren für $\mathcal{ALC}_{\mathcal{R}^+}$ (mit R^+ der Operator für transitive Rollen(s. u.)) auch der Buchstabe $\mathcal{S}$ verwendet.

10. Hinzu kommen u.a. Untersuchungen in Bezug auf generelle Inklusionsaxiome (GCI), in denen nicht nur terminologische Axiome benutzt werden können, um primitive Konzeptnamen mit Konzepttermen in Beziehung zu setzen, sondern mit denen Beziehungen zwischen zwei beliebigen Konzepttermen beschrieben werden können, sowie Erweiterungen um modallogische Operatoren und symbolische Anzahlbeschränkungen sowie Aufzählungstypen.

- Anzahlbeschränkungen; $\mathcal{N}$
  $\leq_{\mathsf{n}}\ \mathsf{R};\quad \geq_{\mathsf{n}}\ \mathsf{R};\quad =_{\mathsf{n}}\ \mathsf{R}$ (At-Least-, At-Most-, Exactly-Restriktion)
- Qualifizierte Anzahlbeschränkungen: $\mathcal{Q}$
  $\leq_{\mathsf{n}}\ \mathsf{R} : \mathsf{C};\quad \geq_{\mathsf{n}}\ \mathsf{R} : \mathsf{C};\quad =_{\mathsf{n}}\ \mathsf{R} : \mathsf{C}$ (Qualifizierte At-Least-, At-Most-, Exactly-Restriktion)

Weitere Untersuchungen betrafen die Erhöhung der Komplexität der Rollenterme. Von diesen sollen ebenfalls nur die wichtigsten vorgestellt werden. Neben der bereits vorgestellten und in $\mathcal{ALCF}$ vorhandenen Komposition von Rollen und Attributen sind dies insbesondere die Konjunktion ($\mathcal{R}$) und die Disjunktion von Rollen, und die Bildung von Inversen ($\mathcal{I}$) und Rollenhierarchien ($\mathcal{H}$). Der Operator $(_{R^+})$, erlaubt es, die transitive Hülle von Rollen zu bilden, $\mathcal{N}(\circ, \dots)$ erlaubt komplexe Rollen in Anzahlbeschränkungen.

Weitere Details zu den verschiedenen bislang untersuchten Sprachen der AL Sprachfamilie können in Kapitel 2: »Basic Description Logics« [BN03] und Kapitel 5 : »Expressive Description Logics« [CDG03] des »Description Logic Handbook« nachgelesen werden, Kapitel 3: »Complexity of reasoning« [Don03] des gleichen Buches enthält eine Zusammenfassung der verschiedenen Komplexitätsergebnisse.

3.6. Erweiterung um Basisdatentypen: $\mathcal{ALCF}(\mathcal{D})$

Das Fehlen jeglicher Möglichkeit zur Repräsentation von Attributen, und damit zur Integration von Basisdatentypen, wie Zahlen und Strings, macht den Einsatz von Beschreibungslogiken in technischen Anwendungsbereichen unmöglich.

Basisdatentypen

Entscheidendes Manko von $\mathcal{ALCF}$ im Vergleich mit den Anforderungen ist das Fehlen jeglicher Möglichkeit zur Beschreibung von Aussagen über Basisdatentypen. $\mathcal{ALCF}$ erlaubt wie die meisten anderen Beschreibungssprachen nur Aussagen über einem abstrakten Gegenstandsbereich. Dies bedeutet, daß zwar Objekte und Beziehungen zwischen Objekten wie Spezialisierung, Generalisierung und Assoziationen beschrieben werden können, es aber nicht möglich ist, Attribute über Basisdatentypen zu definieren. Damit ist es nicht möglich, Daten für diese Attribute zu speichern, bzw. Wissen über Zusammenhänge zwischen Attributen zu beschreiben.

Ein beschreibungslogisches System, das dazu dienen soll, bestehende Einschränkungen objektorientierter Systeme zu überwinden, darf dies nicht durch die Einführung neuer, anderer Einschränkungen erreichen.

Weiterhin muß sichergestellt werden, daß diese Datentypen transparent in die Repräsentationsformalismen und Inferenzen der Beschreibungslogik einbezogen werden, und keine der gewünschten Eigenschaften der Beschreibungslogik, insbesondere die Korrektheit und Vollständigkeit der Inferenzverfahren verlorengehen.

3.6.1. Erster Ansatz: »Hostdatentypen«

Der Bedarf an Erweiterungen von Beschreibungslogiken um Basisdatentypen wurde von den Entwicklern der meisten beschreibungslogischen Systeme ebenfalls erkannt und führte zu einer Reihe von ad hoc Erweiterungen um sogenannte »Hostdatentypen«.

Diese Erweiterungen haben allerdings den Nachteil, daß jedes beschreibungslogische System einen anderen Satz von Hostdatentypen realisierte, und die Integration in das beschreibungslogische Kernsystem sehr spezifisch für den jeweiligen Satz an Hostdatentypen ist. In vielen Systemen wird innerhalb der Hostdatentypen zwischen zwei verschiedenen Mengen von Operatoren über dem jeweiligen Basistyp unterschieden:

- Ausdrucksschwache Operatoren mit Berücksichtung bei den Inferenzen,
- Ausdrucksstarke BlackBoxes ohne Berücksichtigung bei den Inferenzen.

Inferenzen berücksichtigen nur ausdrucksschwache Operatoren

In den meisten Systemen gibt es einen sehr eingeschränkten Satz von Operatoren, zumeist Vergleiche mit Konstanten des Basisdatentyps. Ausdrücke auf Basis dieser Operatoren werden in die Inferenzen des beschreibungslogischen Kernes, etwa die Subsumption einbezogen.

Diese Operatoren sind aber viel zu ausdrucksschwach, um das zugrundeliegende Hintergrundwissen zu repräsentieren. Damit sind solche Systeme für die Realisierung von Second-Level-Support-Systemen ungeeignet.

Black Box Mechanismen ungeeignet

Andererseits stellen die Systeme einen sehr allgemeinen Mechanismus, zumeist beliebige Funktionen in der Programmiersprache, die dem beschreibungslogischen System zugrundeliegt, zur Verfügung. Ausdrücke in einer solch allgemeinen Form können aus offensichtlichen Gründen nicht in die Inferenzmechanismen einbezogen werden.

In strukturorientierten Systemen kann eine Integration in gewissem Maß noch im Rahmen der ABox Inferenzen realisiert werden, indem die oben beschriebenen Funktionen als Testfunktionen aufgefasst und im Rahmen des Instanztestes überprüft werden können. Allerdings können diese Funktionen nicht zur Ableitung neuer Informationen, z.B. zur weiteren Einschränkung des Wertebereiches eines Attributes benutzt werden. Im Rahmen des Subsumptionstestes werden diese Ausdrücke als „Black Boxes“ behandelt und nicht berücksichtigt.

Dies führt dazu, daß die berechneten Subsumptionshierarchien falsch sein können bzw. überhaupt nicht berechnet werden können. Implizit muß jedes Konzept, daß eine solche Testfunktion beinhaltet als primitives Konzept behandelt werden.

In beschreibungslogischen Systemen, die auf dem Tableauxkalkül basieren, ist eine solche Zweiteilung prinzipiell nicht möglich. Alle bereitgestellten Operatoren müssen in allen Inferenzen berücksichtigt werden. Unverzichtbares Kernelement ist aber die korrekte Klassifikation von Instanzen und Konzepten. Damit sind »Black Box« Ansätze ebenfalls ungeeignet.

3.6.2. Konkrete Gegenstandsbereiche

Damit sind die von solchen ad hoc Ansätzen zur Verfügung gestellten Mechanismen zur Integration von Basisdatentypen in Beschreibungslogiken ungeeignet. Die Ausdrucksstärke der Operatoren, die bei den Inferenzen berücksichtigt werden, ist viel zu gering, um dem Repräsentationsbedarf zu genügen. Die Ausdrucksmittel, die es erlauben, das vorhandene Hintergrundwissen zu beschreiben, finden keine ausreichende Berücksichtigung in den Inferenzen.

Baader und Hanschke [BH91b, Han96] suchten im Gegensatz zu diesen ad hoc Ansätzen eine allgemeine Lösung des Problems der Erweiterung um Basisdatentypen. Die angestrebte Erweiterung eines beschreibungslogischen Systems sollte die folgenden Eigenschaften haben:

- Formale, deklarative Semantik

  Sie sollte weiterhin eine formale, deklarative Semantik besitzen, wobei die normale Semantik von Konzeptsprachen soweit wie möglich beibehalten werden sollte.

- Bereitstellung eines Schemas statt Entwicklung von ad hoc Lösungen

  Es soll ein Schema zur Erweiterung von Konzeptsprachen um verschiedene konkrete Bereiche bereitgestellt werden anstatt ad hoc Erweiterungen für einen speziellen konkreten Bereich zu realisieren.

- Realisierung der Erweiterung durch Kombination existierender Algorithmen

  Die Realisierung der Inferenzen der Erweiterung soll durch die Kombination existierender Inferenzalgorithmen für Konzeptsprachen mit Algorithmen des konkreten Bereiches möglich sein.

- Spezifikation der Semantik auf Schemaebene

  Die Spezifikation der formalen Semantik der Erweiterung soll bereits auf der Ebene des Schemas möglich sein und nicht für jeden konkreten Gegenstandsbereich neu erfolgen. Dadurch wird auch die Kohärenz der Semantik über alle realisierten konkreten Gegenstandsbereiche gesichert.

- Einfache Schnittstelle zwischen abstraktem und konkretem Gegenstandsbereich

  Das Interface zwischen Konzeptsprache und konkretem Bereich soll einfach sein.

Mit den von ihnen in [BH91b] und [Han96] vorgestellten *zulässigen konkreten Gegenstandsbereiche* entwickelten sie ein solches Schema zur Erweiterung von Konzeptsprachen um Basisdatentypen, welches im folgenden vorgestellt werden soll.

Definition 3.21 (Konkreter Gegenstandsbereich)
Ein *konkreter Bereich* $\mathbf{D}$ ist eine Relationalstruktur $\mathbf{D}=\langle\Delta_\mathbf{D}, \{p_j \mid j \in J_D\}\rangle$, bestehend aus der Domäne, einer Menge $\Delta_\mathbf{D}$ und einer Menge von Prädikatsnamen $\mathsf{N}_D = \{p_j \mid j \in J\}$. Mit jedem $p \in \mathsf{N}_D$ ist eine Stelligkeit n und ein n_p-stelliges Prädikat $P^\mathbf{D} \subseteq (\Delta_\mathbf{D})^{n_p}$ assoziiert.

Nicht jeder konkrete Bereich erfüllt allerdings die obigen Kriterien. Insbesondere um die Erweiterung in die Inferenzen integrieren zu können ohne die Eigenschaften der Vollständigkeit und Korrektheit zu verlieren, müssen zusätzliche Bedingungen erfüllt sein. Ein konkreter Gegenstandsbereich der die Bedingungen erfüllt heißt *zulässig*.

Die folgenden Definitionen geben beschreiben die zusätzlichen Bedingungen:

Definition 3.22 (Abgeschlossenheit unter Negation)
Ein konkreter Bereich heißt *abgeschlossen unter Negation*, falls für jedes Prädikat auch dessen Negation im Bereich enthalten ist. D.h.

$$\forall p \in \mathsf{N}_D.\exists q \in \mathsf{N}_D.q = (\Delta_\mathbf{D})^{n_p} \setminus p$$

Definition 3.23 (Erfüllbarkeit endlicher Konjunktionen)
Seien $P_1, \ldots, P_k \in \mathsf{N}_D$ Prädikate mit Stelligkeiten $n_1, \ldots, n_k$. Die *endliche Konjunktion* K von Prädikaten

$$K = \bigwedge_{i=1}^{k} p_i(\vec{x}_i) = p_1(x_{11}, \ldots, x_{1n_1}) \wedge \cdots \wedge p_k(x_{1k}, \ldots, x_{1n_k})$$

heißt *erfüllbar*, falls eine Belegung der Variablen existiert, so daß K wahr ist.

Definition 3.24 (Zulässiger Konkreter Bereich)
Ein konkreter Bereich heißt *zulässig* gdw.

(i) abgeschlossen unter Negation ist;

(ii) einen Namen $\top$ für $\Delta_\mathbf{D}$ enthält, d.h.:

$$\exists \top \in \mathsf{N}_D.\forall d \in \Delta_\mathbf{D}.(d \in \top^\mathbf{D})$$

(iii) die Erfüllbarkeit endlicher Konjunktionen entscheidbar ist.

Syntax

Konkrete Gegenstandsbereiche werden über zusätzliche Operatoren zur Bildung von Konzepttermen in Beschreibungslogiken integriert. Hanschke führt in [Han96] dazu die generalisierte Wertebereichs- und Existenzbeschränkung ein, mit der er

die verschiedensten Formen der Wertebereichseinschränkung, sowohl über dem abstrakten, als auch dem konkreten Gegenstandsbereich formulieren kann ([Han96], Seite 26).

Aus Gründen der Übersichtlichkeit und der besseren Unterscheidung zwischen abstraktem und konkretem Gegenstandsbereich sollen an dieser Stelle zusätzlich zur Wertebereichs- und Existenzeinschränkung, die über dem abstrakten Gegenstandsbereich operieren, zwei weitere Operatoren für den konkreten Gegenstandsbereich eingeführt werden. Diese Operatoren quantifizieren konkrete Prädikate existentiell bzw. universell, und dürfen daher auch nur auf solche angewandt werden.

Die ABox wird ebenfalls um einen Operator erweitert. Dieser erlaubt es, Objekte der ABox auch durch konkrete Prädikate zueinander in Beziehung zu setzen.

Definition 3.25 (Syntax (Operatoren für konkrete Gegenstandsbereiche))
Sei P ein n-stelliges konkretes Prädikat, und seien $R_1 \dots R_n$ Rollen bzw. Attributketten. Dann sind

- $\exists_{\Delta}\, R_1 \dots R_n.P$ (Existentielles Constraint)
- $\forall_{\Delta} R_1 \dots R_n.P$ (Allgemeines Constraint)

Konzeptterme.

Seien $\mathsf{o_1}, \mathsf{o_n} \in \mathsf{O}$ Objekte der ABox, und P_{Δ_D} ein konkretes Prädikat. Dann ist

- $\langle \mathsf{o_1}, \dots, \mathsf{o_n} \rangle : P$ (Konkrete Beziehung)

eine ABox Assertion.

Bemerkung 3.4
Die obige Definition konkreter Beziehungen erlaubt keine direkte Definition von konkreten Attributwerten. So werden zur Festlegung des Radius eines Rotationsgliedes auf den Wert 10 zunächst ein konkretes Prädikat $= \mathsf{10}$ mit der Extension $\{10\}$ benötigt.

Darauf aufbauend wird der Zusammenhang zwischen Glied und Radius durch eine Beziehungsassertion und eine konkrete Beziehungsassertion beschrieben:

- $\langle$Glied1, radius1$\rangle$: radius,
- $\langle$radius1$\rangle$: $= \mathsf{10}$.

Zur vereinfachten Notation dieses häufig vorkommenden Musters wird die folgende, abkürzende Schreibweise eingeführt. Sei o ein Objekt, v ein Wert des konkretes Gegenstandbereiches und f ein Attribut (eine Attributkette). Dann beschreibt

- $\langle\langle \mathsf{o}, \mathsf{v} \rangle\rangle : \mathsf{f}$ (Konkreter Wert)

die Belegung der funktionalen Rolle f des Objektes o mit dem Wert v.

Semantik

Die Einführung der zusätzlichen Operatoren für den konkreten Gegenstandsbereich macht es natürlich notwendig, eine Semantik für diese Operatoren zu definieren.

Darüber hinaus sind aber auch Anpassungen an die Semantik von $\mathcal{ALCF}$ notwendig. Dies betrifft insbesondere den der Interpretation zugrundeliegenden Wertebereich von funktionalen Rollen, der auf den konkreten Gegenstandsbereich ausgeweitet werden muß.

Definition 3.26 (Semantik (Operatoren für konkrete Gegenstandsbereiche))
Sei P ein n-stelliges konkretes Prädikat, und seien $R_1 \ldots R_n$ Rollen bzw. Attributketten. Die Interpretationsfunktion für $\mathcal{ALCF}(\mathcal{D})$ unterscheidet sich von der für $\mathcal{ALCF}$ wie folgt:

Zusätzlich zur Menge Δ_a, des abstrakten Gegenstandbereiches, existiert eine zweite Menge $\Delta_,$ der konkrete Gegenstandsbereich. $\Delta_a$und $\Delta_\mathbf{s}$ind disjunkt $\Delta_a \cap \Delta_=\emptyset$.

Die Interpretationsfunktion $\cdot^\mathcal{I}$wird wie folgt verändert: Mit jedem Rollennamen $\mathsf{R} \in \mathbf{R}$ wird eine binäre Relation $\mathsf{R}^\mathcal{I} \subseteq \Delta_a \times (\Delta_a \cup \Delta_)$ und mit jedem Feature $\mathsf{f} \in \mathbf{F}$ eine partielle Funktion $\mathsf{f}^\mathcal{I} : \Delta_a \mapsto (\Delta_a \cup \Delta_)$ assoziiert.

Die Semantik der konkreten Wertebereichs- und Existenzbeschränkung ist dann wie folgt definiert:

- $\exists_\Delta R_1 \ldots R_n.P \Leftrightarrow \{d \in \Delta_a \mid \exists e_1, \ldots, e_n.(d,e1) \in {R_1}^\mathcal{I} \wedge \ldots \wedge (d,e_n) \in {R_n}^\mathcal{I} \wedge (e_1, \ldots, e_n) \in P^\mathbf{D}\}$
- $\forall_\Delta R_1 \ldots R_n.P \Leftrightarrow \{d \in \Delta_a \mid \exists e_1, \ldots, e_n.(d,e1) \in {R_1}^\mathcal{I} \wedge \ldots \wedge (d,e_n) \in {R_n}^\mathcal{I} \wedge (e_1, \ldots, e_n) \in P^\mathbf{D}\}$

Entscheidbarkeit

Wesentliches Ergebnis der Arbeiten von Baader und Hanschke [BH91b, Han96] ist der folgende Satz:

Satz 3.3 (Entscheidbarkeit von $\mathcal{ALCF}(\mathcal{D})$)
Sei $\mathcal{D}$ ein zulässiger konkreter Bereich. Dann existiert ein korrekter und vollständiger Algorithmus für die Sprache $\mathcal{ALCF}(\mathcal{D})$, der das Konsistenzproblem für eine ABox entscheidet[11].

Damit sind auch die anderen, im vorhergehenden Abschnitt vorgestellten Probleme, wie Subsumptionsproblem, Instantiierung etc. entscheidbar.

11. $\mathcal{ALCF}$ ist die von R zur Verfügung gestellte Beschreibungssprache.

Direkte Summen konkreter Bereiche

Dieses Ergebnis ist nicht auf einen konkreten Bereich beschränkt. Wie das folgende Lemma zeigt, können unter bestimmten Voraussetzungen konkrete Gegenstandsbereiche kombiniert werden.

Definition 3.27 (Direkte Summe von Konkreten Bereichen)
Seien $\mathbf{D}$ und $\mathbf{E}$ zulässige konkrete Gegenstandsbereiche mit $\mathsf{N}_D = P_{1,1}, \ldots, P_{1,n_1}$ und $\mathsf{N}_E = P_{2,1}, \ldots, P_{2,n_2}$ und $\Delta_{\mathbf{D}} \cap \Delta_{\mathbf{E}} = \emptyset$.

Sei $\mathbf{D} \oplus \mathbf{E}$ wie folgt definiert:

$$\begin{aligned}\Delta_{\mathbf{D}\oplus\mathbf{E}} &= \Delta_{\mathbf{D}} \cup \Delta_{\mathbf{E}} \\ \mathsf{N}_{D\oplus E} &= \{Q_{1,1}, \ldots, Q_{1,n_1}, \widehat{Q}_{1,1}, \ldots, \widehat{Q}_{1,n_1}, \\ &\qquad Q_{2,1}, \ldots, Q_{2,n_2}, \widehat{Q}_{2,1}, \ldots, \widehat{Q}_{2,n_2}\}\end{aligned}$$

mit $(x_1, \ldots, x_{n_j}) \in Q_{\mu,j} \Leftrightarrow (x_1, \ldots, x_{n_j}) \in P_{\mu,j}$ und $(x_1, \ldots, x_{n_j}) \in \widehat{Q}_{\mu,j} \Leftrightarrow (x_1, \ldots, x_{n_j}) \in \overline{P}_{\mu,j} \vee \exists i x_i \in \delta(\mu)$ und $\mu \in \{1,2\}, \delta(1) = \Delta_{\mathbf{E}}, \delta(1) = \Delta_{\mathbf{D}}$.

Es gilt dann das folgende Lemma [BH91b, BH91a]:

Lemma 3.1 (Zulässigkeit von $\mathbf{D} \oplus \mathbf{E}$)
Wenn $\mathbf{D}$ *und* $\mathbf{E}$ *zulässig sind, dann ist* $\mathbf{D} \oplus \mathbf{E}$ *ebenfalls zulässig.*

Die Erweiterung von $\mathcal{ALCF}$ um konkrete Gegenstandsbereiche erlaubt damit die Integration von Basisdatentypen in beschreibungslogische Systeme, die zur Realisierung von Second-Level-Support-Systemen benötigt werden.

Voraussetzung ist allerdings, daß es sich um zulässige konkrete Gegenstandsbereiche handelt. Daher muß geprüft werden, inwieweit die Anforderungen an die Ausdrucksstärke der Repräsentationssprache durch zulässige konkrete Gegenstandsbereiche abgedeckt werden können. Dies ist u.a. Inhalt des nächsten Kapitels der Arbeit.

Komplexität

Obwohl die Möglichkeit zur Erweiterung von Beschreibungslogiken um zulässige konkrete Gegenstandsbereiche bereits seit geraumer Zeit bekannt ist, wurde lange Zeit die Komplexität des bzw. der resultierenden Beschreibungslogiken nicht näher untersucht, der Beweis der Entscheidbarkeit war ausreichend.

Erst in jüngster Zeit wurde eine umfassende Untersuchung der Komplexität von $\mathcal{ALCF}(\mathcal{D})$ durch Lutz [Lut02c, Lut01a] durchgeführt [12].

12. Tatsächlich untersucht Lutz die Beschreibungssprachen $\mathcal{ALC}(\mathcal{D})$ und $\mathcal{ALCF}$ sowie diverse Erweiterungen und Kombinationen der beiden Sprachen. Genau betrachtet unterscheidet sich die Definition von Lutz für $\mathcal{ALC}(\mathcal{D})$ minimal von der in dieser Arbeit verwendeten, in dem er zwischen abstrakten und konkreten Feature unterscheidet. Der Unterschied in der Ausdrucksstärke kann aber insbesondere für Komplexitätsbetrachtungen vernachlässigt werden.

An dieser Stelle sollen die wesentlichen Ergebnisse bzgl. $\mathcal{ALCF}(\mathcal{D})$ zusammengefasst werden:

- Die Erfüllbarkeit von $\mathcal{ALCF}(\mathcal{D})$-Konzepten, und damit auch die Subsumption von $\mathcal{ALCF}(\mathcal{D})$-Konzepten ist PSPACE-vollständig.
- Ein auf der Tracing-Technik basierender Tableaux-Algorithmus zur Entscheidung der Erfüllbarkeit von $\mathcal{ALCF}(\mathcal{D})$-Konzepten etabliert PSPACE als obere Komplexitätsschranke unter den folgenden Voraussetzungen:
  - Die Komplexität der Entscheidungsprozedur für die Erfüllbarkeit endlicher Konjunktionen liegt in PSPACE.
  - Der konkrete Gegenstandsbereich ist zulässig.
- Die PSPACE-Vollständigkeit folgt aus der Tatsache, daß, wie von Schmidt-Schauss und Smolka gezeigt [SSS91], bereits die Erfüllbarkeit von $\mathcal{ALC}$-Konzepten PSPACE-hart ist.
- Ein auf der ≫Precompletion≪-Technik basierender Algorithmus etabliert PSPACE als obere Komplexitätsschranke zur Entscheidung der Konsistenz einer ABox (unter den gleichen Voraussetzungen wie oben).
- Die für $\mathcal{ALCF}(\mathcal{D})$ erzielten Ergebnisse gelten unverändert auch für $\mathcal{ALC}(\mathcal{D})$.

Der Einsatz eines indirekten Algorithmus der die Konsistenz der ABox auf die Erfüllbarkeit von Konzepten reduziert hat die in Abschnitt 3.3.3 beschriebenen Auswirkungen. Auch vor diesem Hintergrund ist das folgende Zitat aus [Lut02c] zu sehen:

> ≫However, for the implementation of DL reasoners that can decide ABox consistency, it may be more appropriate to use a "direct" ABox consistency algorithm instead of reducing this reasoning task to concept satisfiability. Considering the two algorithms developed in this paper, it should be straightforward to devise such a direct algorithm.≪

Leider wird kein direkter Algorithmus präsentiert und es bleibt unklar, ob ein solcher Algorithmus ebenfalls inPSPACE liegt, oder ob, ähnlich wie in [BFT95] ein im worst-case theoretisch schlechterer Algorithmus wegen seiner softwaretechnischen Eigenschaften wie einer höheren Modularität vorgezogen werden sollte.

Komplexität bzgl. TBoxen Die Diskussion der Komplexität der Beschreibungslogiken relativiert sich zudem durch die folgenden, ebenfalls von Lutz gezeigten Ergebnisse für den Fall, daß eine TBox in die Betrachtungen mit einbezogen wird:

- Die Komplexität der Entscheidungsprozedur für die Erfüllbarkeit von $\mathcal{ALCF}$-Konzepten bzgl. einer TBox T liegt in NExpTime-vollständig.
- Die Komplexität der Entscheidungsprozedur für die Erfüllbarkeit von $\mathcal{ALC}(\mathcal{D})$-Konzepten bzgl. einer TBox T liegt für jeden arithmetischen Gegenstandsbereich in NExpTime-hart.

  Ein Gegenstandsbereich ist arithmetisch [Lut01a], wenn er die natürlichen Zahlen enthält und unäre Prädikate für die Gleichheit und Ungleichheit mit der Null, ein binäre Prädikate für Gleichheit und Ungleichheit sowie ein binäres Prädikat für die Addition mit 1 und ein ternäres Prädikate für die Addition enthält.
- Eine obere Schranke kann wie folgt formuliert werden: Die Erfüllbarkeit von $\mathcal{ALC}(\mathcal{D})$-Konzepten bzgl. einer TBox T liegt in NExpTime falls $\mathcal{D}$ zulässig ist, und die Entscheidungsprozedur für $\mathcal{D}$ in NP liegt.

Auf den Einsatz von TBoxen zur Strukturierung des Wissens kann in dem in dieser Arbeit betrachteten Einsatzszenario auf keinen Fall verzichtet werden.

Andere Erweiterungen Lutz untersuchte neben TBoxen weitere Möglichkeiten zur Erweiterung von $\mathcal{ALCF}$ und $\mathcal{ALC}(\mathcal{D})$, u.a. Qualifizierte Anzahlbeschränkungen, Generalisierte TBoxen, Rollenkonjunktionen, transitive Rollen und inverse Rollen [Lut02b, Lut02c, Lut01b, Lut01a, Lut00, Lut99]. Die wesentlichen Ergebnisse sind:

- Die PSpace-Komplexität von $\mathcal{ALC}(\mathcal{D})$ ist nicht robust gegenüber Erweiterungen, da für eine Vielzahl von zunächst harmlos erscheinenden Erweiterungen die Komplexität von PSpace-Vollständigkeit auf NExpTime-Vollständigkeit ansteigt, in einigen Fällen die Probleme sogar unentscheidbar werden.

  Beispiele für solche Erweiterungen sind Rollenkonjunktionen und inverse Rollen – hier ist die Erfüllbarkeit von Konzepten NExpTime-hart für jeden arithmetischen konkreten Gegenstandsbereich.
- Die meisten der Erweiterungen, die die Komplexität nicht über PSpace ansteigen lassen sind derart, daß es keine Interaktionen zwischen der Erweiterung (die natürlich ebenfalls in PSpace liegen muß) und $\mathcal{ALCF}(\mathcal{D})$ gibt.

  Beispiele für solche Erweiterungen sind die Erweiterung um qualifizierte Anzahlbeschränkungen ($\mathcal{Q}$) oder transitive Rollen ($_{\mathcal{R}^+}$). Allerdings wird in [Lut01a] diese Tatsache nicht formal bewiesen.

3.7. Zusammenfassung

Beschreibungslogiken sind ein Wissensrepräsentationsformalismus, der die Vorteile der objektorientierten Repräsentation der Frames mit der formalen Semantik

der Prädikatenlogik vereint.

Sie stellen einen semantisch klaren Ähnlichkeitsbegriff – die Subsumption – für beliebig komplexe Strukturen zur Verfügung, der zugleich die Basis für die verschiedenen Retrievalverfahren und die automatische semantische Indexierung bildet.

Ein Vergleich der Ausdrucks $\mathcal{ALCF}$ mit den im letzten Kapitel herausgearbeiteten Anforderungen eines Second-Level-support-Systems ergibt, daß $\mathcal{ALCF}$ einen Großteil der Anforderungen erfüllt.

Von den fehlenden Ausdrucksmitteln ist die Möglichkeit zur Repräsentation von Attributen und Einschränkungen über Attributen im Kontext der Arbeit essentiell. Ohne die Möglichkeit, insbesondere numerische Attribute und Einschränkungen über Attributen repräsentieren zu können, sind Beschreibungslogiken in einem technischen Anwendungsbereich uninteressant.

Baader und Hanschke [BH91a, Han96] haben mit ihren zulässigen konkreten Gegenstandsbereichen – im wesentlichen Relationalstrukturen, die eine Reihe zusätzlicher Kriterien erfüllen – die Grundlage für Integration primitiver Datentypen, z.B. numerischer Attribute, gelegt.

Die Vorstellung der Architektur eines beschreibungslogischen Systems, daß auf diesen Ideen aufbaut sowie eine systematische Betrachtung zulässiger numerischer konkreter Gegenstandsbereiche ist Gegenstand des nächsten Kapitels. Auch dort liegt der Schwerpunkt auf der Bereitstellung der Ausdrucksstärke und der Mechanismen, die für die Unterstützung der Arbeit von Mitarbeitern des Second-Level-Supports notwendig sind.

So stellt das beschreibungslogische System u.a. Möglichkeiten zur Anzeige der generierten Modelle zur Verfügung. Wird weiterhin die Anzeige der berechneten Parametereinschränkungen auch von den angebundenen konkreten Gegenstandsbereichen unterstützt, so erlaubt dieser Mechanismus die Verwendung des beschreibungslogischen Systems für das Browsing, die Simulation und die What-If-Analyse sowie der modellbasierten Diagnose.

4 CTL – ein beschreibungslogisches System mit ausdrucksstarken konkreten Gegenstandsbereichen

Im letzten Kapitel wurde festgestellt, daß Beschreibungslogiken ein für die Lösung der Aufgabenstellung dieser Arbeit interessanter Wissensrepräsentationsformalismus sind, insbesondere dann, wenn sie um ausdrucksstarke konkrete Gegenstandsbereichen, und hier besonders numerischen Gegenstandsbereichen, erweitert werden.

Dabei lag der Schwerpunkt der Betrachtung auf den theoretischen Ergebnissen, die konkrete Umsetzung der theoretischen Ergebnisse in beschreibungslogischen Systemen ist Inhalt dieses Kapitels.

Aufbau des Kapitels

Übersicht beschreibungslogische System der »zweiten« Generation

Am Anfang des Kapitels werden in Abschnitt 4.1 einige, zum Zeitpunkt des Beginns dieser Arbeit verfügbare beschreibungslogische Systeme vorgestellt und insbesondere die Ausdrucksstärke ihrer numerischen Gegenstandsbereiche untersucht. Es stellt sich heraus, daß diese Systeme zur Realisierung der Anforderungen insbesondere bei der Implementierung von Basisdatentypen nicht ausdrucksstark genug sind.

Vorstellung CTL

Daher wurde, ausgehend von TAXON [BHHM93] das System CTL [KW96a, KW96b] entwickelt, das es erlaubt, existierende Algorithmen und Systeme zur Realisierung von konkreten Gegenstandsbereichen einzusetzen.

CTL wird in Abschnitt 4.2 vorgestellt. Es implementiert die Beschreibungslogik $\mathcal{ALCF}(\mathcal{D})$ und unterscheidet sich von den untersuchten Systemen insbesondere

durch eine Schnittstelle zur modularen Einbindung externer Entscheidungsverfahren für konkrete Gegenstandsbereiche und die Möglichkeit, sich im Falle eine erfolgreichen Konsistenztest das generierte Modell anzeigen zu lassen.

Die so zur Verfügung gestellten Ausdrucksmittel und Inferenzen erlauben es, die in Kapitel 2 aufgestellten Anforderungen an ein Second-Level-Support-System zu erfüllen.

Numerische konkrete Gegenstandsbereiche

Nachdem mit CTL die Grundlagen für die Integration von konkreten Gegenstandsbereichen gelegt sind, werden in Abschnitt 4.3 zunächst einige grundlegende Ergebnisse zur Entscheidbarkeit und Unentscheidbarkeit aus dem Bereich der Theorien über numerischen Grundkörpern vorgestellt und die Verbindung dieser Ergebnisse zu zulässigen konkreten Gegenstandsbereichen hergestellt.

Im zweiten Teil dieses Abschnittes wird zunächst eine, sich an Theoriefragmenten orientierende, Systematik für numerische konkrete Gegentandsbereiche eingeführt. Anhand dieser wird dann eine Reihe in ihrer Ausdrucks Gegenstandsbereiche, zusammen mit den jeweils verfügbaren Entscheidungsverfahren, vorgestellt.

Vorstellung und Bewertung RACER

CTL wurde aufbauend auf dem System TAXON in den Jahren 1996 und 1997 prototypisch implementiert, seine Weiterentwicklung ruht seitdem. Der Schwerpunkt der Implementierung lag dabei auf der Überprüfung der konzeptionellen Ideen, weniger auf der tatsächlichen praktischen Einsetzbarkeit [1].

Neben einer weiteren Steigerung der Ausdrucksstärke standen bei der Entwicklung der beschreibungslogischen Systemen der »dritten« Generation Aspekte wie Stabilität des Gesamtsystems, Anbindung von Drittsystemen und Optimierung der Performance für den »average case« im Mittelpunkt. Ein herausragender Vertreter der beschreibungslogischen Systeme der dieser Generation ist das seit 1998 ebenfalls an der Universität Hamburg entwickelte, System RACER [HM02c, HM02b]. Zum Abschluß dieses Kapitels wird daher in Abschnitt 4.4 RACER vorgestellt und auf seine Eignung für die in dieser Arbeit behandelte Fragestellung untersucht.

4.1. Beschreibungslogische Systeme der zweiten Generation

Ziel dieser Arbeit war es nicht, ein beschreibungslogisches System von Grund auf neu zu entwickeln, sondern existierende beschreibungslogische Systeme für den Einsatz im Bereich des Second-Level-Supports zu verwenden bzw. weiterzuentwickeln. So steht man vor dem Problem, einen optimalen Punkt im Tradeoff zwischen gewünschter Ausdrucksstärke, Realisierbarkeit (d.h. Entscheidbarkeit bzw.

1. Eine Eigenschaft, die CTL mit den meisten anderen beschreibungslogischen Systemen der »zweiten« Generation gemein hat.

Komplexität der Algorithmen) und Verfügbarkeit (d.h. frei zugänglichen terminologischen Systemen) zu finden.

Daher stand die Evaluation einer Reihe von beschreibungslogischen Systemen im Hinblick auf ihre Eignung für den Second-Level-Support am Beginn dieser Arbeit. Dies bedeutete insbesondere, das neben dem abstrakten Gegenstandsbereich ein besonderes Augenmerk auf die Realisierung des konkreten Gegenstandsbereiches gelegt wurde. Untersucht wurden die folgenden Systeme:

- LOOM [Mac91] [2],
- CLASSIC [BMPS+91, PSMB+] [3],
- KRIS[4] [BH90, KRI92] [5],
- TAXON [6].

Alle untersuchten Systeme waren zu Beginn der Arbeit verfügbar und sind der zweiten Generation von beschreibungslogischen Systemen zuzuordnen. Einen guten Überblick über die Historie beschreibungslogischer Systeme und die verschiedenen verfügbaren Systeme enthält das Kapitel 8: »Description Logic Systems« des »Description Logic Handbooks« [MH03].

4.1.1. Systemarchitektur

In gewisser Weise decken die untersuchten Systeme die Extrempositionen in einem Dreieck möglicher Implementationskriterien ab. KRIS und TAXON realisie-

2. LOOM wurde an der University of Southern California am ISI entwickelt. Ziel der Entwicklung war die Bereitstellung einer Entwicklungsumgebung zur Realisierung von Expertensystemen. Loom verfügt über die mit Abstand mächtigste Beschreibungssprache aller untersuchten Systeme. Es ist möglich, Methoden und Regeln an Konzepte anzubinden und damit spezielle Inferenzen zu realisieren. Die von LOOM eingesetzten Algorithmen sind allerdings auch mit Abstand die unvollständigsten.
3. CLASSIC wurde von den Entwicklern von R bei AT&T entwickelt. Ziel war ein benutzbares Wissensrepräsentationswerkzeug mit bekannten theoretischen Eigenschaften. Eines der vorgesehenen Hauptanwendungsgebiete waren Informationssysteme. Neben der Benutzung einer eingeschränkten Beschreibungssprache mit fast vollständigen und effizienten Algorithmen ist es möglich, einfache Regeln zu formulieren.
4. KRIS – Knowledge Representation and Inference System.
5. KRIS wurde am DFKI Saarbrücken entwickelt, mit dem Ziel, ein Testbett für die Entwicklung und den Test neuartiger Algorithmen zur Klassifikation bereitzustellen. KRIS enthält vollständige und korrekte Algorithmen für eine relativ mächtige Beschreibungssprache. Ursprünglich nur über dem abstrakten Gegenstandsbereich operierend, wurde KRIS später um einen einfachen numerischen konkreten Gegenstandsbereich erweitert.
6. TAXON wurde am DFKI Kaiserslautern entwickelt. Der Schwerpunkt lag ähnlich wie bei KRIS in der Bereitstellung vollständiger Inferenzen allerdings unter Einbindung konkreter Bereiche durch Umsetzung der Ideen aus [BH91b, Han96].

ren vollständige Inferenzen für die jeweils bereitgestellten Beschreibungssprachen. LOOM stellt eine sehr mächtige Beschreibungssprache zur Verfügung, die gezwungenermaßen sehr unvollständig ist. CLASSIC wurde im Hinblick auf eine möglichst effiziente Realisierung und einen möglichst geringen Verbrauch von Ressourcen entwickelt.

Dies spiegelt sich auch in der zugrundeliegenden Basistechnologie wieder. LOOM und CLASSIC benutzen strukturorientierte Algorithmen, während KRIS und TAXON tableaux-basierte Systeme sind. Während KRIS das erste tableaux-basierte System überhaupt darstellte, entstand TAXON aus der Untersuchung der Einsetzbarkeit von Beschreibungslogiken in technischen Anwendungsbereichen und legte den Schwerpunkt auf die Realisierung der daraus erwachsenden Anforderungen, insbesondere auf die Einbindung von konkreten Gegenstandsbereichen.

Im Folgenden soll die Ausdrucks beschreibungslogischen Systeme verglichen werden, und zwar sowohl im abstrakten als auch im konkreten Gegenstandsbereich.

4.1.2. Ausdrucksstärke des abstrakten Gegenstandsbereiches

Basis für den Vergleich der Ausdrucksstärke der abstrakten Gegenstandsbereiche ist die Menge der Konstrukte, die in der KRSS-Spezifikation [PSS93] vorgeschlagen wurde. Ähnliche Vergleiche sind bereits in der Literatur zu finden, daher erfolgt an dieser Stelle nur eine Zusammenfassung der Ausdrucksstärken der einzelnen Systeme in tabellarischer Form.

Es fällt auf, daß CLASSIC praktisch den KRSS-Kern plus die Erweiterung um number-restrictions widerspiegelt. Dies ist nicht weiter verwunderlich, da die Hauptarbeit bei der Definition des Standards von den CLASSIC-Entwicklern geleistet wurde.

KRIS und TAXON stellen vollständige Inferenzen für ihre relativ mächtigen und sich nur in wenigen Punkten unterscheidenden Beschreibungssprachen zur Verfügung. KRIS stellt im Gegensatz zu TAXON volle Anzahlbeschränkungen zur Verfügung, TAXON erlaubt dahingegen die Einbindung konkreter Gegenstandsbereiche und basiert nicht auf der ≫unique name assumption≪.

LOOM hat mit Abstand die mächtigste Beschreibungssprache und verfügt auf dem Papier auch über die ausdrucksstärksten konkreten Gegenstandsbereiche.

Tests zeigten aber, daß das System für die in dieser Arbeit verfolgte Aufgabenstellung ungeeignet war. In seinen Inferenzen ist LOOM so unvollständig, daß eine Klassifikation nur sehr selten, wenn überhaupt durchgeführt wurde [7].

7. So ist LOOM auch dort unvollständig, wo eigentlich vollständige Inferenzen möglich sind. Ein Extrembeispiel für diese Tatsache ist die Konzeptdefinition (:or a (:not a)), die nicht mit thing, dem LOOM Äquivalent zu top identifiziert wird.

KRSS	Core	LOOM	CLASSIC	KRIS	TAXON
$\top$	✔	✓	✓	✓	✓
$\bot$	✔	✓	✓	✓	✓
$(C \sqcap D)$	✔	✔	✔	✔	✔
$(C \sqcup D)$		✔		✔	✔
$\neg C$		✔[b]		✔	✔
∀ R . C	✔	✔	✔	✔	✔
∃ R	✔	✔		✔	✓
↑R		✓		✓	✓
$\geq_n$ R	✔[a]	✔	✔	✔	
$\leq_n$ R	✔[a]	✔	✔	✔	
$=_n$ R	✔[a]	✓	✓	✓	
∃ R . C		✔		✔	✔
$\geq_n$ R : C		✓			
$\leq_n$ R : C		✓			
$=_n$ R : C		✓			
$R_1 = R_2$	✔[c]	✔	✓[c]	✔[c]	✔[c]
$R_1 \neq R_2$		✓		✔[c]	✔[c]

✔ explizit enthalten ✓ implizit ausdrückbar
[a] number-restriction Erweiterung [b] seit 2.1 [c] nur Attribute

Tabelle 4.1.

