Natural Language Parsing with Graded Constraints

Dissertation
zur Erlangung des Doktorgrades
am Fachbereich Informatik der Universität Hamburg

vorgelegt von
Ingo Schröder
aus Uelzen

Hamburg im Jahr 2002

Genehmigt vom Fachbereich Informatik der Universität Hamburg auf Antrag von

Betreuer:	Prof. Dr.-Ing. Wolfgang Menzel Fachbereich Informatik Universität Hamburg
2. Gutachter:	Prof. Dr. Christopher Habel Fachbereich Informatik Universität Hamburg
Externe Gutachterin:	Prof. Mary P. Harper, Ph.D. School of Electrical and Computer Engineering Purdue University, IN, USA

Hamburg, den 15. April 2002 Dekan Prof. Dr. Siegfried Stiehl

Dedicated to my mother
Ursula Schröder

Abstract

Based on the working hypothesis that gradation is a central phenomenon in natural language, this thesis proposes the grammar formalism of weighted constraint dependency grammars (WCDG) as a unified framework for the treatment of gradation.

Gradation can be found in different forms in natural language: as a preference for one outcome over another or as an extended notion of grammaticality which accepts not only 'well-formed' sentences but also deviant structures if no other analysis is feasible. It also occurs in inherently uncertain information such as that from automatic recognition of speech or blurred hand-writing. WCDGs not only offer a well-defined formalism to concisely express what the underlying constraints of natural language understanding are but additionally allow one to state which status a constraint has, be it a strict rule which describes what structures are possibly imaginable or be it a slight preference which only selects the preferred one among a set of ambiguous analyses for a sentence.

The formalism of WCDGs is formally defined both syntactically by specifying the constraint language and semantically by providing a concise mapping from the WCDG parsing problem to the well-known class of constraint satisfaction optimization problems. The set of languages generated by WCDGs have been ranked between the mildly context-sensitive and the context-sensitive languages; the word problem has been identified as being $\mathcal{NP}$-complete.

WCDGs are shown to be appropriate to be used for modeling a large variety of linguistic phenomena such as immediate dominance, agreement, valence, aspects of word order, projectivity as well as semantic aspects such as thematic roles.

A spectrum of algorithms that solve the WCDG parsing problem can be devised from which an application can select the most appropriate method for the task at hand. Algorithms exist which are complete and sound; others approximate the correct solution but compare favorably with respect to temporal characteristics.

A large set of experimental results confirms that WCDGs are well-suited for handling gradation in natural language:

- Graded constraints facilitate a robust mode of parsing so that even utterances with considerable deviations can be analyzed.
- The inherent robustness and the availability of characterizations of constraint conflicts make the WCDG parser a suitable candidate for a diagnosis component in applications for computer-assisted language learning.
- The WCDG approach is capable of trading runtime against solution quality.

- External graded assessments of linguistic structures such as those originating from a speech recognizer can be integrated into the grammatical arbitration mechanism.
- Constraint weights can be learned from annotated corpora, thus mitigating the grammar acquisition problem.

WCDGs have been found to offer solutions for a series of problems in natural language processing such as robustness, integration of preferences, information fusion and explanatory clarity. So far WCDGs have been studied in an academic context but realistic applications begin to appear on the horizon.

Zusammenfassung

Ausgehend von der Arbeitshypothese, daß Gradiertheit ein zentrales Phänomen natürlicher Sprache ist, schlägt diese Arbeit den Grammatikformalismus der gewichteten Constraint-Dependenzgrammatiken (weighted constraint dependency grammar, WCDG) als einen universellen Rahmen für die Behandlung von Gradiertheit vor.

Gradiertheit kommt in verschiedenen Formen in natürlicher Sprache vor: als Präferenz für eine Variante gegenüber einer anderen oder als erweiterter Grammatikalitätsbegriff, welcher nicht nur die 'wohlgeformten' Sätze akzeptiert, sondern auch abweichende Strukturen, wenn sonst keine Lösung gefunden werden kann. Gradiertheit begegnet uns auch als inhärent unsichere Information wie zum Beispiel beim automatischen Erkennen von Sprache oder undeutlichen Handschriften. Gewichtete Constraint-Dependenzgrammatiken stellen nicht nur einen präzisen Formalismus dar, in dem die dem Sprachverstehen zugrunde liegenden Bedingungen beschrieben werden können, sondern erlauben auch, den Status einer Bedingung auszudrücken: Sei es eine starre Regel, die beschreibt, was überhaupt als vorstellbar zu gelten hat, oder als schwache Präferenz, die aus einer Menge von ambigen Analysen die bevorzugte auswählt.

Der Formalimus der WCDG wird formal definiert, sowohl syntaktisch durch die Spezifikation der Constraint-Sprache als auch semantisch durch eine präzise Abbildung des WCDG-Parsingproblems auf die bekannten Constraint-Optimierungsprobleme. Die Menge der durch WCDG generierten Sprachen kann zwischen der der mild kontextsensitiven und der der kontextsensitiven eingeordnet werden. Das Wortproblem für WCDG ist als $\mathcal{NP}$-vollständiges Problem erkannt worden.

Es wird gezeigt, daß WCDG für die Modellierung einer großen Menge von linguistischen Phänomen geeignet ist: unmittelbare Dominanz, Kongruenz, Valenz, Aspekte von Wortstellung, Projektivität und semantische Aspekte wie thematische Rollen.

Ein Spektrum von Algorithmen zum Lösen des WCDG-Parsingproblems wird angegeben, aus denen eine Anwendung diejenige Methode auswählen kann, die am geeignetsten erscheint. Neben vollständigen und korrekten Algorithmen existieren auch solche, die eine Lösung nur approximativ berechnen, dafür aber ein besseres Zeitverhalten zeigen.

Eine Anzahl von experimentellen Ergebnissen belegt, daß sich die WCDG gut für die Behandlung von Gradiertheit in natürlicher Sprache eignen:

- Durch den Einsatz von gewichteten Constraints wird robustes Parsing ermöglicht, so daß sogar Äußerungen mit erheblichen Störungen analysiert werden können.

- Die inhärente Robustheit und die Verfügbarkeit von Informationen über Constraint-Verletzungen ermöglichen den Einsatz des WCDG-Parsers als Diagnosekomponente in Anwendungen für das computergestützte Erlernen von Fremdsprachen.
- Der Ansatz erlaubt eine Verringerung der Laufzeit zu Lasten der Lösungsqualität (und umgekehrt).
- Externe gewichtete Bewertungen von linguistischen Strukturen wie zum Beispiel diejenigen aus einem automatischen Spracherkenner können in das grammatische Bedingungssystem integriert werden.
- Die Constraint-Gewichte können anhand eines annotierten Korpus gelernt werden, so daß der Aufwand der Grammatikerstellung gemindert wird.

Es wurde festgestellt, daß die gewichteten Constraint-Dependenzgrammatiken Lösungen für eine Reihe von Problemen der automatischen Verarbeitung von natürlicher Sprache anzubieten haben: robustes Parsing, Integration präferenziellen Wissens, Informationsverschmelzung und Erklärungsfähigkeit. Bisher wurden WCDG im akademischen Rahmen untersucht, aber realistische Anwendungen zeichnen sich bereits am Horizont ab.

Acknowledgments

I am grateful to many people who were directly or indirectly involved in the preparation of this thesis. In particular, I want to thank

- Wolfgang Menzel, for providing a stimulating working environment and for innumerable discussions from which this thesis has greatly benefited;
- Kilian Foth, Horia F. Pop and Michael Schulz, my colleagues in the DAWAI project, for the cool times together, for endless discussions and for their work on the WCDG parsing system;
- Dietmar Fünning, Jochen Hagenström, Stefan Hamerich and Andrzey Walczak, my student workers in the DAWAI project, for spending hours and hours improving the WCDG parsing system;
- Mary Harper and the speech group at Purdue University, for letting me stay with them in 1999 and for several fruitful discussions about the nature of CDG;
- Johannes Heinecke, Jürgen Kunze and Andreas Nolda of the Humboldt Universität zu Berlin, for the good time while we were working together in the first phase of the DAWAI project and for linguistic advice;
- my colleagues at the natural language division of the computer science department of the University of Hamburg, for the professional and enjoyable working atmosphere;
- Daniel Herron and Soenke Ziesche, for proofreading earlier versions of this thesis;
- Ulrike Meerstein, for proofreading, continuous support and encouragement throughout the work on this thesis.

Part of the research reported in this thesis was conducted while the author was employed in the DAWAI project 'Parsing of spoken language with limited resources'[1] that was funded by the German Science Foundation 'Deutsche Forschungsgemeinschaft (DFG)' under grant no. Me 1472/1-1 and Me 1472/1-2.

[1]See `http://nats-www.informatik.uni-hamburg.de/~dawai/` for further information.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis proposes the grammar formalism of weighted constraint dependency grammars (WCDG) as a unified framework for the treatment of gradation in natural language.

Gradation is first of all a phenomenon in natural language data: Not all sentences fit into the binary distinction of right vs. wrong; some are simply slightly better than others without the latter having to be completely rejected. Gradation also concerns structure, i.e., one reading of an utterance can be preferred over another one, be it by conscious reasoning or by perceiving only one of the possibilities in the first place. Gradation is both an intra-personal and an inter-personal phenomenon, i.e., one and the same speaker may argue about graded assessments, or preferences can be due to statistical distributions over a population of speakers. And finally, gradation is also found in classification tasks when the mapping between input and output is inherently uncertain, e.g., in speech recognition.

This introduction presents examples of gradation in natural language and motivates a framework for natural language processing that heavily relies on the concept of graded information. Subsequently the major claims of this work are presented and this thesis is related to prior work. Finally an outline of this thesis is given.

1.1 Gradation in natural language

Gradation is ubiquitous in natural language.

Gradation refers to fine-grained assessments of natural language or to degrees of certainty in classification tasks. A graded judgment can come in different forms, e.g., as a ranking (*This sentence variant is better than that one.*), by analogy with a different scale, such as length of lines (*This sentence is as good as the third line is long.*), or as a numerical score on a pre-defined scale (*This sentence gets a score of 3.4/a probability of .23.*). Both utterances alone and linguistic structures, i.e., analyses of the underlying hierarchical composition of sentences, can be judged by using a graded scale.

Chomsky [1965] distinguishes between the *competence system*, i.e., an ideal speaker-hearer's knowledge of his language, and the *performance system*, i.e., the actual use of the language including all processing difficulties. A sentence is considered *grammatical* if it belongs to the language of the grammar according to the linguistic knowledge, i.e., *grammaticality* is defined in terms of the competence system. Traditionally, theoretical linguistics deals with this aspect of language. *Acceptability* of a sentence additionally depends on the performance system of language, i.e., an utterance is acceptable if an actual native speaker-hearer judges it (spontaneously) as belonging to his language. Performance issues are investigated in, for instance, psycholinguistics and cognitive science.

Since the performance system has to respect restrictions of the processing system, such as memory limitations, and since it should explain *all* (spontaneous) utterances and judgments of utterances, it must include a notion of fuzziness or probability. Hence gradation has always been accepted for this area.[1]

But even for the competence system, gradation has been acknowledged quite early:[2]

> "... an adequate linguistic theory will have to recognize degrees of grammaticalness."
> —— Chomsky [1975, p. 131]

It even has been identified as a central part of the competence system:

> "Die sprachliche Kompetenz äußert sich u. a. in folgendem. Der kompetente Sprecher kann
>
> [...]
>
> (e) Grade der sprachlichen Abweichung unterscheiden, wie sie in zunehmendem Maße in den folgenden Beispielen vorliegt:
>
> [...]" —— Grewendorf, Hamm & Sternefeld [1987, pp. 32-33]

Contrary to intuition, grammatical sentences are not automatically acceptable [Sternefeld 1998]. Repeated self embeddings (cf. Example E-1)[3] are often unacceptable but nevertheless grammatical. Example E-2 is incomprehensible because it misleads the reader in a way similar to garden path sentences.

E-1	Das	[der	[dem	Menschen	wichtigen]	Gesundheit]	schädliche	Rauchen...
	The	[the	[the	human	important]	health]	harmful	smoking...

'Smoking being harmful to the health that is important to humans...'

E-2 Vorschriften$_{\text{nom, }\textbf{dat}}$, die$_{\text{nom}}$ Dich stören, solltest Du Dich widersetzen.
'You should defy regulations which perturb you.'

While ungrammatical utterances are *produced* very frequently in spontaneous speech (and thus must be dealt with in a natural language analysis system), it is harder to find ungrammatical sentences that are systematically judged acceptable. Sternefeld [1998] discusses this case and gives Example E-3 on the next page which lacks the complementizer *daß*.

[1]Another reason why graded information is more common for performance is that experimental data, such as reading time, is not binary per se so that a graded interpretation seems more natural.

[2]Cited after Keller [2000b, p. 18]. Although published in 1975, a manuscript of 'The Logical Structure of Linguistic Theory' [Chomsky 1975] has been widely circulated since 1955.

[3]Examples E-1, E-2 and E-3 on the next page are taken from Sternefeld [1998].

E-3 Ein Satz, den zu lesen Günther glaubt den Peter erfreuen wird, steht auf Seite 428.

While the competence vs. performance dichotomy attracts a lot of attention from a theoretical point of view, it is of only minor relevance to the enterprise of building practical systems for natural language analysis. Ultimately, it is irrelevant to such a system whether a particular preference is justified by the competence system or (only) by the performance system. Linguistic input should be processed in the best way possible. This argument holds in an even broader sense. Pragmatically, one can demand that even distorted sentences that would be judged as both ungrammatical and unacceptable should in fact be included in a system of graded assessment since even those sentences might contribute something to the analysis, possibly in the form of a partial analysis.

For the purpose of this thesis, we therefore assume gradation not to be restricted to competence or performance but to include many more aspects where weighted information proves helpful.

In this broad sense gradation can be found in quite a variety of different phenomena in natural language processing,[4] as can be seen from the examples in the following subsections.

A central characteristic of graded assessments is that they potentially conflict with each other. More than one preference might be triggered in a sentence variant, some of which favor one outcome of the analysis and others another one. This potential for conflicts is deeply rooted in our human language system, as will also be demonstrated by the examples.

Deviation from impeccable sentences is not always undesirable. Speakers deliberately produce utterances that violate certain preferences in order to attract the attention of a hearer, to emphasize certain aspects or to signal the forthcoming end of their contribution to the dialogue. These communicative functions are often achieved by means of not fulfilling the expectations of the hearer. In such cases the conflict not only has to be arbitrated but the conflict itself contributes to the interpretation of the sentence.

1.1.1 Grammaticality, acceptability and beyond

The following examples[5] show that language does not in general adhere to strict criteria. We simply note that all these examples *actually do occur* in natural language and conclude that a robust system for natural language analysis should deal with them in some way.

- The following examples are taken from Keller [2000b, pp. 51-52]. The selection of the auxiliary verb in German passive formation (*haben*/*have* or *sein*/*be*) depends (at least) on the classification of the verb with respect to a semantic hierarchy (change of location, change of state, continuation of state, existence of state, uncontrolled process, motional controlled process and non-motional controlled process), on telecity, on animacy of the subject and on dialect.

[4]We only consider natural language analysis in this thesis. However, we feel that based on similar reasoning it makes sense to argue for a graded approach to natural language production as well.

[5]We omitted the usual acceptability marks '*' and '?' from our own examples since they reflect the duality of grammaticality which is in opposition to our approach. However, when citing an example we also quote the acceptability marks as they appear in the source.

E-4	Die	Frau	?ist/hat	etwas	getorkelt.	
	The	woman	is/has	a-bit	tottered.	
	‘The woman tottered a bit.’					
E-5	Die	Frau	*ist/hat	in	der	Wohnung	getorkelt.
	The	woman	is/has	in	the	flat	tottered.
	‘The woman tottered in the flat.’						
E-6	Die	Frau	ist/*hat	in	die	Wohnung	getorkelt.
	The	woman	is/has	into	the	flat	tottered.
	‘The woman tottered into the flat.’						

Both variants in Example E-4 are acceptable, although *hat* seems to be the preferred option. The detelic variant in Example E-5 prefers *hat* even more while the telic alternative in Example E-6 highly prefers *ist*. However, different native speakers of German prefer different variants; only for some the dispreferred variant is acceptable.

- Genuine grammatical errors are most prominent among graded phenomena. Different deviations have to be weighted against each other, but most importantly they have to be analyzed in the first place. To do so one has to balance out a huge variety of errors including but not limited to spelling errors (Example E-7), punctuation mistakes (Example E-8),[6] use of adjectives instead of adverbs (Example E-9), violated agreement (Example E-10), omission of words (Example E-11), neglect of auxiliary verb (Example E-12), use of comparatives for absolute adjectives (Example E-13) and many more.

E-7 The enviroment around the libary (accept the hi bildings) is exceptable.

E-8 This birthday card say’s everything...

E-9 (How are you?) Thanks, I’m good.

E-10 Physics are my favorite subject.
The whereabouts of the stolen diamond are unknown.

E-11 Incomplete sentences not acceptable.

E-12 Where played you football?

E-13 Clara is more pregnant than last month.

- The acceptability of sentences with topicalized constituents depends on the syntactic structure as well as on the communicative function.

E-14 You’ve been telling me nice things!

E-15 Nice things you’ve been telling me!

E-16 Me you’ve been telling nice things!

- Metaphorical use of language often violates selectional restrictions but helps to concisely express semantic parallels.

[6]Although good examples for real world deviations from ideal language use, spelling or punctuation errors will not be dealt with in this work. However, incorporating these aspects into the proposed framework is feasible, e.g., by adding spelling variants of words with associated penalties proportional to their Levenstein distance. We do feel that real-world applications can greatly benefit from such small improvements that are often considered scientifically uninteresting.

E-17 The car uses up all my money.

E-18 The car eats up all my money.

E-19 The car eats gasoline.

E-20 The car eats all my money.

E-21 The car died.

- The separated prefix of a German discontinuous verb usually follows all other components in a declarative sentence. On the other hand the attempt to keep (semantically) related parts closely together can lead to extraposed constituents.

E-22	Das	hängt		von	Dingen		, die	ich	nicht	verantworte,	**ab**.
	Das	hängt		von	Dingen	**ab**	, die	ich	nicht	verantworte.	
	Das	hängt	**ab**	von	Dingen		, die	ich	nicht	verantworte.	
	That	depends		from	things		that	I	not	account for	

‘That depends on circumstances I do not account for.’

- Sometimes it is more understandable to use a different gender than the grammatical one in pronominal reference (Examples E-23 and E-24[7]). Similarly a plural pronoun is used to refer to a singular antecedent in Example E-25.

E-23	Das	Mädchen$_{\text{neut}}$	sucht	nach	seiner$_{\text{neut}}$	Mutter.
	Das	Mädchen$_{\text{neut}}$	sucht	nach	ihrer$_{\text{fem}}$	Mutter.

‘The girl looks for her mother.’

E-24	Alfo	hielt	die	Zahnbürste$_{\text{fem}}$,	als	wäre	sie$_{\text{fem}}$	eine	Waffe.
	Alfo	hielt	die	Zahnbürste$_{\text{fem}}$,	als	wäre	es$_{\text{neut}}$	eine	Waffe.

‘Alfo held the toothbrush as if it were a weapon.’

E-25 Everyone$_{sg}$ has their$_{pl}$ own role to play.

1.1.2 Preference

Another class of graded assessment is preferences. Preferences simply rank sentences or readings of truly ambiguous sentences but do not distinguish between grammatical and ungrammatical in the traditional sense. Typically, there is no clear-cut division between grammatical constraints and preferences; the topicalization examples above can either be classified as grammatical conditions or as preferences.

- The subject is preferably placed before the object in German although the inverse word order is also perfectly grammatical.

E-26	Die	Frau$_{\text{nom,acc}}$	sieht	die	Tochter$_{\text{nom,acc}}$.
	The	woman	sees	the	daughter.
E-27	Die	Frau$_{\text{nom,acc}}$	sieht	der	Mann$_{\text{nom}}$.
	The	woman	sees	the	man.

‘The man sees the woman.’

[7]Example E-24 has been adapted from Thiel [1987].

E-28 (Wen sieht die Tochter?)
(Whom does the daughter see?)

Die	$\text{Frau}_{\text{nom,acc}}$	sieht	die	$\text{Tochter}_{\text{nom,acc}}$.
The	woman	sees	the	daughter.

'The woman, the daughter sees.'

E-29 Osama Bin Laden und 21 weitere Anführer hätten die eingekesselten al-Qaida-Kämpfer ausliefern sollen, um frei abziehen zu dürfen.

Example E-26 on the preceding page is syntactically ambiguous with regard to the subject and object functions. Nevertheless the reading with *Frau* as subject is preferred as long as no counter-evidence is available as in Example E-27 on the page before (morpho-syntax) and Example E-28 (information structure and optional prosody). Example E-29 is a similar, but real-world example[8] with inverted subject-object word order; in this case world knowledge is needed for the correct analysis.

- Ceteris paribus, a referential pronoun is preferably resolved to the nearest antecedent. Of course, even very long distances between antecedent and pronoun are allowed and actually do occur in natural language data.

E-30 Peter_i hates his boss_j since he_j likes swimming.
E-31 Peter_i hates his boss_j and he_i likes swimming.
E-32 Peter_i hates his boss_j since he_i dislikes authorities.

In Example E-30 one might have a slight tendency to associate *he* with *boss* because the distance to *Peter* is longer and no further evidence enforces the longer reference resolution. As soon as additional evidence is available, however, the longer dependency is preferred, for instance, in the case of syntactically parallel constructions (Example E-31) or semantic hints (Example E-32). Again different kinds of graded evidence have to be weighted against each other.

1.1.3 Combining different levels of analysis

The human language system is often divided into different subsystems.[9] Syntax and semantics are the most prominent ones. Not only do we find graded assessments on each of these levels but the relation between them can be graded as well. The relative 'strength' of a level depends on a variety of conditions and, of course, the lack of good evidence on other levels.

- Chomsky's well-known Example E-33 [Chomsky 1957, p. 15] uses valid lexical items, is syntactically well-formed but contains semantic anomalies. Semantics should probably not play a prominent role during its analysis since almost no (meaningful) semantic evidence is available.

E-33 Colorless green ideas sleep furiously.

[8]Taken from Spiegel Online 2001/12/13 `http://www.spiegel.de/politik/ausland/0,1518,172454,00.html`.
[9]See, for instance, the arguments in Forster [1987], Marslen-Wilson & Tyler [1987], Fodor [1987] and Frazier [1987].

- Example E-34 is a more realistic example which is syntactically distorted but nevertheless understandable. The semantic hints suffice to construct a meaning. In Example E-35, selectional restrictions overrule word order preferences.

  E-34 We cinema tonight going.

  E-35 Steak eat lumberjack.

- The poem Jabberwocky by Carroll [1872] is an extreme case where most of the lexical items (except function words) are non-words and therefore almost no semantics is available. Nevertheless people are able (at least partially) to assign a syntactic structure and even assume a very vague and underspecified meaning.

  E-36 'Twas brillig, and the slithy toves
  Did gyre and gimble in the wabe:
  All mimsy were the borogoves,
  And the mome raths outgrabe.
  —— Carroll [1872, Jabberwocky]

- Sometimes one level is needed to resolve ambiguities on a different level. The syntactic attachment of prepositional phrases, such as in Examples E-37 and E-38, is largely influenced by the verb's and the noun's degree of affinity to the prepositional phrase [Hindle & Rooth 1993].

  E-37 Don't start$_V$ a discussion$_N$ [with customers]$_{PP \rightarrow N}$.

  E-38 Please help$_V$ the customer$_N$ [with the decision]$_{PP \rightarrow V}$.

1.1.4 Uncertain information

In the previous section we looked at the relative strength of different linguistic levels. This section considers a related issue, the integration of inherently uncertain information from (possibly extra-linguistic) sources outside of the analysis system.

- If we understand natural language analysis as syntactic parsing and semantic disambiguation, the recognition uncertainty stemming from a speech analyzer can be handled as graded assessment of word hypotheses. See Section 5.6 for more details.

- Prosodic information, e.g., word stress, is inherently uncertain. Nevertheless the two variants in Example E-39 and E-40 have different meanings.

E-39	Lassen	Sie	uns	*noch*	einen	Termin	ausmachen.
	Let	you	us	yet another/still	an	appointment	schedule.
	'Let's schedule yet another appointment.'						
E-40	Lassen	Sie	uns	noch	einen	*Termin*	ausmachen.
	Let	you	us	yet another/still	an	appointment	schedule.
	'We still have to make an appointment.'						

Although these examples mark an extreme case since the difference in meaning is determined by word stress alone (and sentence context if available), the inclusion of graded prosody information can enrich the analysis process in other ways, e.g., by detecting likely phrase boundaries or by supporting an inverted subject-object word order (cf. Example E-28 on page 6).

- Probabilistic part-of-speech (POS) taggers can accurately determine the likely categories of words. Although a syntactic parser operating on word forms and using a lexicon can disambiguate POS tags independently of a POS tagger, it nevertheless makes sense to incorporate into the parsing system such a probabilistic preprocessor that annotates words with tag probabilities. See Section 5.7 for more details.
- Similar to the POS tagger case, it might be beneficial to use a chunker device which determines likely phrase boundaries in an utterance to help the syntactic parser to determine the syntactic structure more efficiently. Again this evidence should be integrated as graded information because of its uncertain nature.

1.1.5 Summary

The examples from the previous sections are all understandable (at least partially or in specific contexts) and they might occur in actual language data. In all cases some kind of graded information is involved that on the one hand prefers some (and hence also disprefers other) readings and on the other hand provides valuable information about inconsistencies.

The causes of deviation from an ideal can be manifold. This thesis does not always make assumptions about the deeper origin of graded information. We simply note that gradation plays an important role in natural language and conclude that an analysis system can benefit from regarding gradation as a vital part of the parsing system.

1.2 Motivation of gradation in natural language processing

The previous section gave evidence for the validity of the working hypothesis that gradation can be found ubiquitously in natural language. This section is based on this working hypothesis (as is the rest of this thesis) and explicates why a grammatical framework for natural language analysis should not only accept the existence of graded phenomena but instead provide means to deal with gradation as a central concept of natural language understanding.

- If one accepts that gradation plays an important role in human natural language processing, it seems only consistent to model it explicitly also in an artificial system dealing with natural language. Note, however, that the human language processing mechanisms are not our primary concern although the proposed grammar system may be used as a modeling tool in psycholinguistics or cognitive science.
- *Robustness* is the ability to deal with unexpected and possibly erroneous input.

  Some (if not all) examples in the previous section can be classified as ungrammatical with respect to some grammar. However, if we are aiming at a system that can analyze natural language as flexibly as a human being can, we have to provide analyses for these examples, and a process assessing these analyses inevitably has to deal with graded information.

- Graded information can be inconsistent with natural language utterances, and robustness can only be achieved when such violations do not lead to parsing failures. On the other hand graded preferences are valid most of the time and can thus be exploited as valuable information to make the disambiguation task more accurate, more reliable and more efficient.
- Some classes of graded information rank competing analyses for a single utterance. Humans usually only perceive a single analysis, even for truly ambiguous sentences. We are also interested in a *complete disambiguation*, i.e., a single analysis instead of an enumeration of possible structures, because subsequent stages of processing cannot reasonably distinguish among multiple candidates. Hence, graded information must be consulted to make a well-founded decision.

  Note that graded information is used for two contrary purposes: Firstly, the coverage of the grammar is extended so that *more* analyses become possible where fewer can be found using a strict grammar. And secondly, the inclusion of preferences leads to *less* (i.e., completely disambiguated) results at the same time.
- If one wants to include every available piece of information that can help to disambiguate a natural language utterance, some arbitration mechanism must be provided that inevitably has to utilize graded information to break ties in a justified way. In contrast to all previous reasons for graded information, this last one of *information fusion* is mainly extra-linguistically motivated but nevertheless important in practical cases.

Hence it is our firm understanding that no artificial system that neglects the concept of gradation will ever converge to or even reach the outstanding human capabilities of natural language understanding. Traditional linguistics might not be wrong when it idealizes natural language and considers only the competence system of a speaker, but it definitely tackles a different problem.

1.3 Central claims

Starting from the assumption that gradation is one of the cornerstones of natural language processing, the grammar formalism of weighted constraint dependency grammars (WCDG), which is centered around the idea of graded evidence, is proposed and formally defined (cf. Chapter 2). Three main claims are put forward.

1. The proposed grammar formalism is suitable for modeling a non-trivial subset of natural language. See Chapter 3.
2. Algorithms can be devised that are capable of solving the parsing problem for a natural language utterance and for a grammar that has been formulated in the new formalism. Different solution methods exhibit different characteristics, and are therefore suitable for a number of applications with varying requirements. Implementations are feasible and computationally tractable in practical cases. See Chapter 4.

3. A number of attractive properties can be experimentally confirmed for the new grammar formalism. See Chapter 5.

    - Given an appropriate grammar, sentences that would otherwise be declared ungrammatical or uncovered can now be analyzed. This robust behavior can be proven not only for genuine syntactic errors but also for other deviations, such as semantic anomalies. See Section 5.3.
    - The formalism can not only parse deviant input but also allows the location of the faults and the identification of the type of error without further effort. Such diagnostic power is indispensable in applications for computer-aided natural language learning. See Section 5.4.
    - The robustness against errors in language use can be carried over to robustness against processing restrictions, such as time restrictions. See Section 5.5.
    - The formalism is flexible enough to integrate and exploit graded evidence originating from external sources. See Sections 5.6 and 5.7.
    - The parameters of a grammar can be estimated from an annotated corpus either for local clusters of constraints or for a complete grammar. See Sections 5.8 and 5.9.

1.4 Prior and related work

Gradation in natural language has many different aspects, some of which have been studied in different research areas such as theoretical linguistics, psycholinguistics, computational linguistics, artificial intelligence, cognitive science and computer science.

The notion of gradation does not play a major role in theoretical linguistics, in particular generative linguistics, although its importance has been recognized early [Chomsky 1964; Chomsky 1965]. Traditionally, linguists and especially syntacticians argue about the binary decision of whether a sentence is grammatical (competence), while deviation in actual language data is explained with generic references to performance aspects. Sells [1987] summarizes this attitude towards gradations in acceptability ratings:

> "Fuzziness of intuitions is noise in the data to the syntactician, and one must simply find ways to deal with it." —— Sells [1987, pp. 6-7]

Optimality theory [Prince & Smolensky 1993, OT] for the first time rests upon the concept of possibly conflicting, violable constraints. The constraints are assumed to be universally valid and only their specific strict orderings in various languages determine the differences between individual languages. Grammaticality is defined in terms of least constraint violation, i.e., a linguistic realization or structure is optimal (and thus grammatical) if it does not violate higher ranked constraints than any other structure. The view is usually a generative one, i.e., different realizations are ranked by constraints but the models do not aim at the structural disambiguation of a single utterance. The focus of optimality theory research is still on crisp grammaticality:

> "[...] eine Struktur ist grammatisch, wenn sie besser als ihre Konkurrenten ist, und ungrammatisch sonst. Feinere Differenzierungen sind nicht vorgesehen."
> —— Fanselow & Féry [forthcomingb, p. 13]

Only lately has gradation in grammaticality attracted more attention [Keller 2000b; Sternefeld 1998, for instance]. This shift has been influenced by new methods of collecting empirical judgment data. In particular the ubiquity of conflicts [Fanselow & Féry forthcomingb] between (and within) linguistic rule systems and linguistic data leads to the introduction of violability of constraints and preference rules. For instance, Keller [2000a] further elaborates on Uszkoreit [1987]'s system of weighted constraints for German word order and presents supportive data that was collected using a new experimental methodology [Keller 2001]. Graded grammaticality slowly finds its way into linguistic research [Fanselow & Féry forthcominga] although the phenomena are still often explained by means of performance models [Schlesewsky forthcoming].