Systeme im Vergleich (Konzeptterme)

4.1.3. Ausdrucksstärke der bereitgestellten numerischen Gegenstandsbereiche

Im KRSS-Standard sind neben den Zahlen auch Strings als Basisdatentyp eines konkreten Gegenstandsbereiches vorgesehen. Im Folgenden soll der Gegenstandsbereich der Zahlen daraufhin untersucht werden, inwieweit es möglich ist, Sätze über Ungleichungssystemen über Polynomen aus $\mathbb{Q}[x_1, \ldots, x_n]$ zu formulieren und zu behandeln. Dazu wird zuerst auf den den Systemen zugrundeliegenden Basisdatentyp eingegangen. Daran schließt sich eine Untersuchung der Ausdrucksmächtigkeit und Vollständigkeit der verschiedenen Systeme in Bezug auf lineare und nichtlineare Sätze an.

Basisdatentyp

In KRSS sind als numerische Basisdatentypen Number und Integer vorgesehen. Als Extension von Number wird »the numbers« angegeben. Was sind aber »the numbers«? Aufgrund der Bemerkung: »The syntax of names, numbers, integers and strings are the same as in R«[8], könnte man annehmen, daß damit der R-

8. Unter R soll mit größter Wahrscheinlichkeit CommonLisp verstanden werden.

Datentyp number gemeint ist. Dieser umfaßt aber $\mathbb{C}$.

Da aber weiterhin mit minimum und maximum Vergleiche über diesem Datentyp definiert sind, und auf $\mathbb{C}$ keine mit der algebraischen Struktur verträgliche Anordnung definierbar ist, kann mit Number maximal $\mathbb{R}$ gemeint sein.

Von der Verwendung von $\mathbb{R}$ im Rahmen eines terminologischen Systems ist allerdings abzusehen, da dies unweigerlich Rechenungenauigkeiten und Rundungsfehler nach sich zieht. Daher ist $\mathbb{Q}$ die geeignete Basis für den numerischen Teil des Gegenstandsbereiches, insbesondere im Hinblick auf die Realisierung von Ungleichungssystemen. Dies stimmt mit einer Bemerkung die in [PSS93] an anderer Stelle gemacht wird, überein:

> »All domains contain the rationals and strings over some alphabet of size at least 2. This is called the concrete part of the domain.«

Daher wird im weiteren davon ausgegangen, daß $\mathbb{Q}$ mit dem Datentyp Number in KRSS gemeint ist.

Tabelle 4.2. Ausdrucksstärke: Basisdatentyp

KRSS	Core	LOOM	CLASSIC	KRIS	TAXON
Basisdatentyp	$\mathbb{Q}$,$\mathbb{Z}$	$\mathbb{R}$,$\mathbb{Z}$	$\mathbb{R}$	$\mathbb{N}$	$\mathbb{Q}$

Ausdrucksstärke

Tabelle 4.3 gibt eine Übersicht über die Möglichkeiten, die die betrachteten Systeme im Hinblick auf die Repräsentation von Sätzen der elementaren Algebra über dem jeweiligen Basisdatentyp bieten. Bei einer Analyse dieser Tabelle fallen folgende Punkte auf:

Atomare Terme Konstanten a des Basisdatentyps wie etwa 0 und 20 lassen sich in allen Systemen verwenden, wobei ihre Verwendung in TAXON auf die Definition der konkreten Prädikate beschränkt ist. Variablen x, y werden ebenfalls von allen Systemen zur Verfügung gestellt. Die Verbindung mit dem terminologischen System geschieht über die Referenz auf Attributketten. Besonders deutlich wird dies in dem zweistufigen Verfahren von TAXON. In der Definition des konkreten Prädikats werden die Variablen eingeführt. Bei der Verwendung des konkreten Prädikates in den Konzeptdefinitionen werden die Variablen dann an die verschiedenen Attributketten gebunden. Innerhalb von LOOM und CLASSIC ist auch eine implizite Bindung der Variablen möglich, und zwar wenn Untertypen des Basisdatentyps definiert werden.

KRSS	Core	LOOM	CLASSIC	KRIS	TAXON
Terme					
Konstanten a	✔	✔	✔	✔	✔
Variablen x	✔	✔	✔	✔	✔
Funktionssymbole					
$(x + a)$					
$(x + y)$					
$(a \cdot x)$					
$(x \cdot y)$					
Atomare Formeln					
$(\alpha < \beta)$		✔		✔	✔
$(\alpha > \beta)$		✔		✔	✔
$(\alpha = \beta)$	✓	✔	✓	✔	✔
$(\alpha \neq \beta)$		✔		✔	✔
$(\alpha \leq \beta)$	✔	✔	✔	✔	✔
$(\alpha \geq \beta)$	✔	✔	✔	✔	✔
Logische Operatoren					
$\neg\varphi$		✗		✗	✗
$(\varphi \vee \psi)$		✗		✗	✗
$(\varphi \wedge \psi)$	✗	✗	✗	✗	✗
Quantoren					
$(\forall x\varphi)$	✗	✗	✗	✗	✗
$(\exists x\varphi)$		✗			✗

✔ explizit enthalten ✓ implizit ausdrückbar
✗ über den abstrakten Gegenstandsbereich

Tabelle 4.3.
Ausdrucksstärke

Terme Durch die Einschränkung auf lineare Sätze genügt die Betrachtung von komplexen Termen, die durch die Verwendung von + entstehen [9].

Als einziges System sieht LOOM in seiner Beschreibungssprache die Relationen + und − vor, erlaubt also im Prinzip die Definition von Termen. Bei genauerem Hinsehen stellt sich allerdings heraus, daß diese Relationen nur für das Retrieval eingesetzt werden können [10].

In allen anderen Systemen ist von vornherein keine Möglichkeit zur Formu-

9. Ein Term der Form $a \cdot x$ ist bekanntlich äquivalent zu $\underbrace{a + \cdots + a}_{x}$.

10. So führt die Definition (defconcept foo :is (:and (= x (+ y y)))) zu einem Syntaxfehler.

lierung von komplexen Termen vorgesehen. Damit ist es in keinem der Systeme möglich, selbst einfachste, d.h. univariate lineare Terme wie etwa $x + 2$ zu definieren, die Definition multivariater linearer Terme wie $x + y$ ist damit natürlich auch nicht möglich [11].

Atomare Formeln Alle Systeme bis auf CLASSIC stellen den vollen Satz von Vergleichsrelationen $=, \neq, <, \leq, >, \geq$ zur Verfügung. In CLASSIC und im KRSS-Kern sind nur Vergleiche mittels $\leq$ und $\geq$ erlaubt. Dort ist es auch nicht möglich Vergleichsrelationen zwischen Variablen, etwa $x \leq y$, zu formulieren, da die Konstrukte minimum und maximum nur Konstanten zulassen. Damit ist etwa die Definition eines Quadrates durch die Gleichsetzung der Länge und Breite über $=$ unmöglich. Es verbleibt die unbefriedigende Möglichkeit zur Gleichsetzung der Attribute über das abstrakte Gleichheitsprädikat equal.

Formeln Keines der Systeme definiert logische Operatoren wie $\neg\varphi, (\varphi \vee \psi), (\varphi \wedge \psi), (\varphi \rightarrow \psi), (\varphi \leftrightarrow \psi)$ oder Quantoren $(\forall x\varphi)$ und $(\exists x\varphi)$ innerhalb des konkreten Bereiches. Stattdessen wird auf die entsprechenden Konstrukte des abstrakten Gegenstandsbereiches zurückgegriffen [12]. Dadurch ist man nicht in der Lage, spezialisierte und damit meist auch schnellere Algorithmen des konkreten Bereiches zu benutzen. Darüber hinaus schränkt man so evtl. die Ausdrucksmächtigkeit des Gesamtsystems ein. So besteht etwa im KRSS-Kern und CLASSIC keine Möglichkeit $<$ durch Negation von $\geq$ zu definieren.

Zusammenfassung

Insgesamt läßt sich in den untersuchten Systemen nur ein sehr kleiner Teil der linearen Sätze der elementaren Algebra über den reellen Zahlen beschreiben.

CLASSIC und der KRSS-Kern bieten die geringsten Ausdrucksmöglichkeiten. Sie stellen nicht alle Vergleichsrelationen zur Verfügung und verlangen, daß einer der zu vergleichenden Terme eine Konstante ist.

KRIS, TAXON und LOOM erlauben beliebige Vergleiche zwischen atomaren Termen. Diese werden in KRIS und TAXON bei den T- und ABox-Inferenzen vollständig berücksichtigt, während die Einbeziehung von Ausdrücken des kon-

11. In gewissem Maß können diese Terme durch eine Umformulierung der sie enthaltenden atomaren Formeln repräsentiert werden. So sind bekanntlich die atomaren Formeln $x + 2 > 0$ und $x > -2$ äquivalent. Erstere ist nicht repräsentierbar, letztere schon. Die Umformung muß allerdings auf jeden Fall durch den Entwickler der Repräsentation erfolgen und geschieht nicht automatisch.

12. Sogar logische Operatoren, die in den Definitionen konkreter Prädikate von TAXON benutzt werden, werden auf den abstrakten Bereich abgebildet.

kreten Gegenstandsbereiches in LOOM doch sehr zu wünschen übrig läßt [13]. So wird z.B. die Konzeptbeschreibung (defconcept foo :is (:and a (= x y) (> x y))) nicht als inkonsistent erkannt [14].

Keines der Systeme erlaubt nichtatomare Terme. Daher ist es nicht möglich selbst einfachste Beziehungen zwischen Attributen wie $x = y + 2$ zu formulieren. Multivariate lineare Sätze wie $x + y$ sind genausowenig möglich wie nichtlineare Sätze.

Die Realisierung logischer Operatoren und Quantoren über dem konkreten Gegenstandsbereich durch Rückgriff auf die entsprechenden Konstrukte des abstrakten Gegenstandsbereiches kann Auswirkungen auf die Performanz haben und führt im Fall von CLASSIC zu einer Einschränkung der Ausdrucksmächtigkeit.

4.1.4. Fazit

Von den untersuchten Systemen ist das auf der Beschreibungssprache $\mathcal{ALCF}$ basierende TAXON [BH91a, Han96] am besten geeignet. Es vereinigt eine relativ große Ausdrucksstärke über dem abstrakten Gegenstandsbereich mit einer sauberen Einbindung von Basisdatentypen durch zulässige konkrete Gegenstandsbereiche. Gerade Basisdatentypen sind aber im untersuchten Anwendungsbereich extrem wichtig, etwa um Hintergrundwissen in Form von physikalischen Gesetzen zu beschreiben, aber um Meßwerte erfassen zu können.

Außerdem macht TAXON im Gegensatz zu den anderen beschreibungslogischen Systemen nicht die sogenannte Unique Name Assumption. Auch dieses Faktum ist, wie im letzten Kapitel gezeigt wurde, für den Einsatz von Beschreibungslogiken für Second-Level-Support-Systeme von entscheidender Bedeutung.

Allerdings ist die Ausdrucksstärke von TAXON im konkreten Gegenstandsbereich nicht groß und die Möglichkeiten zur Anbindung unterschiedlicher zulässiger konkreter Gegenstandsbereiche nicht flexibel genug. Darüber hinaus werden die Möglichkeiten, die aus der Fähigkeit tableaux-basierender Verfahren, Modelle zu generieren, erwachsen, nicht erschöpfend genutzt.

Ausgehend von TAXON wurde daher das System CTL [KW96b] entwickelt, das im folgenden Absatz beschrieben werden soll.

4.2. CTL

Es unterscheidet sich von den untersuchten Systemen insbesondere durch seine Schnittstelle zur modularen Einbindung externer Entscheidungsverfahren für kon-

13. Leider kann hier nicht im Detail auf die Vollständigkeit der Inferenzen in den verschiedenen Systemen eingegangen werden.

14. x und y sind Attribute von a, die Wissensbasis selbst wird als monoton gekennzeichnet. Dies erlaubt es LOOM, maximale Inferenzen zu ziehen.

krete Gegenstandsbereiche und die Möglichkeit, sich im Falle eine erfolgreichen Konsistenztest das generierte Modell anzeigen zu lassen.

Die Schnittstelle erlaubt die Einbindung eines Entscheidungsverfahrens für einen konkreten Gegenstandsbereich, der genau die Ausdrucksstärke realisiert, die im konkreten Anwendungsszenario benötigt wird. Im darauf folgenden Abschnitt werden daher zulässige konkrete Gegenstandsbereiche unterschiedlicher Ausdrucksstärke und die dahinter liegenden Entscheidungsverfahren vorgestellt.

In bestehenden beschreibungslogischen Systemen wird das Hauptaugenmerk auf die Berechnung des Konsistenztestes resp. des Subsumptionstestes gerichtet. Bei tableaux-basierten Systemen bedeutet dies, daß festgestellt wird, ob ein Modell existiert.

Insbesondere in technischen Domänen ist dies allerdings nicht die herausragende Fragestellung, vielmehr ist von Interesse, wie das Modell genau aussieht. In Verbindung mit konstruktiven Verfahren zur Entscheidung der konkreten Gegenstandsbereiche, besonders mit Verfahren zur Quantorenelimination, läßt sich CTL so erweitern, daß im Falle eines erfolgreichen Konsistenztests die generierten Modelle angezeigt werden.

4.2.1. Abstrakter Gegenstandsbereich und Inferenzmaschine

CTL [KW96b] basiert auf TAXON und stellt daher die im letzten Kapitel beschriebene Beschreibungssprache $\mathcal{ALCF}$ als Repräsentationssprache des abstrakten Gegenstandsbereiches zur Verfügung. CTL benutzt die KRSS-Syntax als Eingabesyntax für den abstrakten Gegenstandsbereich. Abbildung 4.4 faßt den Umfang der TBox Beschreibungssprache und die konkrete Syntax zusammen.

Kurzschreibweise	ASCII-Syntax
$\top$	top
$\bot$	bottom
$(C_1 \sqcap \ldots \sqcap C_n)$	(and C_1 ... C_n)
$(C_1 \sqcup \ldots \sqcup C_n)$	(or C_1 ... C_n)
$\neg C$	(not C)
$\exists$ R . C	(some R C)
$\forall$ R . C	(all R C)
$(R_1 \circ \ldots \circ R_n)$	(compose R_1 ... R_n)
$CN \doteq C$	(define-concept CN C)
$CN \dot{\sqsubseteq} C$	(define-primitive-concept CN C)
$RN \dot{\sqsubseteq} \top \times \top$	(define-primitive-role RN toprole)
$A \dot{\sqsubseteq} \top \mapsto \top$	(define-primitive-attribute A topattr)

Tabelle 4.4. Die Syntax des abstrakten Gegenstandsbereiches von CTL

Konkrete Objekte werden als Instanzen der Abox repräsentiert. Dabei werden neue Instanzen o mittels (define-distinct-individual o) in die ABox eingeführt. Assertionen können bezüglich der Zugehörigkeit zu einem Konzept durch (state (instance o CN)) und bezüglich einer existierenden Relation zwischen zwei bzw. mehr Objekten durch (state (related $o_1 \ldots o_n$ r)) gemacht werden.

Die Inferenzmaschine basiert auf dem Tableaux-Kalkül und einem modellgenerierenden Konsistenztest, so daß korrekte und vollständige Inferenzen realisiert werden. Basierend auf dem Konsistenztest werden die verschiedenen Inferenzen wie Konzept- und Objektklassifikation und Retrieval realisiert(siehe Tabelle 4.5 und Tabelle 4.6).

(show-hierarchy)
(show-lower CN)
(show-upper CN)
(show-immediate-lower CN)
(show-immediate-lower CN)

Tabelle 4.5.

Die verschiedenen TBox - Inferenzen von CTL

Während (show-hierarchy) die komplette Konzepthierarchie ausgibt, zeigen (show-lower CN) und (show-upper CN) die Mengen der unteren bzw. der oberen Nachbarn, (show-immediate-lower CN) und (show-immediate-upper CN) die Mengen der direkten unteren und oberen Nachbarn im Konzeptgraphen an.

(show-realizations o)
(show-weak-realizations o)
(show-instances CN)
(show-weak-instances CN)

Tabelle 4.6.

Die verschiedenen ABox - Inferenzen von CTL

Mittels (show-realizations o) und (show-weak-realizations o) wird die (schwache) Objektklassifikation für das Objekt o, mittels (show-instances CN) bzw. (show-weak-instances CN) das (schwache) Retrieval für das Konzept CN angestossen.

CTL übernimmt dabei die modulare Realisierung der benötigten Inferenzen von TAXON. Dies bedeutet, daß die unterschiedlichen Konstrukte der Beschreibungssprache in ihre entsprechenden ABox-Expansionsregeln übersetzt werden. Jede einzelne ABox-Expansionsregel wird dann als eigenständige funktionale Einheit realisiert, die unabhängig von den anderen Elementen der Beschreibungssprache angewandt werden kann.

Dies bedeutet, daß es einerseits leicht möglich ist, weitere Regeln in dieses modulare Rahmenwerk zu integrieren. Andererseits ist es genau so leicht möglich, die Regeln, die in einem konkreten Anwendungsfall nicht benötigt werden, wegzulassen. Damit ist es möglich, das beschreibungslogische System an die konkreten Anforderungen der Aufgabenstellung anzupassen.

So kann in einer Reihe von Fällen die Performanz des Gesamtsystems dadurch gesteigert werden, daß nur die ABox-Expansionsregeln in das System eingepflegt werden, die für die im konkreten Szenario verwendeten Konstrukte der Beschreibungssprache benötigt werden. Durch Entfernung der nicht benötigten Expansionsregeln müssen diese nicht auf ihre Anwendbarkeit auf ein konkretes Modell überprüft werden, so daß dadurch die Performanz gesteigert wird.

Eine weitere Performanzsteigerung ist dadurch möglich, daß die allgemeinen Expansionsregeln durch die im konkreten Szenario benötigten Expansionsregeln ersetzt und diese Regeln kompiliert werden.

4.2.2. Konkrete Gegenstandsbereiche

Vorüberlegungen

Die Realisierung konkreter Gegenstandsbereiche ist im wesentlichen die Aufgabe der Integration eines Entscheidungsverfahrens für endliche Konjunktionen von Prädikaten aus dem jeweiligen Gegenstandsbereich (siehe Definition 3.24). Diese Integration kann im grundsätzlich auf zwei Arten erfolgen:

- Das Entscheidungsverfahren wird durch ein Modul realisiert, das fest in die Inferenzmaschine des beschreibungslogischen Systems integriert ist.
- Existierende *Implementationen* der Entscheidungsverfahren werden mittels einer wohldefinierten Schnittstelle an die Inferenzmaschine des beschreibungslogischen Systems angebunden.

Wird der erste Ansatz gewählt, so ist bei jeder Änderung des konkreten Gegenstandsbereiches, z.B. Hinzufügen eines neuen Gegenstandsbereiches oder Austausch des Entscheidungsverfahrens eines bestehenden Gegenstandsbereiches, eine Modifikation der Inferenzmaschine der Beschreibungslogik notwendig.

Wählt man den zweiten Ansatz, wird der Aufwand auf die Anpassung eines bereits existierenden Entscheidungsverfahrens für den konkreten Gegenstandsbereich an die Schnittstelle reduziert. Die Existenz dieser Schnittstelle erlaubt es u.a., simultan mehrere konkrete Gegenstandsbereiche, etwa über Zahlen, Symbolen und Strings zur Verfügung zu stellen. Eine solche Schnittstelle erlaubt es darüber hinaus, relativ leicht von Weiterentwicklungen zu profitieren, die z.B. Entscheidungsverfahren für ausdrucksstarke Gegenstandsbereiche bereitstellen (siehe Abschnitt 4.3).

Eine Schnittstelle erlaubt es auch, je nach Aufgabenstellung die Ausdrucksstärke der konkreten Gegenstandsbereiche an die Anforderungen der Anwendung anzupassen. Wird z.B. nur eine geringe Ausdrucksstärke im konkreten Gegenstandsbereich der Zahlen benötigt, kann ein dafür optimiertes Entscheidungsverfahren eingesetzt werden. Wachsen die Anforderungen an die Ausdrucksstärke des Gegenstandsbereiches, so kann das Entscheidungsverfahren für diesen Gegenstandsbereich modular ausgetauscht werden.

Insgesamt wird durch die Schnittstelle zur Ankopplung konkreter Gegenstandsbereiche das Konzept eines modularen, konfigurierbaren beschreibungslogischen Systemes, das von TAXON und CTL verfolgt wird, konsequent auf den Bereich der konkreten Gegenstandsbereiche erweitert. Daher wird dieser Ansatz auch in den verbleibenden Abschnitten dieses Kapitels weiter verfolgt.

Eine Schnittstelle für konkrete Gegenstandsbereiche

Definition 3.24 dient als Richtlinie für den Entwurf der Schnittstelle. Die Schnittstelle selbst besitzt zwei Ebenen: Auf der Repräsentationsebene werden zusätzliche Konzeptoperatoren zur Definition von Prädikaten über konkreten Gegenstandsbereichen definiert. Auf der Inferenzebene muß ein Satz von Funktionen bereitgestellt werden, der die Integration der Entscheidungsprozeduren ermöglicht.

Repräsentationsebene

Ein erster Ansatz zur Definition der Repräsentationsebene der Schnittstelle ist der Rückgriff auf existierende Standards, z.B. KRSS. Der konkrete Gegenstandsbereich von KRSS ist fest vorgegeben [PSS93] und nicht sehr ausdrucksstark.

Gesucht ist jedoch ein Repräsentationsformalismus, der im Hinblick auf die Ausdrucksstärke der in einem Szenario benötigten konkreten Gegenstandsbereiche flexibel ist. Daher muß ein Weg gefunden werden, der einen solchen Mechanismus in KRSS integriert.

Anstatt neue Konzeptoperatoren für jedes neue konkrete Prädikat zu definieren (wie es KRSS mit minimum und maximum tut), wird ein einziger, generischer Konzeptoperator (constrain $r_1 \ \ldots \ r_n \ PN$) eingeführt. Dieser erlaubt es, eine Anzahl von Attributen und Rollen miteinander in Verbindung zu setzen. PN ist ein Constraintname, der mit Hilfe des zusätzlichen terminologischen Axioms (define-constraint PN (V T E)) eingeführt wurde.

E ist ein Ausdruck des konkreten Gegenstandsbereiches, in dem die Variablen aus V der Basistypen T[15] verwendet werden können. Tabelle 4.7 gibt einen Überblick über die Erweiterungen der Repräsentationsebene.

15. Momentan werden nur numerische Variablen im Rahmen eines CLP($\mathcal{R}$) Systems unterstützt.

Tabelle 4.7.

CTL Erweiterungen: Die Repräsentationsebene

Kurzschreibweise	ASCII-Syntax
$\exists_{\Delta}\ R_1 \ldots R_n\ .\ PN$	(constrain $R_1 \ldots R_n\ PN$)
$PN\ \dot{\sqsubseteq}\ PD$	(define-constraint PN ($V\ T\ E$))

Inferenzebene

Wie bereits erwähnt basiert die Inferenzmaschine von TAXON und damit von CTL auf dem Tableauxkalkül. Benutzt werden die Transformationsregeln aus [Han93].

Um eine generische Schnittstelle für konkrete Gegenstandsbereiche, die externe Entscheidungsverfahren benutzt, in das Entscheidungsverfahren integrieren zu können, müssen die für den konkreten Gegenstandsbereich zuständigen Transformationsregeln das Entscheidungsverfahren der Implementierung des jeweiligen konkreten Gegenstandsbereiches aufrufen.

Jedes mal, wenn eine solche Regel aufgerufen wird, muß dann:

- die Menge der Attribut- und Rollenketten, die an die Variable des konkreten Prädikates gebunden werden, berechnet werden;
- die möglichen Füller für diese Attribut- und Rollenketten ermittelt werden;
- der Prädikatsterm des konkreten Prädikates für die möglichen Kombinationen der Füller instantiiert werden;
- der instantiierte Prädikatsausdruck einem Pool der instantiierten Ausdrücke hinzugefügt werden (durch Aufruf der Schnittstellenfunktion add-expr);
- das Entscheidungsverfahren des konkreten Gegenstandsbereiches aufgerufen werden, um zu überprüfen, ob der veränderte Pool noch konsistent ist (durch Aufruf der Schnittstellenfunktion consistent);
- im Falle einer Inkonsistenz müssen im Rahmen des Backtracking die zwischen dem letzten Choice-point und der Inkonsistenz hinzugefügten Ausdrücke aus dem Pool entfernt werden (durch Aufruf der Schnittstellenfunktion remove-expr);
- im Rahmen der ABox Transformation wird die Negation von Konzepttermen so weit wie möglich nach innen geschoben. Da diese Konzeptterme auch konkrete Prädikatsaufrufe beinhalten können, muß der konkrete Gegenstandsbereich zusätzlich noch noch eine Schnittstellenfunktion negate zur Verfügung stellen, die die Negation eines konkreten Prädikates berechnet.

Eine Implementierung eines konkreten Gegenstandsbereiches muß damit nur die folgende, schlanke Schnittstelle zur Verfügung stellen, um in CTL integriert werden zu können:

- consistent, eine Funktion zum Aufruf des Entscheidungsverfahren;
- add-expr, eine Funktion zum Hinzufügen eines Ausdrucks zu einem Pool von Ausdrücken;
- remove-expr, eine Funktion zum Entfernen eines Ausdrucks aus einem Pool von Ausdrücken;
- negate, eine Funktion zur Negation eines Ausdrucks.

Tabelle 4.8 gibt noch einmal eine Übersicht über die Funktionen der Schnittstelle.

```
(consistent dom)
(add-expr expr dom)
(remove-expr expr dom)
(negate expr dom)
```

Tabelle 4.8.

Die Schnittstelle zu den konkreten Gegenstandsbereichen

Nachdem mit der Schnittstelle die Basis für die Einbindung unterschiedlichster zulässiger konkreter Gegenstandsbereiche geschaffen ist, werden im nächsten Abschnitt eine Reihe zulässiger numerischer konkreter Gegenstandsbereiche vorgestellt. Zuvor soll aber noch kurz die Erweiterung zur Anzeige der generierten Modelle präsentiert werden.

4.2.3. Anzeige generierter Modelle

Abstrakter Gegenstandsbereich

Normalerweise liefert der ABox-Konsistenztest eines beschreibungslogischen Systems als Ergebnis einen bool'schen Wert, der anzeigt ob ein Modell existiert oder nicht. Insbesondere bei Anwendungen in technischen Domänen, aber auch z.B. in der Konfigurierung, ist die Tatsache, daß ein Modell existiert aber nur von untergeordnetem Interesse.

Viel wichtiger als die pure Existenz eines Modelles ist in vielen Fällen sein konkretes Aussehen. Dies ist insbesondere deswegen interessant, weil das Modell das zusätzlich abgeleitete Wissen expliziert.

Das generierte Modell enthält in der Regel neue Individuen, die erst während des Vervollständigungsprozesses angelegt wurden, aber auch zusätzliche, während

des Inferenzprozesses abgeleitete Relationen zwischen den einzelnen Individuen des Modells.

Bei modellgenerierenden Verfahren zum Konsistenztest wird, wie der Name schon sagt, ein Verfahren eingesetzt, das im Fall eines erfolgreichen Konsistenttestes ein explizites Modell der ABox generiert. In CTL wurde daher die zusätzliche Funktion show-model realisiert, die es erlaubt, im Fall eines erfolgreichen Konsistenztests das generierte Modell anzuzeigen.

Konkreter Gegenstandsbereich: Berechnung zulässiger Parameterwerte

Von besonderem Interesse ist das generierte Modell dann, wenn nicht nur die Ableitungen aus dem abstrakten Gegenstandsbereich, sondern auch die aus dem konkreten Gegenstandsbereich berücksichtigt werden können. Denn dies bedeutet insbesondere, daß alle Einschränkungen an den verschiedenen Parametern, die sich während des Inferenzprozesses durch die Verwendung konkreter Prädikate ergeben, expliziert werden.

Im Prinzip ist dies möglich: Um die Konsistenz einer ABox zu überprüfen, wird für den konkreten Gegenstandsbereich überprüft, ob eine endliche Konjunktion konkreter Prädikate erfüllbar ist, d.h. ob der Satz $S = \exists x_1 \ldots \exists x_k (p_1 \wedge \cdots \wedge p_n)$ wahr ist. Ist nun das im jeweiligen konkreten Gegenstandsbereich implementierte Verfahren zur Überprüfung der Erfüllbarkeit des Satzes S konstruktiv, so werden die Einschränkungen der einzelnen Parameter x_k berechnet und können bei der Anzeige des generierten Modelles angezeigt werden.

Verfahren zur Quantorenelimination sind besonders interessant Besonders vielversprechend sind in diesem Zusammenhang die Methoden zur Quantorenelimination aus dem Gebiet der Computer Algebra. Im Bereich des Constraint Logic Programming werden diese Verfahren Variablen auch Eliminationsverfahren bzw. Projektion genannt.

Ihren Ursprung haben diese Verfahren bereits in Tarskis Beweis zur Entscheidbarkeit der Theorie der elementaren Algebra über den reellen Zahlen [Tar19] (siehe Abschnitt 4.3.1. Der Beweis beruht auf folgender Idee: Zunächst wird jeder Satz S in einen äquivalenten quantorenfreien Satz $\hat{S}$ transformiert. Die Überprüfung der Gültigkeit des Satzes $\hat{S}$ ist dann einfach.

Neben der Überprüfung der Gültigkeit eines Satzes kann die Quantorenelimination aber auch dazu benutzt werden, um die Einschränkungen, die sich für die einzelnen Variablen ergeben, zu berechnen. Anstatt alle Variablen zu eliminieren, werden alle Variablen bis auf den interessanten Parameter aus der Formeln eliminiert. Dadurch erhält man einen Satz mit den zulässigen Werten des variablen Parameters.

Konkret lässt sich das angewandte Verfahren wie folgt beschreiben:

- Sei $X = x_1, \ldots, x_n$ die Liste der Parameter im Modell M.
- Für $x_i \in X$
  - Eliminiere alle Parameter außer x_i aus S. D.h. transformiere S in einen Satz $S_i = \exists x_i R_i, R_i$ quantorenfrei.
  - Transformiere S_i in disjunktive Normalform, d.h. transformiere S_i in $\hat{S}_i = \mathrm{dnf}(S_i)$.
  - Sei R_i die sich aus den $\hat{s}_i$ ergebende Matrix.
- Liefere $R = R_1, \ldots R_n$, die Liste der R_i, als Ergebnis des Aufrufes zurück.

Da R eine Liste der zulässigen Werte für jeden Parameter beschreibt, kann diese Liste der Einschränkungen zusammen mit dem durch die ABox Inferenzen berechneten Modell als Ergebnis des Aufrufes show-model zurückgeliefert werden.

Natürlich können auch mehrere Variablen offen gelassen werden. Als Ergebnis der Eliminationsprozedur erhält man dann einen Satz der die Abhängigkeiten unter den offen gelassenen Parametern beschreibt.

Tabelle 4.9.

Die Schnittstelle zur Anzeige der Modelle

```
(show-model dom)
(eliminate varlist dom)
```

Voraussetzung zum Einsatz der Quantoreneliminationsverfahren ist natürlich, daß sie Entscheidungsprozeduren für zulässige konkrete Gegenstandsbereiche darstellen. Dies soll im nächsten Abschnitt untersucht werden.

4.2.4. Fazit CTL

CTL überwindet Mängel existierender beschreibungslogischer Systeme, indem es die flexible Anbindung zulässiger konkreter Gegenstandsbereiche erlaubt und Mechanismen zur Anzeige des generierten Modells bereitstellt.

Dabei kann die Flexibilität bei der Anbindung konkreter Gegenstandsbereiche über eine Schnittstelle auf verschiedene Arten genutzt werden:

- Zunächst kann je nach Anwendungsfall der optimale konkrete Gegenstandsbereich, d.h. der Gegenstandsbereich der genau die benötigte Ausdrucksstärke bereitstellt,angebunden werden.
- Sollten sich dann im Lauf Zeit die Anforderungen des Anwendungsfalles an den konkreten Gegenstandsbereich verändern, z.B. eine größere Ausdrucksstärke benötigt werden, so wird einfach ein anderer zulässiger konkreter Gegenstandsbereich über die Schnittstelle eingebunden, die bisherige Repräsentation des Wissens muß nicht verändert werden.

- Weiterhin können neue, effizientere Verfahren zur Realisierung der Ausdrucksstärke eines bereits verwendeten konkreten Gegenstandsbereiches, z.B. neue Quantoreneliminationsverfahren, ebenfalls problemlos in das Gesamtsystem eingebunden werden.

Status und mögliche Weiterentwicklungen

CTL wurde aufbauend auf dem System TAXON in den Jahren 1996 und 1997 prototypisch implementiert, seine Weiterentwicklung ruht seitdem. Der Schwerpunkt der Implementierung lag dabei primär auf der Überprüfung der konzeptionellen Ideen, nicht auf einer Optimierung der Performanz bzw. auf der Erzielung einer Produktnahen Softwarequalität inklusive Support, User Manual etc.

Betrachtet man sich die Architektur von CTL, so zeichnet sich CTL sowohl durch eine einfache, *schmale* Schnittstelle zu den Entscheidungsprozeduren der konkreten Gegenstandsbereiche, als auch durch eine einfache Syntax zur Repräsentation und Abfrage des Wissens selbst aus.

Verteilte Architektur durch Webservices Nimmt man den Fakt hinzu, daß zur Berechnung der Inferenzen sowohl in den konkreten Gegenstandsbereichen als auch im abstrakten Gegenstandsbereich nicht unerhebliche Ressourcen benötigt werden, so erscheint aus heutiger Sicht eine Migration hin zu einer verteilten (Client/Server) Architektur unter Einsatz von Webservices äußerst sinnvoll.

Webservices haben im Vergleich zu »klassischen« Mechanismen für verteilte Architekturen wie RemoteProcedureCalls (RPC) und CORBA ihre Vorteile insbesondere in Szenarien, in denen die angebotenen Dienstleistungen sehr grobgranular sind, d.h. eine verhältnismäßig lange Rechenzeit benötigen und die Interoperabilität zwischen in verschiedenen Programmiersprachen implementierten Systemkomponenten hergestellt werden muß.

Beides ist in dem hier betrachteten Szenario der Fall. Sowohl sind die Basisinferenzen eines beschreibungslogischen Systems sind sehr rechenintensiv als ist LISP als klassische Implementierungssprache beschreibungslogischer System nicht die Standardimplementierungssprache im betrachteten Umfeld.

Die Verwendung von Webservices bietet sich sowohl für die Bereitstellung der Inferenzen als auch für die Implementierung der der Schnittstellen an.

Nicht nur könnte so das Gesamtsystem auf transparente Art und Weise von externen Systemen genutzt werden, auch die Anbindung externer Entscheidungsprozeduren für konkrete Gegenstandsbereiche könnten dadurch auf standardisierte Art und Weise erfolgen. Darüber hinaus ist es leicht möglich, ein Spektrum unterschiedlich ausdrucksstarker konkreter Gegenstandsbereiche zur Verfügung zu stellen, die dann über UDDI[16] gefunden und über WSDL[17] automatisch an das

16. UDDI – Universal Description, Discovery and Integration of webservices.

Kernsystem angebunden werden.

Bevor im übernächsten Abschnitt untersucht werden soll, inwieweit mit RACER ein beschreibungslogisches System der «dritten» Generation diese Ansätze bereits umsetzt, wird im nächsten Abschnitt eine Systematik für numerische konkrete Gegenstandsbereiche vorgestellt.

4.3. Numerische konkrete Gegenstandsbereiche

> *In fact, for everyday wear and tear, the number 1 is a "stronger" abstraction than any definition resembling "the set containing the fewest elements that is not the empty set". Small numbers are most easily defined by holding up fingers.*
>
> *– Unknown –*

Einer der Hauptgründe zur Entwicklung von CTL war die mangelnde Ausdrucksstärke der numerischen konkreten Gegenstandsbereiche existierender Systeme. Diese erlaubten es nicht, das zur Realisierung des Leitbeispiels notwendige Wissen zu beschreiben. Das CTL-Kernsystem stellt eine Schnittstelle zur Verfügung, daß die flexible Anbindung von zulässigen konkreten Gegenstandsbereichen der unterschiedlichsten Ausdrucksstärken erlaubt. In diesem Abschnitt soll daher das Spektrum der Möglichkeiten untersucht werden.

Dazu werden zunächst einige fundamentale Entscheidbarkeits- und Unentscheidbarkeitsresultate aus dem Bereich der Theorien erster Ordnung über numerischen Grundkörpern vorgestellt und die Verbindung zu zulässigen konkreten Gegenstandsbereichen hergestellt [18]. Damit geben diese Resultate mehr oder weniger vor, welche maximale Ausdrucksstärke für zulässige numerische Gegenstandsbereiche über den unterschiedlichen Grundkörpern erreichbar ist.

4.3.1. Grundlegende Entscheidbarkeits- und Unentscheidbarkeitsresultate

Entscheidbarkeit der elementaren Algebra über den reellen Zahlen

Definition 4.1 (Theorie der elementaren Algebra)
Die *Theorie der elementaren Algebra über* $\mathbb{R}$ ist die Theorie erster Ordnung $\langle\langle \mathbb{R}, +, \cdot, =, <, 0, 1\rangle\rangle$ [19].

17. WSDL – Web Services Description Language.

18. Im Folgenden wird ein Basiswissen aus den Bereichen Modelltheorie und Logik vorausgesetzt. Eine Zusammenfassung der wichtigsten Definitionen findet sich in Anhang B.

19. $+,\cdot$ und $<$ werden wie üblich interpretiert.

Bemerkung 4.1
In der Praxis wird man natürlich ein anderes Alphabet benutzen, insbesondere zusätzliche Vergleichsoperatoren zum Aufbau atomarer Formeln und alle $q \in \mathbb{Q}$ als Individuenkonstanten zulassen.

Die Theorie der elementaren Algebra über den reellen Zahlen wird häufig auch als Theorie der reell abgeschlossenen Körper bezeichnet.

Bereits 1930 wurde von Tarski der folgende Satz bewiesen:

Satz 4.1 (Entscheidbarkeit der Theorie der elementaren Algebra)
Die Theorie der elementaren Algebra über $\mathbb{R}$ ist entscheidbar, d.h. es gibt ein Verfahren, das für jede Formel der elementaren Algebra in einer endlichen Anzahl von Schritten entscheidet, ob diese Formel wahr ist.

Zulässigkeit der elementaren Algebra als konkrekter Gegenstandsbereich

Nun gilt es noch zu klären, ob die elementare Algebra auch ein zulässiger konkreter Gegenstandsbereich ist. Da das Entscheidungsverfahren die wesentliche Hürde darstellt, ist dies einfach.

Lemma 4.1
Die Theorie der elementaren Algebra über den reellen Zahlen ist ein konkreter Bereich über der Domäne $\mathbb{R}$.

Beweis. Die Prädikate entstehen auf natürliche Art und Weise aus den Formeln der Theorie. So definiert z.B. die Formel $\exists z(x + z^2 = y)$ ein zweistelliges Prädikat $\{(r, s) : r, s \in \mathbb{R} \wedge r \leq s\}$. □

Daraus folgt die Zulässigkeit der Theorie der elementaren Algebra als konkreter Bereich:

Satz 4.2 (Zulässigkeit der Theorie der elementaren Algebra)
Die Theorie der elementaren Algebra über den reellen Zahlen ist ein zulässiger konkreter Bereich.

Beweis. Folgt aus der Entscheidbarkeit der Theorie der elementaren Algebra. □

Weitere Entscheidbarkeits- und Unentscheidbarkeitsresultate

Neben den reellen Zahlen sind auch andere numerische Grundkörper und eingeschränkte Operatormengen interessant. Es gelten die folgenden Resultate (siehe [Tar51, TMR71, BS81, HU79] für die Beweise):

Satz 4.3
(i) Die Zahlentheorie, d.h. $\langle\langle \mathbb{N}, +, \cdot, =, <, 0, 1\rangle\rangle$ ist unentscheidbar, sogar die eingeschränkte Theorie $\langle\langle \mathbb{N}, +, \cdot, =, 1\rangle\rangle$ ist schon unentscheidbar.

(ii) Die Theorie der Ringe, d.h. $\langle\langle \mathbb{Z}, +, \cdot, =, 1\rangle\rangle$ *ist unentscheidbar.*

(iii) Die Presburger Arithmetik, d.h. $\langle\langle \mathbb{Z}, +, =, <, 0, 1\rangle\rangle$ *ist entscheidbar.*

(iv) Die Theorie der Körper, d.h. $\langle\langle \mathbb{Q}, +, \cdot, =, 1\rangle\rangle$ *ist unentscheidbar.*

(v) $\langle\langle \mathbb{Q}, +, =, <, 0, 1\rangle\rangle$ *ist entscheidbar.*

(vi) Die Theorie der algebraisch abgeschlossenen Körper, d.h. $\langle\langle \mathbb{C}, +, \cdot, =, <, 0, 1\rangle\rangle$ *ist entscheidbar.*

4.3.2. Auswirkungen auf die Zulässigkeit von konkreten Gegenstandsbereichen

Welche Schlüsse können aus diesen Ergebnissen für die Realisierbarkeit numerischer konkreter Gegenstandsbereiche gezogen werden?

Unentscheidbare Theorien

Es gilt zunächst, daß jede Theorie, die unentscheidbar ist, kein zulässiger konkreter Gegenstandsbereich sein kann. Das bedeutet konkret, daß es keinen zulässigen konkreten Gegenstandsbereich geben kann, der die Zahlentheorie (resp. $\langle\langle \mathbb{N}, +, \cdot, =, 1\rangle\rangle$ und $\langle\langle \mathbb{Z}, +, \cdot, =, 1\rangle\rangle$) in voller Mächtigkeit in eine Beschreibungslogik einbettet. Da $\langle\langle \mathbb{N}, +, \cdot, =, 1\rangle\rangle$ und $\langle\langle \mathbb{Z}, +, \cdot, =, 1\rangle\rangle$ die nichtlinearen Gleichungssysteme über $\mathbb{N}$ bzw. $\mathbb{Z}$ beinhalten, ist es zumindest nicht möglich, zulässige konkrete Gegenstandsbereiche für die nichtlinearen Gleichungssysteme über $\mathbb{N}$ resp. $\mathbb{Z}$ durch entsprechende entscheidbare Theorien zu definieren.