In the meantime, a substantial portion of research in computational linguistics performed a shift away from devices that recognize the grammatical sentences of a language, towards natural language applications that deal with large amounts of actual linguistic data.[10] Driven by the enormous increase of computational power and memory capacity in modern computers, the success stories have mainly taken place in the field of probabilistic grammar research. Different attempts to reconcile the two views [Klavans & Resnik 1996] have been made from varying perspectives but the dualism between Chomskian rationalism and Shannon's empirism [Pereira 2000], theory and application [Menzel 2000] or scientific understanding and approximation [Abney 1996b] still persists.

The technological challenge of dealing with all kinds of language data has been tackled mainly by two means: shallow and partial parsing as well as statistical disambiguation. The former circumvents the tough problems by reducing the depth or the completeness of an analysis. Finite-state techniques [Abney 1990; Abney 1996a; Abney 1997; Karttunen et al. 1997; Karttunen, Gaál & Kempe 1997; Järvinen & Tapanainen 1997] are particularly successful in this area. Statistical approaches [Black et al. 1992; Jelinek 1990; Chelba & Jelinek 1998; Eisner 1996; Collins 1996; Collins 1997; Collins 1999; Charniak 1993; Carroll & Charniak 1992; Magerman 1994; Magerman 1995; Ratnaparkhi 1997a; Ratnaparkhi 1998; Bangalore & Joshi 1999; Joshi & Srinivas 1994] not only overcome the problem of too small a grammatical coverage and the lack of error tolerance because they always yield the 'most plausible' analysis as a result, but they also offer solutions to some of the intrinsic and difficult problems of strict manual grammars: The labor-intensive tasks of acquiring and porting grammars can be avoided as long as a suitable annotated corpus is available from which probabilistic grammars are usually automatically induced. They also facilitate a complete disambiguation by resolving even real ambiguities with preferences (encoded by probabilities).

Two disadvantages can be identified for stochastic approaches (apart from the ubiquitous data sparseness problem). They often come as a closed system, i.e., it is not feasible to integrate external preferences into the probability models, in particular when the external ratings are incomplete, i.e., when they do not cover all cases of the probability model. The second drawback concerns the methodological status of the model. In statistical models, the underlying principles are not explicitly represented but distributed over a large set of numerical values so that they are not accessible for inspection. The individual learned parameters themselves cannot be meaningfully interpreted.[11] Furthermore the problem of language acquisition, i.e., the question

[10]For early accounts to weighted grammars in artificial intelligence see, for instance, Thiel [1987].

[11]Though not strictly belonging to the class of statistical approaches to natural language processing, rule induction systems somewhat are an exception. Error-driven learning of transformation-based rules [Brill 1995; Brill 1996], for instance, yields relatively small sets of rules from annotated examples that are in fact interpretable. Nevertheless, they cannot explain different ratings for different readings.

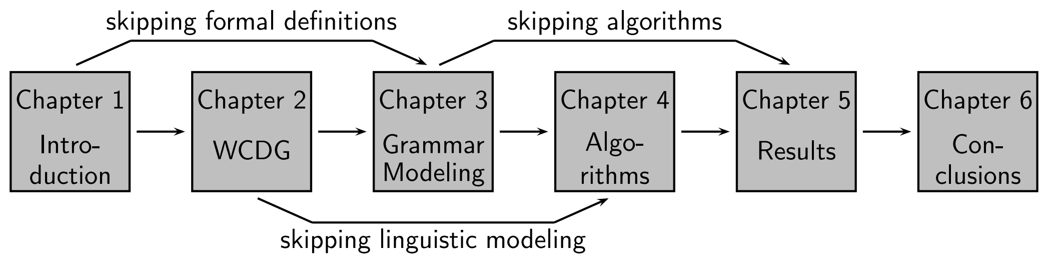


Figure 1.1: Structure of this thesis.

of how a human can learn a language given the typical amount and quality of language data to which he is exposed, is rarely addressed. The holistic explanation of emergent events is shared by neural networks.

The framework proposed in this thesis can in this context be seen as a step towards a grammatical framework which combines the advantages of stochastic models of language and of traditional rule-based grammar systems. We bring together the successful concept of graded assessment with the transparent explanatory power of a rule system. We allow the inclusion of information from heterogeneous knowledge sources, both within the linguistic system, for instance syntax and semantics, and from external sources, for instance acoustical preferences and domain knowledge. Finally, we also point to a way of how free parameters can be estimated from corpora.

1.5 Overview of this thesis

This thesis is structured as follows.

- The introduction featured examples for gradation in natural language that serve as a motivation for a concept of gradation in natural language processing. It also put forward the main claims of the thesis.
- Chapter 2 introduces the formalism of weighted constraint dependency grammars. Formal definitions for both the formalism and the proposed constraint language are given.
- The subsequent Chapter 3 describes the general techniques that were used in a grammar for a non-trivial German corpus. It starts with non-graded grammar fragments and extends the use to graded conditions.
- A set of algorithms capable of solving the parsing problem for weighted constraint dependency grammars is introduced in Chapter 4. A discussion of their advantages and disadvantages as well as their applicability for particular tasks is included.
- A compilation of different experimental results is presented in Chapter 5. Some aspects of the examples from this introduction are revisited and discussed in the light of the proposed grammatical framework for gradation. The section contains extensive experimental data.

- Chapter 6 summarizes the thesis, highlights the achievements and describes possible further lines of research.

Although reading the thesis from cover to cover is the preferred sequence, Figure 1.1 on the preceding page outlines alternative possibilities. The technically uninterested reader can initially skip over the formal definitions in Chapter 2 and the algorithmic part in Chapter 4 and return to these sections later. Alternatively, a linguistically uninterested reader can at first omit the linguistic modeling in Chapter 3.

Chapter 2

Constraint Dependency Grammar

2.1 Overview

This chapter introduces the grammar formalism of weighted constraint dependency grammars (WCDG).

Since the parsing problem for WCDG can be viewed as a (partial) constraint satisfaction problem (CSP), this class of problems is introduced first. Sections 2.3 through 2.5, the main part of this chapter, contain the formal definitions for WCDGs along with illustrative examples. Section 2.7 summarizes differences between WCDGs and CDGs as defined by Maruyama [1990b]; Section 2.8 clarifies complexity issues. The subsequent section then reviews related work and finally a summary of the whole chapter is presented.

2.2 Constraint satisfaction problems

This section introduces constraint satisfaction problems (CSP), both intuitively and formally (as far as it is required for this thesis). Algorithms that are capable of solving these kinds of problems are described in Chapter 4.

2.2.1 Definition

Meseguer [1989] and Kumar [1992] feature introductory overview articles on CSP. However, the formal definitions used here occasionally depart from theirs in order to better fit in the framework of constraint dependency grammars. More recent introductions to CSP can be found in Miguel & Shen [2001b] and Miguel & Shen [2001a].

Definition 2.1 *A* constraint satisfaction problem *(CSP) is represented by a triple* $P = \langle V, D, C \rangle$ *where*

V	*is a finite set of constraint variables* $V = \{v_1, ..., v_n\}$*;*
D	*is a finite set of finite domains* $D = \{D_1, ..., D_n\}$ *corresponding to possible values for the constraint variables* V*. The* degree *of a variable* v_i *is the number of possible values in its domain* D_i*.*
C	*is a finite set of constraints* $c = \langle V_c, D_c\rangle \in C$ *where* $V_c = \langle v_{i_1}, ..., v_{i_k}\rangle$ *is the tuple of constrained variables and* $D_c \subseteq D_{i_1} \times ... \times D_{i_k}$ *is the set of assignments to those variables that are allowed by constraint* c*. The number* k *of variables put in relation is the* arity *of constraint* c*. An alternative to explicitly specifying a separate domain for each variable is to assume a single uniform domain for all variables and restrict the values of individual variables using unary constraints, i.e., constraints of arity one. The* relative satisfiability *of a constraint* $c = \langle V_c, D_c\rangle \in C$ *is the relative frequency of permitted value tuples:* $$s_{CSP}(c) = \frac{|D_c|}{|D_{i_1} \times ... \times D_{i_k}|} \quad \textit{for } V_c = \langle v_{i_1}, ..., v_{i_k}\rangle$$ *The* tightness *of a constraint is the complement of the relative satisfiability. The higher the tightness of a constraint, the more constraining information is expressed by it.*

The maximum degree of all variables is the degree of the CSP and the maximum arity of all constraints is the arity of the CSP. A (partial) assignment of values to constraint variables (also called labeling of constraint variables) is a (partial) function $\alpha : V \mapsto \bigcup_{1\leq i\leq n} D_i$ with $\alpha(v_i) \in D_i$. A solution to a constraint satisfaction problem is a complete assignment α^* that satisfies all constraints:

$$\bigwedge_{\substack{\langle V_c, D_c\rangle \in C \\ V_c = \langle v_{i_1}, ..., v_{i_k}\rangle}} \langle \alpha^*(v_{i_1}), ..., \alpha^*(v_{i_k})\rangle \in D_c$$

A tuple of value assignments $\alpha(v_{i_1}), ..., \alpha(v_{i_k})$ is said to be *inconsistent* or *in conflict* if it fails to fulfill at least one constraint:

$$\bigvee_{\substack{\langle V_c, D_c\rangle \in C \\ V_c = \langle v_{i_1}, ..., v_{i_k}\rangle}} \langle \alpha(v_{i_1}), ..., \alpha(v_{i_k})\rangle \notin D_c$$

The *density* of a problem is only defined for CSP with an arity of two, i.e., binary CSP, and denotes the ratio between the number of actual binary constraints and number of variable pairs:

$$d_{\text{CSP}} = \frac{|\{\langle V_c, D_c\rangle \in C | V_c = \langle v_i, v_j\rangle\}|}{n \cdot (n-1)} \quad \text{where } n \text{ is the number of variables (as above)}$$

There is a tendency that a CSP with a high density is more constrained than one with a low density although more parameters, especially the tightness of the constraints, influence the constrainedness. The *tightness* of a CSP is computed from the corresponding figure of the constraints, usually the mean is taken.

CSPs belong to the set of $\mathcal{NP}$-complete problems [Hopcroft & Ullman 1979, cf. also Section 2.8].

2.2.2 Example

Cryptarithmetic puzzles are typical examples of constraint satisfaction problems. An arithmetic equation is presented with the digits replaced by letters (cf. Figure 2.1a). The task is to assign a decimal digit to each (possibly repeatedly occurring) letter; each digit must not be used more than once.

```
    S E N D          S  E  N  D          9 5 6 7
  + M O R E        + M  O  R  E        + 1 0 8 5
                     C3 C2 C1            0 1 1
  ---------        -------------       ---------
  M O N E Y        M O  N  E  Y        1 0 6 5 2

     (a)               (b)                (c)
```

Figure 2.1: A cryptarithmetic puzzle as a typical CSP problem.

The direct mapping of letters to constraint variables is straight-forward. However, introducing additional constraint variables for the carry bits (`C1`, `C2` and `C3` in Figure 2.1b) facilitates a more convenient constraint formulation. Following that idea, the set of constraint variables is $V = L \cup B$ with $L = \{S, E, N, D, M, O, R, Y\}$ and $B = \{C1, C2, C3\}$. The domain of the variables corresponding to carry bits is $D_B = \{0, 1\}$ and the set of decimal digits $D_L = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ for the rest. The degree of those variables from set L is ten and of those from set B it is two; hence the degree of the CSP is ten.

Specifying constraints by enumerating all allowed value tuples – as suggested by definition 2.1 – is sometimes inconvenient. Instead, constraints are often expressed by means of logical formulae, lists of forbidden value tuples, expressions in a specific constraint language or even code in a programming language. The only restriction that a constraint formulation must fulfill is that the constraint must be efficiently computable and that its meaning must be unambiguous.[1] For the example, mathematical formulae seem to be the most useful for constraint formulation.

The set of constraints for cryptarithmetic puzzles can be divided into two groups:

- **Uniqueness:** No two variables must be assigned the same digit. This constraint is applicable to all instances of cryptarithmetic puzzles.

$$\bigwedge_{\substack{x,y \in L \\ x \neq y}} \alpha(x) \neq \alpha(y)$$

  Note that this constraint is actually not a single constraint but a constraint template. The meta-variables x and y can be bound to any of the variables from set L, i.e., the template stands for $|L| \cdot (|L| - 1) = 56$ constraints (half of which are redundant because the inequality relation is symmetric). The arity of these constraints is two because they constrain the values of two variables; hence they are also called binary constraints.

[1]Constraint evaluations are considered *basic* operations, i.e., a constraint which just states that a value must be such that it is part of a global solution is useless since evaluating that constraint is as hard as solving the overall problem.

- **Arithmetic validity:** These constraints are instance specific since each puzzle imposes different restrictions.

  In order to ease the exact formulation of the constraints we define two functions: modulo and integer division.

$$\begin{aligned} x \bmod y &\overset{\text{def}}{=} (x/y - \lfloor x/y \rfloor) \cdot y \\ x \operatorname{div} y &\overset{\text{def}}{=} \lfloor x/y \rfloor \end{aligned}$$

  For our example, the following constraints are appropriate.

$$\begin{aligned} (\alpha(S) + \alpha(M) + \alpha(C3)) \bmod 10 &= \alpha(O) \\ (\alpha(S) + \alpha(M) + \alpha(C3)) \operatorname{div} 10 &= \alpha(M) \\ (\alpha(E) + \alpha(O) + \alpha(C2)) \bmod 10 &= \alpha(N) \\ (\alpha(E) + \alpha(O) + \alpha(C2)) \operatorname{div} 10 &= \alpha(C3) \\ (\alpha(N) + \alpha(R) + \alpha(C1)) \bmod 10 &= \alpha(E) \\ (\alpha(N) + \alpha(R) + \alpha(C1)) \operatorname{div} 10 &= \alpha(C2) \\ (\alpha(D) + \alpha(E)) \bmod 10 &= \alpha(Y) \\ (\alpha(D) + \alpha(E)) \operatorname{div} 10 &= \alpha(C1) \end{aligned}$$

  These constraints have an arity between three and four; therefore the arity of the CSP is four.

Figure 2.1c on the preceding page shows a solution to the example problem. If the problem is defined as above, there are 24 additional solutions with zero assigned to variable `M`, which is usually forbidden by an additional unary constraint because the variable `M` stands for a leading digit.

2.2.3 Partial constraint satisfaction problems

Real-world requirements often do not allow one to formulate a problem as a CSP or solve it completely.

- Real-world problems are sometimes over-constrained, i.e., there exists no solution that fulfills all constraints.

- It is often desirable to use different kinds of constraints in a problem formulation. While some constraints are absolutely indispensable, others may simply encode properties that a solution preferably but not necessarily has.

- Some problems are inherently intractable, i.e., they cannot be solved completely with current techniques and computational resources during the lifetime of the problem. Other problems could be solved if enough time were available, but external requirements put temporal limits on the process, e.g., a flight schedule restricts the time that is available for luggage sorting on an airport.
- Some problems simply do not need an exact solution but one is willing to settle for a solution that is 'good enough'.

In such situations one can try to find an assignment which at least partially fulfills the requirements. *Partial constraint satisfaction problems* (PCSP) [Freuder & Wallace 1992] formalize the notion of partial satisfaction. The traditional (crisp) CSP can be seen as a special case of the partial constraint satisfaction problem class.

PCSP can be divided [Tsang 1993] into *minimal violation problems* (MVP), where one wants to find a labeling such that a minimum number of constraints is violated, and the *maximal utility problem* (MUP), the solution of which assigns values to a maximum subset of the variables with no constraints violated. Both views are relevant in practice:

- A typical instance of the MVP type is the problem of assigning a number of passengers with different needs and preferences to a number of available flights. While it might be tolerable if some individual wishes cannot be fulfilled, all passengers have to get a flight eventually.
- The decision of which cargo to load on a specific airplane and which freight to leave for a later plane is a similar problem but belongs to the MUP type. There is a strict upper bound for the weight and possibly other strict constraints.

As the example suggests, the classification as MUP or MVP sometimes depends on the problem formulation.

In the context of natural language parsing, a MVP corresponds to assigning a structure to the complete sentence but allows some constraints to be violated. Partial parsing, on the other hand, which uses only a subset of the input words for building a structure, more directly maps to a MUP. However, the decision of which words may be omitted from the solution cannot be properly controlled. Since partial parsing can also be handled reasonably in the context of MVPs,[2] partial constraint satisfaction problems from here on are understood as minimal violation problems, i.e., as the problem of complete value assignment with possibly some constraints violated in some minimal way.

Since PCSP are a special case of optimization in CSP, we postpone the formal definition until Section 2.2.4.

2.2.4 Constraint satisfaction optimization problems

In practice, even PCSP are often not flexible enough because they do not distinguish between more and less important constraints. *Constraint satisfaction optimization problems* (CSOP)

[2]See Section 3.5.4 for a discussion of how partial parsing can be achieved as a minimal violation problem.

incorporate a more fine-grained distinction and therefore overcome this deficiency. PCSP (at least the MVP) are a special case of CSOP where all constraints are of equal importance.

Tsang [1993] calls this kind *stochastic CSP* and postulates a function that maps assignments to fine-grained assessments. However, associating the assessment with the constraints is equivalent to this approach and more convenient here.

Definition 2.2 *A* constraint satisfaction optimization problem *(CSOP) is represented by a tuple* $P = \langle V, D, C, \phi, E, \succ, \top, \bot, \otimes \rangle$ *where*

$\langle V, D, C \rangle$	*is a constraint satisfaction problem as in definition 2.1;*
ϕ	*is a function* $\phi : C \mapsto E$ *that assigns a valuation to each constraint; this is related to the 'importance' of the constraint;*
E	*is a valuation set;*
$\succ$	*is a total order on* E*;*
$\top$	*is the maximum element in* E *with regard to* $\succ$*;*
$\bot$	*is the minimum element in* E *with regard to* $\succ$*;*
$\otimes$	*is a commutative, associative and monotonic (with respect to* $\succ$*) function* $\otimes : E \times E \mapsto E$ *that aggregates elements from the valuation set.*

Let $\bar{C}(\alpha, P)$ be the set of constraints in P that are violated by an assignment α.

$$\bar{C}(\alpha, P) = \{\langle V_c, D_c \rangle \in C | V_c = \langle v_{i_1}, ..., v_{i_k} \rangle, \langle \alpha(v_{i_1}), ..., \alpha(v_{i_k}) \rangle \notin D_c\}$$

Let $\phi_\otimes$ be a generalized assessment function for assignments.

$$\phi_\otimes(\alpha) = \otimes_{c \in \bar{C}(\alpha,P)} \phi(c)$$

The solution of a CSOP is an assignment α^* of values to constraint variables such that the aggregated valuation of all violated constraints is minimal and lower than the top element $\top$ of E.

$$\alpha^* = \min_{\alpha \ \succ} \phi_\otimes(\alpha)$$

Schiex, Fargier & Verfaillie [1995] provide an algebraic framework for different types of constraint-based optimization problems called *valued constraint satisfaction problems* (VCSP).

Table 2.1 presents some instances of the VCSP framework. The classical VCSP corresponds to the traditional crisp CSP: Constraints are all of equal importance and are either completely satisfied or fail altogether. The other instances are relevant in the context of partial constraint satisfaction. Optimality theory [Prince & Smolensky 1993, cf. Section 1.4] uses the lexicographic VCSP instance [Fanselow & Féry forthcominga, Section 5] that suffers from a kind of the so-called 'drowning effect' [Schiex, Fargier & Verfaillie 1995] because one important constraint can overrule any number of weaker constraints (cf. Section 3.5.7). The probabilistic VCSP [Fargier & Lang 1993] understands the constraint score as the probability that the constraint actually

Name	**Notation**	**E**	**Order $\succ$**	$\top$	$\perp$	$\otimes$
Classical	$\wedge$-VCSP	{true, false}	false $\succ$ true	false	true	$\wedge$
Possibilistic	max-VCSP	$\mathbb{N}_\infty$	$>$	$+\infty$	0	max
Lexicographic	lex-VCSP	$\mathbb{N}^*_\infty$	$>^*$	$\{+\infty\}$	$\emptyset$	$\cup$
Additive	Σ-VCSP	$\mathbb{N}_\infty$	$>$	$+\infty$	0	$+$
Probabilistic	Π-VCSP	$[0, 1]$	$>$	0	1	$1-(1-x)(1-y)$
Multiplicative	CDG-VCSP	$[0, 1]$	$<$	0	1	$\cdot$

Table 2.1: Instances of the valued constraint satisfaction problem framework. The information in this table has been compiled from de Givry & Verfaillie [1997], Schiex, Fargier & Verfaillie [1995] and Schiex [1992]. The instance in the last line is used in the WCDG paradigm.

exists. The possibilistic VCSP [Schiex 1992] is to an even greater degree subject to the drowning effect than the lexicographic one and exclusively considers the maximum element among a set of violated constraints.

We propose the new multiplicative CDG-VCSP which is similar to the additive instance but avoids the problems with infinite penalties which are more difficult to handle computationally. Scores taken from the interval $[0, 1]$ are assigned to constraints and the scores of violated constraints are aggregated multiplicatively. An aggregated score of zero means complete failure while a score of one indicates an ideal solution. Although different instances are possible in general, we will assume this kind of CSOP from here on. Section 3.5.7 motivates this choice linguistically and discusses some of the consequences for a grammar writer. In particular, the aggregation function takes into account all constraint violations (not just the most severe one) and also allows several weaker constraints to overrule a stronger one.

It was suggested [de Givry & Verfaillie 1997] that by simplifying a more complex problem to a simpler one (e.g., by modifying the constraint graph or by using a different VCSP scheme) and solving the easier problem first, a lower bound for the original problem can be found. In conjunction with the easily obtainable upper bound, an anytime bounding of the optimum in an optimization problem can be achieved. However, no experimental results are available, neither from de Givry & Verfaillie [1997] nor for our special case.

2.3 WCDG as a grammar formalism

This section formally describes the framework on which the grammar formalism WCDG is based. The formalism is deliberately introduced separately from the parsing algorithms to make clear that the meaning is completely descriptive, i.e., the semantics is formally defined without any reference to operational issues. A grammar writer is therefore conceptually independent of the procedures that are used for parsing. However, a specific grammar style[3] may have consequences

[3]Grammar style may differ in a number of aspects, e.g., number of postulated representational levels, number of function labels, integration of redundant constraining information, rate of the violable constraints, coverage etc.

on other characteristics than pure semantics, such as, for instance, efficiency; Chapter 4 discusses efficiency issues in greater detail.

The upcoming sections will define the key components in the WCDG framework:

- Word graphs which are used as the input to a parsing component and represent utterances.
- Lexica which associate orthographic word forms with additional information.
- Constraints which are used to express (grammatical) knowledge.
- Constraint dependency grammars which glue the items together.
- Structure candidates which represent possible analyses for particular word graphs.

2.3.1 Word graph

Word graphs account for the individual parsing problems. Usually, the grammar and the lexicon are relatively static once the development has been finished while word graphs define new problem instances.

Definition 2.3 *A* word graph *is represented by a set WG of edges of the form $\langle a, z, w, s\rangle$ where $a \in \mathbb{N}$ is the start point (in time) and $z \in \mathbb{N}$ is the end point (with $a < z$) of word form w and $s \in \mathbb{R}^+$ is a score associated with the confidence that the word form has actually been uttered between the start point and the end point. This score is usually an acoustic measure taken from a speech recognizer (cf. Section 5.6). The minimum start point of all edges is the start point and the maximum end point of all edges is the end point of the word graph:* $\min_{WG} = \min\{a|\langle a, z, w, s\rangle \in WG\}$ *and* $\max_{WG} = \max\{z|\langle a, z, w, s\rangle \in WG\}$. *A* path *is a sequence of edges $\langle e_1, e_2, ..., e_n\rangle$ with $e_i = \langle a_i, z_i, w_i, s_i\rangle$ such that adjacent edges in the path are adjacent in the word graph:*

$$\bigwedge_{0\leq i<n} z_i = a_{i+1}$$

A complete path *is a path $\langle e_1, e_2, ..., e_n\rangle$ where $a_0 = \min_{WG}$ and $z_n = \max_{WG}$. We usually require all edges to be connected to both the start and end point of the word graph by a path.*

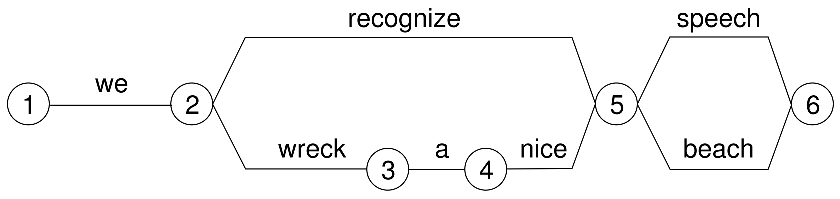
Figure 2.2 on the next page exemplifies the idea of word graphs. Nine edges corresponding to nine word hypotheses are shown that can be combined to four complete paths.

The scores are mentioned for completeness here; they will not become important before the integration of acoustic scores in Section 5.6.

2.3.2 Lexicon

A lexicon associates a word form (an orthographic form in our case) with additional morphosyntactic, syntactic and semantic information.

Definition 2.4 *A* lexicon *is represented by a quadruple $L = \langle W, \lambda, A, V\rangle$ where*



word	we	wreck	a	nice	recognize	speech	beach
start	1	2	3	4	2	5	5
end	2	3	4	5	5	6	6
score	-	-	-	-	-	-	-

Figure 2.2: A simple word graph.

W	*is a finite set of word symbols;*
A	*is a finite set of attribute symbols;*
V	*is a finite set of value symbols. Let AVM_L (attribute-value matrices) be the set of all partial functions $A \mapsto V$;*
λ	*is a function $\lambda : W \mapsto 2^{AVM_L}$ that maps words to a set of attribute-value matrices. For convenience, we assume that the attribute* word *exists, i.e.,* word $\in A$, *and furthermore*

$$\bigwedge_{w \in W} \bigwedge_{\rho \in \lambda(w)} \rho(\mathsf{word}) = w$$

Note that an attribute-value matrix is a flat structure in this definition: Values are symbols, i.e., no recursion is allowed, nor is structure sharing. Therefore, we neither have a type-token distinction nor do we support structure sharing as in standard feature structures [Shieber 1986; Carpenter 1992, for instance]. Figure 2.3a on the following page shows a simple lexical entry l as it is usually depicted at the top, and its formal meaning at the bottom. For convenience, we sometimes write lexical entries as if they contained embedded structures; however, this syntactic sugar can always be resolved to the canonical representation (cf. Figure 2.3b on the next page) as long as no cyclic structures are allowed. Every sequence of attributes is collapsed into a single attribute containing colons.[4]

Negation is not allowed in lexical entries since it would require either a closed world assumption for feature values or an extended feature unification mechanism [Carpenter 1992, for instance]. We nevertheless sometimes use negation in the examples (cf. Figure 2.4 on page 27) when the meaning is clear, e.g., ¬`third-sg` for value combinations of number and person except third person singular.

The function λ is necessary to allow for ambiguous word form like, for instance, *run* as a verb or a noun, i.e., $\lambda(run) = \{run_V, run_N\}$ with $run_V(\mathsf{cat})$ = 'VERB' and $run_N(\mathsf{cat})$ = 'NOUN'. However, we will sometimes simplify the notation and write a subscript that uniquely identifies the correct lexical reading.

[4]Even co-reference between substructures is feasible although the type-token distinction is lost when flattening the structures.

$$\begin{bmatrix}\text{word: `}\textbf{bellen}\text{'}\\ \text{cat: `FINVERB'}\\ \text{number: `pl'}\\ \text{person: `third'}\end{bmatrix}$$

$$\begin{aligned} l(\text{word}) &= \text{`bellen'}\\ l(\text{cat}) &= \text{`FINVERB'}\\ l(\text{number}) &= \text{`pl'}\\ l(\text{person}) &= \text{`third'}\end{aligned}$$

(a) Flat AVM

$$\begin{bmatrix}\text{word: `}\textbf{bellen}\text{'}\\ \text{cat: `FINVERB'}\\ \text{morph: } \begin{bmatrix}\text{number: `pl'}\\ \text{person: `third'}\end{bmatrix}\end{bmatrix}$$

$$\begin{aligned} l(\text{word}) &= \text{`bellen'}\\ l(\text{cat}) &= \text{`FINVERB'}\\ l(\text{morph:number}) &= \text{`pl'}\\ l(\text{morph:person}) &= \text{`third'}\end{aligned}$$

(b) Structured AVM

Figure 2.3: Two lexical entries.

2.3.3 Definition CDG

In this section, we finally define weighted constraint dependency grammar in a formal way.

Definition 2.5 *A* weighted constraint dependency grammar *(WCDG) is represented by a tuple $WCDG = \langle LEV, LAB, \alpha, C, \phi\rangle$ where*

LEV	*is a finite set of level symbols (corresponding to role-ids in the definition of Maruyama [1990a]);*
LAB	*is a finite set of label symbols;*
α	*is a function $\alpha : LEV \mapsto 2^{LAB}$ that assigns a subset of the label symbols to each level symbol. Sometimes, we assume that the subsets are pairwise disjoint*

$$\bigwedge_{l,l' \in LEV} l \neq l' \to \alpha(l) \cap \alpha(l') = \emptyset$$

and partition the set of all labels

$$LAB = \bigcup_{l \in LEV} \alpha(l)$$

so that one can tell the level given the label, e.g., that the `subj` *belongs to the syntactic level.*

C	*is a set of constraints;*
ϕ	*is a function $\phi : C \mapsto [0,1]$ that assigns a penalty factor between zero and one to each constraint.*

This definition departs from the original one by Maruyama [1990b] in two respects. First it assumes that labels are valid for certain levels only and – more importantly – provides a weighting function for the constraints which was first suggested by Menzel [1994].

A second main difference between CDG as Maruyama [1990b] and also Harper & Helzerman [1994] use it and our version of WCDG applies to the mapping from CDG parsing problems to constraint satisfaction problems and is discussed in detail in Section 2.4.

Structure candidates are possible forests of dependency trees, one tree (maybe disconnected) for each level. They represent complete alternative structures that have to be considered when parsing a sentence (see examples in Section 2.3.4).