Theoriefragmente

Ferner ist zu sehen, daß eine Einschränkung (bzw. eine geringfügige Modifikation) einer unentscheidbaren Theorie dazu führen kann, daß sie danach entscheidbar ist. In praktischen Problemstellungen wird im allgemeinen nicht die Entscheidbarkeit der generellen Theorie benötigt, sondern nur die Entscheidbarkeit eines Fragmentes der Theorie. So treten z.B. im Leitbeispiel maximal quadratische Polynome auf. Es genügt also, einen zulässigen konkreten Gegenstandsbereich zu finden, der Ungleichungen über multivariaten quadratischen Polynomen bereitstellt.

Daher ist man allgemein an der Zulässigkeit bestimmter Fragmente der allgemeinen Theorie interessiert. Diese Fragmente können u.a. durch folgende Einschränkungen entstehen:

- die zugelassenen Funktionssymbole $(+, \cdot)$,
- die zugelassenen Vergleichsoperatoren $(=, \neq, <, \leq, >, \geq)$,

- die zugelassenen logischen Operatoren ($\vee, \wedge, \neg$),
- die erlaubten Quantoren ($\forall, \exists$),
- den Grad $\deg(p)$ der Polynome,
- die Anzahl n der variablen $x, y, \ldots$ eine Polynoms, oder
- die Anzahl der Quantorenwechsel $\forall\exists\forall$.

Wenn bereits die allgemeine Theorie entscheidbar ist, genügt es zu zeigen, daß das Fragment abgeschlossen unter Negation ist und ein Prädikat für $\top$ enthält.

Lösbarkeit von Ungleichungssystemen

Weiter oben wurde bereits angedeutet, daß sich die Lösbarkeit von Ungleichungssystemen in der Theorie der elementaren Algebra resp. den anderen Theorien ausdrücken lässt. Wegen der Bedeutung für die in Abschnitt 2 formulierten Anforderungen; diese wurden über logischen Sätzen über (nicht)linearen, multivariaten Ungleichungssystemen formuliert, nicht über Theorien; soll diese Tatsache hier detailliert werden.

Lemma 4.2
Die Theorie der elementaren Algebra umfaßt beliebige Sätze der Prädikatenlogik erster Stufe über Ungleichungssystemen über multivariaten Polynomen mit rationalen Koeffizienten.

Beweis. Multivariate Polynome $p(x_1, \ldots, x_n) \in \mathbb{Q}[x_1, \ldots, x_n]$ sind Terme der elementaren Algebra. Mit den logischen Junktoren und $<$ lassen sich die anderen Vergleichsoperatoren definieren. □

Definition 4.2 (Existentielle Sätze, Sätze der Ordnung n)
Unter **existentiellen Sätzen** sollen Sätze der Form

$$\overbrace{\exists \ldots \exists}^{n} x_1, \ldots, x_n \varphi$$

φ offene Formel verstanden werden, d.h. alle Variablen des Satzes sind existentiell quantifiziert.

Sätze der Ordnung n sind Sätze, die nur Terme enthalten, deren Grad kleiner gleich n ist.

Wie das folgende Lemma zeigt, läßt sich die Lösbarkeit eines Ungleichungssystems über $\mathbb{Q}[x_1, \ldots, x_n]$ auf einfache Art und Weise in der Sprache der elementaren Algebra der reellen Zahlen ausdrücken.

Lemma 4.3
Die Lösbarkeit eines Ungleichungssystemes $\mathcal{U} = \{g_1, \ldots, g_m\}$ kann in der Sprache der elementaren Algebra ausgedrückt werden.

Beweis. Enthalte $\mathcal{U}$ die Variablen $x_1, \ldots, x_n$. Dann ist $\mathcal{U}$ lösbar gdw. der existentielle Satz

$$\overbrace{\exists \ldots \exists}^{n} x_1, \ldots, x_n \; g_1 \wedge \cdots \wedge g_m$$

wahr ist. □

Satz 4.4
Die Lösbarkeit eines Ungleichungssystem $\mathcal{U} = \{g_1, \ldots, g_m\}$ über den Polynomen $p_1, \ldots, p_m, q_1, \ldots, q_m$ in den Variablen $x_1, \ldots, x_n$ ist entscheidbar.

Beweis. Folgt trivialerweise aus Lemma 4.3 und Satz 4.1. □

Damit ist klar, daß die Anforderung 2.5 aus Kapitel 2, die verlangt, logische Formeln über (nicht)linearen Ungleichungen zwischen numerischen Attributen beschreiben zu können, erfüllt werden kann.

Komplexität der Verfahren

Ein weiterer Grund zur Betrachtung von Fragmenten einer allgemeinen Theorie ist die Komplexität der Entscheidungsverfahren. Der Beweis von Tarski zur Entscheidbarkeit der Theorie der elementaren Algebra ist konstruktiv, d.h. es wird ein Verfahren angegeben mit dem die Gültigkeit beliebiger Formeln überprüft werden kann, und nicht nur die generelle Entscheidbarkeit bewiesen. Die Komplexität des Verfahrens ist allerdings hyperexponentiell, d.h. es hat keine durch einen endlichen Turm von Exponentionen beschreibbare obere Schranke. Es ist daher für realistische Aufgabenstellungen nicht einsetzbar.

Das im Algorithmus von Tarski verwendete Verfahren ist ein Quantoreneliminationsverfahren. In diesem Verfahren wird eine Formel solange durch bestimmte Regeln in äquivalente Formeln umgewandelt, bis man eine zur ursprünglichen Formel äquivalente, quantorenfreie Formel erhält. Es kann dann leicht überprüft werden, ob die Formel wahr ist.

Im Lauf der Jahrzehnte seit Tarskis Beweis entwickelte sich die Untersuchung von Quantoreneliminationsverfahren zu einem zentralen Gegenstand der algorithmischen Algebra[Cha93]. Es sind zwei Forschungsrichtungen zu beobachten:

Zum einen wurden (insbesondere im letzten Jahrzehnt) die generischen Verfahren zur Quantorenelimination stark verbessert. Zu nennen sind hier insbesondere das Verfahren zur zylindrisch algebraischen Dekomposition (CAD) [Col75, CH91] und die Verbesserung von Hong zur partiellen CAD [Hon93], die beide doppelt exponentiell in der Anzahl der Variablen sind.

Des weiteren gibt es Verfahren, die zwar einfach exponentiell in der Anzahl der Variablen und doppelt exponentiell in der Zahl der Quantorenwechsel sind, aber für jede sinnvolle Eingabe ein schlechteres Zeitverhalten als die doppelt exponentiellen Verfahren zeigen. Die Komplexität der Theorie $\langle\langle \mathbb{Z}, +, =, <, 0, 1\rangle\rangle$ ist ebenfalls doppelt exponentiell [HU79].

Zum anderen wurden optimierte Verfahren für Fragmente der Theorie entwickelt. Zu nennen sind hier besonders die Verfahren zur Quantorenelimination über linearen Ungleichungssystemen. Insbesondere das Fourier-Motzkin Verfahren wird in weiten Bereichen des Constraint Logic Programming zur Projektion bzw. Variablenelimination benutzt [Las90].

Im Bereich der nichtlinearen Ungleichungssysteme sind die Elimination Set Verfahren die von Weispfenning entwickelt wurden besonders interessant. Ausgehend von der Zielsetzung, Lösungsverfahren für real interessante Fragestellungen wie Geometriebeweise und Simulation zu finden, entwickelte Weispfenning Verfahren u.a. für die quadratische Elimination [Wei96], die von Dolzmann [DS95, DS96] im System Redlog realisiert wurden. Auch die Elimination Set Verfahren sind Quantoreneliminationsverfahren und können daher zur Anzeige der generierten Modelle verwandt werden.

Quantoreneliminationsverfahren sind daher der aussichtsreichste Ansatz zur Realisierung ausdrucksstarker numerischer konkreter Gegenstandsbereiche. Nicht in allen Anwendungsfällen werden allerdings zulässige konkrete Gegenstandsbereiche benötigt, die die linearen Ungleichungssysteme beinhalten. Daher sollen im folgenden eine Reihe von zulässigen konkreten Gegenstandsbereichen unterschiedlicher Ausdrucksstärke vorgestellt werden.

4.3.3. Zulässige numerische konkrete Gegenstandsbereiche

Nachdem im letzten Abschnitt der theoretische Rahmen geschaffen wurde, sollen in diesem Abschnitt eine Reihe von Theoriefragmenten vorgestellt werden, die zulässige konkrete Gegenstandsbereiche darstellen [Kam97b].

Beginnend beim ausdrucksschwächsten Gegenstandsbereich wird die Ausdrucksstärke sukzessive gesteigert.

Beispiel 4.1
In diesem Abschnitt soll noch nicht untersucht werden, wie sich die Anforderungen des Leitbeispiels an die Repräsentation mit Hilfe der unterschiedlichen konkreten Gegenstandsbereiche umsetzen lassen. Dies ist Inhalt des nächsten Kapitels. Statt dessen wurde ein Anwendungsfall aus dem Bereich der Recherche in bibliographischen Datenbanken gewählt, der auch in [Kam97b] verwandt wurde..

Vergleich mit Konstanten

Der ausdrucksschwächste zulässige konkrete Gegenstandsbereich besteht lediglich aus einem Prädikat für $\top$ und seiner Negation, d.h.

$$\mathbf{D_0} = \langle \Delta_{\mathbf{D}}, \top, \bot \rangle$$

Abbildung 4.1.

Die Ausdrucksstärke des konkreten Gegenstandsbereiches D_0

Augenscheinlich hat dieser zulässige konkrete Gegenstandsbereich keine sinnvolle Verwendung. Um einsetzbar zu sein, muß ein konkreter Gegenstandsbereich mindestens einige unäre Prädikate enthalten, die einen Vergleich mit Konstanten des Gegenstandsbereiches erlauben. Daher ist der ausdrucksschwächste sinnvoll einsetzbare zulässige konkrete Gegenstandsbereich der folgende:

$$\mathbf{N_1} = \langle \Delta_{\mathbf{N}}, \top, \bot, =_n, \neq_n, n \in \Delta_{\mathbf{N}} \rangle, \Delta_{\mathbf{N}} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}\}.$$

Beispiel 4.2
Bereits dieser konkrete Gegenstandsbereich erlaubt es, etwa das Konzept »*Eine Publikation ist etwas mit einem Publikationsjahr*« durch die Terminologie in 4.3.1 zu beschreiben und eine Anfrage nach »*Alle Publikationen im Jahre 1995*« an das System durch den Konzeptterm: (and publication (constrain year (number (?x) (=1995 ?x)))) zu formulieren.

```
(define-primitive-attribute year)

(define-primitive-concept publication
  (and (constrain year top)))
```

TBox 4.3.1

Numerische Gegenstandsbereiche (I)

Ist der Gegenstandsbereich total geordnet – das sind alle numerischen Grundkörper außer $\mathbb{C}$– setzt man eher einen konkreten Gegenstandsbereich ein, der beliebige Vergleiche zwischen Variablen und Konstanten zuläßt, d.h. einen Gegenstandsbereich:

Abbildung 4.2.

Die Ausdrucksstärke des konkreten Gegenstandsbereiches N_1

$$\mathbf{N_2} = \langle \Delta_{\mathbf{N}}, \top, \bot, =_n, \neq_n, <_n, \leq_n, >_n, \geq_n, n \in \Delta_{\mathbf{N}} \rangle, \Delta_{\mathbf{N}} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$$

Beispiel 4.3
$\mathbf{N_2}$ erlaubt es, die Definition einer Publikation um ein Attribut für die Seitenzahl zu erweitern, sowie eine Veröffentlichung in einem Proceedingsband als Spezialisierung einer Publikation mit Attributen für die Startseite und die Endseite zu definieren (siehe 4.3.2).

```
(define-constraint x>=1 ((?x) (number) (>= 1 ?x)))

(define-primitive-attribute year)
(define-primitive-attribute pages)
(define-primitive-concept publication
  (and
    (constrain year top)
    (constrain pages x>=1)))

(define-primitive-attribute start-page)
(define-primitive-attribute end-page)
(define-concept inproceedings
  (and publication
    (constrain start-page x>=1)
    (constrain end-page x>=1)))
```

TBox 4.3.2

Numerische Gegenstandsbereiche (II)

Über diesem zulässigen konkreten Gegenstandsbereich lassen sich dann Abfragen wie: »*Alle Publikationen zwischen 1992 und 1994*« oder »*Alle Publikationen mit mehr als 20 Seiten*« durch die folgenden Konzeptterme beschreiben:

Der selbe konkrete Gegenstandsbereich kann auch dafür genutzt werden, um andere Attribute wie »Volume Number« etc. zu beschreiben. Insgesamt hat er also

```
(and publication
   (constrain year ((?x) (number) (>= ?x 1992)))
   (constrain year ((?x) (number) (<= ?x 1994))))

(and publication
   (constrain pages (number (?x) (> ?x 20))))
```

TBox 4.3.3

Numerische Gegenstandsbereiche (IIb)

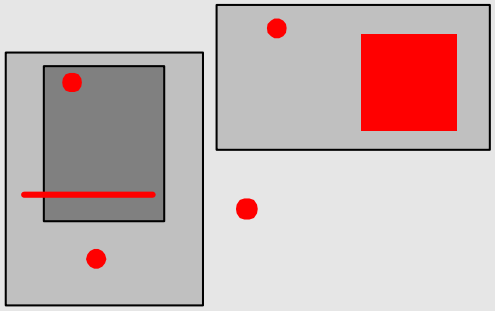


Abbildung 4.3.

Die Ausdrucksstärke des konkreten Gegenstandsbereiches N_2

eine große Zahl sinnvoller Einsatzmöglichkeiten. Daher wird dieser zulässige konkrete Gegenstandsbereich von einigen beschreibungslogischen Systemen, wie z.B. von Classic eingesetzt.

Wenn allerdings Hintergrundwissen beschrieben werden soll, werden ausdrucksstärkere konkrete Gegenstandsbereiche benötigt. Sowohl für die Konzeptbeschreibungen als auch für das Retrieval. So können zum Beispiel die folgenden einfachen Zusammenhänge in $\mathbf{N_2}$ nicht repräsentiert werden: »*Die Seitennummer der letzten Seite einer Inproceedings-Veröffentlichung ist größer oder gleich der Seitennummer der ersten Seite dieser Veröffentlichung*« bzw. »*Die Anzahl der Seiten einer Inproceedings-Veröffentlichung ist gleich der Differenz zwischen Seitennummer der letzten Seite und Seitennummer der ersten Seite plus 1*«.

Im Folgenden werden weitere zulässige konkrete Gegenstandsbereiche vorgestellt, die die Repräsentation dieser Fakten erlauben.

Betrachtet man die beiden zulässigen konkreten Gegenstandsbereiche $\mathbf{N_1}$ und $\mathbf{N_2}$ genauer, so stellt man fest, daß sie im Falle von $\Delta_{\mathbf{N}} = \mathbb{R}$ extrem eingeschränkte Fragmente der Theorie der elementaren Algebra darstellen.

$\mathbf{N_2}$ ist das Fragment, das keine Funktionsymbole, logischen Operatoren und Quantoren enthält. Da keine Funktionssymbole zur Verfügung stehen, sind alle Polynome univariat. Zudem sind nur Anwendungen der Vergleichsoperatoren in der Form $\circ(x, n)$ erlaubt, d.h. der zweite Subterm muß eine Konstante sein. $\mathbf{N_1}$ ist die Einschränkung von $\mathbf{N_2}$ auf die Vergleichoperatoren $=$ und $\neq$.

Wohlordnung

Der erste Schritt zur Verallgemeinerung von $\mathbf{N_2}$ ist die Aufhebung der Einschränkung, daß der zweite Subterm (d.h. die rechte Seite der Gleichung / Ungleichung) ein Grundterm sein muß. Daraus resultiert der folgende zulässige konkrete Gegenstandsbereich:

$$\mathbf{N_3} = \langle\langle \Delta_{\mathbf{N}}, < \rangle\rangle, \Delta_{\mathbf{N}} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$$

Beispiel 4.4
Im Beispiel ist es so möglich, die Tatsache: »*Die Seitenzahl der letzten Seite eine Veröffentlichung ist größer oder gleich der Seitenzahl der ersten Seite dieser Veröffentlichung*« auszudrücken (siehe Terminologie 4.3.4). Durch die Verankerung dieses Hintergrundwissens wird garantiert, daß alle Publikationen in der Daten/Wissensbasis korrekte Seitenbereiche haben, und daß fehlerhafte Einträge automatisch erkannt werden.

TBox 4.3.4

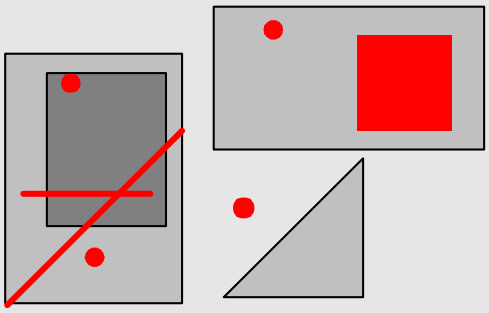
Numerische Gegenstandsbereiche (III)

```
(define-constraint x>=y
   ((?x ?y) (number) (>= ?x ?y)))

(define-primitive-attribute start-page)
(define-primitive-attribute end-page)
(define-concept inproceedings
   (and publication
      (constrain start-page x>=1)
      (constrain end-page x>=1)
      (constrain end-page start-page x>=y)))
```

Abbildung 4.4.

Die Ausdrucksstärke des konkreten Gegenstandsbereiches N_3



Dieser zulässige konkrete Gegenstandsbereich ist allgemein auch unter dem Namen »Wohlordnung« bekannt und in den beschreibungslogischen Systemen KRIS und TAXON implementiert. Er stellt das Maximum an Ausdrucksstärke dar, das

in diesen beiden Systemen behandelt werden kann. $\mathbf{N_3'} = \langle\langle \mathbb{Q}, < \rangle\rangle$ ist zugleich der ausdrucksschwächste konkrete Gegenstandsbereich, der an CTL angebunden wurde.

Lineare Sätze über Polynomen

Obwohl $\mathbf{N_3}$ ohne Zweifel bereits sinnvoll eingesetzt werden kann, ist er nicht ausdrucksstark genug, um z.B. die im letzten Abschnitt angeführte Beziehung zwischen der Anzahl der Seiten und den Seitenzahlen der ersten und letzten Seite auszudrücken. Der Grund hierfür liegt darin, daß durch den Verzicht auf Funktionen nur univariate lineare Polynome zulässige Terme sind. Zur Darstellung der obigen Beziehung wird aber ein multivariates lineares Polynom: $x-y+1$ benötigt.

Der folgende zulässige konkrete Gegenstandsbereich stellt beliebige multivariate lineare Polynome als Terme zur Verfügung und ist daher ausdrucksstark genug um auch die obige Beziehung darzustellen.

$$\begin{aligned}\mathbf{N_4} &= \langle\langle \Delta_{\mathbf{N}}, +, =, <, 0, 1 \rangle\rangle, \Delta_{\mathbf{N}} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\} \\ &= \langle\langle \Delta_{\mathbf{N}}, +, \cdot, =, <, 0, 1 \rangle\rangle, \deg(p) \leq 1\end{aligned}$$

Die Realisierung der Entscheidungsverfahren kann einerseits mit Hilfe von Verfahren und Systemen aus dem Bereich des CLP[20] erfolgen, in die selbst wieder numerische oder algebraische Verfahren aus der Mathematik eingeflossen sind. Andererseits können direkt Systeme aus der Computeralgebra verwandt werden. Zu nennen sind hier insbesondere die beiden folgenden Verfahren, die über $\mathbb{R}$ operieren:

- Gauss-Jordan Eliminationsverfahren,
- Fourier-Motzkin Eliminationsverfahren.

Da es beides Eliminationsverfahren sind, können sie für die Ermittlung der zulässigen Parameterwerte bei der Anzeige des generierten Modells eingesetzt werden.

In CTL wird der konkrete Gegenstandsbereich $N_4' = \langle\langle \mathbb{R}, +, \cdot, =, <, 0, 1 \rangle\rangle$, $\deg(p) \leq 1$ realisiert, indem ein in Lisp realisiertes CLP(R)-System namens Epilytis angebunden wird, Das CLP(R)-System selbst verwendet wiederum das Gauss-Jordan-Verfahren als internes Verfahren.

Sollte ein konkreter Gegenstandsbereich benötigt werden, der Lösungen über $\mathbb{N}$ resp. $\mathbb{Z}$ statt über $\mathbb{R}$ liefert, so ist es möglich, das Computeralgebra-System Presburger über die Schnittstelle anzubinden. Presburger ist ein auf dem allgemeinen

20. CLP – Constraint Logic Programming.

Computeralgebrasystem REDUCE aufbauendes, an der Universität Passau entwickeltes System, das ein Entscheidungsverfahren für die Presburger-Arithmetik bereitstellt.

Beispiel 4.5
Mit Hilfe von $\mathbf{N_4}$ wird es möglich, die Beziehung zwischen der Seitenanzahl und den Seitenzahlen der ersten und der letzten Seite wie in Terminologie 4.3.5 gezeigt zu beschreiben.

TBox 4.3.5

Numerische Gegenstandsbereiche (IV)

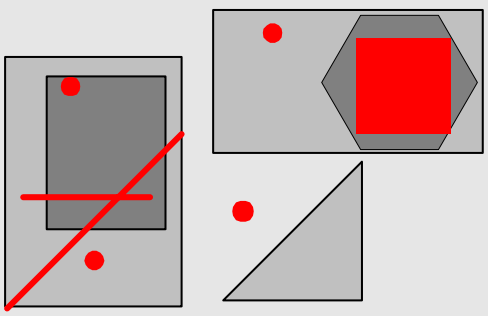
```
(define-constraint x>=y
   ((?x ?y) (number number) (>= ?x ?y)))
(define-constraint x=y-z+1
   ((?x ?y ?z) (number number number) (= ?x (+ (- ?y ?z) 1))))

(define-primitive-attribute start-page)
(define-primitive-attribute end-page)

(define-concept inproceedings
   (and publication
      (constrain start-page x>=1)
      (constrain end-page x>=1)
      (constrain end-page start-page x>=y)
      (constrain pages end-page start-page x=y-z+1)))
```

Abbildung 4.5.

Die Ausdrucksstärke des konkreten Gegenstandsbereiches N_4



Nichtlineare Sätze über Polynomen

Wie wir gesehen haben werden zur Repräsentation des Leitbeispiels multivariate nichtlinearer Polynome, genauer nichtlinearer quadratischer Polynome, benötigt.

Der Gegenstandsbereich $\mathbf{N_5}$ stellt genau diese Ausdrucksstärke zur Verfügung.

$$\mathbf{N_5} = \langle\langle \mathbb{R}, +, \cdot, =, <, 0, 1\rangle\rangle, \deg(p) \leq 2.$$

Realisiert werden kann dieser konkrete Gegenstandsbereich z.B. durch die Anbindung des Computeralgebra Systems REDLOG [DS96]. Dieses System stellt eine Entscheidungsprozedur für genau dieses Theoriefragment über $\mathbb{R}$ zur Verfügung, indem es ein Elimination Set Verfahren von Weispfenning implementiert.

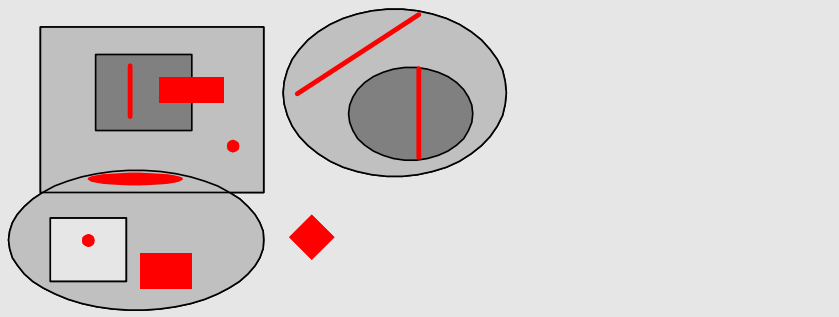


Abbildung 4.6.

Die Ausdrucksstärke des konkreten Gegenstandsbereiches N_5

Weitere Mögliche Anknüpfungspunkte für die Realisierung dieser konkreten Gegenstandsbereiche sind Simplikationsverfahren auf der Basis des Verfahrens von Buchberger zur Berechnung von Gröbner-Basen. Die Nachteile dieser Verfahren liegen allerdings darin, daß die Lösungen nur über algebraischen Körpern, d.h. $\mathbb{C}$ berechnet werden.

Über quadratische Ungleichungssysteme hinausgehende konkrete Gegenstandsbereiche wie

$$\mathbf{N_6} = \langle\langle \mathbb{R}, +, \cdot, =, <, 0, 1\rangle\rangle, \deg(p) \leq 3.$$

oder die volle Theorie der reell abgeschlossenen Körper $\mathbf{N_7}$=$\langle\langle \mathbb{R}, +, \cdot, =, <, 0, 1\rangle\rangle$ können dann realisiert werden, wenn Implementierungen der entsprechenden Entscheidungsprozeduren frei verfügbar sind. Ein Herausragender Kandidat ist das System Risc-CLP, das den Algorithmus von Hong zur partiellen zylindrisch algebraischen Dekomposition implementiert.

Beispiel 4.6
In der Beispieldomäne, dem Retrieval von bibliographischen Daten besteht im Gegensatz zum Leitbeispiel kein konkreter Bedarf an der Repräsentation nichtlinearer multivariater Polynome.

Die in diesem Abschnitt vorgestellte Vorgehensweise zur Realisierung konkreter Gegenstandsbereiche ist nicht auf den Bereich der numerischen konkreten Gegenstandsbereiche eingeschränkt. In Anhang C wird gezeigt, wie auf ähnliche Art und Weise eine Hierarchie von zulässigen konkreten Gegenstandsbereichen über Strings definiert werden kann.

4.4. RACER

CTL wurde aufbauend auf dem System TAXON in den Jahren 1996 und 1997 prototypisch implementiert, seine Weiterentwicklung ruht seitdem. Der Schwerpunkt lag dabei primär auf der Überprüfung der konzeptionellen Ideen zur Integration von Konkreten Gegenstandsbereichen und der Nutzung von Beschreibungslogiken im Anwendungsbereich Service und Support, weniger die tatsächliche praktische Einsetzbarkeit [21].

Nachdem zunehmend klarer wurde, daß die Ausdrucksstärke der zugrundeliegende beschreibungslogischen Systeme groß genug ist, um sie nicht nur im akademischen Umfeld zum praktischen Einsatz zu bringen, begann die Entwicklung von beschreibungslogischen Systemen der dritten Generation.

Hier standen Aspekte wie Stabilität des Gesamtsystems, Dokumentation und Optimierung der Performance für den »average case« weit stärker im Vordergrund als die Bereitstellung einer Plattform für die Überprüfung der theoretischen Ergebnisse.

Die beiden herausragenden Systeme der dritten Generation sind das an der University of Manchester (in Zusammenarbeit mit der RWTH Aachen) entwickelte System FaCT[22] [Hor99, Hor98] – mit seiner kommerziellen Variante Cerebra – und das an der Universität Hamburg entwickelte System RACER sowie dessen Vorgänger RACE[23] [HM02c, HM02b, HM02a, Moe01, HM01, HMT01, HMM01].

Im Gegensatz zu FaCT und Cerebra die nur TBox-Inferenzen bereitstellen [24], wurden RACE und RACER von Beginn so konzipiert um sowohl TBox- als auch ABox-Inferenzen zu unterstützen.

Ohne ABox-Inferenzen ist ein beschreibungslogisches System für den in dieser Arbeit betrachteten Anwendungsbereich aber nicht bzw. wenig geeignet.

Somit ist eine Konzentration auf RACER zur Untersuchung der Eignung beschreibungslogischer Systeme der dritten Generation mit CTL vollkommen ausreichend.

4.4.1. Abstrakter Gegenstandsbereich und Inferenzmaschine

RACER realisiert eine sehr ausdrucksstarke Beschreibungslogik auf Basis eines hoch optimierten Tableauxkalküls. Die Inferenzen werden sowohl für multiple TBoxen als auch multiple ABoxen zur Verfügung gestellt.

In der Version 1.7.6 [25] [HM02c] wird die Beschreibungslogik $\mathcal{ALCQHI}_{\mathcal{R}^+}$ (auch

21. Diese Eigenschaft hat CTL mit der überwiegenden Mehrzahl der beschreibungslogischen Systemen der »zweiten« Generation gemein.

22. FaCT – Fast Classification of Terminologies.

23. RACE – Reasoner for ABoxes and Concept Expressions.

24. Laut [MH03] unterstützt FaCT auch zu Beginn des Jahres 2003 noch keine ABox-Inferenzen.

$\mathcal{SHIQ}$), d.h. $\mathcal{ALC}$ erweitert um qualifizierte Anzahlbeschränkungen, Rollenhierarchien, inverse und transitive Rollen.

RACER erlaubt die Definition von »generalisierten TBoxen«. Diese unterscheiden sich von den in dieser Arbeit verwendeten TBoxen u.a. dadurch, daß sie die Definition von »general concept inclusions« erlauben, die Subsumptionsbeziehung zwischen zwei Konzept*termen* beschreiben. Weiterhin sind in generalisierten TBoxen multiple Definitionen von Konzepten sowie zyklische Definitionen möglich [26].

In der Beschreibungssprache von RACER fehlen von den von CTL unterstützten Konstrukten lediglich Feature-Agreementsments und Feature-Disagreements sowie Rollen- und Featureketten. Beide Konstrukte erlauben, wie im nächsten Kapitel gezeigt wird, die objektorientierte und modularisierte Repräsentation des Wissens, indem sie Mechanismen zur Konstruktion komplexer Mechanismen aus einfachen Modulen erlauben.

Während evtl. noch auf Feature-(Dis)agreements verzichtet werden könnte, zeigt die Umsetzung des Leitbeispiels im nächsten Kapitel, daß Rollen- und Featureketten insbesondere im Zusammenspiel mit konkreten Gegenstandsbereichen essentiell für die Repräsentierbarkeit des Wissens im Gegenstandsbereich sind.

4.4.2. Konkrete Gegenstandsbereiche

Eine wesentliche Erweiterung von RACER im Vergleich zu seinem Vorgänger RACE ist die Unterstützung von konkreten Gegenstandsbereichen in Form von $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D})^-$, das ist die Erweiterung um konkrete Gegenstandsbereiche unter Verzicht auf Featureketten.

Dies hat seinen Grund darin, daß die Erweiterung der von RACER implementierten Beschreibungssprache $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D})^-$ um Featureketten unentscheidbar ist.

Wie in [Lut01b, Lut02b] gezeigt wird, ist bereits die Erfüllbarkeit von $\mathcal{ALC}(\mathcal{D})$-Konzepten bzgl. einer generalisierten TBox für relativ ausdrucksschwache konkrete Gegenstandsbereiche (wie z.B. $\mathbf{N_4}$) unentscheidbar.

Will man weiterhin generalisierte TBoxen beibehalten, kann die Entscheidbarkeit dadurch wiedergewonnen werden indem man entweder die Ausdrucksstärke der Beschreibungslogik – genauer, der Konstruktoren für den konkreten Gegenstandsbereich – einschränkt, oder die Ausdruckstärke der konkreten Gegenstandsbereiche einschränkt. Beide Ansätze wurden verfolgt:

- Durch die Einschränkung der Ausdrucksstärke der Beschreibungslogik über den Verzicht auf Featureketten läßt sich, wie in [HMM01] gezeigt wird, die Ent-

25. Anfang Mai 2003 ist dies die aktuell verfügbare Version von RACER.

26. Aus diesem Grunde werden die in dieser Arbeit verwendeten TBoxen auch als »azyklische TBoxen« bezeichnet.

scheidbarkeit für die Erfüllbarkeit von Konzepten bzgl. einer generalisierten TBox für $\mathcal{ALC}(\mathcal{D})$ bzw. $\mathcal{ALCHN}_{\mathcal{R}^+}(\mathcal{D})^-$ wieder herstellen.

- Für den Gegenstandsbereich $\mathbf{N_3}$ (über $\mathbb{Q}$) und den abstrakten Gegenstandsbereich $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D})^-$ ist die Entscheidbarkeit der Erfüllbarkeit von Konzepten bzgl. einer generalisierten TBox entscheidbar (genauer: sie liegt ist NExpTime-vollständig) [27].

RACER nimmt diese beiden Ergebnisse praktisch auf, und stellt numerische konkrete Gegenstandsbereiche für lineare (Un)gleichungssysteme über den reellen Zahlen und min/max Einschränkungen über den ganzen Zahlen bereit.

Will man, wie in dem in dieser Arbeit betrachteten Kontext sowohl Featureketten verwenden als auch ausdrucksstarke konkrete Gegenstandsbereiche einsetzen, so ist man auf die Verwendung azyklischer TBoxen, wie sie bereits in CTL bereitgestellt werden eingeschränkt.

4.4.3. Architektur

RACER ist in CommonLisp implementiert und steht sowohl als Stand-Alone Version als auch als kompilierte Lisp Datei für die meisten CommonLisp-Systeme zur Verfügung.

Die Stand-Alone Version von RACER wird auch als RACER-Server bezeichnet und stellt ein echtes Client/Server System dar. Clients können sich über die folgenden Wege an den Server konnektieren:

- Dateischnittstelle,
- TCP-Socketschnittstelle,
- HTTP-Schnittstelle.

Über die Dateischnittstelle können sowohl die zu betrachtende T- und ABox als auch eine Menge von Anfragen an den Server übergeben werden. Die Ergebnisse der Anfragen werden je nach Option entweder auf der Konsole oder in einer Datei ausgegeben. Unterstützt werden sowohl die KRSS-Syntax als auch RDF[28], DAML[29] und DIG[30]-Syntax. Die zu verwendende Syntax wird anhand der Dateiendung erkannt.

27. In [Lut02a] wird die resultierende Beschreibungslogik auch $\mathbb{Q}$-$\mathcal{SHIQ}$ genannt.
28. RDF – Resource Description Framework.
29. DAML – DARPA Agent Markup Language.
30. DIG – Description Logic Implementation Group.

Die TCP-Socketschnittstelle stellt einen Lisp SocketStream zur Verfügung, auf den die Clients schreiben und von dem sie lesen können. Mit Racer wir ein Java-basierter Client zur Verfügung gestellt, der die TCP-Schnittstelle nutzt und die Racer Funktionalität in Java-Methoden kapselt.

Die HTTP-Schnittstelle nutzt den Lisp-basierten HTTP-Server CL-HTTP und stellt über HTTP-als Transportebene ein XML-basiertes Protokoll nach der DIG-Spezifikation. DIG lehnt sich an die ebenfalls XML-basierten Standards für Webservices SOAP[31] und Remote Procedure Calls XML-RPC[32] an, implementiert allerdings ein eigenes Protokoll.

Die konkreten Gegenstandsbereiche sind ebenfalls in Lisp realisiert und Bestandteil des monolithischen RACER-Server.

4.4.4. Fazit RACER

RACER ist das beschreibungslogisches System der dritten Generation, welches eine sehr ausdrucksstarke Beschreibungssprache und sowohl T- als auch ABox-Inferenzen über dieser Beschreibungssprache zur Verfügung stellt. Darüberhinaus implementiert RACER zwei numerische konkrete Gegenstandsbereiche. RACER ist zusätzlich hocheffizient implementiert und genügt den Ansprüchen an ein kommerzielles Produkt. Zudem unterstützt es die wesentlichen im Rahmen des ≫sematischen Webs≪ definierten Anfrage-und Beschreibungssprachen.

Wesentliches Manko der aktuellen RACER Version im Vergleich zu CTL ist die fehlende Unterstützung von Featureketten, auf die zugunsten der anderen Konstrukte verzichtet wurde. Das Fehlen von Featureketten verhindert die objektorientierte Repräsentation des Hintergrundwissens und damit die Einsetzbarkeit von RACER in dem in dieser Arbeit betrachteten Szenario.

Dieses Manko soll allerdings in zukünftigen Versionen von RACER ebenso behoben werden wie ein konkreter Gegenstandsbereich für nicht-lineare multivariate Ungleichungssysteme eingebunden werden soll [HM02c, HM02b].Konkret werden die folgenden Erweiterungen von RACER vorbereitet:

≫Future releases of RACER will provide:

- Role equality (in particular for the DAML interface),
- Feature chains for $\mathcal{ALC}(\mathcal{D})$ knowledge bases,
- Feature chain equality for $\mathcal{ALCF}(\mathcal{D})$ knowledge bases,
- A built-in domain for strings,
- A built-in concrete domain for non-linear multivariate polynomial (in-)equations≪.

31. SOAP – Simple Object Access Protocol.

32. XML-RPC – XML - Remote Procedure Call.

Bereits fertiggestellt, aber noch nicht integriert [33] ist der konkrete Gegenstandsbereich für nicht-lineare multivariate Ungleichungssysteme, allerdings über $\mathbb{C}$ basierend auf einem Gröbner-Basen Algorithmus.

Featureketten können ohne Einschränkungen an die Ausdrucksstärke des konkreten Gegenstandsbereiches prinzipiell nur unter Verzicht auf generalisierte TBoxen verwendet werden. Damit kann RACER im Vergleich zu CTL keine drastisch ausdrucksstärkere Beschreibungslogiken anbieten, bereits die mit CTL realisierte Ausdruckstärke ist sehr nahe am theoretisch machbaren angelangt.

Damit ist, nach einer Integration von Featureketten, RACER das System der Wahl für den in dieser Arbeit betrachteten Kontext. Es stellt im Vergleich zu CTL eine wesentlich modernere Systemarchitektur zur Verfügung und ist hoch optimiert. Um die Verwendbarkeit im nicht-wissenschaftlichen Umfeld zu fördern, ist es allerdings dringend geraten, eine Standard-Webserviceschnittstelle zu RACER zur Verfügung zu stellen und den Server threadsave zu implementieren. Letzteres wird bereits durch den RACER-Proxy, der gleichzeitig Loadbalancer ist, angestrebt.

Empfehlenswert, wäre es auch, zu prüfen, inwieweit eine Webservices-basierte Schnittstelle zu konkreten Gegenstandsbereichen möglich ist.

Ein zweiter wesentlicher Unterschied zwischen RACER und CTL ist die fehlende Möglichkeit zur Darstellung der generierten Modelle, ein für den in dieser Arbeit betrachteten Kontext unverzichtbares Feature von CTL. Möglich ist die Beschreibung einzelner Instanzen.

Inwieweit geplant ist, eine generelle Anzeige der generierten Modelle in eine der nächsten Versionen von RACER zu integrieren, ist nicht bekannt.

4.5. Zusammenfassung

CTL stellt die zur Umsetzung der in Kapitel 2 postulierten Anforderungen notwendigen Mechanismen zur Verfügung.

Die Ausdrucksstärke des abstrakten Gegenstandsbereiches in Verbindung mit der Einbindung von Mechanismen zur Anzeige der generierten Modelle, insbesondere dann, wenn über Quantoreneliminationsverfahren auch die konkreten Gegenstandsbereiche mit einbezogen werden können, lassen ein deduktives Informationssystem entstehen, das ein viel breiteres Anwendungsgebiet als die klassische fallbasierte Wartungsunterstützung hat.

Die flexible Anbindung konkreter Gegenstandsbereiche erlaubt es, genau die jeweils benötigte Ausdrucksstärke des konkreten Gegenstandsbereiches zur Verfügung zu stellen und bildet eine ideale Basis zur Verteilung der Applikation, z.B. über Webservices.

33. Mdl. Kommunikation mit Ralf Möller.

Interpretiert man CTL als ein generisches constraint-basiertes Datenbanksystem, verallgemeinert es CLP(R) Systeme, indem es uniforme Möglichkeiten zur strukturierten Repräsentation und mächtige Inferenzen bietet.

Während R prototypisch implementiert wurde, ist RACER das beschreibungslogisches System der dritten Generation, welches eine sehr ausdrucksstarke Beschreibungssprache und sowohl T- als auch ABox-Inferenzen über dieser Beschreibungssprache zur Verfügung stellt. Darüberhinaus implementiert RACER zwei numerische konkrete Gegenstandsbereiche.

RACER ist zusätzlich hocheffizient implementiert und genügt den Ansprüchen an ein kommerzielles Produkt. Allerdings fehlt RACER die Möglichkeit zur Repräsentation von Featureketten und Mechanismen zur Anzeige der generierten Modelle. Damit ist es mit der aktuellen Version von RACER weder möglich, den Gegenstandsbereich objektorientiert zu beschreiben, noch das Ergebnis der Inferenzen plastisch darstellen zu können.

Beide Komponenten sind, wie im nächsten Kapitel gezeigt wird, in dem betrachteten Gegenstandsbereich absolut unverzichtbar. In diesem Kapitel wird vorgestellt, wie neben der Bestimmung zusätzlicher Parameterwerte, die Simulation, die modellbasierte Diagnose und die Pflege und Wartung von Modellbibliotheken mit den Mechanismen von CTL realisiert werden können.

5 Umsetzung des Leitbeispiels mit CTL

Angestoßen wurde die Entwicklung von CTL durch den erhöhten Wissenrepräsentationsbedarf für die Unterstützung des Second-Level-Supports, wie er im Leitbeispiel in Kapitel 2 herausgearbeitet wurde. In diesem Kapitels wird gezeigt, daß CTL die dort postulierten Anforderungen an ein Second-Level-Support-System erfüllt.

Dazu wird zunächst in Abschnitt 5.1 gezeigt, wie das Wissen des Leitbeispiels mit den Mitteln von CTL repräsentiert werden kann. Neben der Beschreibung der Gerätestruktur wird gezeigt, wie auch das Hintergrundwissen und Verhaltensmodelle repräsentiert werden können.

Die folgenden Abschnitte zeigen dann, wie sich auf Grundlage des repräsentierten Wissens, insbesondere des Hintergrundwissens, und unter der Verwendung der Basisinferenzen, Konzeptklassifikation, Retrieval und Objektklassifikation, ein großer Teil der im Second-Level Support benötigten Anwendungen und Anforderungen realisieren lassen. Neben dem ähnlichkeitsbasierten Retrieval sind dies: Browsing und Retrieval, Simulation, What-If-Analyse, Konsistenzbasierte Diagnose und Wissensmanagement durch Pflege von Modellbibliotheken.