Definition 2.6 *A* structure candidate *of an input word graph WG given a lexicon* $L = \langle W, \lambda, A, V\rangle$ *and a weighted constraint dependency grammar* $WCDG = \langle LEV, LAB, \alpha, C, \phi\rangle$ *is represented by a triple* $SC = \langle P, \pi, \delta\rangle$ *such that*

P	*is a complete path* $P = \langle e_1, e_2, ..., e_n\rangle$ *in word graph WG. The set of all edges in* P *is* $P_S = \{e_i \vert 0 \leq i \leq n\}$ *and* $P_S^\top = P_S \cup \{\top\}$ *is the set of all edges plus one otherwise unused symbol* $\top$*;*
π	*is a function* $\pi : P_S \mapsto AVM_L$ *such that* $\pi(e) \in \lambda(w)$ *for all* $e = \langle a, z, w, s\rangle$*, i.e.,* π *selects the lexical reading for the edges in the selected path;*
δ	*is a function* $\delta : P_S \times LEV \mapsto LAB \times P_S^\top$ *which assigns structures to the edges in the complete path on all levels. A function value* $\delta(e_i, l) = \langle l', e_j\rangle$ *means that on level* l *the word contained in edge* e_i *is subordinated under the word contained in edge* e_j *with label* l'*. One such mapping represents a dependency edge in a dependency tree.*

The function ξ : WG $\times$ AVM $\times$ LEV $\times$ LAB $\times$ WG $\times$ AVM $\times$ $C \mapsto \{0, 1\}$ is then used as an indicator of whether a dependency edge fulfills a constraint or not.

$$\xi(e_i, \pi(e_i), l, l', e_j, \pi(e_j), c) = \begin{cases} 1 & : \text{ dependency edge } \delta(e_i, l) = \langle l', e_j\rangle \text{ satisfies } c \text{ under } \pi \\ 0 & : \text{ else} \end{cases}$$

The following sets are defined for a dependency edge as the sets of satisfied and violated constraints, respectively.

$$\begin{aligned} \xi_0(e_i, \pi(e_i), l, l', e_j, \pi(e_j)) &= \{c \in C \vert \xi(e_i, \pi(e_i), l, l', e_j, \pi(e_j), c) = 0\} \\ \xi_1(e_i, \pi(e_i), l, l', e_j, \pi(e_j)) &= \{c \in C \vert \xi(e_i, \pi(e_i), l, l', e_j, \pi(e_j), c) = 1\} \end{aligned}$$

These sets are then extended to the multi-sets Ξ_0 and Ξ_1 for complete solution candidates. Note that a constraint might be violated multiple times by a solution candidate; hence we need to use multi-sets here.

$$\begin{aligned} \Xi_0(\text{SC}) &= \bigcup_{\substack{1\leq i\leq n \\ l\in \text{LEV}}} \xi_0(e_i, \pi(e_i), l, l', e_j, \pi(e_j)) \\ \Xi_1(\text{SC}) &= \bigcup_{\substack{1\leq i\leq n \\ l\in \text{LEV}}} \xi_1(e_i, \pi(e_i), l, l', e_j, \pi(e_j)) \\ & \text{for} \quad \text{SC} = \langle P, \pi, \delta\rangle \text{ and } \delta(e_i, l) = \langle l', e_j\rangle \end{aligned}$$

In order to aggregate the weights of all violated constraints into a single number, one has to define an aggregation function as described in Section 2.2.4. As mentioned earlier, we use a simple multiplicative aggregation scheme which results in the following aggregated scoring function:

$$\phi_{\otimes}(\mathrm{SC}) = \prod_{c \in \Xi_0(\mathrm{SC})} \phi(c)$$

Definition 2.7 *A* solution structure *for an input word graph WG given a lexicon* $L = \langle W, \lambda, A, V\rangle$ *and a weighted constraint dependency grammar* $WCDG = \langle LEV, LAB, \alpha, C, \phi\rangle$ *is a structure candidate* $SC^{*} = \langle P, \pi, \delta\rangle$ *that, intuitively, is minimal with respect to constraint violations among all structure candidates. More formally, the solution structure is the solution candidate where the aggregated scoring function is maximal:*

$$SC^{*} = \arg\max_{SC} \phi_{\otimes}(SC)$$

2.3.4 Example

In this section, a complete example including a lexicon, a word graph and a weighted constraint dependency grammar is presented to illustrate the ideas behind the definitions. Assume that the lexicon from Figure 2.4 on the next page is given which provides a subset of lexical entries for the words occurring in Figure 2.2 on page 23.

Two possible structure candidates for the word graph of Figure 2.2 on page 23 are depicted in Figure 2.5 on page 28. Whenever unambiguous, we present the path P implicitly by a sequence of words; otherwise the complete edge information must be provided. The lexical selection (function π) is suggested by the part-of-speech symbols given below the word information; again, if this kind of presentation is not unambiguous, more distinguishing elements or even the full lexicon entry must be provided. The structural information of function δ is depicted as trees: a mapping $\delta(e_i, l) = \langle l', e_j\rangle$ corresponds to an arrow on level l labeled l' from a node belonging to e_i to a node belonging to e_j. An arrow pointing to none of the nodes means a function value containing the special symbol $\top$. In Figure 2.5 on page 28, two levels are shown: SYNTAX and FUNCTOR-ARGUMENT.

The two left trees in Figure 2.5 on page 28 and the following formal notation are equivalent.

$$\begin{aligned}
\delta(e_{\mathsf{we}}, \mathsf{SYNTAX}) &= \langle \mathit{subj}, e_{\mathsf{recognize}}\rangle \\
\delta(e_{\mathsf{recognize}}, \mathsf{SYNTAX}) &= \langle \mathit{none}, \top\rangle \\
\delta(e_{\mathsf{speech}}, \mathsf{SYNTAX}) &= \langle \mathit{obj}, e_{\mathsf{recognize}}\rangle \\
\\
\delta(e_{\mathsf{we}}, \mathsf{FUNCTOR\text{-}ARGUMENT}) &= \langle \mathit{agent}, e_{\mathsf{recognize}}\rangle \\
\delta(e_{\mathsf{recognize}}, \mathsf{FUNCTOR\text{-}ARGUMENT}) &= \langle \mathit{none}, \top\rangle \\
\delta(e_{\mathsf{speech}}, \mathsf{FUNCTOR\text{-}ARGUMENT}) &= \langle \mathit{theme}, e_{\mathsf{recognize}}\rangle
\end{aligned}$$

Note that there exist many more solution candidates than the two that we showed in Figure 2.5 on page 28.

$$\left[\begin{array}{ll} \text{word: ‘\textbf{we}’} & \\ \text{sem: ‘*speaker*’} & \\ \text{syn:} & \left[\begin{array}{ll} \text{cat: ‘PPRO’} & \\ \text{morph:} & \left[\begin{array}{l} \text{number: ‘pl’} \\ \text{person: ‘first’} \\ \text{case: ‘nom’} \end{array}\right] \end{array}\right] \end{array}\right] \quad \left[\begin{array}{ll} \text{word: ‘\textbf{recognize}’} & \\ \text{sem: ‘*recognize*’} & \\ \text{syn:} & \left[\begin{array}{ll} \text{cat: ‘VFIN’} & \\ \text{morph: } \neg & \left[\begin{array}{l} \text{number: ‘sg’} \\ \text{person: ‘third’} \end{array}\right] \\ \text{tense: ‘present’} & \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{word: ‘\textbf{wreck}’} & \\ \text{sem: ‘*damaged object*’} & \\ \text{syn:} & \left[\begin{array}{ll} \text{cat: ‘NOUN’} & \\ \text{morph:} & \left[\begin{array}{l} \text{number: ‘sg’} \\ \text{case: ‘}\neg\text{ gen’} \end{array}\right] \end{array}\right] \end{array}\right] \quad \left[\begin{array}{ll} \text{word: ‘\textbf{wreck}’} & \\ \text{sem: ‘*damage*’} & \\ \text{syn:} & \left[\begin{array}{ll} \text{cat: ‘VFIN’} & \\ \text{morph: } \neg & \left[\begin{array}{l} \text{number: ‘sg’} \\ \text{person: ‘third’} \end{array}\right] \\ \text{tense: ‘present’} & \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{word: ‘\textbf{a}’} & \\ \text{sem: ‘*indefinite*’} & \\ \text{syn:} & \left[\begin{array}{l} \text{cat: ‘INDEF’} \\ \text{morph: } \left[\text{number: ‘sg’}\right] \end{array}\right] \end{array}\right] \quad \left[\begin{array}{l} \text{word: ‘\textbf{nice}’} \\ \text{sem: ‘*beautiful*’} \\ \text{syn:: } \left[\text{cat: ‘ADJ’}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \text{word: ‘\textbf{beach}’} & \\ \text{sem: ‘*sandy waterfront*’} & \\ \text{syn:} & \left[\begin{array}{ll} \text{cat: ‘NOUN’} & \\ \text{morph:} & \left[\begin{array}{l} \text{number: ‘sg’} \\ \text{case: ‘}\neg\text{ gen’} \end{array}\right] \end{array}\right] \end{array}\right] \quad \left[\begin{array}{ll} \text{word: ‘\textbf{speech}’} & \\ \text{sem: ‘*spoken language*’} & \\ \text{syn:} & \left[\begin{array}{ll} \text{cat: ‘NOUN’} & \\ \text{morph:} & \left[\begin{array}{l} \text{number: ‘sg’} \\ \text{case: ‘}\neg\text{ gen’} \end{array}\right] \end{array}\right] \end{array}\right]$$

Figure 2.4: An example lexicon.

The affinity of the verb *wreck* to the direct object *beach* in a neutral context is probably smaller than that of *recognize* to *speech* (especially in a thesis about natural language processing). Hence, one might argue that, all other things being equal, the solution candidate (a) in Figure 2.5 on the next page is superior to solution candidate (b). If no further solution candidates are considered this candidate (a) would then be the solution structure for the grammar and the word graph in the example.

Of course, the exact determination of the solution depends on the constraints of the grammar which will be dealt with shortly.

2.4 Mapping from WCDG parsing problem to CSOP

It is the relational nature of dependency structures that makes it easy to map the parsing problem for WCDG to CSOP. In order to do so, one has to define exactly what the constraint variables and the domains are and how a solution candidate can be put in relation to a complete assignment of constraint variables.

Given a lexicon $L = \langle W, \lambda, A, V \rangle$, a weighted constraint dependency grammar WCDG $= \langle \mathrm{LEV}, \mathrm{LAB}, \alpha, C, \phi \rangle$ and a word graph $WG = \{\langle a_i, z_i, w_i, s_i \rangle | 1 \leq i \leq n\}$, the set of constraint

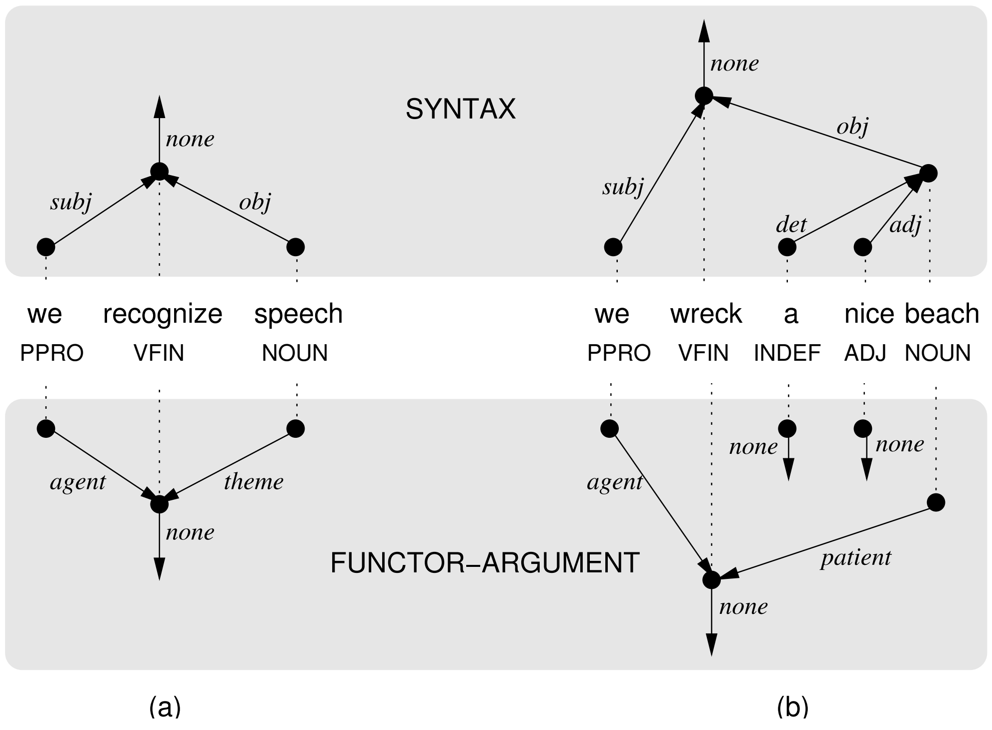


Figure 2.5: Two CDG structure candidates for the example word graph of Figure 2.2 on page 23.

variables consists of all triples $\langle e_i, a, l\rangle$ with $e_i \in \mathrm{WG}$, $a \in \lambda(w_i)$, $l \in \mathrm{LEV}$, i.e., for each lexical reading of a word in an edge and each level an individual variable is constructed. All these variables $v = \langle e_i, a, l\rangle$ have the set $D_\square = \mathrm{LAB} \times \mathrm{WG}_\top \cup \{\square\}$ as their domains.

$$\begin{aligned}
V &= \{\langle e_i, a_i, l\rangle | e_i \in \mathrm{WG}, a_i \in \lambda(w_i), l \in \mathrm{LEV}\}\\
\mathrm{WG}_\top &= \mathit{WG} \cup \{\top\}\\
D &= \{\langle l', e_j, a_j\rangle | l' \in \mathrm{LAB}, e_j \in \mathrm{WG}_\top, a_j \in \lambda(w_i) \leftrightarrow e_j \neq \top, a_j = \top \leftrightarrow e_j = \top\}\\
D_\square &= D \cup \{\square\}
\end{aligned}$$

A structure candidate $SC = \langle P, \pi, \delta\rangle$ is then equivalent to a complete assignment of values from the domains to constraint variables.

Each variable $\langle e_i, a_i, l\rangle$ with $e_i \notin P$ or $\pi(e_i) \neq a_i$ is assigned the dummy value $\square$. Such an assignment marks those elements of the word graph that are not part of the path in the structure candidate and those variables with a different lexical reading respectively. Recall that a structure candidate represents a completely disambiguated parsing problem, i.e., it does not contain any alternatives, neither acoustical, nor lexical nor structural.

If $\delta(e_i, l) = \langle l', \top\rangle$ the special symbol $\top$ is used again to denote that the word in edge e_i depends on no other word and is the root of the dependency tree. This is reflected as a value $\langle l', \top, \top\rangle$ for a constraint variable $\langle e_i, a_i, l\rangle$.

All other variables $\langle e_i, a_i, l\rangle$ get a value $\langle l', e_j, a_j\rangle$ if and only if $\delta(e_i, l) = \langle l', e_j\rangle$, $\pi(e_i) = a_i$ and $\pi(e_j) = a_j$.

$$\alpha(\langle e_i, a_i, l\rangle) = \begin{cases} \Box & : \text{ iff } e_i \notin P \text{ or } \pi(e_i) \neq a_i \\ \langle l', \top, \top\rangle & : \text{ iff } \delta(e_i, l,) = \langle l', \top\rangle \\ \langle l', e_j, a_j\rangle & : \text{ iff } \delta(e_i, l) = \langle l', e_j\rangle \text{ and } \pi(e_j) = a_j \end{cases}$$

Furthermore we demand an assignment to be *path consistent*[5]: The lexical readings must be chosen consistently, i.e., one and the same word graph edge must take the same lexical entry on all levels:

$$\bigwedge_{\substack{\alpha(\langle e_i, a_i, l\rangle) = \langle l', e_j, a_j\rangle \\ \alpha(\langle e_i, a_i^*, l^*\rangle) = \langle l'^*, e_k, a_k\rangle}} a_i = a_i^*$$

and also in modifiee and modifier position:

$$\bigwedge_{\substack{\alpha(\langle e_i, a_i, l\rangle) = \langle l', e_j, a_j\rangle \\ \alpha(\langle e_j, a_j^*, l^*\rangle) = \langle l'^*, e_k, a_k\rangle}} a_j = a_j^*$$

Picking up the example from the previous sections, Figure 2.6 on the following page shows the mapping from a WCDG structure candidate to a CSP assignment. Note that we again abbreviated the notation when the meaning is unambiguous.

Since the notations as a structure candidate or as an assignment are equivalent and the latter is more convenient, we will use constraint variables and values in the next Section 2.5 where we define constraints more formally and exhaustively.

2.5 Constraints

Constraints are the main means to express (linguistic) knowledge. In this section, we will describe how constraints can be used to encode grammatical requirements. We will make direct reference to the constraint language that was developed for the DAWAI project[6] [Heinecke et al. 1998].

Table 2.2 gives the definition of the syntax of the constraint language in extended Backus-Naur form [Wirth 1986]; an informal but complete explanation of the whole input language can be found in the user manual [Schröder 1997d; Schröder 1997a; Foth, Schröder & Schulz 1998] of the parser [Menzel & Schröder 1998c].

In Section 2.2 constraints are defined as an enumeration of feasible value combinations taken from domains of the constraint variables. However, such an exhaustive listing is not possible for [W]CDG parsing for two reasons.

[5]Not to be confused with *path consistency* of CSP, see Section 4.3.1.