5.1. Wissensrepräsentation

5.1.1. Strukturwissen

Wie bereits in Kapitel 2 dargestellt wurde, werden zur adäquaten Repräsentation des Fahrradantriebes aus Abbildung 2.1 Ausdrucksmittel zur Beschreibung der unterschiedlichen Komponententypen eines Getriebes sowie Konstrukte zur Beschreibung der Verbindungsstruktur der Komponenten zum Gesamtgetriebe benötigt. Im Bereich der Kinematik haben sich Glieder und Kinematische Paare als Grundlagen zur Beschreibung dieses Wissens etabliert.

Glieder

Zur Beschreibung eines Fahrradantriebes reicht die Beschreibung von Rotationsgliedern (rotational-link) und Zuggliedern (tension-link) als Spezialisierungen allgemeiner Glieder (link) aus. Die Terminologie in TBox 5.1.1 beschreibt Glieder als Dinge, die eine Kraft (link.force) übertragen. Rotationsglieder sind spezielle Glieder, die zusätzlich noch einen Radius (rot.radius) und ein Drehmoment (rot.torque) aufweisen. Kraft und Drehmoment sind nicht negativ, der Radius strikt positiv. Zusätzlich werden noch eine Reihe von weiteren Komponententypen wie z.B. Zahnräder (gearwheel) definiert, ohne sie genauer zu beschreiben.

TBox 5.1.1

Glieder

```
(define-constraint x>=0
  ((?x) (Number) (>= ?x 0)))

(define-primitive-attribute link.force Top)
(define-primitive-concept link
  (and Top
    (constrain link.force x>=0)))

(define-constraint x>0
  ((?x) (Number) (> ?x 0)))

(define-primitive-attribute rot.radius Top)
(define-primitive-attribute rot.torque Top)
(define-primitive-concept rotational-link
  (and link
    (constrain rot.radius x>0)
    (constrain rot.torque x>=0)))

(define-primitive-concept tension-link link)
(define-primitive-concept crank rotational-link)
(define-primitive-concept wheel rotational-link)
(define-primitive-concept gearwheel rotational-link)
(define-primitive-concept spindle rotational-link)
(define-primitive-concept chain tension-link)
```

Kinematische Paare

Zur Beschreibung der Getriebestruktur werden in der Kinematik sogenannte »kinematische Paare« benutzt. Ein kinematisches Paar (kinematic-pair) beschreibt die Verbindung zweier Glieder (pair.link1 bzw. pair.link2). Je nach Art der Verbindung und den sich ergebenden Freiheitsgraden bei der Bewegung können Typen

von kinematischen Paaren unterschieden werden. Die Terminologie in TBox 5.1.2 beschränkt sich auf die zur Darstellung des Leitbeispiels benötigten Typen: Rotationspaare (rot-pair) und Paare zwischen einem Rotationsglied und einem Zugmittelglied (rot-tension-pair). Zur Beschreibung der Verbindung zwischen einem Rotationsglied und einem Zugmittelglied wird, da sie ungerichtet ist, eine Disjunktion benötigt. Da die Beschreibungssprache von CTL alle bool'schen Operatoren für Konzeptterme zur Verfügung stellt, ist dies problemlos möglich.

```
(define-primitive-attribute pair.link1 top)
(define-primitive-attribute pair.link2 top)
(define-primitive-concept kinematic-pair
  (and (all pair.link1 link)
    (all pair.link2 link)))

(define-concept rot-pair
  (and kinematic-pair
    (all pair.link1 rotational-link)
    (all pair.link2 rotational-link)))

(define-concept rot-tension-pair
  (and kinematic-pair
    (or
      (and (all pair.link1 rotational-link)(all pair.link2 tension-link))
      (and (all pair.link2 rotational-link)(all pair.link1 tension-link)))))
```

TBox 5.1.2

Kinematische Paare

5.1.2. Physikalische Gesetze

Wie in Kapitel 2 dargelegt ist in dem in dieser Arbeit untersuchten Anwendungsbereich die Repräsentation von Hintergrundwissen, etwa physikalischen Gesetzen, zwingend notwendig.

Die Tatsache, daß in terminologischen Systemen die Repräsentation dieses Wissens nicht möglich war, bildete den Ausgangspunkt zur Entwicklung von CTL. In CTL läßt sich das Drehmomentgesetz einfach durch die in TBox 5.1.3 auf der nächsten Seite dargestellte Modifikation des Konzeptes rotational-link repräsentieren.

Durch das Hinzufügen des Constraints ist es möglich, den Wert eines beliebigen dritten Parameters zu bestimmen, unabhängig davon, welches Parameterpaar vorgegeben wird und unabhängig von den konkreten Parameterwerten. Die Möglichkeiten zur Repräsentation von Hintergrundwissen sind einzig und allein durch die Ausdrucksstärke des verwendeten konkreten Gegenstandsbereiches begrenzt.

TBox 5.1.3

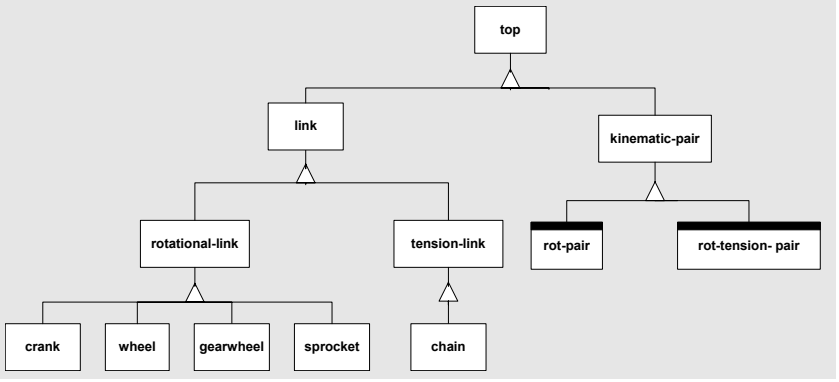
Physikalische Gesetze

```
(define-constraint x∗y=z
   ((?x ?y ?z) (number number number) (?x ∗ ?y = ?z)))

(define-primitive-concept rotational-link
   (and link
      (constrain rot.radius x>0)
      (constrain rot.torque x>=0)
      (constrain rot.radius link.force rot.torque x∗y=z)))
```

Abbildung 5.1.

Glieder und Kinematische Paare



Es ist klar, daß es auf Basis von Constraints ebenfalls möglich ist, mit Wertebereichseinschränkungen, wie ≫Das Drehmoment ist liegt zwischen 20 Nm und 40 Nm≪ etc. umzugehen. Die Einschränkung bzw. Bestimmung zusätzlicher Parameterwerte verbessert weiterhin zusätzlich die Retrievalqualität der ähnlichkeitsbasierten Retrievalverfahren aus Abschnitt 6.1.

In Abschnitt 5.3 wird gezeigt, wie es auf Basis der Möglichkeit, physikalische Gesetze durch Constraints zu definieren, möglich ist, das Verhalten eines Fahrradantriebes zu simulieren.

5.1.3. Verhaltensmodelle

Zur Simulation und konsistenzbasierten Diagnose werden neben einer Beschreibung der Komponententypen und der Komponentenstruktur noch Verhaltensmodelle sowohl des Normalverhaltens als auch der verschiedenen Fehlverhalten benötigt. Die Terminologie in TBox 5.1.4 auf der nächsten Seite beschreibt das Normalverhalten von Rotations- und Rotations-Zug-Paaren (ok-rot-pair bzw. ok-rot-tension-pair): Während bei Rotationspaaren das Drehmoment übertragen wird, ist es bei Rotations-Zug-Paaren die Kraft.

```
(define-constraint x=y
  ((?x ?y) (Number Number) (?x = ?y)))

(define-concept ok-rot-pair
  (and rot-pair
    (constrain (compose pair.link1 rot.torque)
      (compose pair.link2 rot.torque) x=y)))

(define-concept ok-rot-tension-pair
  (and rot-tension-pair
    (constrain (compose pair.link1 link.force)
      (compose pair.link2 link.force) x=y)))
```

TBox 5.1.4

Das Normalverhalten der kinematischen Paare

Die Terminologie in TBox 5.1.5 auf der nächsten Seite zeigt exemplarisch einige Fehlverhalten von Rotationspaaren. Ein Rotationspaar rutscht durch (slipping-rot-pair), wenn sich die Drehmomente der beiden Glieder unterscheiden. Es ist gebrochen (broken-rot-pair), falls ein Drehmoment strikt positiv, das andere dagegen 0 ist. Auch hier wird zwingend die Disjunktion als Konzeptoperator benötigt.

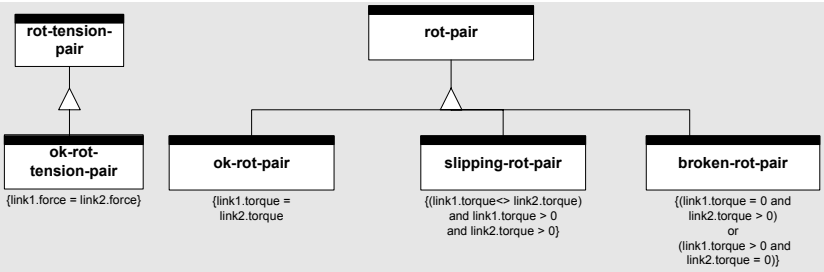


Abbildung 5.2.

Normal- und Fehlverhalten

Zusätzlich sind in der Terminologie in TBox 5.1.6 auf Seite 149 unterschiedliche Grade eines durchrutschenden Rotationspaares modelliert (strong-slipping-rot-pair bzw. weak-slipping-rot-pair).

Durch die Bereitstellung der Negation innerhalb der Beschreibungssprache wird es möglich, ein schwach durchrutschendes Rotationspaar einfach als Komplement eines stark durchrutschenden Paares zu modellieren.

Dadurch wird es nicht notwendig, eine komplexe Modellierung etwa über ein Constraint x>=0.7y durchzuführen. Dies reduziert etwaige Fehler bei der Modellierung ebenso wie den Aufwand bei einer Modifikation der Wissensbasis, etwa wenn der Grenzwert zwischen stark und schwach durchrutschenden Paaren verändert werden soll. Die Negation als Konzeptoperator ermöglicht es zudem,

```
(define-constraint x>y
  ((?x ?y) (number number) (?x > ∗ ?y)))

(define-concept slipping-rot-pair
  (and rot-pair
    (constrain (compose pair.link1 rot.torque) x>0)
    (constrain (compose pair.link2 rot.torque) x>0)
    (or
      (constrain (compose pair.link1 rot.torque)
        (compose pair.link2 rot.torque) x>y)
      (constrain (compose pair.link2 rot.torque)
        (compose pair.link1 rot.torque) x>y))))

(define-constraint x=0
  ((?x) (Number) (?x = 0)))

(define-concept broken-rot-pair
  (and rot-pair
    (or
      (and
        (constrain (compose pair.link2 rot.torque) x>0)
        (constrain (compose pair.link1 rot.torque) x=0))
      (and
        (constrain (compose pair.link1 rot.torque) x>0)
        (constrain (compose pair.link2 rot.torque) x=0)))))
```

TBox 5.1.5

Mögliche Fehlverhalten eines Rotationspaares (I)

unbekanntes Fehlverhalten als nicht bekanntes Fehlverhalten zu modellieren.

5.1.4. Zusammenfassung

Zur Repräsentation des Leitbeispiels, insbesondere zur Repräsentation des vorhandenen Hintergrundwissens, werden einerseits alle vorgestellten und in CTL definierten Konstrukte und Erweiterungen benötigt. Andererseits reichen die enthaltenen Konstrukte zur Repräsentation des vorhandenen Wissens weitestgehend aus, auf weitere, etwa in ausdrucksstärkeren Beschreibungslogiken vorhandene Konstrukte werden nicht unbedingt benötigt.

Daher soll an dieser Stelle auf eine detaillierte Darstellung weiterer Konzepte – z.B. der Definition von »Fahrradantrieb« und »Getriebe« – verzichtet werden.

Stattdessen wird in den nächsten Abschnitten der Fokus darauf gerichtet, in welcher Art und Weise die verschiedenen, in beschreibungslogischen Systemen definierten Inferenzen genutzt werden können, um auf Basis der vorhandenen Repräsentation Aufgaben aus dem Bereich des Second-Level-Supports, wie etwa

```
(define-constraint x<.3y
  ((?x ?y) (number number) (?x < 3/10 * ?y)))

(define-concept strong-slipping-rot-pair
  (and rot-pair
    (constrain (compose pair.link1 rot.torque) x>0)
    (constrain (compose pair.link2 rot.torque) x>0)
    (or
      (constrain (compose pair.link1 rot.torque)
        (compose pair.link2 rot.torque) x<.3y)
      (constrain (compose pair.link2 rot.torque)
        (compose pair.link1 rot.torque) x<.3y))))

(define-concept weak-slipping-rot-pair
  (and slipping-rot-pair (not strong-slipping-rot-pair)))
```

TBox 5.1.6

Mögliche Fehlverhalten eines Rotationspaares (II)

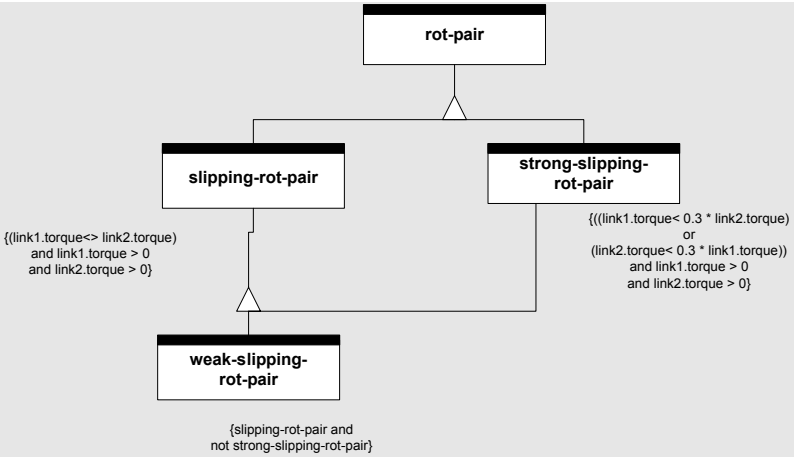


Abbildung 5.3.

Fehlverhalten (II)

Simulation, What-If-Analyse, Diagnose und Wissensmanagement zu realisieren.

5.2. Browsing, Retrieval, Vervollständigung

Bereits in den beiden vorhergehenden Kapiteln wurde deutlich gemacht, daß mit den verschiedenen Inferenzen der Objekt-, und der Konzeptklassifikation sowie des Retrievals eine Reihe von Möglichkeiten existieren, mit denen flexibel auf das in der Wissensbasis gespeicherte Wissen zugegriffen werden kann. Daher soll an dieser Stelle nur noch kurz auf die verschiedenen Nutzungsformen eingegangen

werden.

Die einfachste Nutzungsform der Wissensbasis ist das Browsing in der Wissensbasis. Hier werden dem Nutzer Möglichkeiten geboten entlang der Kanten im Graphen, d.h. entlang der Spezialisierungs- und der Instantiierungsbeziehungen zu navigieren. Ausgehend von einem Konzept kann er zu dessen direkten Vorgängern und direkten Nachfolgern navigieren, sich die Menge der Instanzen, die das Konzept erfüllen anzeigen lassen und die Details der einzelnen Instanzen ansehen.

Beim Retrieval definiert der Nutzer zunächst das ihn interessierende Konzept, bevor er die oben genannten Möglichkeiten zum Browsing in der Wissensbasis hat. Bei der Vervollständigung definiert er eine unvollständige Objektdefinition und läßt sich die durch die Wissensbasis definierten Einschränkungen an den offen gelassenen Parameterwerten berechnen.

Im folgenden sollen einige komplexere Beispiele gezeigt werden, wie diese Mechanismen zur Simulation, der What-If-Analyse und der modellbasierten Diagnose genutzt werden können.

5.3. Simulation, What-If Analyse

Die Verhaltenssimulation technischer Geräte ist eine eigenständige Aufgabenstellung. Sie spielt aber auch im Rahmen der Diagnose technischer Geräte eine wichtige Rolle, etwa um Hypothesen zu überprüfen (What-If-Analyse).

Darüber hinaus wird bei der Verhaltenssimulation eine große Anzahl von Parametern in Abhängigkeit von einer kleinen Menge von Startwerten berechnet; sie stellt damit einen Extrem- bzw. Idealfall für die im letzten Abschnitt angesprochene Vervollständigung von Objektbeschreibungen dar. In diesem Abschnitt soll gezeigt werden, wie der um die Anzeige der generierten Modelle erweiterte Konsistenztest von CTL zur Verhaltenssimulation eingesetzt werden kann.

5.3.1. Einfache Verhaltenssimulation

Ein Teil des Konsistenztests ist die Überprüfung der Erfüllbarkeit der Randbedingungen, die durch die Assertionen in der ABox impliziert werden. Dies gilt insbesondere für die Randbedingungen, die im Rahmen der Definitionen der Verhaltensmodelle und der Glieder spezifiziert wurden.

Wird etwa durch (state (instance link1 rotational-link)) zugesichert, daß link-1 ein Rotationsglied ist[1], so muß in einer konsistenten ABox das Drehmomentgesetz für link-1 gelten, sowie der Radius > 0 und die Kraft und das Drehmoment ≥ 0 sein. Um dies überprüfen zu können, wird im Lauf des Konsistenztests durch

1. Im Folgenden wird von der in TBox 5.1.3 auf Seite 146 vorgestellten, erweiterten Definition eines rotational-link ausgegangen.

den konkreten Gegenstandsbereich ein Constraintnetz aufgebaut, in dem all diese Randbedingungen automatisch eingetragen und propagiert werden.

Wird zusätzlich eine Menge von Parameterwerten vorgegeben (z.B. durch (state (related link1 10 link.force))), so werden diese Werte ebenfalls automatisch in dem Constraintnetz propagiert. Führen sie zu keinen Widersprüchen, ist die ABox also konsistent, so ergeben sich durch die Constraintpropagation Einschränkungen an den Wertebereich anderer Parameter. Für eine Reihe von Parametern können ggf. sogar eindeutige Werte ermittelt werden. Gibt man etwa für link-1 zwei beliebige der drei Parameter vor, so wird der Wert des dritten im Laufe des Konsistenztests ermittelt.

Da der erweiterte Konsistenztest von CTL es erlaubt, die berechneten Einschränkungen aus dem Constraintnetz auszulesen, kann auf diese Art und Weise Verhaltenssimulation erfolgen. Dabei ist es bei diesem Ansatz nicht notwendig, eine feste Menge von Eingabeparametern festzulegen. Im Rahmen des Konsistenztests werden für jede beliebige Kombination von Parameterwerten die Einschränkungen, die sich für die fehlenden Parameterwerte ergeben, automatisch ermittelt.

Da der Konsistenztest nach jeder ABox Assertion, d.h. nach jedem (state ...) angestoßen werden kann, ist es darüber hinaus möglich, die Festlegung von Parameterwerten (z.B. (state (related link1 0.175 rot.radius))) und Verhaltensmodellen (z.B. (state (instance pair1 ok-rot-pair))) interaktiv und inkrementell zu gestalten.

Damit ergibt insgesamt die folgende Vorgehensweise:

❶ Beschreibung der Gerätestruktur,

❷ Angabe der Startwerte einiger Parameter,

❸ Festlegung der Verhaltensmodelle,

❹ Angabe zusätzlicher Parameterwerte.

Beispiel 5.1 (Einfache Verhaltenssimulation)
Diese Vorgehensweise soll anhand des Leitbeispieles verdeutlicht werden.

❶ **Beschreibung der Gerätestruktur**

Auf Basis der in Abschnitt 5.1 eingeführten Terminologie ist es möglich, neben beliebigen anderen Rad-Ketten-Getrieben, den Fahrradantrieb aus Abbildung 2.1 auf Seite 33 zu beschreiben:

❶ In einem ersten Schritt werden die zur Beschreibung des Fahrradantriebes notwendigen Individuen, d.h. Glieder und kinematische Paare erzeugt (siehe ABox 5.3.1 auf der nächsten Seite).

❷ Dann werden sie den entsprechenden primitiven Konzepten zugeordnet (ABox 5.3.2 auf der nächsten Seite).

ABox 5.3.1

Beschreibung der Struktur des Fahrradantriebes (I)

```
(define-distinct-individual crankarm-1)
(define-distinct-individual chainring-1)
(define-distinct-individual bottom-bracket-spindle-1)
(define-distinct-individual chain-1)
(define-distinct-individual sprocket-1)
(define-distinct-individual rear-axle-1)
(define-distinct-individual rear-wheel-1)

(define-distinct-individual pair1)
(define-distinct-individual pair2)
(define-distinct-individual pair3)
(define-distinct-individual pair4)
(define-distinct-individual pair5)
(define-distinct-individual pair6)
```

ABox 5.3.2

Beschreibung der Struktur des Fahrradantriebes (II)

```
(state (and
    (instance crankarm-1 crank)
    (instance chainring-1 gearwheel)
    (instance bottom-bracket-spindle-1 spindle)
    (instance chain-1 chain)
    (instance sprocket-1 gearwheel)
    (instance rear-axle-1 spindle)
    (instance rear-wheel-1 rotational-link)
    (instance pair1 kinematic-pair)
    (instance pair2 kinematic-pair)
    (instance pair3 kinematic-pair)
    (instance pair4 kinematic-pair)
    (instance pair5 kinematic-pair)
    (instance pair6 kinematic-pair)))
```

❸ In einem dritten Schritt (ABox 5.3.3 auf der nächsten Seite) erfolgt die Zuordnung der Glieder zu den kinematischen Paaren und damit der Aufbau der Verbindungsstruktur.

Da zu diesem Zeitpunkt noch keine Verhaltensmodelle festgelegt sind, führt der Konsistenztest zur Berechnung der Parametereinschränkungen für beliebige Verhaltensmodelle (siehe Abb.5.4).

❷ **Angabe von Startwerten** Im Beispiel sollen die Radien des Kurbelarmes, des Kettenrades, des Ritzels und des Hinterrades festgelegt werden[2].

2. Die angegebenen Werte entsprechen in etwa der Realität.

```
(state (and
     (related pair1 crankarm-1 pair.link1)
     (related pair1 bottom-bracket-spindle-1 pair.link2)
     (related pair2 bottom-bracket-spindle-1 pair.link1)
     (related pair2 chainring-1 pair.link2)
     (related pair3 chainring-1 pair.link1)
     (related pair3 chain-1 pair.link2)
     (related pair4 chain-1 pair.link1)
     (related pair4 sprocket-1 pair.link2)
     (related pair5 sprocket-1 pair.link1)
     (related pair5 rear-axle-1 pair.link2)
     (related pair6 rear-axle-1 pair.link1)
     (related pair6 rear-wheel-1 pair.link2)))
```

ABox 5.3.3

Beschreibung der Struktur des Fahrradantriebes (III)

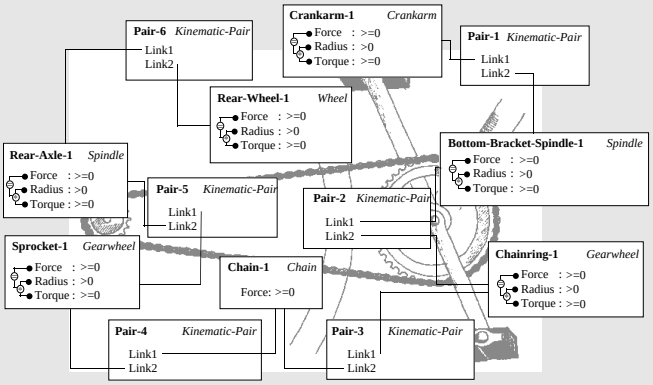


Abbildung 5.4.

Das berechnete Modell nach der Beschreibung der Struktur des Fahrradantriebes

❸ **Festlegung der Verhaltensmodelle** Im Beispiel soll angenommen werden, daß alle kinematischen Paare ihr Normalverhalten zeigen.

Die Festlegung der Verhaltensmodelle fügt einige neue Constraints in das Constraintnetz ein. Der Aufruf des Konsistenztests führt nicht zur Ableitung weiterer Informationen. Dies hat seinen Grund darin, daß noch nicht ausreichend viele Parameter festgelegt wurden (siehe Abb.5.5).

❹ **Festlegung weiterer Parameter** Durch Angabe eines zusätzlichen Parameterwertes, z.B. der Kraft die auf den Kurbelarm einwirkt (siehe ABox 5.3.6 auf Seite 155), sind ausreichend viele Information vorhanden, um die Propagierung der Constraints durch das Constraintnetz anzustoßen.

Ein Aufruf des Konsistenztestes führt dazu, daß die Werte fast aller fehlenden Parameter eindeutig festgelegt werden können (Abb. 5.6).

Einzig für Hinterradachse und Tretlagerwelle können Kraft und Drehmoment nicht wei-

ABox 5.3.4

Vorgabe einiger Startwerte

```
(state (and
      (related crankarm-1 0.175 rot.radius)
      (related chainring-1 0.1 rot.radius)
      (related sprocket-1 0.05 rot.radius)
      (related rear-wheel-1 0.6858 rot.radius)))
```

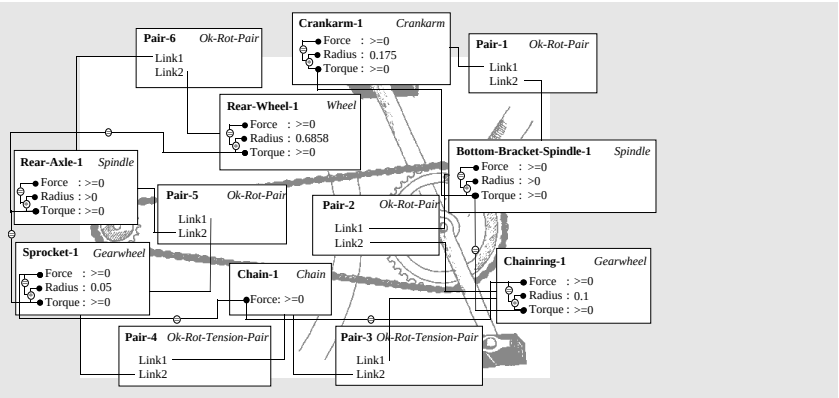
ABox 5.3.5

Festlegung der Verhaltensmodelle

```
(state (and
      (instance pair1 ok-rot-pair)
      (instance pair2 ok-rot-pair)
      (instance pair3 ok-rot-tension-pair)
      (instance pair4 ok-rot-tension-pair)
      (instance pair5 ok-rot-pair)
      (instance pair6 ok-rot-pair)))
```

Abbildung 5.5.

Das berechnete Modell nach der Vorgabe einiger Startwerte und Festlegung der Verhaltensmodelle



ter eingeschränkt werden, da für sie keine Radien angegeben wurden. Trotzdem erfolgt die Propagierung der entsprechenden Drehmomente auf die angeschlossenen Glieder, so daß für diese wiederum die Bestimmung der Kräfte über das Drehmomentgesetz erfolgen kann. Dies zeigt, daß im jeweiligen Fall nur die wirklich benötigten Informationen bereitgestellt werden müssen resp. berechnet werden.

Wie bereits erwähnt, kann die Reihenfolge der Schritte beliebig vertauscht werden. Werden die Schritte noch weiter in Teilschritte zerlegt, so kann die Angabe von Parameterwerten und Verhaltensmodellen noch enger ineinander verzahnt erfolgen.

Die vorgestellten ausdrucksstarken konkreten Gegenstandsbereiche[3] erlauben

ABox 5.3.6

Angabe eines zusätzlichen Parameterwertes

```
(state (related crankarm-1 200 link.force))
```

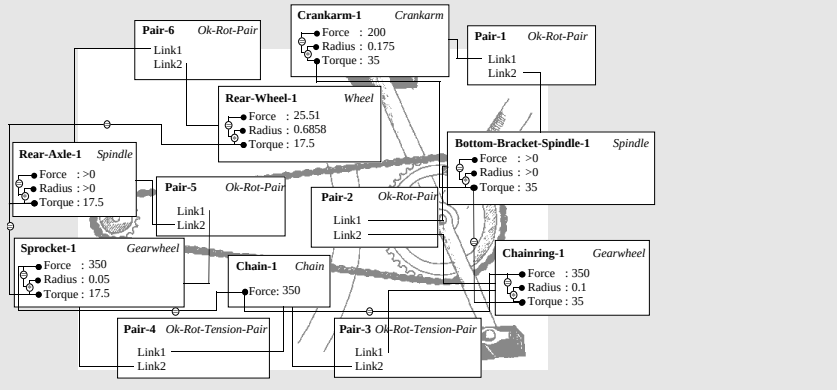


Abbildung 5.6.

Das berechnete Modell nach Angabe der Kraft am Kurbelarm

nicht nur einzelne Werte aus dem Gegenstandsbereich als Parameterwerte für die Simulation vorzugeben. Vielmehr sind beliebige, im Rahmen des konkreten Gegenstandsbereich repräsentierbare Prädikate, z.B. eine Menge von Intervallen oder eine Menge von Werten als Parametereinschränkungen möglich. Diese Tatsache erlaubt die einfache Darstellung komplexer Simulationen, wie sie z.B. im Rahmen der What-If-Analyse in der Designphase eines technischen Gerätes benötigt werden.

5.3.2. Komplexe Verhaltenssimulation

Die Möglichkeit, komplexe Prädikate als Einschränkungen an den Parametern nicht nur der Konzepte sondern auch der Instanzen zu spezifizieren, erlaubt es aber auch, auf einfache Art und Weise komplexe Geräte zu beschreiben. Das folgende Beispiel soll dies verdeutlichen.

Beispiel 5.2 (Komplexe Verhaltenssimulation)
Es soll ein Fahrradantrieb simuliert werden, der nicht nur eine Übersetzung hat, sondern eine Schaltung besitzt. Konkret soll der Antrieb zwei Kettenräder und drei Ritzel besitzen.

❶ Ein solcher Antrieb kann durch die im obigen Beispiel verwendete Antriebsstruktur beschrieben werden, indem entsprechende Disjunktionen für die Parametereinschränkungen an Kettenrad und Ritzel verwendet werden.

❷ Zusätzlich wird die Kraft am Kurbelarm auf ein Intervall eingeschränkt, das für einigermassen trainierte Radfahrer typisch ist.

3. Ab der Ausdrucksstärke von $\mathbf{N_2}$.

ABox 5.3.7

Vorgabe einiger Startwerte (komplexe Simulation)

```
(state (and
     (related crankarm-1 0.175 rot.radius)
     (related rear-wheel-1 0.6858 rot.radius)
     (instance chainring-1
       (constrain rot.radius
         ((?x) (or (= ?x 0.1)(= ?x 0.125)))))
     (instance sprocket-1
       (constrain rot.radius
         ((?x) (or (= ?x 0.05)(= ?x 0.0625) (= ?x 0.075)))))))
```

ABox 5.3.8

Angabe der Kraft am Kurbelarm

```
(state
  (instance crankarm-1
    (constrain link.force
      ((?x) (and (<= ?x 200)(>= ?x 300))))))
```

Diese Einschränkungen resultieren in dem in Abbildung 5.7 gezeigten Modell.

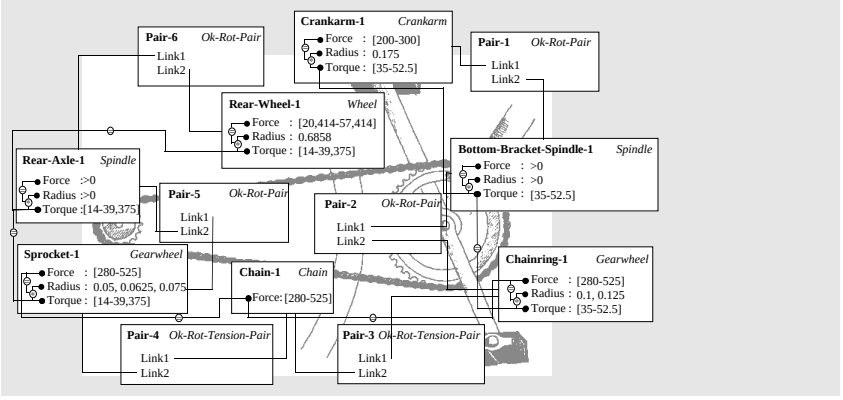


Abbildung 5.7.

Das berechnete Modell nach Angabe der Kraft am Kurbelarm

Tatsächlich liefert das Eliminationsverfahren für quadratische Ungleichungssysteme noch mehr Informationen, da die verschiedenen Übersetzungen separat behandelt werden. So lautet z.B. die berechnete Formel für die Kraft am Hinterrad (frw) wie folgt:

An dieser Formel kann man zwei Tatsachen sofort ablesen. Erstens werden nur 5 Intervalle statt erwarteten $2 * 3 = 6$ Intervalle zurückgeliefert. Eine genaue Betrachtung der angegebenen Radii ergibt, daß $0{,}05 * 0{,}125 = 0{,}625 * 0{,}1$ ist, und daher nur 5 unterschiedliche Übersetzungen vorliegen. Genau dies soll im Normalfall allerdings vermieden werden, die Simulation gibt also Hinweise auf einer Verbesserung des Antriebes. Zweitens existieren nur 8 Intervallendpunkte, was darauf hinweist, daß einige Intervalle direkt aneinander stoßen.

```
(or
  (= (∗ 429 frw) 70000) (= (∗ 3429 frw) 87500)
  (= (∗ 3429 frw) 109375) (= (∗ 2286 frw) 109375)
  (= (∗ 1143 frw) 35000) (= (∗ 1143 frw) 43750)
  (= (∗ 381 frw) 17500) (= (∗ 381 frw) 21875)
  (and (> (∗ 1143 frw) 43750) (< (∗ 381 frw) 21875))
  (and (> (∗ 3429 frw) 109375) (< (∗ 2286 frw) 109375))
  (and (> (∗ 3429 frw) 70000) (< (∗ 1143 frw) 35000))
  (and (> (∗ 1143 frw) 35000) (< (∗ 381 frw) 17500))
  (and (> (∗ 3429 frw) 87500) (< (∗ 1143 frw) 43750)))
```

ABox 5.3.9

Berechnete Kraft am Hinterrad

5.4. Konsistenzbasierte Diagnose

Wie schon in Abschnitt 6.1.1 gezeigt wurde, ist die Objektklassifikation die Basis eines Verfahrens zum ähnlichkeitsbasierten Retrieval. In diesem Abschnitt wollen wir nun zeigen, wie dieses Verfahren – die Repräsentation von physikalischen Gesetzen und Verhaltensmodellen vorausgesetzt – zur konsistenzbasierten Diagnose eingesetzt werden kann.

Dabei zeigt sich deutlich, wie durch den erhöhten Einsatz von Wissen die Komplexität des Ähnlichkeitsmaßes reduziert werden kann, ja sogar auf die Bestimmung ähnlicher Objekte ganz verzichtet werden kann.

Auch hier wird, wie bei der Verhaltenssimulation in einem ersten Schritt das zu diagnostizierende Gerät beschrieben.

Im nächsten Schritt werden dann aber keine konkreten Verhaltensmodelle assertiert, vielmehr werden die beobachteten Parameterwerte in die ABox eingetragen und die Objektklassifikation angestoßen. Diese berechnet nicht nur die Menge der speziellsten Verhaltensmodelle, die durch diese Parameterwerte erfüllt sind. Im Rahmen des eingebetteten Konsistenzalgorithmus werden, wie im vorhergehenden Abschnitt zu sehen war, auch die resultierenden Einschränkungen an weiteren Parametern berechnet.

Die schwache Objektklassifikation berechnet die Verhaltensmodelle, zu denen die Instanzen evtl. noch spezialisiert werden können, und damit Hinweise, welche Parameterwerte als nächste erhoben werden sollten. Ferner schließt die schwache Objektklassifikation eine Reihe von Verhaltensmodellen aus, die zwar Spezialisierungen der durch die Objektklassifikation berechneten Verhaltensmodelle sind, die mit den gemessenen und berechneten Parameterwerten aber nicht mehr erfüllbar sind. So werden durch die Klassifikation und schwache Klassifikation eine obere und eine untere Annäherung an die möglichen Verhaltensmodelle berechnet. Mit jeder Parameterbestimmung bewegen sich diese Annäherungen weiter aufeinander zu.

Beispiel 5.3
Die Vorgehensweise soll am Beispiel eines einfachen Rotationspaares verdeutlicht werden:

❶ Beschreibung des Gerätes.
Wie bei der Verhaltenssimulation muß zuerst die Struktur des zu diagnostizierenden Gerätes, hier des Rotationspaares beschrieben werden (ABox 5.4.1).

ABox 5.4.1
Beschreibung der Struktur eines Rotationspaares

```
(define-distinct-individual rot-pair1)
(define-distinct-individual link1.1)
(define-distinct-individual link1.2)

(state (instance link1.2 rotational-link))
(state (instance link1.1 rotational-link))

(state (and
     (instance rot-pair1 kinematic-pair)
     (related rot-pair1 link1.1 pair.link1)
     (related rot-pair1 link1.2 pair.link2)))
```

❷ Festlegung allgemeiner Verhaltensmodelle.
Im Beispiel besteht dieser Schritt aus der in ABox 5.4.2 gezeigten Festlegung des Paares auf ein Rotationspaar[4].

ABox 5.4.2
Festlegung allgemeiner Verhaltensmodelle

```
(state (instance rot-pair1 rot-pair))
```

❸ Angabe der beobachteten Parameterwerte und Objektklassifikation.
Da unterschiedliche Parametersätze zu unterschiedlichen Klassifikationen und damit zu verschiedenen Diagnosen führen sollten, soll dieser Schritt anhand dreier Parametersätze illustriert werden:

❶ Die Angabe identischer Drehmomente für die beiden Glieder des Rotationspaares führt erwartungsgemäß zur Klassifikation als ok-rot-pair (siehe ABox 5.4.3 auf der nächsten Seite).

❷ Der Diagnoseprozeß wird deutlicher, wenn man den Parametersatz aus Abbildung 5.4.4 benutzt. Die erste Assertion, zusammen mit dem Drehmomentgesetz und der Randbedingung eines strikt positiven Radius, bedingt, daß das Drehmoment von link1.1 strikt positiv sein muß.
Daher reicht die zweite Assertion, die einen Wert von Null für das Drehmoment von link1.2 zusichert, aus, um das Rotationspaar als gebrochen zu klassifizieren.

4. Selbst diese könnte im Beispiel entfallen, da rot-pair1 automatisch zu einem rot-pair klassifiziert werden kann, weil beide Glieder vom Typ rotational-link sind.

```
(state (and
     (related link1.1 10 rot.torque)
     (related link1.2 10 rot.torque)))
```

ABox 5.4.3

Diagnose: Parametersatz I

Auch in diesem Falle genügt also die Angabe von nur zwei der 6 möglichen Parameter zur eindeutigen Diagnose.

```
(state (related link1.1 8 link.force))
(state (related link1.2 0 rot.torque))
```

ABox 5.4.4

Diagnose: Parametersatz II

❸ Der dritte Parametersatz in ABox 5.4.5 verdeutlicht, wie der Diagnoseprozeß zu einer Klassifikation des Rotationspaares als weak-slipping-rot-pair führt. Dies ist möglich, da das Drehmoment für link1.2 durch das Drehmomentgesetz berechnet werden kann. Das Verhältnis der Drehmomente der beiden Glieder führt zur Klassifikation als weak-slipping-rot-pair.

Anhand dieses Beispiels wird auch die Rolle der schwachen Objektklassifikation deutlich: Nach der Assertion (state (related link1.2 8 link.force)) kann broken-rot-pair aus der Liste der noch möglichen Diagnosen gestrichen werden, da – aufgrund der selben Argumentation wie oben – das Drehmoment von link1.2 strikt positiv sein muß. Daher sind beide Drehmomente strikt positiv und eine Klassifikation als broken-rot-pair ist nicht mehr möglich.

```
(state (related link1.1 20 rot.torque))
(state (related link1.2 8 link.force))

(state (related link1.2 3 rot.radius))
```

ABox 5.4.5

Diagnose: Parametersatz III

5.5. Wissensmanagement – Pflege und Wartung von Modellbibliotheken

Abbildung 5.8 zeigt die Konzepthierarchie nach der Klassifikation der Konzepte.

Die folgenden Aussagen, die aus diesem Graph abgelesen werden können, sind im Hinblick auf Anforderungen aus dem Bereich des Wissensmanagements besonders interessant:

- Alle Definitionen sind verschieden von Bottom. Damit sind alle Konzeptdefinitionen erfüllbar, d.h. sie enthalten keine widersprüchlichen Aussagen und erlauben

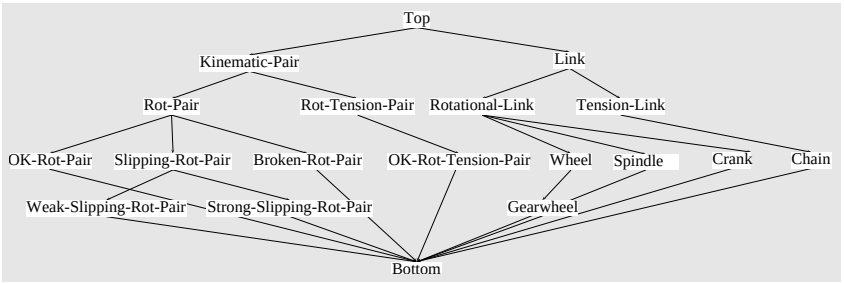


Abbildung 5.8.

Die resultierende TBox

ein Modell. Dies garantiert, daß kein Verhaltensmodell fälschlicherweise so spezifiziert ist, daß es keine Parameterkombination gibt, auf die es zutrifft (etwa durch eine Parameterrestriktion der Form $x > 0 \wedge x < 0$).

- Alle Konzeptdefinitionen sind disjunkt. Dies bedeutet u.a., daß keine zwei Verhaltensmodelle das gleiche Verhalten beschreiben. Eine solche Situation kann leicht auftreten, wenn zwei Modellbibliotheken zusammengeführt werden.

- In der TBox 5.1.6 auf Seite 149 ist strong-slipping-rot-pair nicht als Spezialisierung von slipping-rot-pair, sondern von rot-pair definiert. Solche Fehler sind insbesondere bei einer dynamischen Entwicklung der Verhaltensmodelle und bei komplexen Modellbibliotheken zu erwarten. Der Klassifikationsalgorithmus konnte die intendierte Subsumptionsbeziehung aber aus der Beschreibung der beiden Konzepte ableiten (wie in Abb.5.8 zu sehen). Es könnte allerdings auch sein, daß die ermittelte Subsumptionsbeziehung ihre Ursache in einer fehlerhaften Modellierung hat.

Fehler der oben geschilderten Arten können in großen Modellbibliotheken leicht auftreten. Die Bereitstellung von Inferenzen, die diese Fehler erkennen und anzeigen, ist insbesondere für den Aufbau komplexer Modellbibliotheken unverzichtbar. Durch die Vollständigkeit der Inferenzen innerhalb von CTL kann garantiert werden, daß sämtliche Fehler (zumindest für lineare Systeme) erkannt werden.

5.6. Zusammenfassung

In diesem Kapitels wurde anhand einer Umsetzung des Leitbeispieles gezeigt, daß CTL die in Kapitel 2 postulierten Anforderungen an ein Second-Level-Support-System erfüllt.

Die Repräsentation des Wissens erfolgt objektorientiert und ist erweiterbar, Das Retrieval ist flexibel und durch den Benutzer steuerbar, die Retrievalsematik ist klar und nachvollziehbar und bezieht das Hintergrundwissen mit ein.

Dadurch werden neben dem einfachen Retrieval und der Vervollständigung von Objektbeschreibungen Anwendungen wie Simulation, What-If-Analyse, Modellbasierte Diagnose und Pflege und Wartung von Modellbibliotheken möglich.

RACER ist wie im letzten Kapitel gezeigt wurde, insbesondere aufgrund der noch fehlenden Realisierung von Featureketten noch nicht für die in dieser Arbeit behandelte Aufgabenstellung geeignet.

Voraussetzung für den erfolgreichen Einsatz von CTL ist allerdings, daß bereits eine große Menge an Wissen formalisiert vorliegt.

Ist dies nicht der Fall, dann lassen sich die Basisinferenzen der Konzept- und Objektklassifikation sowie des Retrievals dazu verwenden, um klassische Verfahren des fallbasierten Schließens zu implementieren. Dabei werden die verschiedenen Basisinferenzen dazu benutzt, Indexmengen bzw. Indexvektoren zu berechnen, die dann die Grundlage für die heuristischen Algorithmen bilden.

Wie diese Verfahren genau aussehen können, ist zusammen mit anderen Querbeziehungen zu »klassischen« Fallbasierten Systemen Inhalt des nächsten Kapitels.

6 Zusammenhänge mit ≫klassischen≪ Fallbasierten Systemen

Das letzte Kapitel hat gezeigt, daß CTL die in Abschnitt 1.2 erarbeiteten Anforderungen an ein Second-Level-Support-System erfüllt. Voraussetzung für die Umsetzung von Simulation, What-If-Analyse etc. ist allerdings, daß das Wissen, bereits in einer formalisierten Form, z.B. in einer objekt-orientierten Repräsentation des Domänen- und Hintergrundwissen vorliegt.

Die Probleme, eine solche Repräsentation mit den Mitteln ≫klassischer≪ fallbasierter Systeme abzubilden (siehe Kapitel 1), war neben der Anforderung nach einem semantisch fundierten Retrieval der Ausgangspunkt für diese Arbeit.

Nicht immer liegt das Wissen bereits in mit dem in dem Leitbeispiel dieser Arbeit verwendeten hohen Formalisierungsgrad vor. Oftmals entwickelt sich das Verständnis für das Zusammenspiel der einzelnen Komponenten erst im Laufe der Zeit bzw. mit der Summe der Erfahrungen die gemacht werden: Das formalisierte Wissen ist eine Abstraktion aus der Summe der Einzelerfahrungen.

Gerade bei wenig formalisierten Wissen bietet es sich an, die auf Heuristiken basierenden Ähnlichkeitsbegriffe des ≫klassischen≪ fallbasierten Schließens zu nutzen. Daher ist es wünschenswert, auch diese Verfahren gegebenenfalls einsetzen zu können.

Im diesem, abschließenden, Kapitel soll daher anhand einiger Aspekte gezeigt werden daß sich die beiden, zunächst widersprüchlich erscheinenden Ansätze: einerseits der in dieser Arbeit vorgestellte, auf Logik gestützte, andererseits die approximativer, heuristischen Verfahren des klassischen fallbasierten Schliessens nicht widersprechen, sondern ergänzen.

Nach einigen einleitenden Bemerkungen zum generellen Verhältnis ≫logikbasierter≪ und ≫ähnlichkeitsbasierter≪ Ansätze wird in Abschnitt 6.1 gezeigt wie auf Ähnlichkeitsmaßen basierende heuristische Retrievalverfahren basierend auf den beschreibungslogischen Basisinferenzen Konzept- und Objektklassifikation sowie

Retrieval und den durch die Beschreibungslogik bereitgestellten Indexstrukturen realisiert werden können [Kam96a].

Daran schließt sich ein kurzer Abschnitt über objekt-orientierte Fallbeschreibungen und Indexstrukturen an.

Den Abschluß bildet ein Abschnitt in dem mit dem Ansatz der »generalised cases« eine neuere Entwicklung zur Fallrepräsentation aus dem Bereich des fallbasierten Schliessens mit den in dieser Arbeit verwandten Repräsentationen z.B. für komplexe Attributwerte von Instanzen verglichen.

»Logik« versus »Ähnlichkeit«?

Wie weiter oben bereits bemerkt, werden logikbasierte und ähnlichkeitsbasierte Ansätze oft als widersprüchlich angesehen und der jeweils andere Ansatz von Vertretern eines »Lagers« als wenig geeignet für die Lösung der jeweiligen Aufgabenstellung angesehen.

In den folgenden Abschnitten sollen einige Evidenzen für die in dieser Arbeit vertretene Ansicht, dass sich beide Ansätze eher ergänzen als widersprechen, vorgestellt werden. Dazu zunächst einige allgemeine Bemerkungen:

Weitere Ansätze zum logikbasierten fallbasierten Schließen

Die Idee, Beschreibungslogiken, und die dort zur Verfügung stehenden Mechanismen zur automatischen Klassifikation, für das fallbasierte Schließen zu nutzen, wurde nicht nur in dieser Arbeit und den ihr zugrundeliegenden Publikationen [KPB96, Kam96b, KW96b, Kam96a, Kam97a, KN97, Kam97b] verfolgt.

Ashley [AA94] setzt das System LOOM zur Realisierung eines fallbasierten Systems zum Tutoring des amerikanischen Rechtsystems ein, Koehler benutzt eine stark eingeschränkte Beschreibungslogik zum Retrieval von Plänen im Rahmen eines intelligenten Hilfesystems [Koe94]. Plaza geht in [Pla95] ganz generell auf die Eignung von Featurelogiken[1] für das fallbasierte Schließen ein[2]. Weitere Ansätze werden von Napoli [NLS97], Gonzales [GCDAGA99] und [SCF98, SV98] beschrieben.

Diese Ansätze benutzen im wesentlichen die Basisinferenzen des starken Retrievals und der starken Objektklassifikation wie sie von existierenden beschreibungslogischen Systemen angeboten werden. Zudem bauen sie zumeist auf beschreibungslogischen Systemen auf, die keine korrekten und vollständigen Inferenzen, insbesondere im Bereich der konkreten Gegenstandsbereiche, realisieren. Insofern stellt der in dieser Arbeit vorgestellte Ansatz eine Fortentwicklung der in diesen Arbeiten präsentierten Ansätze dar.

Objektive Definition vs. subjektive Einschätzung

Logikbasierte Ansätze bieten sich insbesondere dann an, wenn der Wissensstand zur Formalisierung von Hintergrundwissen bereits so hoch ist, daß eine objektive

1. Einer Untermenge von Beschreibungslogiken.

2. Allerdings ist seine Präsentation an einigen Stellen sehr unsauber.

Definition des Hintergrundwissen z.B. in Form eines physikalischen Gesetzes oder eines Constraints gegeben werden kann.

Ähnlichkeitsbasierte Ansätze sind besonders geeignet, um subjektive Einschätzungen wie die Ähnlichkeit von Einzelerfahrungen »individuelle Vorlieben« zu operationalisieren.

Effizienz der Verfahren

Logikbasierten Ansätzen wird häufig der Vorwurf der Ineffizienz gemacht, da die veröffentlichten Entscheidbarkeits- bzw. Unentscheidbarkeitsresultate (siehe 3) sehr abschreckend wirken. Insbesondere im Vergleich mit den grossen Datenvolumina fallbasierter und Datenbank-Systeme entstand der Eindruck, dass beschreibungslogische Systeme nur für »Spielanwendungen« zum Einsatz kommen können.

Wie bereits in Kapitel 4 bemerkt handelt es sich bei den Komplexitäts-Ergebnissen um »worst case« Resultate die im wesentlichen die theoretischen Grenzen definieren. Moderne beschreibungslogische Systeme der dritten Generation sind auf den »average case« optimiert und auch mit sehr grossen ($> 100\,000$ Konzepte) Wissenbasen praktisch einsetzbar [Moe01].

Hinzu kommt, daß im Bereich der Beschreibungslogiken im Vergleich zu anderen Bereichen wie etwa dem fallbasierten Schliessen ein besonders hoher Wert auf die Untersuchung der Komplexität der Verfahren gelegt wird. Komplexitätsuntersuchungen nehmen im Bereich des »klassischen« fallbasierten Schließens einen deutlich geringeren Raum ein.

Vergleichbar mit den korrekten und vollständigen Verfahren aus dem Bereich der Beschreibungslogiken sind im Bereich des fallbasierten Schließens Verfahren auf Basis von Voronoi-Diagrammen, die häufig, (z.B. in [Weß95]) zur Abschätzung der Komplexität verwandt werden [3].

Aufgrund der Definition von Voronoi-Diagrammen ist klar, daß Retrievalverfahren die darauf basieren korrekt und vollständig sind. Aus Komplexitätssicht sind die Komplexität des Verfahrens zum Aufbau des Voronoi-Diagramms, die Komplexität für die Suche und die Größe des Voronoi-Diagrammes, d.h. die Anzahl der Zusammenhangskomponenten des Diagramms von Interesse. Die Komplexität

3. Voronoi-Diagramme explizieren praktisch die Suchergebnisse für Nearest-Neighbour und k-Nearest-Neighbour Anfragen. Gegeben eine Menge von Punkten $P = p_1, \ldots, p_n \in \mathbb{R}^d$ ist die Voronoi-Region $\mathrm{VR}(P_i)$ eines Punktes $p_i \in \mathbb{R}^d$ der Teilraum von $\mathbb{R}^d$ für den der nächstgelegene Punkt p_i ist, d.h. $\forall q \in \mathrm{VR}(P_i) : \mathrm{dist}(q, p_i)) \leq \mathrm{dist}(q, p); p \in P$. Das Voronoi-Diagramm (erster Ordnung) ist dann die Partition des $\mathbb{R}^d$ in die Menge der Voronoi-Regionen.

   Zur Beantwortung einer Ähnlichkeitsanfrage für einen Punkt q gilt es dann nur, festzustellen, in welcher Voronoi-Region q liegt: $q \in \mathrm{VR}(p_i)$. Voronoi-Diagramme können in zwei Richtungen verallgemeinert werden. Einerseits kann das Verfahren auf Voronoi-Diagramme k-ter Ordnung verallgemeinert werden. Diese dienen zur Beantwortung des k-Nearest-Neighbour Problems, d.h. die Voronoi-Region $\mathrm{VR_k}(p_1, \ldots, p_k)$ ist der Teilraum, für den die Menge der Punkte $p_1, \ldots, p_k$ die nächstgelegenen Punkte sind. Andererseits können die zugrundeliegenden Objekte verallgemeinert werden, von Punkten zu Geraden, Rechtecken etc. hin zu algebraischen und semialgebraischen Mengen.

dieser Aufgaben ist dabei sehr stark von der Dimension des zugrundeliegenden Raumes $\mathbb{R}^d$ abhängig.

Für den am häufigsten untersuchten Fall gibt es Verfahren zur Generierung des Diagrammes mit eine Komplexität von $O(n \log(n))$ [4] und $O(n\sqrt{n})$ im Durchschnittsfall ($O(n^2)$ im worst case. Für höhere Dimensionen d und höhere k gehen diese komplexitätstheoretisch günstigen Eigenschaften verloren: Ein allgemeines Ergebnis für Voronoi-Diagramme k-ter Ordnung für semi-algebraische Mengen wird in [Rie03] vorgestellt: Eine Sammlung von n (evtl. singulärer), jeweils durch Polynome vom maximalen Grad δ definierte, semi-algebraischer Mengen der Dimension $d-1$ in $\mathbb{R}^d$ hat $\Theta((n\delta)^d)$ Voronoi-Regionen erster Ordnung (für jedes feste d). Für den Fall, daß die maximale Dimension m der semi-algebraischen Mengen $m = d-2$ nicht übersteigt, gilt eine obere Schranke von $O(n^{m+1}\delta^d)$ (für nicht-singuläre Mengen) bzw. $O((n\delta)^d)$ (im allgemeinen Fall).

Die Komplexität des gesamten Voronoi-Diagramms k-ter Ordnung einer generischen Sammlung nicht-singulärer reell algebraischer Mengen im $\mathbb{R}^d$ mit maximaler Dimension $m < d$ und maximalem Grad Δ ist $O(n^{\min(d+k,2d)}\Delta^{2(m+1)d})$.

Damit ergeben sich beispielsweise für den Fall $k = 1, d = 10, m = 0$, d.h. Punkte im 10-dimensionalen Raum eine Komplexität von $O(n^{11})$, und für den Fall $k = 3, d = 3, m = 1$, d.h. Bestimmung der Voronoi-Regionen der drei nächsten Nachbarn zu n Geraden bzw. Kurven im drei-dimensionalen Raum eine Komplexität von $O(n^6 3^{2(3+1)3} = O(3^{24}n^6)$.

Damit ist klar, daß auch im Bereich des »klassischen« fallbasierten Schließens, wenn ähnliche Aufgabenstellungen behandelt werden sollen, mit einer ähnlichen Worst-Case-komplexität zu rechnen ist.

6.1. Ähnlichkeitsbasiertes Retrieval über beschreibungslogischen Repräsentationen

Nach den allgemeinem Betrachtungen zum Verhältnis von Beschreibungslogiken und fallbasierten Schließen sollen in diesem und im nächsten Abschnitt konkrete Aufgabenstellungen und Verfahren vorgestellt werden, die zeigen, daß sich die beiden Gebiete ergänzen bzw. sich immer mehr aneinander annähern.

In diesem Abschnitt sollen unterschiedliche Ansätze zum ähnlichkeitsbasierten Retrieval auf Grundlage der Basisinferenzen von Beschreibungslogiken vorgestellt werden [Kam96a].

Die Grundidee ist bei allen Ansätzen die Gleiche: Die beschreibungslogischen Basisinferenzen liefern die Indexmengen bzw. Indexvektoren, auf deren Basis Methoden zur Ähnlichkeitsbestimmung, und wie sie z.B. in Abschnitt 1.1.2 vorgestellt wurden, aufsetzen können. Damit vereinen diese Ansätze Methoden der Beschrei-

4. Allerdings mit sehr großen Konstanten, die das Verfahren in der Praxis uneinsetzbar machen.

bungslogiken mit Methoden des »klassischen« fallbasierten Schließens ohne diese zu vermischen.

Vielmehr bauen die verschiedenen Retrievalverfahren zwar auf den Basisinferenzen auf, sind aber bewusst nicht in den eigentlichen inferenzbasierten Kern von CTL integriert, sondern als separater Layer realisiert. Dadurch werden deduktive und heuristische Ähnlichkeitsbestimmung klar und sauber voneinander getrennt.

Die verschiedenen Ansätze unterscheiden sich im wesentlichen im Hinblick auf die Art der eingesetzten Basisinferenzen, wie Objektklassifikation, Retrieval, aber auch Konzeptklassifikation, bzw. in dem Grad, in dem die aktuelle ABox, d.h. die Menge der Assertionen über den gespeicherten Objekten, in den Inferenzprozeß einbezogen wird.

Daraus resultieren Ansätze, die sich in ihrer Komplexität, aber auch in ihrer prinzipiellen Benutzung unterscheiden. Dennoch sind alle Ansätze untereinander kompatibel, da sie auf der gleichen Repräsentation operieren. Daher können sie alternativ auf derselben Datenbasis eingesetzt werden.

Während bei der Konzeptklassifikation Assertionen über Objekten überhaupt nicht berücksichtigt werden, sind es bei der Objektklassifikation im wesentlichen die in der Anfrage spezifizierten Objekte, die in den Inferenzprozeß eingehen. Das Retrieval schließlich betrifft alle Objekte. Da die benötigte Rechenzeit direkt von der Größe der ABox abhängt, hat die Wahl der Basisinferenz Auswirkungen auf die Komplexität des Verfahrens. Andererseits beeinflußt die Größe der ABox die Menge der Inferenzen, die gezogen werden können, und damit die Retrievalqualität der Verfahren.

In gewissem Sinn ist dieses Verhalten analog zur Repräsentation von Hintergrundwissen: Wissen, das nicht in der TBox beschrieben oder nicht in der ABox assertiert ist, kann nicht für Inferenzen herangezogen werden, und führt zu einer Verschlechterung der Qualität der Verfahren. Dafür ist es dann aufgrund der geringeren Komplexität der Verfahren möglich, größere Datenmengen zu behandeln.

6.1.1. Verfahren auf der Basis der Objektklassifikation

Die Objektklassifikation ist ein automatisches, dynamisches Verfahren zur semantischen Indizierung der ABox-Objekte o_j in Bezug auf die TBox. Sie bietet sich als Basisinferenz dann an, wenn man die Beschreibungslogik eher in der traditionellen Art eines fallbasierten Systems benutzen möchte: Ein neuer Fall soll gelöst werden, indem nach ähnlichen Fällen gesucht wird, gleichzeitig wird der neue Fall der Fallbasis hinzugefügt.

❶ Hinzufügen von Assertionen über Objekte

Neue Assertionen (z.B. neue Attributwerte) über ein Objekt o_j stoßen automatisch eine erneute Objektklassifikation und damit eine Reindizierung von o_j an.

❷ Indizierung durch Objektklassifikation

Zur Indizierung der Objekte können ebenfalls die starke $I'(o_j) = \overline{C}(o_j)$ oder die schwache Variante $I''(o_j) = \underline{C}(o_j)$ der Objektklassifikation bzw. eine Kombination beider Varianten $I'''(o_j) = (\overline{C}(o_j), \underline{C}(o_j))$ benutzt werden.

❸ Auswahl relevanter Objekte

Die Indizierung der Objekte durch Objektklassifikation erfolgt automatisch mit jeder neuen Assertion und damit im Hintergrund. So kann jederzeit für jedes Objekt das Ergebnis des starken bzw. des schwachen Retrievals abgerufen und angezeigt werden. In unserem Szenario ist es jedoch sinnvoller, die Initiierung der Anzeige und die Auswahl der anzuzeigenden Objekte dem Benutzer zu überlassen: Er wählt die Objekte $o_1, \ldots, o_n$ aus, an denen er zu einem bestimmten Zeitpunkt interessiert ist.

In vielen Anwendungen, insbesondere dann, wenn ausreichend Wissen vorhanden ist (siehe z.B. Abschnitt 5.4) genügt dem Benutzer die Angabe des Ergebnisses der Objektklassifikation, d.h. der Indizes $I, \ldots, I'''$. In Fällen in denen weniger Wissen zur Verfügung steht, ist man allerdings an ähnlichen Objekten interessiert.

❹ Berechnung der Ähnlichkeit zwischen Objekten

Auf Basis der starken und schwachen Objektklassifikation lassen sich eine Reihe von Ähnlichkeitsmaßen definieren:

❶ Übereinstimmende Indizes

Ein einfaches, relationales Ähnlichkeitsmaß liefert die Objekte zurück, die den gleichen Satz von Indizes aufweisen wie das Anfrageobjekt:

$$sim_{I^*}(o_k) = \aleph_{I^*(o_l)=I^*(o_k)}(o_k)$$

Hierbei ist $I^* \in \{I', I'', I'''\}$ und $\aleph_M$ die charakteristische Funktion der Menge M. Da in unserem Szenario der Benutzer Experte ist, und daher die Anfrage (d.h. die ihn interessierenden Objekte) sehr genau formulieren kann, reicht dieses einfache Maß schon für viele Zwecke aus. Offensichtliche Erweiterungen dieses Ähnlichkeitsmaßes, die zugleich auch eine Ordnung auf den zurückgelieferten Objekten berechnen, sind die folgenden:

❷ Quotient der übereinstimmenden Indizes

$$sim_{I^*}(o_l, o_k) = \frac{\mid I(o_l) \cap I(o_k) \mid}{\mid I(o_l) \cup I(o_k) \mid}, \qquad I^* \in \{I', I'', I'''\}$$

❸ Baumdistanz der Indexmengen

Berechnung der Distanz in der Konzepthierarchie [5] Retrieval benutzt.

❹ Winkel zwischen den Konzeptvektoren

Berechnung des Cosinusmaßes wie im Bereich des Information Retrievals üblich.

❺ Anzeige der ähnlichen Objekte.

Dadurch, daß o_j mit anderen Objekten $o_{k_1}, \ldots, o_{k_n}$ in Verbindung steht, müssen auch diese neu klassifiziert und indiziert werden. Im »worst-case« kann so durch die Objektklassifikation eines einzigen Objektes o_j die Neuklassifikation der gesamten ABox $\mathcal{A}$ notwendig werden – ein zeitaufwendiger Prozeß.

Im Normalfall sind allerdings weit weniger Objekte betroffen. Im allgemeinen sind es nur die, die mit einem aktuellen Defekt/Fehler in Beziehung stehen. Daher kann von folgender Unabhängigkeitsannahme ausgegangen werden: Objekte, die der ABox in unterschiedlichen Sitzungen hinzugefügt werden, stehen nicht miteinander in Verbindung, sie sind voneinander unabhängig. Diese Unabhängigkeitsannahme ist eine schwache Form der impliziten Unabhängigkeitsannahme herkömmlicher fallbasierter Systeme: Verschiedene Fälle sind unabhängig voneinander.

Da es aber aus den in Kapitel 1 geschilderten Gründen keine ausgezeichneten »Fallobjekte« gibt, kann nur eine abgeschwächte Unabhängigkeit in der obigen Form angenommen werden. Die Unabhängigkeitsannahme erlaubt eine Variante des in diesem Abschnitt geschilderten Verfahrens, in der die »aktiven« und die »passiven« Teile einer ABox auch unterschiedlich implementiert sind, und das »volle«, Tableaux-Kalkül basierte Verfahren nur auf der aktiven ABox operiert, während für den passiven Teil der ABox ein Lookup zur Bestimmung der Objektindizes genügt.

6.1.2. Verfahren auf der Basis des Retrievals

Sinnvoll können retrievalbasierte Verfahren zum ähnlichkeitsbasierten Retrieval am besten dann eingesetzt werden, wenn keine neuen Informationen der ABox hinzugefügt werden sollen.

❶ Formulierung der Anfrage

Die Anfrage wird analog zum Vorgehen in Abschnitt 6.1.3 als Konzeptterm C_q formuliert.

❷ Aufruf des Retrievals

Das Retrieval wird mit dem Anfragekonzeptterm C_q aufgerufen und die Mengen der $I'(C_q) = \overline{O}(C_q)$, $I''(C_q) = \underline{O}(C_q)$ bzw. $I'''(C_q) = (\overline{O}(C_q), \underline{O}(C_q))$

5. In Anlehnung an die obige Bemerkung zur Distanz zwischen Begriffen in semantischen Netzen.

bestimmt. Dazu kann es, analog zum Verfahren auf Basis der Objektklassifikation, notwendig werden, die Zugehörigkeit aller ABox Objekte zum Anfragekonzept zu berechnen. Im allgemeinen sind aber auch hier nur ein Bruchteil der ABox-Objekte tatsächlich betroffen.

❸ Anzeige der dem Objekt zugehörigen Objekte

Darüber hinaus ist eine Mischform der Objektklassifikation und des Retrievals möglich. In dieser werden die Assertionen über die relevanten Anfrageobjekte zuerst als Konzeptterm reformuliert und das Retrieval für diesen Konzeptterm aufgerufen. Alle existierenden Objekte werden dadurch in Bezug auf dieses Konzept reklassifiziert und reindiziert. Die neuen, spezifischeren Indizes können dann als die Basis für die Ähnlichkeitsmaße aus Abschnitt 6.1.1 verwendet werden.

6.1.3. Verfahren auf der Basis der Konzeptklassifikation

Voraussetzung für Verfahren auf der Basis der Konzeptklassifikation ist eine vorhergehende Indizierung der ABox-Objekte mit Hilfe des Retrievals[6]. Weiterhin ist dieses Verfahren nur dann geeignet, wenn nur Anfragen an eine schon bestehende ABox gerichtet werden sollen, und keine neuen Daten eingetragen werden.

An dieser Stelle soll nur die Grundidee dieser Verfahren skizziert werden:

❶ Vorverarbeitung

Für jedes Konzept C_j des Konzeptgraphens wird das Retrieval aufgerufen. So werden die Instanzen der ABox durch Abspeicherung der Retrievalergebnisse beim Konzept indiziert. Zur Indizierung kann wahlweise das starke Retrieval $I'(C_j) = \overline{O}(C_j)$, das schwache Retrieval $I''(C_j) = \underline{O}(C_j)$ oder eine Kombination von beiden $I'''(C_j) = (\overline{O}(C_j), \underline{O}(C_j))$ benutzt werden.

❷ Konzeptklassifikation

Im ersten eigentlichen Schritt wird das Anfragekonzept klassifiziert, d.h. in den Konzeptgraph einsinken gelassen und die Menge der direkten Nachbarn etc. bestimmt. Der Graph der Konzepte (inklusive des Anfragekonzeptes) wird im folgenden als Netz interpretiert, bei dem die einzelnen Konzepte mit ihren direkten Nachbarn durch (elastische) Seile miteinander verbunden sind.

Da die Anfragesprache mit der Konzeptbeschreibungssprache identisch ist, können beliebige (partielle) Informationen über Objekte als Konzeptterm C_q formuliert und als Anfrage benutzt werden. Der Konzeptterm C_q wird über die Konzeptklassifikation in den Konzeptverband einsortiert und die Menge

6. Es können hier auch andere Verfahren zur Indizierung der ABox-Objekte zum Einsatz kommen, es kann dann aber keine Konsistenz des Retrievalergebnisses mehr garantiert werden.

$\overline{\mathsf{C_q}} = \{\overline{C_{q_1}}, \ldots, \overline{C_{q_n}}\}$ der direkten oberen Nachbarn von C_q, sowie die Menge $\underline{\mathsf{C_q}} = \{\underline{C_{q_1}}, \ldots, \underline{C_{q_m}}\}$ der direkten unteren Nachbarn von C_q berechnet.

❸ Berechnung des Retrievalergebnisses

Zusätzlich zu der Menge der direkten oberen und unteren Nachbarn ist bei jedem der klassifizierten Konzepte auch die Menge der ABox Instanzen vermerkt, die die jeweilige Konzeptbeschreibung erfüllen.

Zur Bestimmung der für die Anfrage relevanten Instanzen der ABox »zieht« man an dem klassifizierten Anfragekonzept. Durch das Ziehen bewegen sich einige Konzepte des Konzeptnetzes. Die Menge der sich bewegenden Konzepte wird durch die Zugrichtung bestimmt. Für die Anfrage relevant sind die Konzepte, und damit die Instanzen, die sich durch das Ziehen am Anfragekonzept um einen bestimmten Betrag bewegen.

Als Ergebnis der Anfrage C_q wird eine Kombination der abgespeicherten Retrievalergebnisse der benachbarten Konzeptterme $\overline{C_q}$ und $\underline{C_q}$ zurückgeliefert. Gute Annäherungen an das exakte Ergebnis des starken bzw. schwachen Retrievals sind $\overline{O'}(C_q) = \bigcap \overline{\mathsf{C_q}}$ bzw. $\underline{O'}(C_q) = \bigcup \underline{\mathsf{C_q}}$.

❹ Anzeige der ermittelten Objekte

6.2. Fallrepräsentationen und Indexstrukturen

Nachdem im letzten Abschnitt gezeigt wurde, wie sich sie in dieser Arbeit entwickelten Techniken mit den »klassischen« Retrievalmethoden des fallbasierten Schliessens kombinieren lassen, soll in diesem Abschnitt untersucht werden, inwieweit sich die insbesondere die Repräsentationsformalismen von CTL dazu geeignet sind, auch klassische Fallrepräsentationen abzubilden bzw. inwieweit sich die Repräsentationsformalismen aneinander angenähert haben.

An dieser Stelle sollen daher zunächst die objekt-orientierten Fallrepräsentationen aus dem Projekt INRECA[7] vorgestellt werden. Sie sind ein Beispiel dafür, wie sich das fallbasierte Schließen bzw. die Sicht darauf im Laufe der Zeit weiterentwickelt hat. Eine Übersicht über die verschiedenen Phasen des fallbasierten Schließens in den letzten 12 Jahren gibt Richter in [Ric03] [8].

7. INRECA – INduction and REasoning from CAses.

8. Das wohl deutlichste Indiz dafür, daß sich die Sicht auf das fallbasierte Schließen gewandelt hat, ist die Tatsache, daß seit kurzem der traditionsreiche »German Workshop on Case Based Reasoning« in »Workshop on Experience Management« umbenannt wurde und die beiden GI Fachgruppen »Knowledge Management« und »Fallbasiertes Schließen« zur Fachgruppe »Experience Management« fusionierten.

6.2.1. INRECA

Repräsentation INRECA[9] [Weß95, Ber01] war eines der ersten fallbasierten Systeme das durchgängig eine objekt-orientierte Repräsentation des Gegenstandsbereiches und der Fälle benutzt. Fälle werden als Sammlungen von Objekten repräsentiert, die jeweils durch eine Menge von Attribut-Wert-Paare beschrieben werden. Die Strukturen der jeweiligen Objekte werden in den Klassendefinitionen festgelegt. Hier wird die Menge der Attribute (auch Slots genannt) und ihr jeweiliger Typ festgelegt. Die Klassen selbst sind in einer Klassenhierarchie angeordnet, die üblichen Vererbungsmechanismen sind implementiert. Neben einfachen Attributen, die als Werte z.B. Zahlen, Symbole oder Strings erlauben, gibt es so genannte relationale Attribute die als Werte Objekte einer beliebigen Klasse der Klassenhierarchie aufnehmen können. Relationale Attribute repräsentieren gerichtete binäre Relationen z.B. eine part-of Relation und werden dazu benutzt um komplexe Fallstrukturen abzubilden.

Basis für die Beschreibung der Klassenhierarchie und der Fälle ist die Repräsentationssprache CASUEL [MBC+94]. CASUEL ist eine frame-artige Sprache zur Definition und zum Austausch von objekt-orientierten Beschreibungen und auf Basis dieser Beschreibungen definierter Fallbasen. In seiner aktuellen Version erlaubt CASUEL zusätzlich die Definition von Regeln zur Fallvervollständigung und Fallanpassung sowie die Definition von Ähnlichkeitsmaßen.

Ähnlichkeitsmaß und Retrieval Ausgangspunkt für die Definition des Ähnlichkeitsmaßes in INRECA war die Überzeugung, daß es sich beim Ähnlichkeitsmaß um kompiliertes Wissen über die Nützlichkeit einer alten Lösung in einem neuen Kontext handelt. Wie anderes Wissen auch, sollte dieses Wissen ebenfalls explizit in dem jeweiligen Anwendungsbereich modelliert werden.

Die Verwendung einer objektorientierten Fallrepräsentation verlangt natürlicherweise nach einem Ähnlichkeitsmaß, daß sich an diese Methodologie anlehnt. In INRECA wird die Ähnlichkeit zweier Objekte auf Basis eines numerischen Ähnlichkeitsmaßes bestimmt. Ähnlich wie in AMS werden dazu Ähnlichkeitsmaße für Attribute zu Ähnlichkeitsmaßen für Objekte kombiniert. In der Terminologie von INRECA heißen die Ähnlichkeitsmaße über Attributen *Lokale Ähnlichkeitsmaße*, die Maße über Objekten *Globale Ähnlichkeitsmaße*.

Die globale Ähnlichkeit wird bottom-up und rekursiv definiert, d.h. für jedes einfache Attribut wird die Ähnlichkeit durch die Ähnlichkeit der Attributwerte definiert, für jedes relationale Attribut wird rekursiv die Ähnlichkeit der jeweiligen

9. INRECA entstand von 1992 bis 1999 mit einem Aufwand von ca. 55 Personenjahren im Rahmen zweier durch die EU Kommision im Rahmen des ESPRIT-Programmes mit einem Volumen von ca. 3 Millionen EUR geförderter Projekte. Während im ersten Projekt die Entwicklung einer integrierten Technologie im Mittelpunkt stand, lag der Schwerpunkt des zweiten Projektes auf der Entwicklung einer Methodologie zur Durchführung von CBR Projekten.

Subobjekte bestimmt. Die einzelnen Ähnlichkeiten werden dann z.B. mit Hilfe einer gewichteten Summe aggregiert ^{10}n [BS98].

Auf der operationalen Ebene wird das Retrieval in INRECA auf Basis eines erheblich erweiterten k-d-Baumes implementiert [Weß95]. In INRECA werden Fälle somit als Punkte in einem k-dimensionalen Raum repräsentiert. Solange die Systeme für den Fallvergleich und die Bestimmung nur bekannte Attribute heranziehen, bereiten fehlende Attributwerte keine Probleme.

Bei der Verwendung einer multidimensionalen Zugriffsmethode wie einem k-d-Baum bereiten fehlende Attributwerte erhebliche Probleme.

Problem: Fehlende und vage Attributwerte

Die Übersetzung eines Falles mit fehlenden Attributwerten resultiert zwangsläufig in einem räumlich ausgedehnten Fall. Für jedes fehlende Attribut bleibt eine Dimension offen, d.h. das entsprechende Objekt hat in dieser Dimension eine unbegrenzte Ausdehnung.

Weß löst dieses Problem auf die von Mehlhorn [Meh84] vorgeschlagene Art und Weise. Die multidimensionale Zugriffsmethode wird modifiziert. Statt eines Binärbaumes in dem bei jedem Split nach $< \mathit{und} \geq$ (bzw. $\leq \mathit{und} >$) unterschieden wird, hat jeder Knoten vier Nachfolger, unterschieden nach $<, =, >$ und $\perp$. Damit ist die Behandlung fehlender Attributwerte möglich.

Allerdings ist auf diese Weise die Behandlung vager und unscharfer Attributwerte, z.B. durch die Angabe eines Intervalles für den Wert des Attributes nicht möglich. Abbildung 6.1 verdeutlicht diesen Sachverhalt.

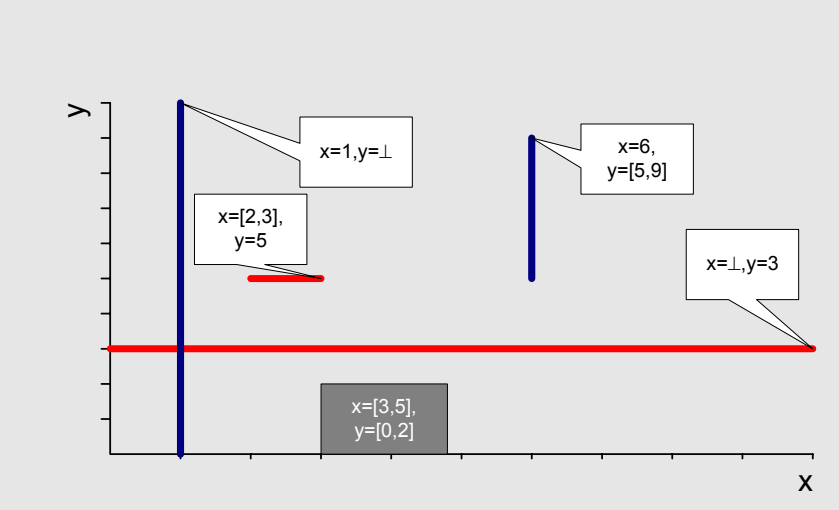


Abbildung 6.1.

Fehlende, vage und unscharfe Attributwerte

10. i

Fazit INRECA stellt einen Meilenstein in der Geschichte des fallbasierten Schließens dar. Als eines der ersten Systeme bietet es dem Nutzer die Möglichkeit einer Objektorientierten Beschreibung sowohl des Gegenstandsbereiches als auch der zu verwendenden Ähnlichkeitsmaße.

Es kombiniert diese Flexibilität in der Beschreibung mit einem hochperformanten Retrieval auf Basis eines erweiterten k-d-Baumes. Allerdings führt dies dazu, daß wie in Kapitel 1 dargestellt, im Regelfalle für einen einzelnen Fall eine große Anzahl von Dimensionen des k-d-Baumes undefiniert bleiben.

Während sich dieses Problem noch durch die Erweiterung des k-d-Baumes lösen läßt, ist eine darüber hinausgehende flexiblere Beschreibungen der Fälle (etwa durch Angabe komplexer Attributwerte etwa einer Menge von Alternativen bzw. eines Intervalles, wie sie z.B. bei der komplexen Verhaltenssimulation im letzten Kapitel verwandt wurden) ist nicht möglich.

6.2.2. Konkrete Gegenstandsbereiche und Indexstrukturen

Am besten lassen sich die Zusammenhänge anhand der geometrischen Interpretation der Ausdrucksstärke der verschiedenen betrachteten konkreten Gegenstandsbereiche erläutern.

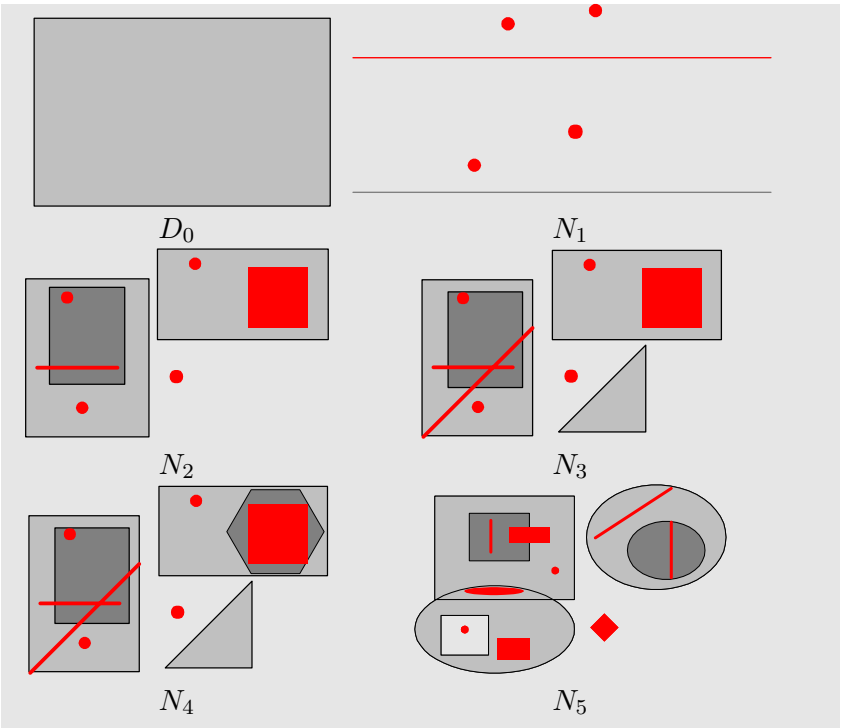


Abbildung 6.2.

Die Ausdrucksstärke numerischer konkreter Gegenstandsbereiche im Vergleich

Diese Interpretation läßt sich wie folgt beschreiben: Die Ausdrucksstärke der Repräsentationssprache im Zusammenspiel mit der Ausdrucksstärke der konkreten Gegenstandsbereiche bestimmt, welche Mengen im n-dimensionalen Raum beschrieben werden können.

Klassische Fälle des fallbasierten Schliessens (d.h. Einzelerfahrungen) sind, gewöhnlicherweise Punkte im n-dimensionalen Raum. Bereits der Gegenstandsbereich $\mathbf{N_1}$ stellt die Ausdruckstärke bereit, um Punkte im n-dimensionalen Raum zu beschreiben. Diagnosen können als Disjunktionen über den verschiedenen sie definierenden Einzelereignissen definiert werden.

Analysiert man, welche Komplexität für die Beschreibung der Fälle in INRECA benötigt wird, so reicht ein konkreter Gegenstandsbereich der Ausdrucksstärke des Gegenstandsbereiches $\mathbf{N_1}$ aus [11].

$\mathbf{N_2}$ besitzt die Ausdrucksstärke, um im Zusammenspiel mit den Ausdrucksmitteln des abstrakten Gegenstandsbereiches achsenparallele n-dimensionale Würfel zu beschreiben.

Achsenparallele Hyperwürfel werden häufig in Indexstrukturen für Nearest-Neighbour-Algorithmen benutzt. Auch ein k-d-Baum ist eine auf achsenparallelen Hyperwürfeln beruhende Indexstruktur. Damit kann die Indexstruktur von INRECA in einer um einen konkreten Gegenstandsbereich $\mathbf{N_2}$ erweiterten Beschreibungslogik definiert werden [12].

Dies bedeutet, daß möglich ist, parallel zu der sukzessiven Erstellung einer semantischen Konzepthierarchie eine syntaktische Konzepthierarchie, die die Indexstruktur eines syntaktischen Indexierungsverfahren darstellt, zu beschreiben. Diese kann dann entweder durch die Basisalgorithmen oder den darauf aufbauenden Algorithmen zum heuristischen ähnlichkeitsbasierten Retrieval genutzt werden. Die Algorithmen zur Berechnung der einzelnen syntaktischen Konzepte, d.h. der konkreten Indexstrukturen, müssen

Die Tatsache, daß auch syntaktische Konzepte innerhalb des Repräsentationsformalismus abgebildet werden können, könnte darüber hinaus auch dann interessant werden, wenn es darum geht, verschiedene Indexierungsverfahren zu vergleichen.

Mit zunehmender Ausdrucksstärke der betrachteten numerischen Gegenstandsbereiche steigt die "Komplexität"der repräsentierbaren Mengen. Anstatt achsenparalleler Hyperwürfel lassen sich sukzessive beliebige Polytope ($\mathbf{N_4}$) bis hin zu sog. semi-algebraischen Mengen ($\mathbf{N_7}$) beschreiben.

11. Bei dieser Betrachtung werden nicht numerische Attributwerte vernachlässigt. Weiterhin soll angemerkt werden, daß zur Abbildung der Klassenhierarchie in einer Beschreibungslogik weder Negation noch Disjunktion aus dem abstrakten Gegenstandsbereich benötigt werden. Relationale Attribute können als Rollen bzw. Features realisiert werden.

12. Aufgrund der Tatsache, daß sich die Fälle einer Klasse in mehreren Buckets befinden können wird in diesem Falle im abstrakten Gegenstandsbereich die Disjunktion benötigt.