[6]See `http://nats-www.informatik.uni-hamburg.de/~dawai/` for further information.

$$\begin{array}{rcl}
\delta(e_{\mathsf{we}},\mathsf{SYN}) = \langle \mathit{subj}, e_{\mathsf{recog}}\rangle & \equiv & \alpha(\langle e_{\mathsf{we}}, \mathsf{we}_{\mathrm{PPRO}}, \mathsf{SYN}\rangle) = \langle \mathit{subj}, e_{\mathsf{recog}}, \mathsf{recog}_{\mathrm{V}}\rangle \\
\delta(e_{\mathsf{recog}},\mathsf{SYN}) = \langle \mathit{none}, \top\rangle & \equiv & \alpha(\langle e_{\mathsf{recog}}, \mathsf{recog}_{\mathrm{V}}, \mathsf{SYN}\rangle) = \langle \mathit{none}, \top, \top\rangle \\
\delta(e_{\mathsf{speech}},\mathsf{SYN}) = \langle \mathit{obj}, e_{\mathsf{recog}}\rangle & \equiv & \alpha(\langle e_{\mathsf{speech}}, \mathsf{speech}_{\mathrm{N}}, \mathsf{SYN}\rangle) = \langle \mathit{obj}, e_{\mathsf{recog}}, \mathsf{recog}_{\mathrm{V}}\rangle \\
 & & \\
 & & \alpha(\langle e_{\mathsf{wreck}}, \mathsf{wreck}_{\mathrm{V}}, \mathsf{SYN}\rangle) = \Box \\
 & & \alpha(\langle e_{\mathsf{wreck}}, \mathsf{wreck}_{\mathrm{N}}, \mathsf{SYN}\rangle) = \Box \\
 & & \alpha(\langle e_{\mathsf{a}}, \mathsf{a}_{\mathrm{INDEF}}, \mathsf{SYN}\rangle) = \Box \\
 & & \alpha(\langle e_{\mathsf{nice}}, \mathsf{nice}_{\mathrm{ADJ}}, \mathsf{SYN}\rangle) = \Box \\
 & & \alpha(\langle e_{\mathsf{beach}}, \mathsf{beach}_{\mathrm{N}}, \mathsf{SYN}\rangle) = \Box \\
 & & \\
\delta(e_{\mathsf{we}},\mathsf{F\text{-}ARG}) = \langle \mathit{agent}, e_{\mathsf{recog}}\rangle & \equiv & \alpha(\langle e_{\mathsf{we}}, \mathsf{we}_{\mathrm{PPRO}}, \mathsf{F\text{-}ARG}\rangle) = \langle \mathit{agent}, e_{\mathsf{recog}}, \mathsf{recog}_{\mathrm{V}}\rangle \\
\delta(e_{\mathsf{recog}},\mathsf{F\text{-}ARG}) = \langle \mathit{none}, \top\rangle & \equiv & \alpha(\langle e_{\mathsf{recog}}, \mathsf{recog}_{\mathrm{V}}, \mathsf{F\text{-}ARG}\rangle) = \langle \mathit{none}, \top, \top\rangle \\
\delta(e_{\mathsf{speech}},\mathsf{F\text{-}ARG}) = \langle \mathit{theme}, e_{\mathsf{recog}}\rangle & \equiv & \alpha(\langle e_{\mathsf{speech}}, \mathsf{speech}_{\mathrm{N}}, \mathsf{F\text{-}ARG}\rangle) = \langle \mathit{theme}, e_{\mathsf{recog}}, \mathsf{recog}_{\mathrm{V}}\rangle \\
 & & \\
 & & \alpha(\langle e_{\mathsf{wreck}}, \mathsf{wreck}_{\mathrm{V}}, \mathsf{F\text{-}ARG}\rangle) = \Box \\
 & & \vdots
\end{array}$$

Figure 2.6: Mapping from [W]CDG structure candidate to a CSP assignment. Various words and category symbols have been abbreviated.

- Firstly, the domains and also the variables themselves are instance specific since they make reference to the input tokens and therefore depend on the utterance being parsed. A number of variables are constructed for each word in the input and the values represent dependency edges which certainly differ from sentence to sentence. No grammar writer can anticipate all possible inputs to the parser.

- Secondly, it is essential for a natural language grammar to abstract from individual sentences and describe the general principles. Therefore, constraints must express characteristics of a whole class of configurations and not just those of a single sentence.

Therefore, we will use constraint templates with logical formulae (cf. example in Section 2.2.2) which are checked for every value (combination) when a sentence is analyzed.

A constraint (template) is satisfied by a specific value or value combination if and only if the constraint formula when applied to the value combination evaluates to true. The constraints are implicitly universally quantified, i.e., all constraints must be applied to all edges and to all edge combinations respectively, at least for the time being (cf. Section 4.13.3). Since constraints are asymmetric in general, even the edge ordering is significant, i.e., not only the pair $\langle e_i, e_j\rangle$ must be tested with a binary constraint but also the inverse pair $\langle e_j, e_i\rangle$. Of course, most constraints are specific to particular constructions so that the constraint formulae often take the form of an implication where the antecedent restricts the application to the special configurations.

The rest of this section defines the semantics of the constraint language.

```
Attr        ::=  String | Number
ConId       ::=  String
Constraint  ::=  ‘{’ VarList ‘}’ ‘:’
                 ConId ‘:’ [ [ Section ] ‘:’ Penalty ‘:’ ]
                 Formula
Formula     ::=  Term Relation Term
              |  Formula Junctor Formula
              |  Predicate ‘(’ TermList ‘)’
              |  ‘∼’ Formula
              |  ‘(’ Formula ‘)’
              |  ‘true’
              |  ‘false’
Function    ::=  String
Junctor     ::=  ‘&’ | ‘|’ | ‘->’ | ‘<->’
Operator    ::=  ‘+’ | ‘–’ | ‘*’ | ‘/’
Path        ::=  Path ‘:’ Attr | Attr
Penalty     ::=  Number | Term
Predicate   ::=  String
Relation    ::=  ‘=’ | ‘>’ | ‘<’ | ‘>=’ | ‘<=’ | ‘!=’
Section     ::=  String
Term        ::=  Variable ‘^’ Path
              |  Variable ‘@’ Path
              |  Variable ‘.label’
              |  Variable ‘.level’
              |  Term Operator Term
              |  Function ‘(’ TermList ‘)’
              |  ‘[’ Term ‘]’
              |  String
              |  Number
TermList    ::=  Term | TermList ‘,’ Term
Variable    ::=  String
VarList     ::=  Variable | Variable ‘,’ VarList
```

Table 2.2: EBNF definition of the constraint syntax.

2.5.1 General structure

A constraint (element Constraint) consists of up to five parts (cf. Table 2.2): variables (element VarList), name (element ConId), class (element Section), penalty (element Penalty) and formula (element Formula).

The variables are placeholders which are replaced by actual dependency edges during evaluation. The name is an identifier and the class facilitates a partitioning of the constraint set for modularization purposes. The penalty is the weight, of course. The formula is a logical expression with extended propositional logics syntax and determines whether the constraint is satisfied or not in a specific context.

Constraints can be classified depending on the kind of penalty they carry. Hard constraints are those with a weight of zero; soft constraints have weights greater than zero. Soft ones can further be subdivided depending on whether their penalty is fixed by the grammar writer or sensitive to different contexts, i.e., it dynamically changes its numeric value in different sentence types. Section 3.5.6 motivates dynamic constraints linguistically and gives typical examples. They are used in Sections 5.6 and 5.7 to import external assessments into the constraint system and in Section 5.8 to model linguistic preferences involving distances between words and length of constituents.

Note that while the meaning of constraint weights can usually be best described as the importance of the constraints, it also carries aspects of the degree of satisfaction in dynamic constraints. In different approaches, constraints may have more than a single weight associated with them, e.g., separate ones for importance, degree of satisfaction, relevance, level of formality etc. respectively [Miguel & Shen 2001a, Section 5.1.1].[7] But since these weights have to be combined in some way eventually, we decided to stick to a single weight and let the grammar writer choose the appropriate value, possibly using dynamic constraints.

Constraint C-1 is a unary example constraint named `SubjectVerbNoun` belonging to the constraint class `Subject`. Unary constraint always contain a single variable which is named `X` in the example. The penalty is fixed at 0.1.

```
C-1 {X} : SubjectVerbNoun : Subject : 0.1 :
      X.level=SYNTAX & X.label=subj -> X@cat=NOUN & X^cat=FINVERB;
```

The subject is a noun and depends on a finite verb.

During constraint evaluation every assignment, i.e., every combination of constraint variable and value, is bound to the syntactic component VARIABLE in the signature of the constraint. Hence this component is a representative of a complete dependency edge with information about the dependent and the governing words including lexical information, the label and the level: `X` $\equiv \langle e_i, a_i, l \rangle \leftarrow \langle l', e_j, a_j \rangle$.[8] Section 2.5.3 explains how these information bits are accessed.

Constraint C-2 is an example for a binary constraint since two variables `X` and `Y` are used.

```
C-2 {X, Y} : OneSubjectOnly : Subject : 0.0 :
      X.level=SYNTAX & Y.level=SYNTAX &
      X^id=Y^id &
      X.label=subj ->
        Y.label!=subj
```

A finite verb does not have more than one subject.

The score is fixed at 0.0, i.e., the constraint is a hard one. While every dependency edge has to be tested against a unary constraint, all *combinations* of two dependency edges must be checked with the binary constraints.

[7]See also Schröder [1997e].

[8]For dependency edges that lead to root the corresponding equivalence `X` $\equiv \langle e_i, a_i, l \rangle \leftarrow \langle l', \top, \top \rangle$ holds. We assume from here on that e_j also subsumes the special symbol $\top$.

2.5.2 Formulae

The formula is the central and most complicated component of a constraint. This section describes how formulae evaluate to true or false in the context of a binding, i.e., when applied to an edge or a tuple of edges.

Truth constants

The simplest formulae are the truth constants `true` and `false`; they are satisfied either always or never.

```
C-3 {X} : NonRoot : General : 0.1 :
      root(X^id) -> false;
```

Nothing should be the root.

Negation

The negation is the only unary operator and is defined in the standard way: `~a` is satisfied if and only if `a` is not.

```
C-4 {X} : NonVerbNonRoot : General : 0.1 :
        X.level=SYNTAX & X@cat!=FINVERB -> ~root(X^id);
```

Something that is not a finite verb is not the root.

Boolean junctors

The Boolean operators conjunction, disjunction, implication and bi-implication are defined in the standard way; the following table lists the truth tables for satisfying (+) or violating (-) conditions.

a	b	a&b	a|b	a->b	a<->b
-	-	-	-	+	+
-	+	-	+	+	-
+	-	-	+	-	-
+	+	+	+	+	+

Predicates

Predicates take terms (cf. Section 2.5.3) as arguments and yield a truth value.

Six special purpose predicates are defined. We assume that `X` $\equiv \langle e_i, a_i, l\rangle \leftarrow \langle l', e_j, a_j\rangle$ in all examples.

- `start(X@id)` takes a word graph edge e_{id} as its only parameter and succeeds if and only if the word graph starts with that edge: $e_{\mathrm{id}} = \langle a_{\mathrm{id}}, z_{\mathrm{id}}, w_{\mathrm{id}}, s_{\mathrm{id}}\rangle$ and $min_{\mathrm{WG}} = a_{\mathrm{id}}$

- `stop(X^id)` takes a word graph edge e_{id} as its only parameter and succeeds if and only if the word graph ends with that edge: $e_{\text{id}} = \langle a_{\text{id}}, z_{\text{id}}, w_{\text{id}}, s_{\text{id}} \rangle$ and $max_{\text{WG}} = z_{\text{id}}$

- `root(X^id)` takes a word graph edge or $\top$ as its only parameter and succeeds if and only if the parameter equals $\top$.

- `exists(X@foo:bar)` takes an attribute f (`foo:bar`) in a lexical entry e (`X@`) as its parameter and succeeds if and only if the lexical entry contains a value at that attribute: $e(f) \neq \text{undef}$.

- `subsumes(h, a, b)` takes three strings as its parameters. The first is the name of an ontology[9] and is satisfied if symbol `a` is a superclass of symbol `b` in the ontology.

- `print(a, b, ...)` takes an arbitrary number of parameters, prints them and always succeeds.

Predicates can be added easily to the system in order to extend the system depending on future demands. However, the calculation needed to determine the result of a predicate should be efficiently computable and independent of the sentence context as a whole; otherwise the considerations about the complexity (cf. Section 2.8.2 and Chapter 4) can be invalidated. For instance, a fictitious predicate `solution(X@id)` which is satisfied if the argument edge is part of the overall solution moves the complete solution process into a predicate and requires considerable computational effort.

Relations

Relations are special purpose predicates which are used with an inline syntax for the operator.

There are six relational predicates defined: ‘equal =’ and ‘not equal !=’ (defined on numbers, symbols and lexical ids) as well as ‘greater >’, ‘greater or equal >=’, ‘less <’ and ’less or equal <=’ (defined only for numbers). All these predicates behave as expected.

2.5.3 Terms

There are three types of terms: numbers, strings (also called symbols) and lexical ids.

Constants

The simplest terms in the constraint language are constant numbers and strings. The latter can be written with or without delimiting double quotes.

[9]See Foth et al. [1999] for details about how to use ontologies.

Lexical information

Lexical information of the dependent and the governing word forms can be accessed using the `@` and `^` operators. If `X` $\equiv \langle e_i, a_i, l\rangle \leftarrow \langle l', e_j, a_j\rangle$ then `X@foo` tries to look up the attribute `foo` in a_i, i.e., the resulting term is determined by a_i(`foo`). If the attribute is not defined for a_i, the behavior is undefined.[10] Lexical information for the governor can analogously be retrieved with the `X^foo` construct in which case a_j is the relevant attribute-value matrix.

The lexicon only contains static information about word forms. However, there are a couple of special attributes that can be used to retrieve information that is lexicon-independent as the information is only induced specifically by the problem instance itself. The following table lists these attributes for the dependent edge.

Attribute	**Semantics**
`X@id`	identification for edge e_i, can be compared for (in)equality
`X@from`	start node of edge e_i, i.e., a if $e_i = \langle a, z, w, s\rangle$
`X@to`	end node of edge e_i, i.e., z if $e_i = \langle a, z, w, s\rangle$
`X@pos`	positional index (used sometimes when analyzing word strings)

As in all cases, the information is also available for the governor: `X^id`, `X^from`, `X^to` and `X^pos`.

Label and level information

The dot operator (‘`.`’) is used to access the information about the label and the level. If again `X` $\equiv \langle e_i, a_i, l\rangle \leftarrow \langle l', e_j, a_j\rangle$ then `X.level` and `X.label` stand for l and l' respectively.

Arithmetic operations and functions

The four standard arithmetic operations (‘`+`’, ‘`-`’, ‘`*`’, ‘`/`’) are available for numbers; they behave as expected.

Some special purpose functions are used to compute new terms from given ones.

- `min(a, b, ...)` returns the smallest of the numbers `a`, `b`... .
- `max(a, b, ...)` returns the largest of the numbers `a`, `b` ... .
- `abs(a)` returns `-a` if `a` is negative and `a` itself otherwise.
- `distance(X@id, X^id)` returns the ‘distance’ of two edges (corresponding to the ids `X@id` and `X^id`) in the word graph under consideration and can be used to gather information about linear ordering. If `X` $\equiv \langle e_i, a_i, l\rangle \leftarrow \langle l', e_j, a_j\rangle$ and $e_i = \langle a_i, z_i, w_i, s_i\rangle$ as well as $e_j = \langle a_j, z_j, w_j, s_j\rangle$ the above functions returns $z_j - a_i$.

[10]See Section 2.5.2 for details about the predicate `exists` which can be used safely to check whether a value for an attribute exists.

- `acoustics(X@id)` returns the adjusted acoustic score of the word hypothesis in the input word graph that corresponds to the given lexical id. Section 5.6 describes the integration of acoustic scores into the parsing process in detail.
- `pts(X@id)` returns the assessment of an external POS tagger for the lexical item corresponding to the given lexical id. Section 5.7 describes the integration of an external POS tagger into the WCDG parsing system in detail.

2.6 Restriction of constraint arity

Recall that the arity of a CSP is the maximum number of variables in any of the constraints and that the degree of a CSP is the maximum size of any of the domains (cf. Section 2.2.1).