Diese Komplexität wird einerseits benötigt, um Abstraktionen der Einzelerfahrungen, wie z.B. die im letzten Kapitel definierten Konzepte inklusive des vorhandenen Hintergrundwissens beschreiben zu können.

Andererseits erlauben solch ausdrucksstarke konkrete Gegenstandsbereiche auch die Repräsentation von Alternativen, Messungenauigkeiten, und Unschärfe bei der Beschreibung von Einzelerfahrungen. Etwa dann, wenn wie in der komplexen Verhaltenssimulation des letzten Kapitels, ein Attribut alternativ einen von mehreren Werten (evtl. in Abhängigkeit vom Wert eines anderen Attributes) annehmen kann.

Aber auch dann, wenn nur ungenaue Beobachtungen gemacht werden konnten oder Lücken in den Erinnerungen bestehen.

Insbesondere sie erste Art der Anforderung an eine flexiblere Fallrepräsentation wurde auch im Bereich des fallbasierten Schließens erkannt und dort unter dem Begriff ≫generalised cases≪ behandelt.

6.2.3. Generalisierte Fälle

≫Generalised cases≪ wurden von Bergmann entwickelt [Ber96, BVW99, BV99, MB02], und haben ihren Ausgangspunkt in den Anwendungsbereichen: ≫Wiederverwendung existierender Designs≪ und ≫Verwaltung von Intellectual Properties≪.

Generalisierte Fälle werden nicht nur durch einen Punkt im Raum repräsentiert sondern durch einen Teilraum. Damit stellt ein einzelner Generalisierter Fall eine Lösung für eine ganze Menge artverwandter Probleme dar. Die Lösungen die durch den generalisierten Fall repräsentiert werden sind im wesentlichen nur leichte Variationen der gleichen ≫prinzipiellen≪ Lösung [13]. Vereinfachend dargestellt ist ein generalisierter Fall eine implizite Repräsentation einer (möglicherweise unendlichen) Menge von Einzelfällen.

Es ist unmittelbar klar, daß mit den in dieser Arbeit entwickelten Methoden der Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen die Ausdruckstärke zur Beschreibung von generalisierten Fällen zur Verfügung steht. Das Beispiel 5.2 zur komplexen Verhaltenssimulation zeigt, wie ein generalisierter Fall in CTL repräsentiert werden kann [14].

Es werden die folgenden Arten von generalisierten Fällen unterschieden:

- Ein ≫Constant Solution Generalised Case≪ beschreibt eine Menge von Problemen die alle die gleiche Lösung besitzen.

13. Hier wird ähnlich wie bei INRECA explizit zwischen ≫Problem Space≪ und ≫Solution Space≪ unterschieden die Anfrage an das fallbasierte System bezieht sich immer nur auf den ≫Problem Space≪.

14. Es stellt sich CTL dann eher die Frage, welche Kriterien entscheidend sind, um aus einem generalisierten Fall ein Konzept zu machen.

- Ein ≫Functional Solution Generalised Case≪ repräsentiert eine Menge von Problemen für die eine Abbildung (Funktion) existiert, so daß die jeweilige Lösung in Abhängigkeit von den Eingangsparametern definiert werden kann.
- Ein ≫Independent Alternative Solution Generalised Case≪ repräsentiert einen Menge von Problemen mit der gleichen Menge alternativer Lösungen
- Ein ≫Dependant Alternative Solution Generalised Case≪ repräsentiert eine Menge von Problemen für den für jeden Satz von Parametern eine andere Menge von Lösungen existieren kann.

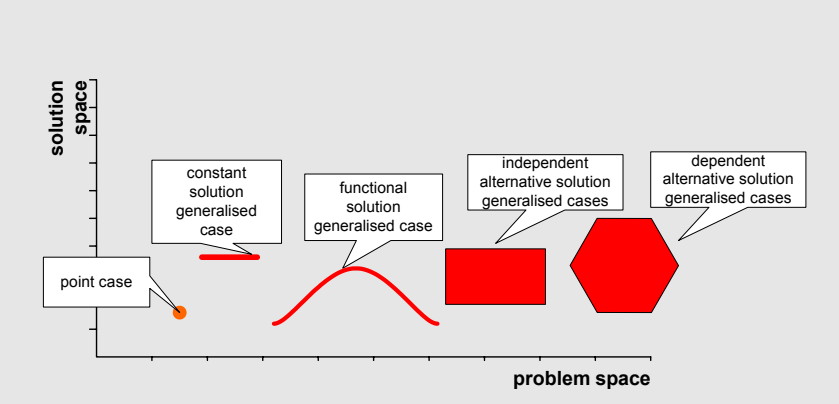


Abbildung 6.3.

Die verschiedenen Arten generalisierter Fälle

Die verschiedenen Typen von generalisierten Fällen sind in Abbildung 6.3 visualisiert. Im Vergleich mit der Ausdruckstärke der verschiedenen konkreten Gegenstandsbereiche wird deutlich, daß die verschiedenen Arten der generalisierten Fälle in der Regel durch einen konkreten Gegenstandsbereich entsprechender Ausdruckstärke beschrieben werden können.

Besonders interessiert ist man nun an kompakten Repräsentationen generalisierter Fälle. Vorgeschlagen und eingesetzt werden einerseits Repräsentation auf Basis von Hyperwürfeln bzw. Hyperrechtecken, wie sie mit dem konkreten Gegenstandsbereich $\mathbf{N_2}$ beschrieben werden können. Andererseits werden auch auf Constraint-Systemen basierende Repräsentationen untersucht. Hier können konkrete Gegenstandsbereiche mit einer Ausdrucksstärke ab $\mathbf{N_4}$ zum Einsatz kommen. Für die von Bergmann verwendeten Beispiele aus dem Bereich der ≫Intellectual Properties≪ ist, das es sich bei den Constraints um multivariate quadratische Polynome handelt, der konkrete Gegenstandsbereich $\mathbf{N_5}$ ausreichend.

Fazit ≫Generalised cases≪ entstanden aus den gleichen Anforderungen heraus, die in dieser Arbeit zur Einführung von ausdrucksstarken konkreten Gegenstands-

bereichen führten. daher ist es nicht verwunderlich, daß sie auch mit Hilfe dieser repräsentiert werden können. Ein Beispiel dafür wurde in Abschnitt 5.3.2 gegeben.

Im Unterschied zu dieser Arbeit entfällt bei den »generalised cases« die Möglichkeit der Repräsentation von funktionalen und anderen Zusammenhängen in der Repräsentation der zugrundeliegenden Klassen [15].

Des weiteren sind »generalised cases« nur als Bestandteil der Fallbasis zugelassen. Die Anfrage selbst ist in jedem Falle punktförmig. Der in dieser Arbeit vorgestellte Ansatz ist vollkommen symmetrisch und erlaubt auch mengenwertige Anfragen (siehe Abschnitt 5.3.2).

6.3. Zusammenfassung

In diesem Kapitel wurde gezeigt, daß sich logikbasierte und ähnlichkeitsbasierte Ansätze des fallbasierten Schließens ergänzen und nicht widersprechen. In vielen Bereichen nähern sie sich sogar deutlich an.

Nah einigen Grundsätzlichen Bemerkungen zum Verhältnis der beiden Ansätze wurde im ersten Abschnitt dieses Kapitels gezeigt, wie die Basisinferenzen der Konzept- und Objektklassifikation sowie des Retrievals dazu verwendet werden können, um klassische Verfahren des fallbasierten Schließens zu implementieren. Dabei werden die verschiedenen Basisinferenzen dazu benutzt, Indexmengen bzw. Indexvektoren zu berechnen, die dann die Grundlage für die heuristischen Algorithmen bilden.

Die verschiedenen Retrievalverfahren bauen zwar auf den Basisinferenzen auf, sind aber bewusst nicht in den eigentlichen inferenzbasierten Kern von CTL integriert, sondern als separater Layer realisiert. Dadurch werden deduktive und heuristische Ähnlichkeitsbestimmung klar und sauber voneinander getrennt.

Während die deduktive Ähnlichkeitsbestimmung innerhalb des Beschreibungslogischen Systems erfolgt, setzt die heuristische Ähnlichkeitsbestimmung auf den Ergebnissen der deduktiven Ähnlichkeitsbestimmung auf. So ist es möglich zu einem späteren Zeitpunkt zusätzliches Wissen in den beschreibungslogischen Kern einzufügen, ohne daß die Gefahr besteht, daß dieses Wissen durch heuristische Ähnlichkeitsbegriffe verwässert wird.

Im zweiten Abschnitt wurde dann gezeigt, daß sich einerseits die verwendeten objekt-orientierten Beschreibungssprachen aneinander annähern und andererseits mit den in dieser Arbeit entwickelten Repräsentationsmechanismen auch die klassischen Indexstrukturen fallbasierter Systeme abbilden lassen.

Zwar empfiehlt es sich nicht, das tatsächliche ähnlichkeitsbasierte Retrieval auf Basis dieser Indexstrukturen durchzuführen, für einen Vergleich verschiedener Ansätze kann aber eine Einordnung in das durch diese Arbeit vorgegebene

15. Zumindest wird in der Literatur nicht näher darauf eingegangen.

Rahmenwerk durchaus nützlich sein, wie es mit der Einordnung des Ansatzes der »generalised cases« im letzten Unterabschnitt geschehen ist.

Zusammenfassen läßt sich feststellen, daß CTL das in der Motivation am Anfang dieser Arbeit geforderte breite Spektrum an Verfahren zum ähnlichkeitsbasierten Retrieval zur Verfügung stellt. Aus diesem Spektrum an Retrievalmethoden kann der Benutzer dann die Verfahren auswählen, die seine Aufgabenstellung und seinen aktuellen Wissensstand am besten unterstützen, bzw. die Retrievalmethoden dem wachsenden Wissenstand anpassen.

Ist z.B. das Hintergrundwissen noch nicht vorhanden, reduziert sich die deduktive Ähnlichkeitsbestimmung auf die Identität, die gesamte Ähnlichkeitsbestimmung erfolgt auf Basis von Heuristiken. Mit wachsendem Wissen (insbes. Hintergrundwissen) können fehlende Attributwerte deduktiv bestimmt oder eingegrenzt werden und bilden somit eine bessere Basis für die heuristische Ähnlichkeitsbestimmung.

7 Zusammenfassung

Eine Zusammenfassung der erzielten Ergebnisse schließt die Arbeit ab. Gleichzeitig soll diese Gelegenheit auch dazu genutzt werden, den Prozeß und den Kontext zu schildern, in dem die Forschungsarbeiten stattfanden.

Ausgangspunkt und Motivation zur Beschäftigung mit dem Bereich der Unterstützung von Wartungstechnikern waren zwei Projekte am Hamburger Labor für Künstliche Intelligenz:

- AMS – Unterstützung von Servicechemikern

  Im Rahmen des Drittmittelprojektes AMS [Kam94a] wurde ein objektorientiertes Unterstützungssystem für die Servicechemiker eines Kühlschmierstoffherstellers entwickelt. Zielsetzung war es, den Servicechemiker sowohl bei der Auswahl geeigneter Kühlschmierstoffe für neue Anlagen, als auch bei der Untersuchung und Behebung von Problemen an bereits existierenden Anlagen zu unterstützen.

- Falldaten – Grundlagenforschung zur fallbasierten Diagnose technischer Systeme

  Gegenstand des daran anschließenden Projektes Falldaten war Grundlagenforschung im Bereich geeigneter Repräsentationen und Inferenzen für die fallbasierte Diagnose technischer Geräte. In dieses von der DFG geförderte Projekt flossen einerseits die in AMS gewonnenen Erkenntnisse ein, andererseits wurden in ihm die während des Einsatzes von AMS deutlich gewordenen, grundsätzlichen Probleme und Beschränkungen des in AMS gewählten Ansatzes untersucht und nach einer Lösung dieser Beschränkungen gesucht.

Ausgangspunkt: Fallbasiertes Schließen

Am Anfang stand der Entschluß, für die Realisierung von AMS einen fallbasierten Ansatz zu wählen. Getragen wurde dieser Entschluß zum einen durch die offensichtlichen Nachteile anderer Diagnoseansätze – etwa regel- oder modellbasierte Diagnose – im Szenario von AMS. Zum anderen wurden fallbasierte Systeme in anderen Aufgabenstellungen aus dem Anwendungsbereich des Service und Supports, erfolgreich eingesetzt. Insbesondere die Unterstützung der Mitarbeiter so

genannter Help-Desks bei der Beantwortung der Fragen, die an eine Hotline eines Geräteherstellers gerichtet werden, ist hier zu nennen.

Für den fallbasierten Ansatz sprachen im Falle von AMS insbesondere die Möglichkeit, durch die direkte Nutzung des Fallwissens den Flaschenhals der Wissensakquisition zu umgehen bzw. zu erweitern, und das Potential, gespeichertes Wissen auf unterschiedliche Art und Weise zu nutzen.

Experten benötigen einen neuartigen Ansatz zur Diagnoseunterstützung

Eine Anforderungsanalyse ergab, daß Experten im Vergleich zu den angelernten Kräften des Help-Desks einen veränderten Ansatz zur fallbasierten Diagnoseunterstützung benötigen [KK92, Kam93a, Kam94c]. Feste Fallrepräsentationen in Form von Attribut-Wert-Vektoren und numerische Ähnlichkeitsmaße, wie sie in Help-Desk Systemen verwendet werden, sind ungeeignet. Stattdessen wird ein objektorientierter Wissensrepräsentationsmechanismus zur Beschreibung des vorhandenen Wissens, zusammen mit einer flexiblen, auf einer klaren Retrievalsemantik beruhenden Anfragesprache und Verfahren zur Ableitung neuer Informationen benötigt[KPB96].

AMS – Ein objektorientierter Ansatz zum ähnlichkeitsbasierten Retrieval

Daher wurde im Rahmen von AMS ein objektorientierten Ansatz für den inhaltsorientierten Zugriff auf situationsrelevante Daten entwickelt [Kam94a, Kam94b, Kam94c]. Am Beginn der Entwicklung eines Systems steht in diesem Ansatz die Modellierung des Gegenstandsbereiches, d.h. des vorhandenen Wissens über zu unterstützende Geräte. Nach der Modellierung des Gegenstandsbereiches werden die verschiedenen von einem Servicechemiker betreuten Kühlschmierstoffanlagen beschrieben und die einzelnen Komponenten der Anlage entsprechend ihrer Position im Klassenverband indexiert. Dies geschieht ebenfalls mit den verschiedenen Beobachtungen (etwa Meßwerten), die der Servicechemiker im Laufe der Zeit an einer Anlage macht. So entsteht eine Indexstruktur, bei der die Indizes der einzelnen Einträge nicht aufgrund syntaktischer Kriterien berechnet werden, sondern semantisch motiviert sind.

Ausgehend von einer Hierarchie von Vergleichsrelationen für primitive Datentypen wie Zahlen und Strings werden Vergleichsrelationen für Objekte realisiert. Die Vergleichsrelationen werden dann zur Formulierung von inhaltsorientierten Anfragen benutzt, die anhand der Indexstruktur beantwortet werden. Die Anfragen selbst werden über eine graphische Benutzeroberfläche formuliert und interaktiv modifiziert. Dies geschieht durch Navigation in der Domänenbeschreibung bzw. der Hierarchie der Vergleichsrelationen.

Neue Datentypen, z.B. Texte oder Multimediadatentypen, lassen sich leicht in diesen Rahmen einbetten. Damit ist es leicht möglich z.B. Techniken des Information Retrieval integrieren [Kam94d]. Insgesamt stellt AMS somit ein objektorientiertes Framework zum ähnlichkeitsbasierten Retrieval zur Verfügung, das neben der Wartungsunterstützung auch für andere Anwendungsbereiche eingesetzt werden kann. Es erlaubt den flexiblen Zugriff auf die Informationen und eine adäquate Repräsentation der Gerätetypen und der Gerätestruktur.

Gleichzeitig zu AMS entstanden eine Reihe weiterer objektorientierter Syste-

me zum fallbasierten Schließen. In diesen Systemen lag der Schwerpunkt jedoch auf der Erweiterung numerischer Ähnlichkeitsmaße auf Objektstrukturen und der Entwicklung effektiver Indexstrukturen, für diese Art der Ähnlichkeitsbestimmung. Stellvertretend sei hier das in Kapitel 6 kurz vorgestellte System INRECA erwähnt.

Objektorientierte Ansätze erfüllen die Anforderungen nur teilweise

AMS und andere objektorientierte Ansätze zum fallbasierten Schließen stellen im Vergleich zu Attribut-Wert-Vektor basierten Systemen einen großen Schritt vorwärts dar. Sie weisen aber dennoch eine Reihe von Mängeln auf.

Auf numerischen Ähnlichkeitsmaßen basierende Systeme bieten einen automatischen Indexierungsmechanismus, sie sind aber oft zu unflexibel und die Semantik des Retrievals auf Basis des numerischen Ähnlichkeitsmaßes ist nicht transparent genug. AMS bietet eine flexible Anfragesprache und die Semantik des Retrievals ist aufgrund der Definition der Vergleichsrelationen leicht vermittelbar. Dafür werden in AMS die Objekte nicht automatisch indexiert, die Zuweisung der Indizes geschieht durch den Benutzer [Kam94d].

Zusätzlich zu dem Wissen über Gerätetypen und Strukturen, die sehr gut in objektorientierten Repräsentationen beschrieben werden können, existiert in technischen Bereichen Hintergrundwissen, etwa in Form von physikalischen Gesetzen wie dem Ohm'schen Gesetz oder Verhaltensmodellen [Kam93b, Kam94e].

Keiner der objektorientierten Ansätze bietet ausreichende Möglichkeiten zur Integration dieses Hintergrundwissens. Dieses Wissen ist aber nicht nur für die Diagnoseunterstützung wichtig, es wird auch zur Unterstützung anderer wichtiger Aufgaben eines Experten der zweiten Supportebene benötigt, wie etwa Simulation, What-If-Analyse etc.

Falldaten – Von objektorientierten Systemen zu Beschreibungslogiken

Die Entwicklung eines Ansatzes zum intelligenten, ähnlichkeitsbasierten Retrieval, das diese Mängel überwindet, war Ziel des Projektes Falldaten. Insbesondere wurde die transparente Integration des Hintergrundwissens, die Entwicklung einer durchgehend deklarativen Semantik und die automatische semantische Indizierung von Objekten angestrebt.

Erreicht wurden diese Ziele durch den in dieser Arbeit beschriebenen Übergang von objektorientierten Repräsentationen zu Beschreibungslogiken mit ausdrucksstarken konkreten Gegenstandsbereichen.

Beschreibungslogiken sind ein Wissensrepräsentationsformalismus zur strukturierten Repräsentation eines Fragmentes der Prädikatenlogik erster Stufe und stellen eine, auf einer formalen Semantik beruhende Erweiterung objektorientierter Repräsentationen dar. Auf Basis einer modell-theoretischen Semantik ist es möglich, leistungsstarke Inferenzen zur Konzept- und Objektklassifikation zu definieren. Diese können zur automatischen semantischen Indexierung von Klassen und Instanzen eingesetzt werden.

Für eine Reihe von Beschreibungssprachen wurden korrekte und vollständige Algorithmen für die verschiedenen Inferenzen entwickelt. Werden diese eingesetzt, so kann garantiert werden, daß die automatisch berechneten Indizes immer korrekt und optimal sind.

Der Einsatz von Beschreibungslogiken für das fallbasierte Schließen wurde auch von anderer Seite in einer Reihe von weiteren Projekten untersucht. Diese Ansätze zum fallbasierten Schließen auf Basis von Beschreibungslogiken bauen auf Systemen auf, die nur unvollständige Inferenzen bieten. Der Einsatz unvollständiger Algorithmen führt aber dazu, daß die Berechnung der Indizes nicht nur suboptimal sondern evtl. auch falsch sind. In dem in dieser Arbeit entwickelten System werden im Gegensatz dazu korrekte und vollständige Algorithmen eingesetzt.

Andere Ansätze zum fallbasierten Schließen auf Basis von Beschreibungslogiken benutzen im wesentlichen die Basisinferenz der Objektklassifikation. In dieser Arbeit wurde gezeigt, daß sich eine ganze Reihe weiterer Inferenzen für das ähnlichkeitsbasierte Retrieval einsetzen lassen [Kam96a]. So ist das Retrieval eine zusätzliche, zur Objektklassifikation duale Basisinferenz, die Konzeptklassifikation kann als Annäherung an das Retrieval benutzt werden.

„Schwache“ Varianten der Objektklassifikation und des Retrievals, lassen sich ebenfalls für das ähnlichkeitsbasierte Retrieval einsetzen. Damit bieten schon allein die verschiedenen Basisinferenzen die Grundlage für eine Reihe allgemeiner Verfahren zum ähnlichkeitsbasierten Retrieval, die schon für einen großen Teil der Anwendungen ausreichend sind.

Saubere Erweiterung um Retrievalverfahren des ≫klassischen≪ fallbasierten Schließens

Die Basisinferenzen bilden darüber hinaus die Grundlage für ein breites Spektrum weiterer Retrievalverfahren [Kam96a]. Alle Inferenzen sind semantisch exakte Indexierungsverfahren, die Objekte und Klassen indexieren und Indexmengen für die zu indexierenden Objekte berechnen. Eine Vielzahl von Ähnlichkeitsbegriffen aus dem fallbasierten Schließen oder dem Information Retrieval setzen auf Indexmengen bzw. Indexvektoren auf und können somit problemlos auf die von den Basisinferenzen berechneten Indexmengen aufgesetzt werden.

Beispiele sind die unterschiedlichen Assoziationsmaße über Mengen wie der Jaccard Koeffizient, der Dice Koeffizient, die im fallbasierten Schließen benutzte Contrast Rule von Tversky [BHW92, Tve77] oder das im Information Retrieval verbreitete Kosinusmaß [Rij79, Fuh93].

Das breite Spektrum an Retrievalverfahren auf Basis einer formalen Semantik erfüllt die Anforderungen an ein flexibles Retrieval in Verbindung mit einer automatischen semantischen Indexierung der Objekte und einer durchgehend deklarativen Semantik.

Konkrete Gegenstandsbereiche – Eine essentielle Erweiterung von Beschreibungslogiken

Dennoch sind Beschreibungslogiken in dieser Form – und damit auch alle anderen bislang vorgestellten Ansätze zum ähnlichkeitsbasierten Retrieval auf Basis von Beschreibungslogiken – für die Realisierung eines Second-Level-Support-Systems ungeeignet, da in ihnen nur die Definition von Konzepten über einem sogenannten abstrakten Gegenstandsbereich möglich sind.

Übertragen auf objektorientierte Repräsentationen bedeutet dies, daß nur Klassen definiert werden können, die keine Attribute enthalten. D.h. es können zwar Beziehungen zwischen Objekten durch Assoziationen beschrieben werden, die Definition von Attributen und damit die Repräsentation und Speicherung von Zah-

len, Strings etc. als Werte dieser Attribute ist aber nicht möglich. Zur Repräsentation und Speicherung realer Daten sind primitive Datentypen wie z.B. Zahlen aber unverzichtbar. So ist ein Ansatz zur Unterstützung von Wartungstechnikern, in dem es nicht möglich ist, Meßwerte von Parametern abzulegen, einfach unvorstellbar.

Darüberhinaus besteht im Anwendungsbereich des Service-Supports ein großer Teil des Hintergrundwissens aus Wissen über Zusammenhänge zwischen konkreten Parametern, etwa in Form von physikalischen Gesetzen. Eine Grundvoraussetzung zur Repräsentation dieses Hintergrundwissens ist die Integration solcher „konkreter“ Parameter.

Die Motivation zum Übergang von objektorientierten Repräsentationen zu Beschreibungslogiken war die Tatsache, daß letztere einen semantisch klaren Ähnlichkeitsbegriff – die Subsumption – für beliebig komplexe Strukturen zur Verfügung stellen, der zugleich die Basis für die verschiedenen Retrievalverfahren und die automatische semantische Indexierung bildet. Zugunsten dieser Eigenschaften kann aber nicht auf Attribute und primitive Datentypen verzichtet werden.

Ein solcher Verzicht würde dazu führen, daß praktisch keine Instanzen abgelegt und kein Hintergrundwissen repräsentiert werden kann. Um also Beschreibungslogiken einsetzen zu können, müssen primitive Datentypen in den Formalismus integriert werden [KNPB94, Kam94d].

Eine solche Integration darf aber ihrerseits nicht dazu führen, daß die positiven Eigenschaften der deklarativen Retrievalsemantik und der automatischen semantischen Indexierung verloren gehen. Daher müssen bei einer Integration die primitiven Datentypen volle Berücksichtigung in den Inferenzmechanismen der Beschreibungslogik finden. Erst durch eine solche Integration konkreter Gegenstandsbereiche werden Beschreibungslogiken erstmals zu einer ernstzunehmenden Erweiterung objektorientierter Systeme.

Baader und Hanschke [BH91a, Han96] haben mit ihren zulässigen konkreten Gegenstandsbereichen die theoretische Grundlage für eine solche Integration primitiver Datentypen gelegt. Zulässige konkrete Gegenstandsbereiche sind im wesentlichen Relationalstrukturen, die eine Reihe zusätzlicher Kriterien erfüllen.

In AMS wurde für das Retrieval eine Hierarchie von Vergleichsrelationen benutzt, die über den jeweiligen Attributtypen definiert waren. Zulässige konkrete Gegenstandsbereiche stellen eine Verallgemeinerung dieser Technik dar, indem sie in der Lage sind, die Subsumptionshierarchie beliebiger, in dem jeweiligen Gegenstandsbereich formulierbarer Prädikate zu berechnen.

Ausreichende Ausdrucksstärke der konkreten Gegenstandsbereiche vorausgesetzt, läßt sich sogar das Hintergrundwissen (z.B. physikalische Gesetze, Normal- und Fehlverhalten) in diesem Formalismus beschreiben. Somit ist eine um konkrete Gegenstandsbereiche erweiterte Beschreibungslogik eine logische Weiterentwicklung von AMS, die die dort vorhandenen Mängel überwindet und damit die Anforderungen an ein System zur Unterstützung von Wartungstechnikern erfüllt.

Existierende beschreibungslogische Systeme genügen nicht den Anforderungen

Ein Vergleich existierender beschreibungslogischer Systeme [KW96a, KW96b] zeigte, daß keines die Anforderungen an konkrete Gegenstandsbereiche erfüllte. So realisieren die meisten Systeme primitive Datentypen nicht über konkrete Gegenstandsbereiche, sondern über Ad-hoc-Ansätze. Diese Ansätze sind nicht vollständig in die Inferenzen des beschreibungslogischen Systems integriert. Zudem sind sie im wesentlichen über speziell dafür entwickelte und eng mit dem jeweiligen System verwobene Module realisiert. Dies macht die Implementierung neuer konkreter Bereiche kompliziert und aufwendig.

Einzig das beschreibungslogische System TAXON benutzt die Technik der zulässigen konkreten Gegenstandsbereiche zur Integration primitiver Datentypen und bietet eine generische Schnittstelle zur Integration neuer konkreter Gegenstandsbereiche in die Beschreibungslogik. In TAXON sind die konkreten Gegenstandsbereiche im Rahmen des beschreibungslogischen Systems selbst implementiert. Dieser Ansatz erschwert die Realisierung ausdrucksstärkerer konkreter Gegenstandsbereiche erheblich. Werden z.B. im Bereich der Computeralgebra oder des Constraint Logic Programming neue Entscheidungsprozeduren entwickelt, müssen diese jeweils im Rahmen des beschreibungslogischen Systems reimplementiert werden, ein unvertretbarer Aufwand. Wie für alle anderen in dieser Arbeit untersuchten beschreibungslogischen Systeme gilt allerdings auch für TAXON, daß die realisierten konkreten Gegenstandsbereiche bei weitem nicht ausdrucksstark genug sind, um das Hintergrundwissen zu repräsentieren.

CTL – Ein modulares beschreibungslogisches System mit ausdrucksstarken konkreten Gegenstandsbereichen

Aus diesen Gründen wurde auf Basis von TAXON das beschreibungslogische System CTL entwickelt [KW96b, KW96a], in dem die Realisierung neuer konkreter Gegenstandsbereiche wesentlich vereinfacht wird. Sie reduziert sich im wesentlichen auf eine Anpassung der Schnittstellenfunktionen. Der Rest des beschreibungslogischen Systems bleibt unverändert. Damit erweitert CTL die modulare Architektur tableauxbasierter Beschreibungslogiken auf den Bereich der konkreten Gegenstandsbereiche. Je nach Aufgabenstellung ist es damit möglich, nur die dort benötigten konkreten Gegenstandsbereiche mit dem beschreibungslogischen Kern zu koppeln.

Darüber hinaus ist es in CTL möglich, nicht nur die Subsumptionshierarchie der Klassen und Objekte zu berechnen, sondern sich das bei der Berechnung dieser Hierarchie ermittelte Modell auch anzeigen zu lassen.

Eine Systematik für ein weites Spektrum konkreter Gegenstandsbereiche

Unterschiedliche Aufgabenstellungen benötigen unterschiedliche konkrete Gegenstandsbereiche, sowohl in Bezug auf die Art und Anzahl der primitiven Datentypen als auch in Bezug auf die Ausdrucksstärke der konkreten Prädikate über einem bestimmten primitiven Datentyp.

Daher wurde untersucht, welche zulässigen konkreten Gegenstandsbereiche für verschiedene primitive Datentypen zur Verfügung stehen. Ergebnis dieser Untersuchungen ist ein breites Spektrum zulässiger konkreter Gegenstandsbereiche sowohl über den Zahlen als auch über Strings [Kam97b]. Aus der Menge der zulässigen konkreten Gegenstandsbereiche können diejenigen ausgewählt werden, die der vor-

liegenden Aufgabenstellung in Bezug auf die Ausdrucksstärke und Komplexität am besten gerecht werden.

Wesentliches Kriterium für die Zulässigkeit eines konkreten Gegenstandsbereiches ist die Entscheidbarkeit des folgenden Problems: „Ist eine endliche Konjunktion von Prädikaten aus dem konkreten Gegenstandsbereich erfüllbar?“. Neben diesem theoretischen Kriterium ist aber auch die Komplexität der Entscheidungsprozeduren und die Verfügbarkeit von Implementationen dieser Prozeduren interessant.

Einfache konkrete Gegenstandsbereiche – etwa die bislang in beschreibungslogischen Systemen realisierten – stellen relativ geringe Anforderungen an die Entscheidungsprozeduren.

Zur Bestimmung der Zulässigkeit komplexer konkreter Gegenstandsbereiche, wie sie etwa zur Beschreibung des Hintergrundwissens benötigt werden, muß auf grundlegende Ergebnisse der Informatik und der Mathematik zurückgegriffen werden. So stellen die Entscheidbarkeit der Theorie der elementaren Algebra über den reellen Zahlen, der Presburger Arithmetik und des Schnittproblems für reguläre Sprachen positive Ergebnisse dar, die Unentscheidbarkeit der Zahlentheorie und des Schnittproblems für kontext-freie Sprachen negative.

Untersprachen der Theorie der elementaren Algebra sind besonders interessant

Aufgrund der Art des zu repräsentierenden Hintergrundwissens, häufig physikalische Gesetze und Randbedingungen in Form von (Un)gleichungen über multivariaten Polynomen, sind Entscheidungsprozeduren für Untersprachen der Theorie der elementaren Algebra über den reellen Zahlen besonders interessant. Hier stellt sich heraus, daß Implementierungen insbesondere für lineare und quadratische Systeme von Ungleichungen zwischen Polynomen verfügbar sind.

Im Bereich der linearen Systeme sind dies u.a. Implementierungen aus dem Bereich des Constraint Logic Programming, für nichtlineare Systeme Lösungen aus dem Bereich der Computeralgebra. Beide Ansätze beruhen auf dem bereits von Tarski zum Beweis der Entscheidbarkeit der Theorie der elementaren Algebra über den reellen Zahlen benutzten Prinzip der Quantorenelimination.

Quantorenelimination erlaubt die Ermittlung zulässiger Parameterwerte

Quantorenelimination – im Bereich des Constraint Logic Programming auch Projektion oder Variablenelimination genannt – kann darüber hinaus nicht nur zur Überprüfung der Erfüllbarkeit einer endlichen Konjunktion konkreter Prädikate benutzt werden. Im Rahmen beschreibungslogischer Systeme wird die Erfüllbarkeit einer endlichen Konjunktion konkreter Prädikate für die Bestimmung der Konzept- und Objektklassifikation benötigt. Dazu reicht die Aussage, die der Erfüllbarkeitstest liefert, z.B. ob ein System von Ungleichungen eine Lösung besitzt, aus.

Normalerweise ist man aber nicht nur an der Existenz einer Lösung, sondern an den konkreten Werten der Lösung interessiert. Auch dies ist mit der Quantorenelimination möglich. Eliminiert man, im Gegensatz zur Vorgehensweise beim Erfüllbarkeitstest, nicht alle Quantoren – und damit nur einen Teil der Variablen – aus der Ausgangsformel, erhält man als Ergebnis der Quantorenelimination eine zur Ausgangsformel logisch äquivalente Formel in den verbleibenden Variablen.

Diese Formel beschreibt die resultierenden maximalen Einschränkungen an die Wertebereiche der verbleibenden Variablen. Dadurch ist es, ausreichendes Hintergrundwissen vorausgesetzt, möglich, die Menge der möglichen Werte für unbekannte Parameter drastisch einzuschränken und evtl. sogar genau zu bestimmen [Kam96a, KN97]. Die Ermittlung der Parametereinschränkungen konkreter Parameter wird so in die Modellberechnung von CTL integriert.

Browsing, Retrieval, Verhaltenssimulation, What-If-Analyse, Diagnose auf Basis von CTL möglich

Die Berechnung unbekannter Parameterwerte, zusammen mit den Repräsentations- und Inferenzmechanismen von Beschreibungslogiken erfüllen die Anforderungen an ein Unterstützungssystem für Wartungstechniker:

Die Ausdrucksmittel der Repräsentationssprache erlauben es, Rad-Ketten-Getriebe inklusive des Hintergrundwissens in Form von physikalischen Gesetzen sowie des Normal- und des Fehlverhaltens in einem geschlossenen Formalismus zu beschreiben.

Durch die Ableitung neuer Parameterwerte in Kombination mit der automatischen Klassifikation von Instanzen und Konzepten wird ein ähnlichkeitsbasiertes Retrieval mit einer exakt definierten Semantik möglich. Gleichzeitig bildet das Ergebnis der automatischen Klassifikation eine exzellente Basis zur Definition weiterer, mehr an die Verfahren traditioneller fallbasierter Systeme angelehnter Retrievalverfahren,z.B. auf Basis von Ähnlichkeitsmaßen über Indexmengen.

Damit ist es möglich, ein breites Spektrum möglicher Repräsentationen und Retrievalverfahren zu realisieren, die sich sehr stark in der Verwendung von Hintergrundwissen unterscheiden. Am unteren Ende des Spektrums steht der völlige Verzicht auf Verhaltensmodelle und physikalische Gesetze und die Verwendung numerischer Ähnlichkeitsmaße über Indexmengen. Im anderen Extremfall kann, die Repräsentation entsprechender Modelle für das Normal- und Fehlverhalten und physikalischer Gesetze vorausgesetzt, eine konsistenzbasierte Diagnose allein mit den Basisinferenzen der Beschreibungslogik realisiert werden. Es wird eine Menge gemessener Parameterwerte vorgegeben und die Menge der Verhaltensmodelle berechnet, die konsistent mit den gemessenen Werten sind.

Der durchgehende Repräsentationsformalismus der Beschreibungslogik erlaubt es, das System bzw. die Anwendung im Laufe der Zeit von einem einfachen System ohne Repräsentation von Hintergrundwissen hin zu einem komplexen System zu entwickeln. So können, aufbauend auf den gemachten Erfahrungen, sukzessive die unterschiedlichen Verhaltensmodelle identifiziert, entwickelt und der Wissensbasis hinzugefügt werden.

Durch die automatische Berechnung der Parametereinschränkungen werden weiterhin Simulationen und What-If Analysen, sowie die intelligente Suche in Katalogen möglich. Anders als bei der konsistenzbasierten Diagnose gibt man hier einige wenige Parameterwerte vor und spezifiziert die Verhaltenmodelle, die der Simulation bzw. der What-If Analyse zugrundegelegt werden sollen. Es werden dann die maximalen Einschränkungen an die abhängigen Parameterwerte berechnet, die durch die angenommenen Verhaltensmodelle induziert werden.

Im Gegensatz zu numerischen Simulationen ist es möglich, die Randbedingungen an die Parameterwerte frei im Rahmen der Ausdrucksmächtigkeit des konkreten Gegenstandsbereiches zu wählen. So können in ausdrucksstarken konkreten Gegenstandsbereichen beliebige logische Sätze über linearen bzw. quadratischen Polynomen formuliert werden.

Zusätzlich ist es leichter möglich, die durch Parameterberechnung vervollständigten Objektbeschreibungen auf andere Verfahren des ähnlichkeitsbasierten Retrievals zu übertragen. Da diese Verfahren meistens exakte Parameterwerte voraussetzen und Probleme mit fehlenden Parameterwerten haben, ist es leichter, vervollständigte Objektbeschreibungen zu übertragen.

Ein Beitrag zum Thema ≫Experience Management≪

In letzter Zeit sind Tendenzen zu beobachten, den Begriff ≫Experience Management≪ als erweiterte Interpretation des Begriffes ≫Fallbasierten Schließen≪ zu etablieren. Unterstützt wird dies z.B. durch die Fusion der beiden GI-Fachgruppen ≫Fallbasiertes Schließen≪ und ≫Knowledge Management≪ zur Fachgruppe ≫Experience Management≪.

Diese Arbeit entstand letztendlich im Spannungsfeld zwischen ≫Fallbasierten Schließen≪, ≫Knowledge Representation≪ und ≫Knowledge Management≪. Die erzielten Ergebnisse sollten daher auch vor diesem Hintergrund interessant sein.

Ausblick

Aus der Reihe der Möglichkeiten zur Weiterentwicklung der in dieser Arbeit vorgestellten Ergebnisse sollen an dieser Stelle zwei Themen hervorgehoben werden, die sich jeweils mit der Schnittstelle zu einem anderen Forschungs- bzw. Tätigkeitsgebiet befassen:

Beschreibungslogiken – Auch aus Sicht der Quantorenelimination interessant

Aber nicht nur aus Sicht der Beschreibungslogiken sind konkrete Gegenstandsbereiche mit Quantorenelimination besonders interessant. Aus Sicht der Quantorenelimination bietet die Kombination mit Beschreibungslogiken die beiden folgenden wesentlichen Vorteile:

Erstens ist es möglich, die komplexen Formeln, die sich zu Beispiel beim Aufbau der Struktur eines Fahrradantriebes ergeben, objektorientiert zu repräsentieren. Dies erleichtert den Umgang mit diesen Formeln enorm. Die Vereinfachung der Repräsentation komplexer Formeln war letztendlich ja auch der Grund für die Einführung von Beschreibungslogiken.

Zweitens realisieren die Basisinferenzen Algorithmen zur planmäßigen Konstruktion einer Folge von Formeln, die zur Simulation bzw. zur Diagnose genutzt werden kann. Dadurch wird die manuelle Entwicklung spezialisierter Algorithmen für die jeweiligen Aufgaben häufig überflüssig.

So können Systeme aus dem Bereich der Computeralgebra, die Quantoreneliminationsverfahren realisieren ebenfalls von den Ergebnissen dieser Arbeit profitieren.

Die Interaktion zwischen Beschreibungslogiken und der Quantorenelimination

sind ein Beispiel für Querbeziehungen und Überschneidungen unterschiedlichen Forschungsrichtungen und Forschungsfeldern. Ähnlich verhält es sich mit dem Bereich des fallbasierten Schliessens insgesamt. In dieser Arbeit wurde intensiv nur auf den die Querbeziehungen zwischen dem Fallbasierten Schliessen und dem Bereich der Beschreibungslogiken eingegangen. Darüberhinaus bestehen intensive Verflechtungen u.a. zu den Bereichen des Information Retrieval, der Datenbanken und des Machine Learnings. In [KLG98] werden diese Beziehungen etwas genauer beleuchtet.

Integration über WebServices

Betrachtet man sich die in dieser Arbeit vorgestellte Architektur, so zeichnet sich CTL sowohl durch eine einfache, "schmale" Schnittstelle zu den Entscheidungsprozeduren der konkreten Gegenstandsbereiche, als auch durch eine einfache Syntax zur Repräsentation und Abfrage des Wissens selbst aus.

Nimmt man den Fakt hinzu, dass zur Berechnung der Inferenzen sowohl in den konkreten Gegenstandsbereichen als auch im abstrakten Gegenstandsbereich nicht unerhebliche Ressourcen benötigt werden, so erscheint eine Migration auf eine XML-Syntax und die Bereitstellung der Inferenzen als Webservices äußerst sinnvoll.

Nicht nur könnten so externe Entscheidungsprozeduren für konkrete Gegenstandsbereiche auf standardisierte Art und Weise angebunden werden, sondern auch das Gesamtsystem könnte auf transparente Art und Weise genutzt werden.

Insbesondere entsteht durch die Bereitstellung der Verfahren als WebServices die Möglichkeit die entwickelten Verfahren in die unterschiedlichen Systeme zum Knowledge Management zu integrieren. Zumindest die kommerziellen Systeme beschränken sich auf die Bereitstellung von Browsing und Navigationsmechanismen (z.B. Hyperbolic Trees) über einfachen Taxonomien und Thesauri (z.B. WordNet).

Anhänge

A Die Object Modeling Technique

Mit dem Boom objektorientierter Techniken entstanden in den letzten Jahren eine Vielzahl an Verfahren zum objektorientierten Softwareentwurf.

Nach einem gewissen kreativen Wildwuchs, wie er in allen stark expandierenden Bereichen zu beobachten ist, ist mittlerweile eine Konsolidierung festzustellen. Mitte der 90er Jahre hatten sich eine kleine Anzahl von Verfahren wie etwa die OMT von Rumbaugh u.a. [RBP+91], die Objektorientierte Analyse und Objektorientiertes Design (OOA/OOD) von Coad und Yourdon [CY91a, CY91b] und die Object Oriented Analysis and Design (OOAD) von Booch [Boo94] etabliert, die dann zur Unified Modeling Language (UML) [Cor97] integriert wurden.