A large part of constraint satisfaction research has exclusively considered – both theoretically and empirically – the binary CSP, i.e., a CSP with arity two. There are two reasons:

- A relation between the arity and the degree of a CSP facilitates a transformation of arbitrary n-ary CSP ($n > 2$) to binary CSP with enlarged degree [Nudel 1983, p. 138]. The idea of a conversion is simple:[11] A subset $\{v_{i_1}, ..., v_{i_m}\}$ of the variables is replaced by a new variable v' with an extended domain consisting of the cross-product of the domains of the original variables: $D' = D_{i_1} \times ... \times D_{i_m}$. Thus, there is a trade-off between the number and the complexity of variables: Either a large number of variables each represent an elementary property or a smaller set of variables encode more complex configurations.
- Any solution algorithm for CSP has to check the compatibility of value combinations whose variables are connected by a constraint [Mackworth & Freuder 1985, for instance]. Obviously, it suffices to check at most $O(n^2)$ pairs of values in binary CSP while potentially $O(n^3)$ triples have to be tested in ternary CSP. Some algorithms (cf. Section 4.6) also employ data structures of size $O(n^{|\text{arity}|})$. Therefore, a formulation of a problem as a binary CSP is preferable and in fact indispensable for some algorithms.

From here on, we will consider at most binary CSP and accordingly allow at most two placeholders for dependency edges in WCDG constraints. Section 3.3.3 discusses what this decision means for a grammar writer.

2.7 Comparison of WCDG to CDG as defined by Maruyama

This section summarizes the differences between WCDGs and the variants of CDG as originally defined by Maruyama [1990b] and as used by Harper et al. [1999a].

- Constraints are weighted in WCDG while CDG exclusively uses hard constraints. Therefore WCDGs are a generalization of CDGs.

[11] Section 3.4.2 shows both that the problem is real and that the solution approach is relevant in WCDGs.

- WCDGs define which labels are appropriate for which level.[12]

- WCDG constraints have a greater expressiveness than the constraints suggested by Maruyama [1990b] and Harper et al. [1999a]. Constraints in WCDGs can freely access any lexical information of any involved modifier or modifiee. Lexical ambiguity is possible both in terms of different categories and of differing feature values (cf. Section 2.5.3). Maruyama [1990b] in contrast assumed a completely disambiguated sequence of categories as the input to the parser. No form of lexical ambiguity is allowed and the only accessible information about the word forms is the category. Harper et al. [1995, Section 3.1] deal with ambiguities regarding categories by duplicating constraint values for each possible part-of-tag. Lexical ambiguity besides that of categories is handled by means of delayed constraint application [Harper et al. 1995, Section 3.3]. Constraints that access the ambiguous lexical features are later applied to the remaining variables after unambiguous values have been created by duplication of constraint variables.

  Nevertheless the expressiveness of their constraints falls short of those of WCDGs since modifier and modifiee are not accessed symmetrically. Agreement between subject and finite verb, for instance, cannot be expressed by a unary constraint [Harper et al. 1995, p. 23] which is easily possible with a unary WCDG constraint. We decided to include the extra expressiveness since we feel that the constraints are already restricted enough (cf. Section 3.3.3). Obviously, the resulting CSP becomes more expensive to solve when allowing fully disambiguated governor nodes because the numbers of constraint variables and values increase.

  Only recently White et al. [1997] have also proposed a kind of generalized constraints, so-called *aggregated constraints* [White 2000, p. 38], which can at least partly access information about the governor of a dependency edge.

- The formalism of WCDG provides dynamic constraints which do not have a static score but receive different weights depending on the context in which they are evaluated (cf. Sections 3.5.6, 5.6 and 5.7). Dynamic constraints are not necessarily soft constraints.

- The WCDG constraint language is richer than that of Maruyama [1990b]. It not only includes more powerful access functions (see above) but additionally predicates, functions and simple arithmetic operators. This added-value is particularly useful when used in combination with information stored in the lexicon and with external ontologies [Foth et al. 1999].

- A further difference does not directly concern the definition of the grammar formalism but the solution methods. Both Maruyama [1990b] and Harper et al. [1999a] deploy a single consistency-based method known as filtering (cf. Section 4.6) in their parser, while several algorithms including filtering have been implemented for WCDG (cf. Chapter 4). Experimental evaluations have shown that filtering is in fact not appropriate for WCDG parsing (cf. Section 4.6).

- Again not being directly linked to the definitions, a further difference is connected to the use of [W]CDGs. While Maruyama [1990a, Section 3.4] and Harper et al. [1993] propose the inclusion of semantic evidence in the form of semantic constraints, they do not aim at separate semantic structures as in WCDG (cf. Section 3.4.6).

[12] Harper & Helzerman [1995, Section 3.2] use a lookup table for the same purpose.

2.8 Complexity

Three different aspects of complexity are distinguished in this section:

1. Relation of the constraint language to logic
2. Complexity of the parsing problem when using [W]CDG
3. Classification of the set of languages that can be generated with [W]CDG with respect to the Chomsky hierarchy of formal languages [Hopcroft & Ullman 1979, chapter 9]

Each of these aspects will be discussed in the following subsections.

2.8.1 Relation of the constraint language to logic

The constraint language defined in Section 2.5 strikingly resembles the language of first-order predicate logic formulae. Function and predicate symbols as well as variables and junctors can be mapped directly. Constraint formulae are implicitly assumed to be universally quantified. The lack of existential quantifiers does not seem to be important since every first-order predicate formula can be converted into an equivalent (with respect to satisfiability) Skolem formula in prenex form [Gallier 1987, pp. 305 & 340] where all existential quantifiers have been replaced and all universal quantifiers precede the actual formula body [Schöning 1992a, p. 67].

A well-known theorem by Church [Schöning 1992a, for instance] proves that the satisfiability and the validity problems for first-order predicate logic are undecidable.

The syntactical equivalence between CDG formulae and first-order predicate logic formulae suggests that the satisfiability and the validity problems are undecidable for [W]CDG formulae as well. But we only looked at the syntax of the formulae so far. The crucial point, however, is the semantics behind the syntax. If ones defines a semantics for [W]CDG constraints equivalent to that of predicate logic formulae [Gallier 1987, p. 158], the questions of whether at least one (satisfiability) or whether all structures (validity) fulfill a particular constraint become undecidable.

However, such a general semantics is usually not assumed for [W]CDG.[13] The space of possible structures $\mathcal{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ is highly limited in CDG. The domain $\mathcal{D}$ is the set of labeled edges between words and, in particular, infinite domains are expressly not allowed, which would correspond to an infinite input. The interpretation function $\mathcal{I}$ is determined by the fixed definitions of the few functions and predicates; no free variables are allowed.

A formal proof that the satisfiability and validity problems can always be answered for all [W]CDG (given the restrictions listed here and the usual semantics defined in Section 2.5.2) is still missing mainly because such a proof is completely irrelevant to the problem of natural language parsing which only involves the application of the formulae to a finite set of pre-defined dependency edges.

[13]It would be if we were looking for a procedure that can always answer the following question in finite time: Given an arbitrary grammar, is there a sentence that can be parsed by that grammar. Obviously, this problem is at least semi-decidable: Enumerate all sentences in a kind of diagonal triangle, try to parse the sentence and answer ‘yes’ if successful.

2.8.2 Complexity of the parsing problem

Computational complexity refers to the amount of time and memory that an algorithm needs to solve a problem in the worst case. A problem itself falls into a specific complexity class if it can be proven that no algorithm exists that both solves the problem and has a smaller complexity class.

The parsing problem for [W]CDG is $\mathcal{NP}$-complete. This can easily be shown by reducing the chromatic number problem [Hopcroft & Ullman 1979, p. 341] to a [W]CDG parsing problem. Given a graph G and an integer k, construct a [W]CDG with a single lexical entry for a dummy word, a single level, k distinct labels and the general initialization Constraint C-5.

```
C-5 {X} : 0.0 : root(X^id);
```

Assume that the vertices in G are numbered. For each edge in G that connects the vertices i and j, add a constraint like Constraint C-6:

```
C-6 {X, Y} : 0.0 : X@pos=i & Y@pos=j -> X.label != Y.label;
```

The parsing problem for an input sequence of k words is then equivalent to the chromatic number problem for G and k. If a parse can be found, the labels determine the color of the corresponding vertices.

Given that the parsing problem for [W]CDG is $\mathcal{NP}$-complete it can be assumed that every complete and sound algorithm that solves the parsing problem has exponential time complexity $O(2^n)$. Note that this is only a worst case consideration; Chapter 4 deals with actual running time and approximate algorithms.

The exponential complexity of CDG is not unexpected. It is presumed that, while a fully projective[14] dependency grammar is weakly equivalent to a context-free grammar [Gaifman 1965; Rambow & Joshi 1994, p. 9] and the recognition problem for this restricted type of dependency grammars is therefore in $O(n^3)$,[15] the parsing problem for general dependency grammars which allow discontinuous structures is probably exponential [Covington 1990, Section 5.3]. Neuhaus & Bröker [1997] prove $\mathcal{NP}$-completeness for the recognition problem of a dependency grammar which separates the immediate dominance relation from the linear precedence relation (which is possible in [W]CDG); see also [Barton, Berwick & Ristad 1987, Chapter 7]. It should be noted that the polynomial complexity $O(n^3)$ of the recognition problem for context-free grammars [Hopcroft & Ullman 1979] is also fragile. As soon as the formalism is extended to handle agreement and lexical ambiguity, as in feature unification grammars, the recognition problem becomes $\mathcal{NP}$-complete [Barton, Berwick & Ristad 1987, Chapter 3]. Furthermore, a context-free grammar may assign an exponential number of parses to an ambiguous input string so that the extraction and enumeration of structures can also take exponential time. Nevertheless, the word problem for CFGs is computable in polynomial time (while it is exponential for WCDG). Plaehn [2000, p. 10] presumes that his parser [Plaehn 1999] for *discontinuous phrase structure grammars* is also exponential in runtime.

[14] Section 3.4.5 defines projectivity.

[15] But note the following caveat:

> "[...], *context-free generative power does not guarantee efficient parsability.* Every ID/LP grammar technically generates a context-free language, but the potentially large size of the corresponding CFG means that we can't count on that fact to give us efficient parsing."
>
> —— Barton, Berwick & Ristad [1987, p. 208]

2.8.3 Classification of the set of languages defined by WCDG

Maruyama [1990a, Section 4] shows that the weak generative capacity of CDG with at least two levels is strictly greater than that of context-free grammars. He specifies a CDG which generates the language $L_{\omega\omega} = \{\omega\omega | \omega \in \Sigma^*\}$ which is not context-free. Furthermore, he gives general instructions on how to construct a CDG from a CFG in Greibach normal form which accepts the same language as the CFG.[16]

Harbusch [1997] extends the proof of Maruyama [1990a] from CFG to tree-adjoining grammars [Joshi, Levy & Takahashi 1975, TAG for short] showing that every language recognized by a TAG can also be generated by a CDG.[17] Furthermore, she gives a CDG which generates the non TAG language L-6 $= a^n b^n c^n d^n e^n f^n$ proving that $L_{\text{TAG}} \subset L_{\text{CDG}}$. The set of languages recognized by CDG therefore contains at least the *mildly context-sensitive* languages.[18]

Since the word problem for CDG is $\mathcal{NP}$-complete but the word problem for context-sensitive languages is $\mathcal{PSPACE}$-complete [Schöning 1992b, p. 169] it follows that the set of languages generated by a CDG is properly contained in the set of context-sensitive languages.[19]

2.9 Related work

There exists a large spectrum of *constraint-based* formalisms [Shieber 1992] for natural language processing that in part vary considerably. Hess [1996] compares the notion of constraints as it is used in term [Robinson 1965] or feature unification [Kay 1984; Shieber 1986] formalisms and in constraint satisfaction problems (cf. Section 2.2). An alternative interpretation of constraint satisfaction in NLP is due to Blache [2000a].

The WCDG approach goes back to constraint dependency grammars of Maruyama [1990b; 1990a].[20] The original idea of CDG has been extended at Purdue University [Harper et al. 1994; Harper & Helzerman 1994; Harper & Helzerman 1995; Harper et al. 1995; Harper et al. 1999a] in the context of word graph filtering in speech applications. The enhancements

[16]White [1995] describes an implemented mono-directional conversion method from CFG to CDG.

[17]Harbusch [1997, p. 40] assumes that "... the runtime of the parser for CDG of arity 2 is $O(n^4)$". Consequently, she motivates her conversion from TAGs (for which the fastest parsing algorithm has complexity $O(n^5)$) to CDGs by conjecturing that parsers for TAGs with better runtime might become available. However, she is mistaken in that only the filtering step of the consistency-based algorithm for CDGs (cf. Section 4.6) has a complexity of $O(n^4)$ while the subsequent extraction step still has exponential complexity in the worst case (cf. Section 2.8.2). The filtering step generally only reduces the problem and does not solve it entirely although there are cases where filtering suffices.

[18]Cf. Joshi, Vijay-Shanker & Weir [1991] or `http://www.kornai.com/MatLing/mcs.html`.

[19]We are grateful to Mary Harper (personal communication) for pointing us in this direction.

[20]The 'inventor' of CDG, Maruyama [1990a] of IBM Tokyo, first used the formalism in an interactive Japanese-to-English machine translation system for newspaper texts [Maruyama, Watanabe & Ogino 1990]. The main component (entered after morphological disambiguation) for the source language analysis is a syntactic parser that assigns a dependency structure to the sequence of morphological units which is relatively easy for Japanese because a common theory demands that all dependency links strictly branch to the right. The parser maintains a consistent state of the analysis during the parsing process using constraint propagation algorithms [Maruyama 1990b, see also Section 4.3.1]. It not only considers grammatical constraints, but also allows the user to specify dependency links directly using a computer mouse in a graphical environment. It can be argued that the human operator dynamically adds constraints to the system by doing so. Due to the ease of interaction, Maruyama, Watanabe & Ogino [1990] claim that the system is eligible for non-expert users and give evidence thereof in terms of numbers from a comparative usability study.

include an extension of the filtering algorithm to multiply-segmented constraint satisfaction problems which allows the analysis of word graphs and lexically ambiguous sentences [Zoltowski et al. 1992; Helzerman & Harper 1996; Harper et al. 1999b], a parallel implementation of the parsing algorithm [Helzerman & Harper 1992; Harper et al. 1995], an integration of semantic constraints [Harper et al. 1993], automatic grammar induction from corpora [White et al. 1997; White 2000; Harper et al. 2000] and various other optimizations [Harper & Helzerman 1995; Harper, Hockema & White 1999]. The use of constraint scores that allow the classification of certain constraints as violable has first been suggested by Menzel [1994].

A similar step from strictly eliminating constraints to graded constraints is proposed by Qu, Rosé & Eugenio [1996] in the context of plan-based discourse processing. High-level constraints operate on discourse structures and on background knowledge and assign penalties to certain inference chains. However, graded constraints are only used to resolve ambiguities that result from previous processing steps.

Duchier [1999a] proposes an alternative eliminative approach to constraint dependency parsing that is inspired by the original formulation of Maruyama [1990a] and based on constraint satisfaction techniques as well. A prototypical parser and a grammar fragment for German have been implemented using the constraint programming language Oz.[21] For an account to German word order in their framework see also Section 3.4.4. A separate axiomatization is given for parsing of phrase structure trees with tree descriptions [Duchier 1999b; Duchier & Niehren 2000; Duchier & Thater 1999].

Blache [2000c] presents the formalism *property grammars* which introduces the notion of properties (instead of constraints) which are subdivided into local properties represented as feature structures (complex categories) in the lexicon and global properties which constrain co-occurring categories with respect to a number of aspects such as constituency, obligation, unicity, requirement, exclusion, linearity and dependency. The properties take effect directly at category level without the need to postulate a local configuration first [Blache 2000b] which is the reason why Blache [2000a] distinguishes active constraints (properties) from passive constraints which are relegated to a secondary role (as for instance in constraint grammar [Karlsson 1990], HPSG [Pollard & Sag 1987; Pollard & Sag 1994] and also [W]CDG). The notion of grammaticality is replaced by so-called characterizations, i.e., partitionings of the set of constraints into satisfied and violated ones. This concept is found in a similar way in WCDG and accounts for a robust system behavior. Blache & Morawietz [2000] hint at an implementation of property grammars.[22]

Constraint grammars [Karlsson 1990; Karlsson et al. 1995] and subsequent parsers [Tapanainen & Järvinen 1997; Järvinen & Tapanainen 1997] share with [W]CDG the concept of eliminating ungrammatical structures instead of constructing grammatical structures, but they employ finite-state techniques.