Zur Beschreibung des Leitbeispiels werden im wesentlichen die Elemente der Analysephase des objektorientierten Entwurfs, und dort vor allem die Ausdrucksmittel für die statische Modellierung, das *Objektmodell* benötigt. Das Objektmodell ist zugleich der Aspekt der verschiedenen Entwurfsmethoden, in dem sich diese am wenigsten unterscheiden. Da zu Beginn dieser Arbeit die UML noch nicht vollständig spezifiziert war, mußte eine Festlegung auf eine der zu diesem Zeitpunkt gängigen Objektmodelle erfolgen. Die Entscheidung fiel zugunsten der OMT.

Um dem Leser die Einarbeitung in die OMT zu erleichtern, soll in diesem Anhang eine kurze Übersicht über die für die Zwecke der Arbeit relevanten Teile des OMT Objektmodelles gegeben werden. Für eine detaillierte Einführung in die OMT wird der Leser auf [RBP+91] und andere Literatur zur objektorientierten Entwicklungsmethodik, etwa [Sch94b] verwiesen.

Objekt, Klasse, Instanz

Zentrale Elemente jeder objektorientierten Repräsentation sind die Begriffe *Objekt*, *Klasse* und *Instanz*.

Definition A.1 (Objekte, Klassen und Instanzen)
Ein **Objekt** ist ein Konzept, eine Abstraktion oder ein Gegenstand mit klaren Abgrenzungen und einer präzisen Bedeutung für das real anstehende Problem. Objekte dienen zwei Zielen: Sie erleichtern es, die reale Welt zu verstehen, und sie sind ein praktikabler Ausgangspunkt für die Implementierung auf einem Compu-

ter. Die Dekomposition eines Problems in Objekte ist Ermessenssache und hängt von der Art des Problems ab. Es gibt niemals nur eine korrekte Repräsentation.

Das Wort Objekt wird in der Literatur oft ohne eindeutige Definition verwendet. Manchmal meint ein Objekt eine einzelne Sache, in anderen Fällen bezieht es sich auf eine Gruppe ähnlicher Dinge. Normalerweise erklären sich etwaige Unklarheiten durch den Kontext. In unklaren Situationen wird der Begriff **Objektinstanz** verwendet, wenn präzise auf genau eine Sache verwiesen werden soll.

Eine **Objektklasse** beschreibt eine Gruppe von Objekten mit ähnlichen Eigenschaften (Attributen), gemeinsamen Verhalten (Operationen), gemeinsamen Relationen zu anderen Objekten und einer gemeinsamen Semantik.

Bemerkung A.1
Häufig wird statt Objektklasse die Abkürzung Klasse und statt Objektinstanz die Abkürzung Instanz verwendet.

Objekte einer Klasse besitzen die gleichen Attribute und Verhaltensmuster. Die Individualität der meisten Objekte ergibt sich aus unterschiedlichen Attributwerten und Relationen zu anderen Objekten. Aber auch Objekte mit identischen Attributwerten und Relationen sind möglich.

Diagramme

Als Formalismus zur Notation der Zusammenhänge zwischen den verschiedenen Objekten werden Diagramme benutzt.

Definition A.2 (Objekt-, Klassen- und Instanzdiagramme)
Objektdiagramme stellen eine formale graphische Notation für die Modellierung von Objekten, Klassen und ihren Relationen zueinander bereit.

Ein **Klassendiagramm** ist ein Schema, Muster oder Template (Schablone) zur Beschreibung vieler möglicher Dateninstanzen. Ein Klassendiagramm beschreibt Objektklassen.

Ein **Instanzendiagramm** beschreibt, wie eine bestimmte Menge von Objekten zueinander in Relation stehen. Ein Instanzendiagramm beschreibt Objektinstanzen. Instanzendiagramme sind besonders nützlich, um Testfälle (vor allem Szenarios) zu dokumentieren und Beispiele zu diskutieren.

Notation A.1
Klassen werden in der OMT-Notation als Rechtecke mit dem Klassennamen in fetter Schrift beschrieben. Instanzen werden als abgerundete Rechtecke beschrieben (siehe Abb. A.1). Der Klassenname steht hier fett in runden Klammern oben in der Instanzbox.

Klassen- und Instanzen können natürlich auch gemeinsam in einem Diagramm vorkommen.

Attribute

Zur detaillierten Beschreibung von Objekten werden *Attribute* benutzt.

Abbildung A.1.

Die OMT Notation für Klassen und Instanzen

Definition A.3 (Attribut)

Attribute sind Datenwerte, die die Objekte der Klasse besitzen. Im Gegensatz zu Objekten besitzen reine Datenwerte keine Identität. Attribute werden über **Attributnamen** identifiziert. Diese sind für jede Klasse eindeutig.

Bemerkung A.2

Der Begriff des Datenwertes ist in der obigen Definition bewußt schwammig gehalten. Im allgemeinen werden unter Datenwerten Werte der Basistypen einer Programmiersprache bzw. einer Datenbank verstanden. Typische Datenwerte sind Zahlen, Zeichenketten (Strings) etc.

So besitzen Objekte einer Klasse *Person* z.B. die Attribute *Name*, *Vorname* und *Alter*.

Bemerkung A.3

Im Sinne der ≫reinen≪ objektorientierten Lehre und der von ihr vertretenen Auffassung, ≫everything is an object≪, scheint die Unterscheidung zwischen Objekten und Datenwerten ein Widerspruch zu sein.

Zu unterscheiden ist hier allerdings zwischen der objektorientierten Modellierung eines Gegenstandsbereiches, bei der durch Objekte im wesentlichen durch eine physikalische Entsprechung in der realen Welt charakterisiert sind, und der Umsetzung dieser Modellierung mit einer objektorientierten Programmiersprache. So ist auch der Begriff der Identität nicht der einer Identitätsrelation der Programmiersprache.

Die Umsetzung einer objektorientierten Modellierung ist natürlich auch mit einer konventionellen Programmiersprache möglich, wenn auch aufwendiger.

Notation A.2

Attribute werden in einem zweiten Teil der Klassenbox aufgeführt. Zwischen den beiden Teilen wird eine Linie gezogen. Hinter jedem Attributnamen können optionale Details stehen, z.B. ein Typ. Vor dem Typ steht ein Doppelpunkt.

In Objektboxen wird zur deutlicheren Unterscheidung von Klassenboxen auf die Trennlinie verzichtet.

Assoziationen

Attribute beschreiben Beziehungen zwischen Objekten und Datenwerten. Assoziationen beschreiben Beziehungen bzw. Relationen zwischen Objekten.

Definition A.4 (Verknüpfung, Assoziation)

Eine **Verknüpfung** ist eine physikalische oder konzeptuelle Verbindung zwischen zwei Objektinstanzen. Mathematisch ist eine Verknüpfung als Tupel definiert.

Abbildung A.2.

Die OMT Notation für Klassen und Instanzen mit Attributen

Eine **Assoziation** beschreibt eine Gruppe von Verknüpfungen mit einer gemeinsamen Struktur und Semantik. Mathematisch gesehen ist daher eine Assoziation eine Relation [1].

Bemerkung A.4

Eine Assoziation beschreibt eine Menge potentieller Verknüpfungen auf die gleiche Weise, in der eine Klasse eine Menge potentieller Objekte beschreibt.

Assoziationen sind inhärent bidirektional. Der **Assoziationsname** drückt in der Regel eine bestimmte Richtung aus, die binäre Assoziation kann jedoch in jeder Richtung durchlaufen werden.

Assoziationen können binär, ternär oder höherer Ordnung sein. In der Praxis sind fast alle Assoziationen binär oder qualifiziert (eine Sonderform einer ternären Assoziation innerhalb der OMT, die im Kontext der Arbeit jedoch nicht relevant ist).

Notation A.3

Eine Assoziation wird durch eine Linie zwischen den Klassen dargestellt, eine Verknüpfung durch eine Linie zwischen den Objekten (siehe Abb. A.3). Der Assoziationsname kann weggelassen werden, wenn ein Klassenpaar nur eine einzige Assoziation besitzt, deren Bedeutung offensichtlich ist. Es empfiehlt sich, Klassen möglichst so anzuordnen, daß Assoziationen von links nach rechts gelesen werden.

Abbildung A.3.

Die OMT Notation für Assoziationen

Definition A.5 (Rollen)

Eine **Rolle** ist ein Ende einer Assoziation. Die beiden Rollen einer binären Assoziation können jeweils einen Rollennamen haben. Ein **Rollenname** ist ein Name, der ein Ende einer Assoziation eindeutig identifiziert.

Notation A.4

Ein Rollenname steht an der Assoziationslinie neben der Klasse, die die Rolle spielt.

1. Natürlich können diese Relationen auch funktional sein.

Bemerkung A.5
Die Verwendung von Rollennamen ist optional, es ist aber oft einfacher und weniger verwirrend, Rollennamen anstelle von, oder zusätzlich zu Assoziationsnamen zu verwenden.

Für Assoziationen zwischen zwei Objekten der gleichen Klasse sind Rollennamen zwingend notwendig.

Alle Rollennamen am anderen Ende der einer Klasse zugeordneten Assoziationen müssen eindeutig sein.

Theoretisch können beliebig viele Instanzen einer Klasse mit einer Instanz einer assoziierten Klasse verknüpft sein. Die *Multiplizität* dient dazu die Zahl der verknüpften Objekte zu beschränken.

Definition A.6 (Multiplizität)
Die **Multiplizität** begrenzt die Zahl verknüpfter Objekte. Der Multiplizitätswert umfaßt im allgemeinen ein Intervall, er kann aber auch einer Menge unzusammenhängender Intervalle bestehen.

Notation A.5
In Objektdiagrammen wird die Multiziplität durch bestimmte Symbole an den Enden der Assoziationslinien angezeigt (siehe Abb. A.4). Im allgemeinsten Fall kann die Multiziplität durch eine Zahl oder eine Menge von Intervallen angegeben sein. Dabei bezeichnet n+ das halbseitig offene Intervall $[n, \infty)$ und n-m das Intervall $[n, m]$.

Bestimmte, häufig vorkommende Multiziplitäten werden durch spezielle Linienenden gekennzeichnet. Ein schwarzer Punkt ist das Symbol für das Intervall $[0, \infty)$, ein Kreis das Symbol für $[0, 1]$, d.h. eine optionale Verknüpfung. Ein Linienende ohne Symbole oder Beschriftung steht für die Multiziplität 1, also genau eins.

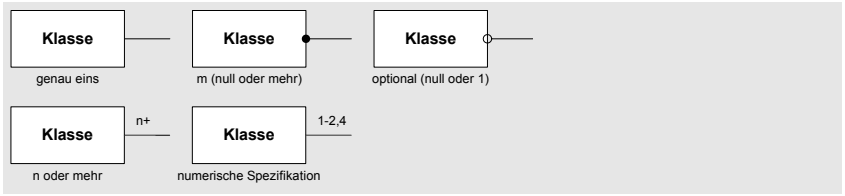


Abbildung A.4.
Die OMT Notation für Multiziplitäten

Aggregationen

Aggregationen sind eine häufig vorkommende Form der Assoziation mit einer speziellen Semantik. Sie bezeichnet die sogenannte »Teil-Ganzes-« oder »ist-Teil-von« Relation.

Definition A.7 (Aggregation, Komponenten, Komponentengruppen)
Aggregation bezeichnet die »Teil-Ganzes« oder »ist-Teil-von-Relation«, in der

Objekte, die die **Komponenten** einer Sache repräsentieren, mit einem Objekt assoziiert **Komponentengruppe** repräsentiert.

Bemerkung A.6

Eine Aggregationsrelation ist die Beziehung zwischen einer **Komponentengruppenklasse** und *einer* Komponentenklasse. Eine Komponentengruppe mit vielen verschiedenen Komponenten hat dementsprechend viele Aggregationsrelationen.

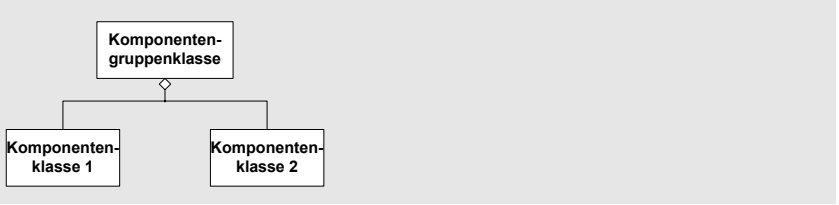
Damit die Multiplizität jeder Komponente innerhalb der Komponentengruppe spezifiziert werden kann, wird jede einzelne Paarrelation als Aggregation modelliert [2].

Notation A.6

Eine Aggregation wird wie eine Assoziation dargestellt, zusätzlich kennzeichnet eine kleine Raute das Komponentengruppenende der Relation. Wenn eine Menge von Komponenten dargestellt werden sollen, die alle der gleichen Komponentengruppe angehören, können die Aggregationen im Diagramm zu einem Aggregationsbaum verbunden werden (siehe Abb. A.5).

Abbildung A.5.

Die OMT Notation für Aggregationen



Generalisierungen

Generalisierungen beschreiben Relationen zwischen Klassen und sind ein mächtiger Abstraktionsmechanismus, um Gemeinsamkeiten von Klassen zu teilen und gleichzeitig ihre Unterschiede zu erhalten.

Definition A.8 (Generalisierung, Spezialisierung)

Eine **Generalisierung** ist die Relation zwischen einer Klasse und einer oder mehreren verfeinerten Versionen davon. Die Klasse, die verfeinert wird, heißt **Oberklasse**, jede verfeinerte Version **Unterklasse**.

2. Dies ist die Entscheidung innerhalb der OMT. In anderen Repräsentationsformalismen z.B. in R [CNJ+91] ist die komplexe Spezifikation von Zerlegungsstrukturen über *eine* Relation möglich.

   Eine weitere Möglichkeit zur Repräsentation der Tatsache, daß es sich eigentlich um verschiedene Relationen der gleichen Aggregation handelt, wäre die Verwendung von Relationenhierarchien wie sie etwa in Beschreibungslogiken Verwendung finden.

   Da aber die Semantik der Relationen innerhalb der OMT bewußt – nicht zuletzt um die universelle Einsetzbarkeit nicht zu gefährden – sehr vage formuliert ist und das intuitive Verständnis der Zusammenhänge gewährleistet ist, ist der in der OMT gewählte Kompromiß die beste Lösung.

Jede Unterklasse **erbt** die Merkmale (Attribute) ihrer Oberklasse. Attribute, die für alle Unterklassen gelten, werden daher der Oberklasse zugewiesen und von den Unterklassen gemeinsam genutzt.

Die Generalisierung ist transitiv, **Vorfahre** und **Nachkomme** beschreiben die Generalisierungsbeziehung von Klassen über mehrere Ebenen hinweg.

Eine Klasse erbt nicht nur die Merkmale aller Vorgänger, sondern fügt diesen auch seine individuellen Attribute hinzu.

Bemerkung A.7

Die Unterklassen einer Generalisierung können disjunkt sein oder nicht, nicht disjunkte Generalisierungen heißen überlappend.

Notation A.7

Die Notation für die Generalisierung ist ein Dreieck, das eine Oberklasse mit ihren Unterklassen verbindet. Die Oberklasse wird durch die Linie mit der Spitze des Dreiecks verbunden. Die Unterklassen werden durch Linien mit einem horizontalen Balken verbunden, der über die Grundlinie des Dreiecks verläuft. Das Dreieck kann aus Gründen der Bequemlichkeit umgedreht werden und Unterklassen können sowohl oberhalb als auch unterhalb des Balkens angefügt werden. Wenn möglich sollten die Oberklasse jedoch oben und die Unterklassen unten stehen.

Ein nicht ausgefülltes Dreieck beschreibt eine disjunkte Generalisierung, ein gefülltes Dreieck eine überlappende (siehe Abb. A.6).

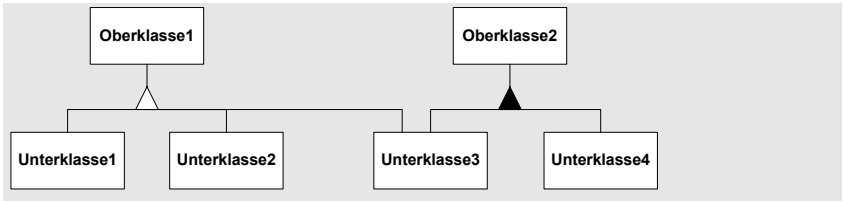


Abbildung A.6.

Die OMT Notation für Generalisierungen

Definition A.9 (Mehrfachvererbung, Vereinigungsklassen)

Eine Klasse kann Attribute auch von mehreren Oberklassen erben (sog. **Mehrfachvererbung**). Eine solche Klasse heißt **Vereinigungsklasse**.

Bemerkung A.8

Normalerweise wird innerhalb der OMT zwischen der konzeptuellen Relation »Generalisierung« und dem Mechanismus »Vererbung« unterschieden. Da aber der Begriff Mehrfachvererbung sich auch für die Beschreibung der oben beschriebenen konzeptuellen Relation eingebürgert hat, wurde auf die Verwendung des Begriffes »Mehrfachgeneralisierung« für diesen Begriff verzichtet.

Instantiierungen

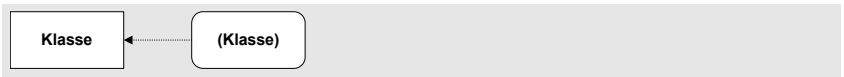
Instantiierungen beschreiben Beziehungen zwischen Instanzen und Klassen.

Definition A.10 (Instantiierung)
Die **Instantiierung** verbindet eine Klasse mit ihren Instanzen.

Notation A.8
Innerhalb der OMT ist die Instantiierung eindeutig und in der Notation durch die Angabe des Klassennamens in der Instanznotation ersichtlich. Soll die Beziehung jedoch explizit gemacht werden, so wird die Instantiierung durch einen gepunkteten Pfeil von der Instanz zur Klasse beschrieben (siehe Abb. A.7).

Abbildung A.7.

Die OMT Notation für Instantiierungen



Einschränkungen

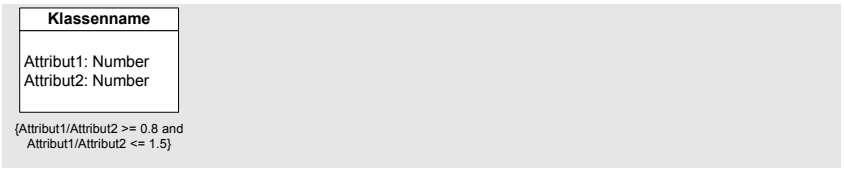
Einschränkungen sind funktionale Beziehungen zwischen Elementen des Objektmodelles.

Definition A.11
Eine **Einschränkung** begrenzt die Werte, die **Entitäten** des Objektmodells annehmen können. Klassen, Objekte, Attribute, Verknüpfungen und Assoziationen sind Entitäten eines Objektmodelles.

Notation A.9
Einschränkungen werden in geschweiften Klammern angegeben und stehen neben der eingeschränkten Entität. Eine gepunktete Entität verbindet mehrere eingeschränkte Entitäten.

Abbildung A.8.

Die OMT Notation für Einschränkungen



Bemerkung A.9
Innerhalb des Objektmodelles der OMT werden somit keinerlei Aussagen darüber gemacht, was zulässige Einschränkungen sind, und wie diese behandelt werden. Die OMT stellt praktisch einen leeren Container zur Verfügung, der von der jeweiligen Anwendung nach Belieben gefüllt werden kann und muß. Die zulässige Syntax und die Semantik der Constraints muß also von der jeweiligen Anwendung bzw. dem jeweiligen Anwendungsbereich definiert werden. In [RBP+91] wird diese Tatsache folgendermaßen behandelt:

≫Wir bevorzugen es, Einschränkungen deklarativ auszudrücken. Normalerweise müssen Einschränkungen in eine prozedurale Form umgewandelt werden, bevor sie in einer Programmiersprache ausgedrückt werden können. Idealerweise sollte diese Umwandlung automatisch erfolgen, dies läßt sich jedoch oft nur schwierig oder überhaupt nicht realisieren.
(...) Es wird immer Einschränkungen geben, die in natürlicher Sprache formuliert werden müssen.≪

B Grundbegriffe der Modelltheorie

Die Modelltheorie ist ein auf die Arbeiten von Tarski zurück gehendes Gebiet der Mathematik zur Untersuchung formaler Sprachen. Die dort erstmals eingeführte Semantik solcher Sprachen fußt auf der Erfüllbarkeit von Formeln dieser Sprachen in bestimmten Strukturen, den sogenannten Modellen. Sie heißt daher modelltheoretische Semantik, wird aber oft aus Respekt gegenüber ihrem Erfinder auch Tarski-Semantik genannt.

R charakterisiert das Gebiet der Modelltheorie in [Ric81] als:

»Modelltheorie = Logik + universelle Algebra«

In der folgenden Einführung in die elementaren Begriffe der Modelltheorie sind daher die zentralen Begriffe der Logik (Prädikatenlogik erster Stufe, Interpretation, ...) sowie der universellen Algebra (Algebra, Struktur, ...) wiederzufinden.

Die Einführung beschränkt sich auf die Begriffe, die im Rahmen der Arbeit benötigt werden. Dies bedeutet aber auch, daß einige Begriffe wie Polynom und Ungleichungssystem, die nicht der Modelltheorie entstammen, an dieser Stelle definiert werden.

Für eine vertiefte Einführung in die Modelltheorie, Logik und universelle Algebra empfiehlt sich die Lektüre von [RSS81, BS81, Ric81, Ihr93, Ric78, Sch95].

Prädikatenlogik

Struktur, Typ

Definition B.1 (Operation, Relation)
Sei $n \in \mathbb{N}^1$ und A eine Menge. Dann heißt die Abbildung $f : A^n \mapsto A$ **(n-stellige) Operation**. Die Teilmenge $r \subseteq A^n$ heißt **n-stellige Relation**.

Bemerkung B.1
(i) n heißt auch **Stelligkeit** von f (bzw. r).

1. Im folgenden soll unter $\mathbb{N}$ die Menge der natürlichen Zahlen inklusive der 0 verstanden werden.

(ii) Ist $n = 1, 2, 3$, so spricht man resp. von **unären**, **binären** und **ternären** Operationen (Relationen). Eine einstellige Relation heißt auch **Prädikat**. Relation!unäre] Relation!binäre] Relation!ternäre]

(iii) Ist $n = 0$, so gilt nach Definition $A^0 = \emptyset$. Daher zeichnet die Abbildung $f : A^0 \mapsto A$ genau ein Element $a \in A$ aus, und man identifiziert die 0-stellige Operation f mit a. Eine solche Abbildung heißt auch **Konstante**.

Definition B.2 (Struktur)
Eine **(mathematische) Struktur** ist ein Tripel $\mathbf{A} = \langle A, F, R\rangle$.

(i) Dabei ist $\mathbf{A}$ eine nichtleere Menge, das **Universum** (, ,),

(ii) $F = \langle f_i \mid i \in I_{\mathbf{A}}\rangle$ eine mit $I_{\mathbf{A}}$ indizierte Familie von n_i-stelligen Operationen, und

(iii) $R = \langle r_j \mid j \in J_{\mathbf{A}}\rangle$ eine mit $J_{\mathbf{A}}$ indizierte Familie von m_j-stelligen Relationen.

Ist $\mathsf{F} = \emptyset$, so heißt $\mathbf{A} = \langle A, \emptyset, R\rangle$ **Relationalstruktur** und man schreibt auch $\mathbf{A} = \langle A, R\rangle$.

Ist $\mathsf{R} = \emptyset$, so heißt $\mathbf{A} = \langle A, \emptyset, R\rangle$ **(allgemeine) Algebra** und man schreibt auch $\mathbf{A} = \langle A, F\rangle$.

$\sigma = \langle\langle n_i \mid i \in I_{\mathbf{A}}\rangle, \langle m_j \mid j \in J_{\mathbf{A}}\rangle\rangle$ heißt **Signatur (der Struktur A)**.

Bemerkung B.2
(i) Solange keine Verwechslungsgefahr besteht, schreibt man im allgemeinen f bzw. r für f_i und r_j.

(ii) Sind F und R endlich, so schreibt man für $\mathbf{A}$ auch $\mathbf{A} = \langle A, f_1, \ldots, f_n, r_1, \ldots, r_m\rangle$.

Terme, Termalgebra

Beispiel B.1 zeigt, daß verschiedene Trägermengen ähnliche Operationen tragen können. Daher verallgemeinert man den Begriff der Struktur, indem man von der Trägermenge abstrahiert.

Definition B.3 (Typ)
Ein **Typ (von Strukturen) T** ist ein Tripel $\mathsf{F} = \langle \mathsf{F}, \mathsf{R}, \sigma\rangle$. Dabei ist

(i) F eine Menge von **Funktionssymbolen**,

(ii) R eine Menge von **Relationssymbolen** (Prädikatssymbolen), und

(iii) $\sigma = \langle\langle n_{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}\rangle, \langle m_{\mathsf{r}} \mid \mathsf{r} \in \mathsf{R}\rangle\rangle$; für $n, m \in \mathbb{N}$

Bemerkung B.3

(i) σ ordnet jedem $\mathsf{f} \in \mathbf{F}$ ($\mathsf{r} \in \mathbf{R}$) eine **Stelligkeit** zu, man schreibt auch $\sigma(\mathsf{f}) = n_{\mathsf{f}}$ (bzw. $\sigma(\mathsf{r}) = m_{\mathsf{r}}$).

(ii) $\mathbf{L} := \langle \mathbf{F}, \mathbf{R} \rangle$ heißt auch **Sprache (erster Stufe)** (des Typs), σ heißt **Signatur (des Typs)**.

(iii) Analog wird der **Typ von Algebren** $\langle \mathbf{F}, \sigma \rangle$ und der **Typ von Relationalstrukturen** $\langle \mathbf{R}, \sigma \rangle$ definiert.

(iv) Eine Struktur $\mathbf{A}$ ist vom Typ $\mathbf{T}$, falls es zu jedem Symbol $\mathsf{f} \in \mathbf{F}$ eine $\sigma(\mathsf{f})$-stellige Operation f_i^A und und zu jedem $\mathsf{r} \in \mathbf{R}$ eine $\sigma(\mathsf{r})$-stellige Relation r_j^A gibt.

(v) Ist $\mathbf{A}$ eine Struktur des Typs $\mathbf{T}$, dann nennt man f_{i}^A **Interpretation** von f und r_j^A Interpretation von r.

(vi) Besteht keine Verwechslungsgefahr, so schreibt man auch f_A für f_i^A und r_A für r_i^A.

Definition B.4 (Terme, Termalgebra)
Sei $\langle \mathbf{F}, \sigma \rangle$ ein Typ von Algebren und $X = \{x_1, \dots\}$ eine abzählbare Menge von **Variablen** (Individuenvariablen ($\mathbf{F} \cap X = \emptyset$). Dann ist die Menge $T(X)$ definiert durch:

(i) Für alle $x \in X$ ist $\langle x \rangle \in T(X)$;

(ii) Für $\mathsf{f} \in \mathbf{F}$ mit $\sigma(\mathsf{f}) = n$ und $t_1, \dots, t_n \in T(X)$ ist auch $\langle \mathsf{f}, t_1, \dots, t_n \rangle \in T(x)$.

Die Elemente von $T(X)$ heißen **Terme** des Typs $\langle \mathbf{F}, \sigma \rangle$. Terme, die keine Variablen enthalten, heißen **Grundterme**.

Die **Termalgebra** des Typs $\langle \mathbf{F}, \sigma \rangle$ ist definiert als Algebra
$\mathbf{T(X)} = \langle T(X), F_{T(X)} \rangle$, wobei die fundamentalen Operationen $f_{T(X)}$ folgendermaßen definiert sind:

$$f_{T(X)}(t_1, \dots, t_n) := \langle f, t_1, \dots, t_n \rangle$$

Bemerkung B.4

(i) Für $t \in T(X)$ schreibt man auch $t(x_1, \dots, x_n)$, um anzudeuten, daß alle bei der Bildung von t verwendeten Variablen aus $\{x_1, \dots, x_n\}$ stammen.

(ii) Im allgemeinen identifiziert man die Tupel $\langle x_i \rangle$ mit x_i und man schreibt statt $\langle f, t_1, \dots, t_n \rangle$ auch $f(t_1, \dots, t_n)$, für binäre Operationen $\circ$ auch $x_i \circ x_j$ statt $\circ(x_i, x_j)$.

Definition B.5 (Variablenbelegung, Auswertung von Termen)
Sei $\mathbf{A}$ eine Algebra des Typs $\mathbf{T}$ und $\rho : X \mapsto A$ eine Operation. Dann heißt ρ **(Variablen)belegung**.

ρ läßt sich induktiv eindeutig auf die Termalgebra T_X erweitern: $\lfloor \quad \rfloor_\rho : T_X \mapsto A$ ist dann definiert durch:

(i) $\lfloor x \rfloor_\rho := \rho(x)$ für $x \in X$,

(ii) $\lfloor \mathsf{f}(t_1, \ldots, t_n) \rfloor_\rho := f_A(\lfloor t_1 \rfloor_\rho, \ldots, \lfloor t_n \rfloor_\rho)$ für $\mathsf{f} \in \mathbf{F}$.

Bemerkung B.5

Seien ρ und τ zwei Belegungen, dann bedeutet $\rho =_{x_k} \tau$, daß $\rho(x_j) = \tau(x_j)$ für alle $j \neq k \in X$.

Prädikatenlogik

Auf Basis dieser Begriffe kann nun die Prädikatenlogik erster Stufe definiert werden:

Definition B.6 (Sprache der Prädikatenlogik erster Stufe)
Sei $\mathbf{T} := \langle \mathbf{F}, \mathbf{R}, \sigma \rangle$ ein Typ einer Struktur. Ferner seien:

(i) $T(X)$ die Termalgebra der Sprache $\mathbf{F}$,

(ii) die Menge der **atomaren Formeln** definiert durch:

$$At := \{\langle \mathsf{r}, t_1, \ldots, t_{\sigma(\mathsf{r})} \rangle \mid \mathsf{r} \in \mathbf{R}, t_1, \ldots, t_{\sigma(\mathsf{r})} \in T(X)\},$$

(iii) die **Formelalgebra** die Termalgebra über dem Universum At und der Sprache

$$\mathbf{Log} := \langle \wedge, \vee, \neg, \exists x_k, \forall x_k \mid x_k \in X \rangle.$$

Dann ist die **Sprache der Prädikatenlogik erster Stufe** die Sprache der gemäß (i) – (iii) konstruierten Formelalgebra.

Bemerkung B.6

(i) Die Elemente der Formelalgebra heißen auch **Formeln der Prädikatenlogik erster Stufe**.

(ii) Abkürzend für Prädikatenlogik erster Stufe schreibt man auch $\mathcal{PL}_{\mathbf{T}}$.

(iii) Die Elemente von $\mathbf{Log}$ heißen auch **logische Symbole**, $\wedge, \vee, \neg$ **logische Operatoren** und $\exists$ bzw. $\forall$ **Quantoren** ($\exists$ **Existenzquantor**, $\forall$ **Allquantor**).

(iv) Es gilt $\mathsf{Log} \cap (\mathsf{F} \cup \mathsf{R}) = \emptyset$.

Definition B.7

(i) Die Menge der in einem Term t **vorkommenden Variablen** $\mathrm{Var}(t)$ wird durch

$$\mathrm{Var}(x_k) = \{x_k\}$$
$$\mathrm{Var}(f(t_1, \ldots, t_n)) = \mathrm{Var}(t_1) \cup \ldots \cup \mathrm{Var}(t_n)$$

definiert.

(ii) Die Menge der in einer Formel φ **vorkommenden Variablen** $\mathrm{Var}(\varphi)$ ist definiert durch:

$$\mathrm{Var}(\mathsf{r}(t_1, \ldots, t_n)) = \mathrm{Var}(t_1) \cup \ldots \cup \mathrm{Var}(t_n)$$
$$\mathrm{Var}(\varphi \wedge \psi) = \mathrm{Var}(\varphi \wedge \psi) = \mathrm{Var}(\varphi) \cup \mathrm{Var}(\psi)$$
$$\mathrm{Var}(\neg\varphi) = \mathrm{Var}(\varphi)$$
$$\mathrm{Var}(\exists x\varphi) = \mathrm{Var}(\forall x\varphi) = \mathrm{Var}(\varphi) \cup \{x\}.$$

Definition B.8 (Freie und gebundene Variablen, Satz)

Sei φ eine Formel.

(i) Die Menge der in φ **frei vorkommenden Variablen** $\mathrm{fr}(\varphi)$ wird definiert durch:

$$\mathrm{fr}(\varphi) = \mathrm{Var}(\varphi) \text{ für } \varphi \text{ atomar}$$
$$\mathrm{fr}(\varphi \wedge \psi) = \mathrm{fr}(\varphi \wedge \psi) = \mathrm{fr}(\varphi) \cup \mathrm{fr}(\psi)$$
$$\mathrm{fr}(\neg\varphi) = \mathrm{fr}(\varphi)$$
$$\mathrm{fr}(\exists x\varphi) = \mathrm{fr}(\forall x\varphi) = \mathrm{fr}(\varphi) \setminus \{x\}.$$

(ii) Die Menge der in φ **gebunden vorkommenden Variablen** $\mathrm{bd}(\varphi)$ wird definiert durch:

$$\mathrm{bd}(\varphi) = \emptyset \text{ für } \varphi \text{ atomar}$$
$$\mathrm{bd}(\varphi \wedge \psi) = \mathrm{bd}(\varphi \wedge \psi) = \mathrm{bd}(\varphi) \cup \mathrm{bd}(\psi)$$
$$\mathrm{bd}(\neg\varphi) = \mathrm{bd}(\varphi)$$
$$\mathrm{bd}(\exists x\varphi) = \mathrm{bd}(\forall x\varphi) = \mathrm{bd}(\varphi) \cup \{x\}.$$

(iii) φ heißt **offene Formel** falls $\mathrm{bd}(\varphi) = \emptyset$, φ heißt **Satz** falls $\mathrm{fr}(\varphi) = \emptyset$.

Definition B.9 (Normalformen)

Eine Formel heißt **pränex** (resp. in pränexer Normalform), wenn sie die Gestalt

$$Q_1 x_{i_1} \ldots Q_n x_{i_n} \psi; \quad Q_j \in \{\forall, \exists\} \text{ für } 1 \leq j \leq n$$

hat und ψ eine offene Formel ist.

Modell, Theorie, Entscheidbarkeit einer Theorie

Definition B.10 (Erfüllbarkeit einer Formel)
Sei $\mathbf{A}$ eine Relationalstruktur des Typs $\mathbf{T}$ und ρ eine Belegung in $\mathbf{A}$. ρ erfüllt eine Formel $\varphi \in \mathcal{PL}_{\mathbf{T}}$ genau dann, wenn gilt:

(i) ρ erfüllt die atomare Formel $\langle \mathsf{r}, t_1, \ldots, t_n \rangle$ genau dann, wenn $r(\lfloor t_1 \rfloor_\rho, \ldots, \lfloor t_n \rfloor_\rho)$ gilt, d.h. $\{\lfloor t_1 \rfloor_\rho, \ldots, \lfloor t_n \rfloor_\rho\} \in r_A$.

(ii) hat φ die Form $\neg\varrho$, so erfüllt ρ φ genau dann, wenn ρ ϱ nicht erfüllt.

(iii) hat φ die Form $\varrho \wedge \chi$, so erfüllt ρ φ genau dann, wenn ρ ϱ und χ erfüllt.

(iv) hat φ die Form $\varrho \vee \chi$, so erfüllt ρ φ genau dann, wenn ρ ϱ oder χ erfüllt.

(v) hat φ die Form $\exists x_k \varrho$, so erfüllt ρ φ genau dann, wenn es eine Belegung τ mit $\rho =_{x_k} \tau$ gibt, so daß τ ϱ erfüllt.

(vi) hat φ die Form $\forall x_k \varrho$, so erfüllt $\rho\varphi$ genau dann, wenn für jede Belegung τ mit $\rho =_{x_k} \tau$ gilt, daß τ ϱ erfüllt.

Definition B.11 (Wahrheit einer Formel, Modell)
Eine Formel φ heißt **wahr (in A)** genau dann, wenn φ von jeder Belegung ρ in $\mathbf{A}$ erfüllt wird.

$\mathbf{A}$ heißt dann auch **Modell (für φ)** und man schreibt $\mathbf{A} \models \varphi$.

Bemerkung B.7
Ist eine Formel wahr in $\mathbf{A}$, so sagt man auch, daß sie in $\mathbf{A}$ gilt (resp. gültig ist).

Definition B.12 (Theorie, Sätze einer Theorie, Entscheidbarkeit einer Theorie)
Sei $\mathbf{A}$ eine Struktur. Dann ist die $\mathbf{A}$ zugeordnete **Theorie (erster Ordnung)** $Th(A)$, die Menge aller Formeln die unter $\mathbf{A}$ gelten, d.h. $Th(A) = \{\varphi | A \models \varphi\}$.

Bemerkung B.8
Besteht keine Verwechslungsgefahr, so schreibt man auch $\langle\langle A, f_1, \ldots, f_n, r_1, \ldots, r_m \rangle\rangle$ für $Th(A)$.

Definition B.13 (Entscheidbarkeit einer Theorie)
Sei $\mathbf{A}$ eine Struktur des Typs $\mathbf{T}$. Die Theorie $Th(A)$ heißt **entscheidbar** genau dann, wenn ein Verfahren existiert, daß für jede Formel $\varphi in \mathcal{PL}_{\mathbf{T}}$ in einer endlichen Anzahl von Schritten entscheidet, ob diese Formel wahr ist.

Polynome und Ungleichungssysteme

Gruppe, Ring, Ring mit 1, Körper

Polynome sind über bestimmten Strukturen definiert, daher sollen diese zunächst definiert werden:

Definition B.14 (Gruppe, Halbgruppe)
Eine **Gruppe** ist eine Algebra $\mathbf{G} = \langle G, \cdot, ^{-1}, 1\rangle$ der Signatur $\langle 2, 1, 0\rangle$ in der die folgenden Gleichungen gelten:

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \tag{B.1}$$
$$x \cdot 1 = 1 \cdot x = x \tag{B.2}$$
$$x \cdot x^{-1} = x^{-1} \cdot x = 1 \tag{B.3}$$

Eine Gruppe heißt **abelsch** bzw. **kommutativ** falls zusätzlich noch die folgende Gleichung, gilt:

$$x \cdot y = y \cdot x \tag{B.4}$$

Eine **Halbgruppe** ist eine Algebra $\mathbf{G} = \langle G, \cdot\rangle$ der Signatur $\langle 2\rangle$ in der Gleichung B.1 gilt.

Definition B.15 (Ring, Ring mit Eins, Körper)
Ein **Ring** ist eine Algebra $\mathbf{R} = \langle R, +, \cdot, -, 0\rangle$ der Signatur $\langle 2, 2, 1, 0\rangle$, für die:

$$\mathbf{R} = \langle R, +, -, 0\rangle \text{ eine abelsche Gruppe ist} \tag{B.5}$$
$$\mathbf{R} = \langle R, \cdot\rangle \text{ eine Halbgruppe ist, und} \tag{B.6}$$
$$x \cdot (y + z) = (x \cdot y) + (x \cdot z), \text{ sowie } (x + y) \cdot z = (x \cdot z) + (y \cdot z) \tag{B.7}$$

gelten.

Ein **Ring mit Eins** ist eine Algebra $\mathbf{R} = \langle R, +, \cdot, -, 0, 1\rangle$ der Signatur $\langle 2, 2, 1, 0, 0\rangle$, so daß die Gleichungen B.5 – B.7, sowie B.2 erfüllt sind.

Ein **Körper** ist eine Algebra $\mathbf{K} = \langle K, +, \cdot, -, 0, 1\rangle$ der Signatur $\langle 2, 2, 1, 0, 0\rangle$, so daß:

$$\mathbf{K} = \langle K, +, \cdot, -, 0\rangle \text{ ein Ring ist} \tag{B.8}$$
$$\mathbf{K} \setminus \{\mathbf{0}\} = \langle K \setminus \{0\}, \cdot\rangle \text{ eine kommutative Gruppe ist} \tag{B.9}$$

Beispiel B.1
Die Mengen $\mathbb{Z}$, $\mathbb{Q}$ und $\mathbb{R}$ zusammen mit der gewöhnlichen Addition $+$ und Multiplikation $\cdot$ sind Ringe mit Eins, $\mathbb{Q}$ und $\mathbb{R}$ sogar Körper.

Polynome

Definition B.16 (Univariates Polynom)

Unter einem *univariaten Polynom* über einem Ring mit 1 $\mathbf{R}$ [2] versteht man Ausdrücke der Form

$$p(x) = \sum_{0 \leq i \leq n} a_i x^i$$

wobei die $a_i \in \mathbf{R}$ Elemente des Rings sind und *Koeffizienten* genannt werden. x ist die *Unbestimmte* des Polynoms.

Der *Grad* des Polynoms ist definiert durch

$$\deg(p) := \begin{cases} -\infty & \text{falls } a_i = 0 \text{ für alle } i, \\ max\{\nu \in \mathbb{N} : a_\nu \neq 0\} & \text{sonst} \end{cases}$$

Bemerkung B.9

Polynome der Grade $0, 1, 2, 3$ bezeichnet man auch als *konstante, lineare, quadratische* bzw. *kubische* Polynome, Polynome mit Grad > 1 allgemein als *nichtlineare* Polynome.

Mit $R[x]$ bezeichnet man die Menge aller univariaten Polynome mit Koeffizienten aus R. $R[x]$ ist selbst wieder ein Ring mit 1, der sog. *Polynomring über* R.

Analog definiert man im Falle mehrerer Unbestimmter die multivariaten Polynome:

Definition B.17 (Multivariate Polynome)

Sei $i = (i_1 \ \ldots \ i_n), n \in \mathbb{N}$ ein *Multiindex*, und $|i| = i_1 + \cdots + i_n$ die *Ordnung* von i. Ein *multivariates Polynom* über R (bzw. K) in $x = (x_1, \ldots, x_n)$ des *Grades* m ist definiert als

$$p(x) = p(x_1, \ldots, x_n) = \sum_{|i| \leq m} a_{i_1 \ \ldots \ i_n} x_1^{i_1} \cdot \cdots \cdot x_n^{i_n}$$

Beispiel B.2

$x^2 + x + y + 3$ und $x \cdot y$ sind quadratische Polynome in x und y, während $x^2 \cdot y$ ein kubisches Polynom

$$x^2 \cdot y = \sum_{|i| \leq 3} a_{i_x i_y} x^{i_x} \cdot y^{i_y}$$

mit $a_{21} = 1$ und $a_{i_x i_y} = 0$ für alle anderen Koeffizienten ist.

Die rechte Seite der Gleichung 2.1: $F_{Rotglied} \cdot r_{Rotglied}$ ist ein quadratisches Polynom in $F_{Rotglied}$ und $r_{Rotglied}$

2. natürlich gilt dies auch für Körper.

Ungleichungssysteme

Damit läßt sich der Begriff eines Ungleichungssystems fassen.

Definition B.18 (Ungleichungssystem des Grades n)
Ein *Ungleichungssystem $\mathcal{U}$ des Grades n in den Variablen x* ist eine endliche Menge von Gleichungen und Ungleichungen zwischen Polynomen in x mit maximalem Grad n. D.h.

$$U = \{g_1, \ldots, g_m\} \text{mit} \begin{cases} g_i = p_i \circ q_i & \circ \in \{<, >, =, \neq, \leq, \geq\} \\ \deg(r) \leq n & r \in \{p_1, \ldots, p_m\} \cup \{q_1, \ldots, q_m\} \end{cases}$$

In dieser Definition ist noch nicht festgelegt über welcher Basis die Polynome definiert sind. In Falle des Leitbeispiels und sind es die Polynome über $\mathbb{Q}[x_1, \ldots, x_n]$, d.h. Polynome mit rationalen Koeffizienten.

Beispiel B.3
Das aus dem Leitbeispiel resultierende Ungleichungssystem aus (2.6):

$$\begin{array}{lll} M_{Ca} = F_{Ca} \cdot r_{Ca} & F_{Sp} = F_{Ch} & \\ M_{Ca} = M_{Bbs} & M_{Sp} = F_{Sp} \cdot r_{Sp} & F_{Ca}, F_{Cr}, F_{Bbs}, F_{Ch}, \\ M_{Bbs} = F_{Bbs} \cdot r_{Bbs} & M_{Ra} = M_{Sp} & F_{Sp}, F_{Ra}, F_{Rw} \geq 0 \\ M_{Cr} = M_{Bbs} & M_{Ra} = F_{Ra} \cdot r_{Ra} & r_{Ca}, r_{Cr}, r_{Bbs}, \\ M_{Cr} = F_{Cr} \cdot r_{Cr} & M_{Rw} = M_{Ra} & r_{Sp}, r_{Ra}, r_{Rw} > 0 \\ F_{Ch} = F_{Cr} & M_{Rw} = F_{Rw} \cdot r_{Rw} & \end{array}$$

ist somit ein quadratisches Ungleichungssystem über

$$\begin{array}{c} (F_{Ca}, F_{Bbs}, F_{Cr}, F_{Ch}, F_{Sp}, F_{Ra}, F_{Rw}, \\ M_{Ca}, M_{Bbs}, M_{Cr}, M_{Sp}, M_{Ra}, M_{Rw}, \\ r_{Ca}, r_{Bbs}, r_{Cr}, r_{Sp}, r_{Ra}, r_{Rw}). \end{array}$$

C Konkrete Gegenstandsbereiche über textuellen Daten

Nachdem in Abschnitt 4.3 konkrete Gegenstandsbereiche über Zahlen vorgestellt wurden, soll in diesem Anhang am Beispiel konkreter Gegenstandsbereiche über Strings gezeigt werden, daß die prinzipielle Vorgehensweise auch auf andere konkrete Gegenstandsbereiche übertragen werden kann.

Zunächst werden einfache konkrete Gegenstandsbereiche über Strings vorgestellt, die im wesentlichen eine Übertragung der einfachen numerischen Gegenstandsbereiche sind. Auch eine Übertragung der Entscheidungsprozeduren ist hier möglich.

Danach wird gezeigt, daß komplexe konkrete Gegenstandsbereiche über Strings mit Hilfe von regulären Ausdrücken zu realisierbar sind.

C.1. Elementare Begriffe

Definition C.1 (Wörter)
Sei $\Sigma = \{a^j \mid j \in J\}$ eine nichtleere Menge. Dann heißen nichtleere Folgen $a_1 \ldots a_n, a_i \in \Sigma$ von Elementen aus Σ **Wörter** über dem **Alphabet** Σ, die Elemente $a_i \in \Sigma$ **Buchstaben** des Alphabets. Das **leere Wort** wird durch ϵ bezeichnet.

Σ^* bezeichnet die Menge aller endlichen Wörter, und $\Sigma^+ := \Sigma^* \setminus \{\epsilon\}$ die Menge aller nichtleeren Wörter.

Für ein Wort $w = a_1 \ldots a_n$ bezeichnet $|w| = n$ die **Länge eines Wortes**.

Für zwei Wörter u, v bezeichnet $w = uv$ die Konkatenation der beiden Wörter.

Definition C.2 (Kanonische Ordnung)
Sei $<_\Sigma$ eine Totalordnung auf Σ. Dann ist

$$u \prec v$$
$$\overset{def}{\Leftrightarrow}$$
$$|u| < |v| \vee |u| = |u| \wedge \exists x, y, z \in \Sigma^* \exists a, b \in \Sigma : u = xaz, v = xby \wedge a <_\Sigma b$$

eine Totalordnung auf Σ^*, die **kanonische Ordnung** über Σ^*

C.2. Gegenstandsbereiche über Strings

C.2.1. Einfache konkrete Gegenstandsbereiche über Strings

Es gilt der folgende Satz:

Satz C.1
Sei $|\Sigma| = n$, und $pos(a) : \Sigma \mapsto \{n_1, \ldots, n_n\}$ die Position (Ordinalzahl) von $a \in \Sigma$ bzgl. $<_\Sigma$. Dann ist $f : \Sigma^+ \mapsto \mathbb{N}$

$$\begin{aligned} f(\epsilon) &= 0 \\ f(w) = f(a_1 \ldots a_m) &= \sum_{i=0}^{m_1} pos(a_{i+1}) n^{(m-i)} \end{aligned} \tag{C.1}$$

ein ordnungserhaltender Isomorphismus.

Damit lassen sich die konkreten Gegenstandsbereiche $\mathbf{N_1}$, $\mathbf{N_2}$ und $\mathbf{N_3}$ aus Abschnitt 4.3 direkt in konkrete Gegenstandsbereiche über Σ^* übertragen.

Satz C.2
Die Relationalstrukturen $\mathbf{S_1}$, $\mathbf{S_2}$ und $\mathbf{S_3}$

$$\begin{aligned} \mathbf{S_1} &= \langle \Sigma^\star, \top, \bot, =_\omega, \neq_\omega, \omega \in \Sigma^\star \rangle \\ \mathbf{S_2} &= \langle \Sigma^\star, \top, \bot, =_\omega, \neq_\omega, \prec_\omega, \preceq_\omega, \succ_\omega, \succeq_\omega, \omega \in \Sigma^\star \rangle \\ \mathbf{S_3} &= \langle \langle \Sigma^*, \prec \rangle \rangle \end{aligned}$$

sind zulässige konkrete Gegenstandsbereiche über Σ^*

Beweis. Folgt trivialerweise aus Satz C.1. □

Beispiel C.1
An dieser Stelle soll das Beispiel aus Abschnitt 4.3: Recherche in bibliographischen Datenbanken, wieder zur Verdeutlichung der Ausdrucksstärke der unterschiedlichen konkreten Gegenstandsbereiche über Strings wieder aufgegriffen werden.

So ist es in $\mathbf{S_1}$ möglich Anfragen wie etwa: ≫*Alle Publikationen in Proceedings mit Titel 'SIGIR'*≪ durch (and inproceedings (constrain booktitle (string (?x) (=SIGIR ?x)))) zu formulieren.

$\mathbf{S_2}$ erlaubt es, Präfixe[1] in die Anfragen und Konzeptbeschreibungen aufzunehmen, etwa ≫*Alle Publikationen die eine Titel haben der mit 'SIGIR' beginnt*≪ durch den Konzeptterm (and inproceedings (constrain booktitle (string (?x) (¿=SIGIR ?x)))(constrain booktitle (string (?x) (¡SIGIS ?x))))).

C.2.2. Reguläre Ausdrücke als konkreter Gegenstandsbereich

Eine Möglichkeit zur Bereitstellung ausdrucksstärkerer Gegenstandsbereiche über Strings ist die Benutzung regulärer Ausdrücke. Möglich ist dies, da das Schnittproblem für reguläre Sprachen im Gegensatz zum Schnittproblem für kontextfreie Sprachen entscheidbar ist.

Definition C.3 (Regulärer Ausdruck)
Sei Σ ein Alphabet. Sind die Buchstaben $a \in \Sigma$ die atomaren Terme und die **reguläre Ausdrücke** über Alphabet Σ sind Terme über der Signatur

- $\cup$ (Vereinigung)
- $\cdot$ (Konkatenation)
- * (Kleene-Star)
- δ (Leeres Wort)

und den Buchstaben a des Alphabetes. Es gilt $\Sigma \cap \{\cup, cdot, ^\star, \delta\} = \emptyset$.

Definition C.4 (Sprache eines regulären Ausdruckes)
Die durch den regulären Ausdruck r beschriebene Sprache $L(r)$, oder kurz die **Sprache des Ausdruckes** r, ist induktiv über die Struktur des regulären Ausdruck definiert:

- $L(\delta) := \emptyset$
- $L(a) := \{a\}$
- $L(r \cdot s) := L(r) \cdot L(s)$
- $L(r \cup s) := L(r) \cup L(s)$
- $L(r^\star) := (L(r))^\star$.

1. Aber keine Postfixe.

Definition C.5 (Verallgemeinerter regulärer Ausdruck)
Verallgemeinerte reguläre Ausdrücke sind um die Operationssymbole $\cap$ (Schnitt), $\setminus$ (Differenz), $\sim$ (Komplement bzgl. $\Sigma^\star$) erweiterte reguläre Ausdrücke.

Die beiden folgenden Sätze sind wohlbekannte Ergebnisse aus dem Bereich der Formalen Sprachen und der Automatentheorie (z.B. aus [HU79] and [MMP+95]):

Satz C.3 (Abgeschlossenheit unter Komplement)
Die Klasse der regulären Ausdrücke ist abgeschlossen unter Komplementbildung.

Satz C.4 (Entscheidbarkeit des Schnittproblems)
Seien r und s zwei verallgemeinerte reguläre Ausdrücke. Dann ist das folgenden Problem entscheidbar:

$$\mathcal{L}(r) \cap \mathcal{L}(s) = \emptyset$$

Bemerkung C.1
Der Beweis der Entscheidbarkeit des Schnittproblems ist konstruktiv und benutzt (nicht)deterministische endliche Automaten.

Aus dem eben gesagten folgt die Zulässigkeit regulärer Ausdrücke als konkreter Gegenstandsbereich.

Satz C.5 (Zulässigkeit regulärer Ausdrücke)

($T(\Sigma^\star, c_1, \ldots, c_n)$ beschreibt die Menge der Terme über der Signatur $c_1, \ldots, c_n$.):

$$\mathbf{S_4} = T(\Sigma^\star, \cdot, ^\star, \sim, \delta, a \in \Sigma)$$
$$\mathbf{S_5} = T(\Sigma^\star, \cdot, ^\star, \sim, \cup, \cap, \setminus, \delta, a \in \Sigma)$$

Beispiel C.2
In $\mathbf{S_4}$ ist es daher möglich die Anfrage: »*Alle Publikationen die 'SIGIR' im Buchtitel enthalten*« durch (and inproceedings (constrain booktitle (string (?x) (regexp ?x "*SIGIR*")))))[2].

Andere, komplexere, Anfragen können durch Einsatz anderer Operatoren konstruiert werden. Dies kann sogar durch die Einführung zusätzlicher Operatoren noch weiter gefasst werden, wie das folgende Ergebnis zeigt [MMP+95]:

Satz C.6 (Abgeschlossenheit)
*Die Klasse der regulären Sprachen ist abgeschlossen unter den Operationen **Präfix**, **Suffix**,**Minimum**, **Maximum**, **Linker Quotient** und **Rechter Quotient**. Die einzelnen Operationen sind dabei wie folgt definiert:*

2. Es wird die übliche Notation für reguläre Ausdrücke verwendet, in der *, ? etc. als Metazeichen benutzt werden.

- $\text{reverse}(L) = \{a_1 \dots a_n \mid n \in \mathbb{N}, a_n \dots a_1 \in L\}$
- $\text{prefix}(L) = \{\omega \in \Sigma^\star \mid \exists \upsilon \in \Sigma^\star : \omega\upsilon \in L\}$
- $\text{suffix}(L) = \{\omega \in \Sigma^\star \mid \exists \upsilon \in \Sigma^\star : \upsilon\omega \in L\}$
- $\min(L) = \{\omega \in L \mid \forall \upsilon \in \text{prefix}(\{\omega\}) \setminus \{\omega\} : \upsilon \notin L\}$
- $\max(L) = \{\omega \in L \mid \forall \upsilon \in \Sigma^+ : \omega\upsilon \notin L\}$
- $\text{lquotient}(L_1, L_2) = \{\omega \in \Sigma^\star \mid \exists \nu \in L_1 : \nu\omega \in L_2\}$
- $\text{rquotient}(L_1, L_2) = \{\omega \in \Sigma^\star \mid \exists \nu \in L_1 : \omega\nu \in L_2\}$

Daher ist der Gegenstandsbereich $\mathbf{S_6}$ der $\mathbf{S_5}$ um all diese Operatoren erweitert ebenfalls zulässig. Offensichtlich können selbst kompliziertes Sachverhalte in $\mathbf{S_6}$ formuliert werden. AMoRE [MMP+95] ist ein System, das zur Realisierung des Gegenstandsbereiches $\mathbf{S_6}$ eingesetzt werden kann.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AA94] K. D. Ashley und V. Aleven. A logical Representation for Relevance Criteria. In S. Wess, K.-D. Althoff und M. M. Richter (Herausgeber), „Topics in Case-Based Reasoning – Selected Papers from the First European Workshop on Case-Based Reasoning – EWCBR-93 Workshop“, Seiten 338–352, Kaiserslautern, Germany (1994). Springer Verlag.

[AK84] Hassan Ait-Kaci. „A Lattice-Theoretic Approach to Computations Based on a Calculus of Partially Ordered type Structures“. Dissertation, University of Pennsylvania, Philadelphia, PA (1984).

[Baa90] Franz Baader. Terminological Cycls in KL-ONE-based Knowledge-Representation Languages. In „Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI-90)“, Seiten 621–626, Boston, MA (1990).

[Baa91] Franz Baader. Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles. In „Proc. IJCAI 91“, Seiten 446–451 (1991).

[Baa96] Franz Baader. Terminologische Wissenrepräsentationssprachen. Eingeladener Vortrag auf dem Jahrestreffen der GI-Fachgruppe Logik in der Informatik (Mai 1996).

[BBH+92] F. Baader, H. Bürckert, B. Hollunder, A. Laux und W. Nutt. Terminologische Logiken. *KI* (3), 23–33 (1992).

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuiness, Daniele Nardi und Peter Patel-Schneider (Herausgeber). „The Description Logic Handbook – Theory, Implementation and Applications“. Cambridge University Press (2003).

[Ber96] Ralph Bergmann. „Effizientes Problemlösen durch flexible Wiederverwendung von Fällen auf verschiedenen Abstraktionsebenen“. Dissertation, University of Kaiaserslautern (1996).

[Ber01] Ralph Bergmann. Highlights of the European INRECA Projects. In David W. Aha und Ian Watson (Herausgeber), „Case-Based Reasoning Research and Development, Proceedings of the 4^{th} International Conference on Case-Based Reasoning“, Band 2080 aus „Lecture Notes in Computer Science“, Vancouver, BC (August 2001). Springer Verlag.

[BFT95] Paolo Bresciani, Enrico Franconi und Sergio Tessaris. Implementing and testing expressive Description Logics: a preliminary report. In „Proceedings of the 1995 KRUSE Symposium“, Santa Cruz (August 1995).

[BH90] F. Baader und B. Hollunder. KRIS: Knowledge Representation and Inference System – System Description. Technical Memo TM-90-03, DFKI (11 1990).

[BH91a] F. Baader und P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In „Proc. IJCAI-91“, Seiten 452–457 (1991).

[BH91b] F. Baader und P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. Research Report RR-91-10, DFKI, Kaiserslautern, Germany (April 1991).

[BHHM93] H. Boley, P. Hanschke, K. Hinkelmann und M. Meyer. CoLab: A Hybrid Knowledge Representation and Compilation Laboratory. Research Report RR-93-08, DFKI, Kaiserslautern, Germany (Januar 1993).

[BHW92] Michael Bayer, Bernhard Herbig und Stefan Wess. Ähnlichkeit und Ähnlichkeitsmaße. In „Fallbasiertes Schließen - Eine Übersicht“, Band 1, seki working paper 8, Seiten 135–153. Universität Kaiserslautern (1992).

[BMPS+91] R.J. Brachman, D.L. MacGuiness, P.F. Patel-Schneider, L.A. Resnick und A. Borgida. Living with Classic: when and How to Use a KL-ONE-like Language. In „Principles of Semantic Networks“. Morgan Kaufmann (1991).

[BN03] Franz Baader und Werner Nutt. An introduction to description logics. In Baader und andere [BCM+03], Kapitel 2.

[Boo94] Grady Booch. „Object-Oriented Analysis and Design with Applications“. Benjamin Cummings, Redwood City, CA (1994).

[Bra77] Ron J. Brachman. What's in a Concept: Structural Foundations for Semantic Networks. *International Journal on Man-Machine Studies* **9**(2), 127–152 (1977).

[BS81] Stanley Burris und H.P. Sankappanavar. „A Course in Universal Algebra", Band 78 aus „Graduate texts in mathematics". Springer Verlag, New York (1981).

[BS85] R. J. Brachman und J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science* **9**(2), 171–216 (April 1985).

[BS98] Ralph Bergmann und Armin Stahl. Similarity measures for object-orientes case representations. In Barry Smyth und Padraig Cunningham (Herausgeber), „Advances in Case-Based Reasoning, Proceedings of the 4^{th} European Workshop (EWCBR-98)", Band 1488 aus „Lecture Notes in Computer Science", Seiten 25–36, Dublin, Ireland (September 1998). Springer Verlag.

[BV99] Ralph Bergmann und Ivo Vollrath. Generalized cases: Representation and steps towards efficient similarity assessment. In Wolfram Burgard, Thomas Christaller und Armin B. Cremers (Herausgeber), „KI-99: Advances in Artificial Intelligence, 23rd Annual German Conference on Artificial Intelligence", Band 1701 aus „Lecture Notes in Computer Science", Seiten 195–206, Bonn, Germany (1999).

[BVW99] Ralph Bergmann, Ivo Vollrath und Thomas Wahlmann. Generalized cases and their application to electronic designs. In E. Melis (Herausgeber), „Proceedings of the 7^th German Workshop on Case-Based Reasoning (GWCBR-99)", Würzburg, Germany (1999).

[CDG03] Diego Calvanese und Guiseppe De Giacomo. Expressive description logics. In Baader und andere [BCM$^+$03], Kapitel 5.

[CH91] G. E. Collins und Hoon Hong. Partial cylindrical algebraic decomposition. *Journal of Symbolic Computation* **12**, 299–328 (1991).

[Cha93] Vijay Chandru. Variable Elimination in Linear Constraints. *The Computer Journal* **36**(5), 463–472 (1993). Special Issue on Quantifier Elimination.

[CLN94] Diego Calvanese, Maurizio Lenzerini und Daniele Nardi. A Unified Framework for Class Based Representation Formalisms. In „Proc. of the 4th Int. Conf. on the Priinciples of Knowledge Representation ond Reasoning (KR-94)“, Seiten 109–120, Bonn, Germany (1994). Morgan Kaufmann.

[CNJ+91] R. Cunis, B. Neumann, J. Jaschinsky, W. Heimann und T. Sommer. Fallbasierte Diagnoseunterstützung für ein flexibles Fertigungssystem – Ergebnisse einer Vorstudie –. Bericht LKI-M-2/91, Labor f (1991).

[Col75] G. E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In „Proc. of the Second GI Conference on Automata Theory and Formal Languages“, Nummer 33 in LNCS, Seiten 512–532. Springer (1975).

[Cor97] Rational Software Corporation. Unified Modeling Language Summary. Bericht, Santa Clara,CA (1997).

[CY91a] P. Coad und E. Yourdon. „Object-Oriented Analysis“. Prentice-Hall, Englewood Cliffs, NJ (1991).

[CY91b] P. Coad und E. Yourdon. „Object-Oriented Design“. Prentice-Hall, Englewood Cliffs, NJ (1991).

[Don03] Francesco M. Donini. Complexity of reasoning. In Baader und andere [BCM+03], Kapitel 3.

[DS95] Andreas Dolzmann und Thomas Sturm. Simplification of Quantifier-free Formulas over Ordered Fields. Bericht MIP-9517, Universität Passau, Passau, Germany (Oktober 1995).

[DS96] Andreas Dolzmann und Thomas Sturm. REDLOG – Computer Algebra Meets Computer Logic. Bericht MIP-9603, Universität Passau, Fakultät für Mathematik und Informatik, Passau, Germany (Februar 1996).

[Fuh93] Norbert Fuhr. Information Retrieval – Skriptum zur Vorlesung im SS 93. (1993).

[FV93] Rolf Fischer und Klaus Vogelsang. „Größen und Einheiten in Physik und Technik“. Verlag Vogelsang, Berlin, Germany, 6 Auflage (1993).

[GCDAGA99] Pedro A. González-Calero, Belén Díaz-Agudo und Mercedes Gómez-Albarrán. Applying dls for retrieval in case-based reasoning (1999).

[Han92] P. Hanschke. Specifying Role Interaction in Concept Languages. In B. Nebel, Rich C. und Swartout W. (Herausgeber), „Principles of Knowledge Representation and Reasoning –Proc. of the 3 rd International Conference“, Cambridge, MA (1992).

[Han93] P. Hanschke. „A Declarative Integration of Terminological, Constraint-Based, Data-driven, and Goal-directed Reasoning“. Dissertation, Universität Kaiserslautern (1993).

[Han96] P. Hanschke. „A Declarative Integration of Terminological, Constraint-Based, Data-driven, and Goal-directed Reasoning“, Band 122 aus „Dissertationjen zur künstlichen Intelligenz“. Infix, Sankt Augustin, Germany (1996).

[Hay77] Patrick J. Hayes. In defense of logic. In „Proc. IJCAI77“, Seiten 559–565, Cambridge, MA (1977).

[Her93] U. Herzog. „Fahrradheilkunde“. Moby Dick Verlag, Kiel, Germany, 17 Auflage (1993).

[HM01] Volker Haarslev und Ralf Möller. High performance reasoning with very large knowledge bases: A practical case study. In Bernhard Nebel (Herausgeber), „Proc. of the 17 th International Joint Conference on Artificial Intelligence (IJCAI-01)“, Seattle, WA (August 2001).

[HM02a] Volker Haarslev und Ralf Möller. Optimization strategies for instance retrieval. In I. Horrocks und S. Tessaris (Herausgeber), „Proc. of the 2002 International Workshop on Description Logics (DL2002)“, Band 53 aus „CEUR Publications“, Toulouse, France (2002).

[HM02b] Volker Haarslev und Ralf Möller. Practical reasoning in racer with a concrete domain for linear inequations. In I. Horrocks und S. Tessaris (Herausgeber), „Proc. of the 2002 International Workshop on Description Logics (DL2002)“, Band 53 aus „CEUR Publications“, Toulouse, France (2002).

[HM02c] Volker Haarslev und Ralf Möller. „RACER User's Guide and Reference Manual, Version 1.7.6“. Concordia University and University of Applied Sciences in Wedel, Montreal, Quebec and Wedel, Germany (Dezember 2002).

[HMM01] Volker Haarslev, Ralf Möller und Wessel Michael. The description logic $\mathcal{ALCNHR}+$ extended with concrete domains: A

practically motivated approach. In Rajeev Goré, Alexander Leitsch und Tobias Nipkow (Herausgeber), „Proc. of the1 st International Joint Conference on Automated Reasoning (IJCAR-01)“, Nummer 2083 in Lecture Notes in Artifical Intelligence, Siena, Italy (2001). Springer Verlag.

[HMT01] Volker Haarslev, Ralf Möller und Anni-Yasmin Turhan. Exploiting pseudo models for tbox and abox reasoning in expressive description logics. In Rajeev Goré, Alexander Leitsch und Tobias Nipkow (Herausgeber), „Proc. of the1 st International Joint Conference on Automated Reasoning (IJCAR-01)“, Nummer 2083 in Lecture Notes in Artifical Intelligence, Siena, Italy (2001). Springer Verlag.

[HN90] B. Hollunder und W. Nutt. Subsumption Algorithms for Concept Languages. Research Report RR-90-04, DFKI, Kaiserslautern, Germany (April 1990).

[Hol90] B. Hollunder. Hybrid Inferences in KL-ONE-based knowledge representation systems. In „Proc. GWAI-90“, Informatik Fachberichte, Ehringerfeld, Germany (1990). Springer.

[Hol94] Bernhard Hollunder. „Algorithmic Foundations of Terminological Knowledge RepresentationSystems“. Dissertation, Universität des Saarlandes, Saarbrücken, Germany (1994).

[Hon93] H. Hong. RISC-CLP(Real): Constraint Logic Programming over the real numbers. In „Constraint Logic Programming: Selected Research“. MIT Press, Cambridge, MA (1993).

[Hor98] Ian Horrocks. The FaCT system. In „Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98“, Nummer 1397 in Lecture Notes in Artificial Intelligence, Seiten 307 – 312. Springer-Verlag, Berlin (1998).

[Hor99] I. Horrocks. FaCT and iFaCT. In Patrick Lambrix, Alex Borgida, Maurizio Lenzerini, Ralf Möller und Peter Patel-Schneider (Herausgeber), „Proc. of the 1999 International Workshop on Description Logics (DL99)“, Seiten 133–135 (1999).

[HU79] John E. Hopcroft und Jeffrey Ullman. „Introduction to automata theory, languages and computation“. Addison-Wesley (1979).

[Ihr93] Thomas Ihringer. „Allgemeine Algebra“. Teubner Studienbücher Mathematik. B.G. Teubner, Stuttgart, Germany, 2 Auflage (1993).

[ISO94] ISO. „ISO 10303 Part 105: Integrated Application Ressource: Kinematics“. ISO (1994).

[Kam93a] G. Kamp. Fälle in Unterstützungssystemen für Servicetechniker. In „Fälle in Hybriden Systemen – Beiträge zum 2. Workshop des Arbeitskreises Fallbasiertes Schließen im Fachausschuß Expertensysteme der Gesellschaft für Informatik“ (1993).

[Kam93b] Gerd Kamp. Wissensrepräsentation für Service Support Systeme. In A. B. Cremers und G. Lakemeyer (Herausgeber), „AKI Workshop Wissensrepräsentation“, Bonn, Germany (1993). Rheinische Friedrich-Wilhelms Universität Bonn.

[Kam94a] Gerd Kamp. AMS - A Case-Based Service Support System. In R. V. Rodriguez F. D. Anger und M. Ali (Herausgeber), „Proc. of the 7th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems“, Seiten 677–683, Austin, TX (1994). Gordon and Breach Science Publishers.

[Kam94b] Gerd Kamp. Case-Based Reasoning in Corporate Support. In „MLNet Workshop on Industrial Applications of Machine Learning“, Seiten 147–156, Dourdan, France (1994).

[Kam94c] Gerd Kamp. Integrating Semantic Structure and Technical Documentation in Case-Based Service Support Systems. In S. Wess, K.-D. Althoff und M. M. Richter (Herausgeber), „Topics in Case-Based Reasoning – Selected Papers from the First European Workshop on Case-Based Reasoning – EWCBR-93 Workshop“, Band 837 aus „Lecture Notes in Artificial Intelligence“, Seiten 392–403, Kaiserslautern, Germany (1994). Springer Verlag.

[Kam94d] Gerd Kamp. On the use of CBR in Corporate Service and Support. In M. Keane, M. Manago und J. P. Haton (Herausgeber), „Advances in Case-Based Reasoning, Proceedings of the 2^{nd} European Workshop (EWCBR-94)“, Seiten 175–184. Acknosoft Press (1994).

[Kam94e] Gerd Kamp. Ontologien im Bereich der Wissensintensiven Fallbasierten Diagnose technischer Systeme. In „Workshop Modularisie-

rung großer Wissensbasenäuf der KI-94“, Saarbrücken, Germany (1994).

[Kam96a] Gerd Kamp. Using Description Logics for Knowledge Intensive Case-based Reasoning. In I. Smith und B. Faltings (Herausgeber), „Advances in Case-Based Reasoning, Proceedings of the 3^{rd} European Workshop (EWCBR-96)“, Band 1168 aus „Lecture Notes in Artificial Intelligence“, Seiten 204–218, Lausanne, Switzerland (November 1996). Springer Verlag.

[Kam96b] Gerd Kamp. Wissensbasierte Inferenzmethoden zur Modellierung technischer Systeme. In J. Lunze (Herausgeber), „Proc. VDE FA 1.6 Workshop Intelligente Systeme der Automatisierungstechnik“, Emmelndorf, Germany (1996).

[Kam97a] Gerd Kamp. Admissible concrete domains for CBR based on description logics. In R. Bergmann und W. Wilke (Herausgeber), „Proceedings of the 5^th German Workshop on Case-Based Reasoning (GWCBR-97)“, Seiten 119–128, Bad Honnef, Germany (1997).

[Kam97b] Gerd Kamp. On the Admissibility of Concrete Domains for CBR based on Description Logics. In D. Leake und E. Plaza (Herausgeber), „Proc. ICCBR97“, Providence, RI (1997). Springer.

[KK92] G. Kamp und S. Kockskämper. Positionspapier. In „Proc. Workshop Welche Rolle spielen Fälle“, Bonn, Germany (1992).

[KLG98] Gerd Kamp, Steffen Lange und Christoph Globig. Related areas. In Lenz, Mario and Bartsch-Spörl, Brigitte and Burkhard, Hans-Dieter and Wess, Stefan (Herausgeber), „Case-based Reasoning Technology – From Foundations to Applications“, LNAI State-of-the-Art Survey, Kapitel 13. Springer Verlag, Berlin, Heidelberg, New York (1998).

[KN97] Gerd Kamp und Bernd Neumann. Knowledge-based Inference Methods for Modeling Technical Systems. In R. H. Sprague (Herausgeber), „Proc. 30th Hawaiian International Conference on System Sciences“, Wailea, HA (1997). Computer Science Press.

[KNPB94] Gerd Kamp, Bernd Neumann, Petra Pirk und Hans-Dieter Burkhard. Fallbasierte Diagnose für technische Systeme. Zwischenbericht an die DFG (1994).

[Koe94] Jana Koehler. An Application of Terminological Logics to Case-based Reasoning. In „Principles of Knowledge Representation and Reasoning –Proc. of the 5th International Conference“, Bonn, Germany (1994).

[KPB96] Gerd Kamp, Petra Pirk und Hans-Dieter Burkhard. Falldaten: Case-based Reasoning for the Diagnosis of Technical Devices. In G. Görz und S. Hölldobler (Herausgeber), „KI-96: Advances in Artificial Intelligence (Proc. of the 20th Annual German Conf. on Artificial Intelligence)“, Band 1137 aus „Lecture Notes in Artificial Intelligence“, Seiten 149–161, Dresden, Germany (September 1996). Springer Verlag.

[KRI92] „KRIS Manual“. DFKI, 1.0 Auflage (6 1992).

[KW96a] Gerd Kamp und Holger Wache. CTL – a description logic with expressive concrete domains. Bericht, LKI (1996).

[KW96b] Gerd Kamp und Holger Wache. Using Description Logics for Consistency-based Diagnosis. In L. Padgham, Franconi E., M. Gehrke, D.L. McGuiness und P. Patel-Schneider (Herausgeber), „Proc. of the 1996 International Workshop on Description Logics (DL96)“, Nummer WS-96-05, Seiten 136–140, Boston, MA (November 1996). AAAI Press.

[Las90] Jean-Louis Lassez. Parametric queries, linear constraints and variable elimination. In A. Miola (Herausgeber), „Design and Implementation of Symbolic Computation Systems“, Band 429 aus „LNCS“, Seiten 164–173, Capri, Italy (April 1990). Springer Verlag.

[LOK91] K. von Luck und B. Owsnicki-Klewe. KL-ONE: Eine Einführung. In „Wissensrepräsentation“, Seiten 103–121. Oldenbourg (1991).

[Lut99] Carsten Lutz. Reasoning with concrete domains. In Thomas Dean (Herausgeber), „Proc. of the 16 th International Joint Conference on Artificial Intelligence (IJCAI-99)“, Seiten 90–95, Stockholm, Sweden (July 31 – August 6 1999). Morgan-Kaufmann Publisher.

[Lut00] Carsten Lutz. Nexptime-complete description logics with concrete domains. LTCS-Report 00-01, LuFG Theoretical Computer Science, RWTH Aachen, Germany (2000).

[Lut01a] Carsten Lutz. „The Complexity of Reasoning with Concrete Domains“. Dissertation, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen (2001).

[Lut01b] Carsten Lutz. NExpTime-complete description logics with concrete domains. In Rajeev Goré, Alexander Leitsch und Tobias Nipkow (Herausgeber), „Proc. of the1 st International Joint Conference on Automated Reasoning (IJCAR-01)“, Nummer 2083 in Lecture Notes in Artifical Intelligence, Seiten 45–60, Siena, Italy (2001). Springer Verlag.

[Lut02a] Carsten Lutz. Adding numbers to the $\mathcal{SHIQ}$ description logic – first results. In „Principles of Knowledge Representation and Reasoning –Proc. of the 8 th International Conference“, Breckenridge, CA (2002). World Scientific Publishing Co. Pte. Ltd.

[Lut02b] Carsten Lutz. Description logics with concrete domains – a survey. In „Advances in Modal Logics Volume 4“. World Scientific Publishing Co. Pte. Ltd. (2002).

[Lut02c] Carsten Lutz. PSpace reasoning with the description logic $\mathcal{ALCF}(\mathcal{D})$. *Logic Journal of the IGPL* **10**(5), 535–568 (2002).

[Mac91] R. MacGregor. The Evolving Technology of Classification-based Knowledge Representation Systems. In „Principles of Semantic Networks“. Morgan Kaufmann (1991).

[MB02] Babak Mougouie und Ralph Bergmann. Generalized cases: Representation and steps towards efficient similarity assessment. In Susan Craw und Alun D. Preece (Herausgeber), „Advances in Case-Based Reasoning, Proceedings of the 6^{th} European Conference (ECCBR 2002)“, Band 2416 aus „Lecture Notes in Computer Science“, Aberdeen, Scotland (September 2002). Springer Verlag.

[MBC+94] Michel Manago, Ralph Bergmann, Noël Conruyt, Ralph Traphöner, James Pasley, Jacques LeRenard, Frank Maurer, Stefan Wess, Klaus-Dieter Althoff und Sylvie Dumont. CASUEL: A Common Case Representation Language. Report, INRECA consortium (Mai 1994).

[Meh84] Kurt Mehlhorn. „Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry“. Springer Verlag (1984).

[MH03] Ralf Möller und Volker Haarslev. Description logic systems. In Baader und andere [BCM+03], Kapitel 8.

[Min75] Marvin Minsky. A Framework for Representing Knowledge. In „The psychology of Computer Vision“, Seiten 211–277. McGraw-Hill, New York, NY (1975).

[MMP+95] O. Matz, A. Miller, A. Potthoff, W. Thomas und E. Valkema. Report on the Program AMoRE. Bericht 9507, Christian-Albrechts-Universität Kiel, Kiel, Germany (Oktober 1995).

[Moe01] Ralf Moeller. „Expressive Description Logics: Foundations for Practical Applications“. Habilitation thesis, University of Hamburg, Hamburg, Germany (Juli 2001).

[Neb90] B. Nebel. „Reasoning and Revision in Hybrid Representation Systems“, Band 422. Springer (1990).

[Neb91] B. Nebel. Terminological Cycles: Semantics and Computational Properties. In „Principles of Semantic Networks“. Morgan Kaufmann (1991).

[NLS97] A. Napoli, J. Lieber und A. Simon. A Classification-Based Approach to Case-Based Reasoning. In M.-C. Rousset, R. Brachman, F. Donini, E. Franconi, I. Horrocks und A. Levy (Herausgeber), „Proc. of the 1997 International Workshop on Description Logics (DL97)“, Seiten 104–108, Gyf sur Yvette, France (1997). Université de Paris-Sud, Orsay.

[Nut94] Werner Nutt. „Algorithms for Canstraints in Deduction and Knowledge Representation“. Dissertation, Universität des Saarlandes, Saarbrücken, Germany (1994).

[Pep96a] Jeff Pepper. The Knowledge-Based Support Center (1996).

[Pep96b] Jeff Pepper. The Value of Online Knowledge in Customer Support Applications (1996).

[Pla86] Rob van der Plas. „Fahrradreparaturen“. Falken-Verlag, Niedernhausen/Ts., Germany (1986).

[Pla95] Enric Plaza. Cases as terms: A feature term approach to the structured representation of cases. In M. Veloso und A. Aamodt (Herausgeber), „Case-based Reasoning: Research and Development. – Proc. of the 1st International Conference ICCBR95“, Band 1010, Seiten 265–276, Sesimbra, Portugal (1995). Springer Verlag, Berlin Heidelberg.

[PSMB+] Peter Patel-Schneider, Deborah L. McGuiness, Ron J. Brachman, Lorin A. Resnick und Alexander Borgida. The CLASSIC Knowledge Representation System: Guiding Principles and Implementation Rational. *SIGART Bulletin* **2**(3), 1108–1113.

[PSS93] P. F. Patel-Schneider und B. Swartout. Description Logic Specification from the KRSS Effort. (November 1993).

[Qui66] M. Ross Quillian. „Semantic Memory“. Dissertation, Carnegie Institute of Technology, Pittsburgh, PA (1966). BBN Report AFCRL-66-189.

[RBP+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy und William Lorensen. „Object-Oriented Modeling and Design“. Prentice-Hall, Englewood Cliffs, NJ (1991).

[Reu76] F. Reuleaux. „The Kinematics of machinery - outlines of a theory of machines“. Macmillan & Co, New York, NY (1876).

[Ric78] Michael M. Richter. „Logikkalküle“, Band 43 aus „Leitfäden der angewandten Mathematik und Mechanik“. B.G. Teubner, Stuttgart, Germany (1978).

[Ric81] Michael M. Richter. Universelle Algebra. Schriften zur Informatik und Angewandten Mathematik 74, RWTH Aachen, Aachen, Germany (August 1981).

[Ric95] Michael M. Richter. The Knowledge Contained in Similarity Measures. Invited Talk (1995).

[Ric03] Michael M. Richter. Fallbasiertes Schließen. *Informatik Spektrum* **26**(3), 180 – 190 (Juni 2003).

[Rie03] Joachim H. Rieger. Voronoi Diagrams of Real Algebraic Sets. *Geometriae Dedicata* **98**(1), 81 – 94 (April 2003).

[Rij79] C. J. van Rijsbergen. „Information Retrieval“. Butterworth-Heinemann, London, 2 Auflage (1979).

[Rob81] D. Robson. Object-Oriented Software Systems. *Byte* (August 1981).

[RS89] C.K. Riesbeck und R.C. Schank. „Inside Case-Based Reasoning“. Lawrence Erlbaum Associates, Hillsdale, NJ (1989).

[RSS81] Michael M. Richter, Barbara Schlösser und Dirk Schwarz. Modelltheorie. Schriften zur Informatik und Angewandten Mathematik 73, RWTH Aachen, Aachen, Germany (August 1981).

[SCF98] Sylvie Salotti, Pascal Coupey und Christophe Fouqueré. Formalizing partial matching and similarity in cbr with a description logic. *Applied Artificial Intelligence Journal* **12**(1), 71 – 112 (1998).

[Sch77] Roger C. Schank. „Scripts,Plans, Goals and Understanding“. Lawrence Erlbaum Associates, Hillsdale, NJ (1977).

[Sch94a] Klaus Schild. Terminological Cycles and the propositional mu-calculus. In „Proc. of the 4th Int. Conf. on the Priinciples of Knowledge Representation ond Reasoning (KR-94)“, Seiten 421–431, Bonn, Germany (1994). Morgan Kaufmann.

[Sch94b] Steffen Schäfer. „Objektorientierte Entwurfsmethoden“. Addison-Wesley (Deutschland), Bonn, Germany (1994).

[Sch95] Uwe Schöning. „Logik für Informatiker“. Spektrum Lehrbuch. Spektrum Akademischer Verlag, Heidelberg, Germany, 4 Auflage (1995).

[SSS91] M. Schmidt-Schauss und G. Smolka. Attributive concept descriptions with complements. *AI* **47** (1991).

[SV98] Sylvie Salotti und Véronique Ventos. Study and formalization of a case-based reasoning system using a description logic. In Barry Smyth und Padraig Cunningham (Herausgeber), „Advances in Case-Based Reasoning, Proceedings of the 4^{th} European Workshop (EWCBR-98)“, Band 1488 aus „Lecture Notes in Computer Science“, Seiten 286–297, Dublin, Ireland (September 1998). Springer Verlag.

[Tar19] A. Tarski. A Decision Method for Elementary Algebra and Geometry. In „Collected Works of A. Tarski“, Band 3, Seiten 300–364. (19).

[Tar51] Alfred Tarski. „A Decision Method for Elementary Algebra and Geometry“. University of California Press, Berkeley and Los Angeles, CA (1951).

[TMR71] Alfred Tarski, Andrzej Mostowski und Raphael M. Robinson. „Undecidable Theories“. Sudies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, Netherlands, 1 Auflage (1971).

[Tve77] A. Tversky. Features of Similarity. *Psychological Review* **84**, 327–362 (1977).

[Wei96] Volker Weispfenning. Applying Quantifier Elimination to Problems in Simulation and Optimization. Bericht MIP-9607, Universität Passau, Fakultät für Mathematik und Informatik, Passau, Germany (April 1996).

[Weß95] Stefan Wess. „Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik“. Dissertation, Universität Kaiserslautern, Kaiserslautern, Germany (1995).

[Weß96] Stefan Wess. Intelligente Systeme für den Customer-Support. *Wirtschaftsinformatik* **38**(1), 23–31 (1996).

[Woo75] W. A. Woods. What's in a Link: Foundations for Semantik Networks. In „Representation and Understanding: Studies in Cognitive Science“, Seiten 35–82. Academic Press, New York, NY (1975).

Index