Link grammars [Sleator & Temperley 1991; Sleator & Temperley 1993] are lexicalized dependency grammars and consist of a set of words with specific linking requirements. A grammatical sentence has a projective and connected tree structure that satisfies all linking requirements. Lafferty, Sleator & Temperley [1992] describe a probabilistic version of link grammars.

[21]Duchier, Gardent & Niehren [1999] give a tutorial for natural language processing with Oz. Oz [Smolka 1995] is now maintained by the Mozart Consortium `http://www.mozart-oz.org/`.

[22]According to Blache (personal communication), property grammars do not belong to the generative paradigm, but fall somewhere between generative and equation theories.

2.10 Summary

The grammar formalism of weighted constraint dependency grammars (WCDG) has been formally described.

Abstract definitions of the expected input structures (word graph), the lexical knowledge base (lexicon), the grammatical knowledge base (WCDG) and feasible structural interpretations (structure candidate) of input sentences were given. A precise formulation of a mapping from WCDG parsing problems to constraint satisfaction optimization problems facilitates the transfer of insights, results and algorithms from the well-known area of constraint satisfaction problems (CSP). In particular, the result that the [W]CDG parsing problem belongs to the class of $\mathcal{NP}$-complete problems is based on a reduction of the well-known CSP problem of chromatic numbers of graphs. The syntax and the semantics of the WCDG constraint language have been precisely defined which is a prerequisite for the accurate formulation of optimality among structure candidates which in turn is necessary for determining the solution structure of a parsing problem. We deliberately avoided any reference to parsing algorithms because the view of the parsing problem as a CSP emphasizes the declarative status of a WCDG, in particular the order independence of constraints and the lack of control parameters in the grammar. That does of course not imply that a WCDG only takes competence aspects into account; performance phenomena that often arise from processing difficulties are simply described declaratively, i.e., without any reference to procedural aspects of the processing model.[23] Chapter 4 deals with algorithms for WCDG parsing.

[23]Compare to the discussion of competence and performance vs. declarativity of grammar formalisms in Bröker, Hahn & Schacht [1993, Section 1].

Chapter 3

Grammar Modeling

This chapter is based on an article [Schröder et al. 2000] published in the international journal 'Traitement Automatique des Langues' (Natural Language Processing)[1].

3.1 Overview

The purpose of this chapter is three-fold. Firstly it serves as a tutorial about grammar writing using the formalism of weighted constraint dependency grammars (WCDG). Although some details have to be omitted we present those techniques that have proven themselves helpful during grammar development.

Secondly it describes the backbone of a grammar that we developed for the Stellingen corpus (cf. Section 5.2.1), a subset of the German Verbmobil [Wahlster 1993] corpus of spoken appointment-scheduling dialogues. The analyses had to be simplified for presentation purposes but the underlying ideas are preserved as described.

The third and most important intention of this chapter is to answer the scientifically interesting question of whether the grammar formalism of WCDG is adequate for modeling a non-trivial part of the phenomena occurring in natural language, in this case in German. We will give example constraints and examine how serious certain restrictions of the formalism are and how they can be overcome by different approximation techniques.

The main focus is on practical models for observed phenomena; it is neither the intention to develop a linguistic theory nor to describe a complete grammar in detail.

After a general introduction to grammar modeling and a brief review of the dominant theories in Section 3.2, we start by briefly recapitulating the WCDG formalism; however, this time we will look at it from the point of view of a grammar writer and discuss consequences that arise from its definition for actual grammars (Section 3.3).

In Section 3.4, we present constraint modelings for a series of language phenomena. For clarity of the presentation, only hard constraints are used in this part; it should be easily imaginable how selected crisp constraints can be made soft by assigning an appropriate weight larger than zero.

[1]See `http://www.atala.org/tal/` for further information.

The subsequent Section 3.5 then describes more thoroughly how to use soft constraints for grammar development. A spectrum of application scenarios for soft constraints is given to demonstrate the usefulness of the concept. Again, only the basic principles can be described while details like, for instance, the complete fine-tuned system of exact numerical weights has to be omitted.

Section 3.6 discusses related work, and in Section 3.7 we summarize the possibilities but also the limitations of the approach and suggest further extensions of the grammar and the formalism.

3.2 Introduction

Grammar modeling is understood as the task of describing the underlying regularities of a natural language[2] using a given well-defined formalism. Ideally, the description should be complete, accurate and unambiguous. However, the fuzzy (and generally unlimited) nature of language as well as external (sometimes computational) requirements often enforce a restriction to specific subclasses of language use. The difficulties of artificial components to process natural language both exhaustively and accurately as well as efficiently, often lead to the limitation to a subset of 'real' language, either by using so-called controlled language[3], i.e., explicitly preventing a user from entering 'forbidden' constructions or lexical items, or by simply rejecting uncovered input as unparsable. In anticipation of the upcoming sections, we note here that WCDG allows a grammar writer to extend the coverage of a grammar by means of soft constraints while providing a ranking of the (now possibly ambiguous) admissible analyses at the same time (and by the same means).

Different goals can be pursued when modeling a grammar. For linguistic theories, the classical levels of adequacy [Bußmann 1990, → Adäquatheitsebenen, p. 46] of Chomsky [1965] distinguish between observational, descriptive and explanatory adequacy, depending on whether the sentences, the underlying structures or the process of learning the language respectively can be predicted by a theory of grammar. Natural language parsing as an important subtask of natural language processing usually aims at finding a structural interpretation for the relevant sentences that is sufficient for a subsequent task-dependent processing. Also, engineers of NLP systems must often pay attention to more practical properties such as efficiency, good empirical coverage and maintainability that are essential in real systems for automatic language analysis or generation.

In this work, we are interested in language modeling for analysis, i.e., for parsing. We focus on syntax rather than on phonetics and prosody or semantics and discourse. However, we try to incorporate knowledge from these levels into the syntactic analysis (see, for instance, Sections 3.3 and 3.4.6).

Different grammatical frameworks have been developed for language modeling, among which context-free grammars (CFG) or phrase-structure grammars (PSG) are the most widely adopted for actual applications. Context-free languages are the *type-2* languages in Chomsky's hierarchy of formal languages [Chomsky 1956; Hopcroft & Ullman 1979]. Context-free grammars are characterized by rewrite rules with one non-terminal symbol on the left and a sequence of symbols

[2]This work assumes that a grammar is language specific although one might use the same formalism to describe universal characteristics and provide parameters for specific languages.

[3]See, for instance, the international workshops on controlled language applications (CLAW).

(terminals and non-terminals) on the right. Although there are some syntactic phenomena that are inherently *not* context-free, e.g., discontinuous structures in German, nesting of subordinated clauses in Swiss German [Gazdar & Mellish 1989, p. 133, for instance] or cross-serial dependencies in center-embeddings in Dutch [Rambow & Joshi 1994, p. 12], it is generally accepted (at least for languages like English, not so much for languages with free word order) that the power of CFGs is *basically* sufficient for natural language analysis.[4] CFGs have been used quite early in natural language parsing, for instance, in definite clause grammars (DCG) [Pereira & Warren 1980]. Their predominant position still persists, although CFGs are nowadays often enriched with additional formal frameworks, such as feature unification which was first widely used in PATR-II [Shieber 1984]. Figure 3.1a presents a context-free analysis of an example sentence.

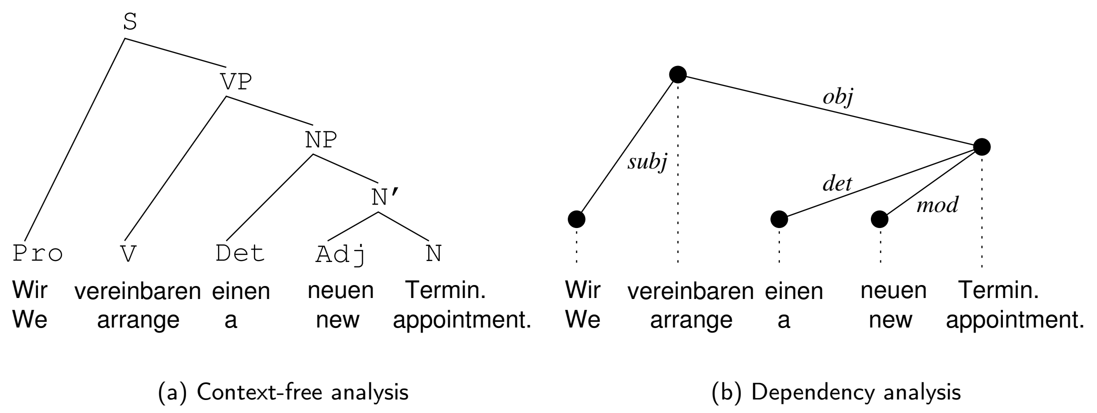

S
VP
NP
N′
Pro V Det Adj N
Wir vereinbaren einen neuen Termin.
We arrange a new appointment.
(a) Context-free analysis
obj
subj
det
mod
Wir vereinbaren einen neuen Termin.
We arrange a new appointment.
(b) Dependency analysis

Figure 3.1: Analyses in the frameworks of context-free and dependency grammars.

Phrase structure rules are also a central part in *transformational grammars* [Chomsky 1957; Chomsky 1965] but they are augmented by *transformations* which map between the deep structure that has been generated by the PSG rules and the surface structure which, among other things, describes the output sentence. The successor yo transformational grammar, *government and binding theory* [Chomsky 1981; Haegeman 1991], reduces the number of transformations to one: move-α. This completely underspecified transformation allows to move anything anywhere [Sells 1987], but its application is restricted by universal constraints which are part of a *universal grammar*. Parameters then tailor constraints to specific languages.

Transformational grammars have played (and still play) a very influential role in theoretical linguistics. However, the focus is not so much on actually occuring language data or even applications, which is partly due to the generative view on grammar. Although some prototypical implementations exist [Fong 2000, for instance] their purpose is rarely a natural language application but mostly a system to verify specific theoretical assumptions about grammar.

Weighted constraint dependency grammars ([W]CDG) – as the name suggests – rely upon a different framework, *dependency grammars* (DG). The latter are based on the concepts of connexion, junction and translation [Weber 1992]. Connexion relates to hierarchical dependency relations between elements of the sentence, i.e., the assumption that one of a pair of directly related words is the governor and the other is the dependant. Figure 3.1b shows an admissible

[4]Compare to Joshi, Vijay-Shanker & Weir [1991].

syntactic dependency structure or – more precisely – the connexion relations of the dependency tree.

Junction and translation are less central in that they are only introduced for the specific phenomena of coordination and category conversion respectively. Both junction and translation are not explicitly supported by particular formal constructs in [W]CDG but have to be implemented by means of regular constraints and possibly additional levels. Note that especially a dependency model of coordination [Kunze 1972; Hesse & Küstner 1985] is inherently difficult because information about the coordinated constituents has to made available at the word node of the coordination which is lexically largely underspecified but must represent the whole coordinated phrase. No satisfactory treatment of coordinations which fulfills all requirements is available so far within the WCDG framework.

A grammar writer will almost always refer to an established dependency theory [Tesnière 1959; Kunze 1975; Hudson 1990], but the formalism of [W]CDG does not enforce a specific one. [W]CDGs may be used as long as only relations between word forms[5] are considered, only a limited number of edge labels is involved and the resulting structures obey the tree property. This last restriction to tree structures could even be relaxed but is usually useful for grammar modeling.

Other semantic representations like the *meaning text model* [Mel'čuk 1988; Kahane forthcoming] employ semantic units more complex than word forms which therefore cannot be modeled directly. Hence, semantic modeling in [W]CDG is restricted, e.g., to functor-argument structures.

[W]CDGs do not support 'competing dependencies', i.e., multiple governors for a single word on a single level, which are proposed for the treatment of discontinuities and non-projective structures in word grammar [Hudson 2000, p. 19]. However, words can have different governors on different levels so that an equivalent account can be modeled.

3.3 WCDG grammar formalism revisited

A brief informal introduction and discussion of the grammar formalism of weighted constraint dependency grammar (WCDG) is presented. For a formal definition see Section 2.3.

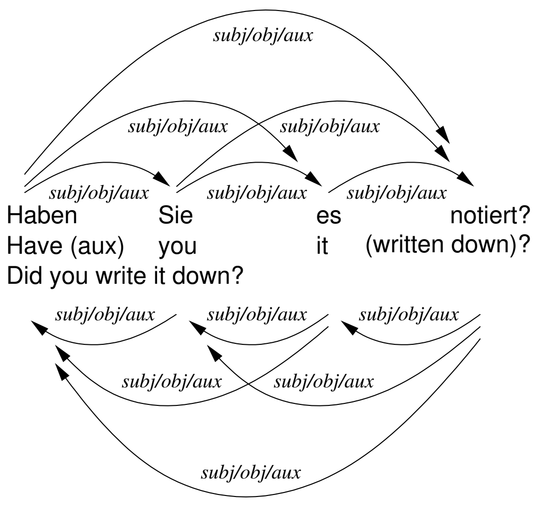
3.3.1 Linguistic structures and disambiguation

Assuming a simplified set of three different dependency relation types `subj(ect)`, `obj(ect)` and `aux(iliary)`, Figure 3.2a on the facing page shows all nine combinatorially possible dependency edges[6] for each word in a simple four word example sentence. This set is a compact representation of all conceivable structures, in this case $(9+1)^4 = 10,000$ of which $4^3 \cdot 3^3 = 1,728$ are connected trees.

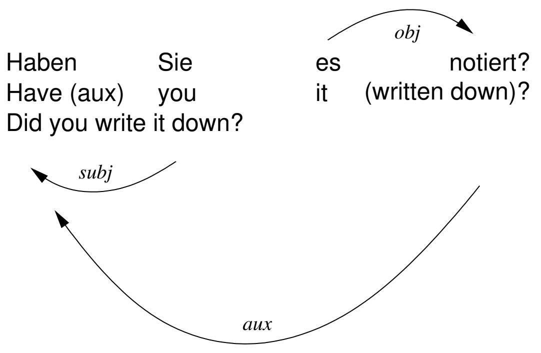
In [W]CDGs, there is no need for an additional component (such as context-free rules) that generates structures which are checked against compatibility constraints. This is a fundamental

[5]Dependency is often defined in purely linguistic terms and captures syntactic as well as semantic aspects of an utterance. The resulting relations of subordinations are usually anti-symmetrical.

[6]Additionally, a word can be the root of the tree which is not shown explicitly in Figure 3.2 on the next page.



(a) Space of possible dependency arcs



(b) Final solution for the example

Figure 3.2: Totally underspecified initial space of possible subordinations consisting of 72 edges and a fully specified unambiguous dependency analysis as one possible outcome of the disambiguation process (cf. Figure 3.3 on the next page).

difference to other constraint-based approaches [Shieber 1992] such as definite clause grammar [Pereira & Warren 1980], functional unification grammar [Kay 1984], PATR-II [Shieber 1984], HPSG [Pollard & Sag 1987; Pollard & Sag 1994] and general feature unification formalisms [Shieber 1986]. Instead, constraints have to be used to *exclude* some of the theoretically possible structures either by discarding them in the first place using hard constraints or by penalizing them with soft constraints so that the desired structures are ranked higher.

It is this eliminative nature of WCDGs that requires a substantially different approach to grammar development. For a context-free grammar to cover new phenomena it is neccessary to *add* rules while constraints have to be *retracted or softened* for a WCDG to analyze additional constructions correctly:[7]

$$\begin{array}{ll} \text{CFG:} & G \subseteq G' \Rightarrow L(G) \subseteq L(G') \\ \text{CDG:} & C \subseteq C' \Rightarrow L(C) \supseteq L(G') \end{array}$$

This counter-intuitive behavior makes it essential to verify all analyses and not just the new ones when extending the coverage. One such example turns up when one wants to allow non-projective structures in exceptional constructions but still requires projectivity to hold in the general case (cf. Section 3.4.5).

[7]Recent work on grammar induction [Schröder 1996; White 2000] focuses on the concept of support for a configuration rather than on that of elimination of unwanted structures. Under this view the traditional perception of grammar extension can be restored.

Figure 3.2b on the preceding page (a different depiction is given in Figure 3.3) shows the remaining edges after applying all constraints; the tree actually describes the unique best solution to this parsing problem.

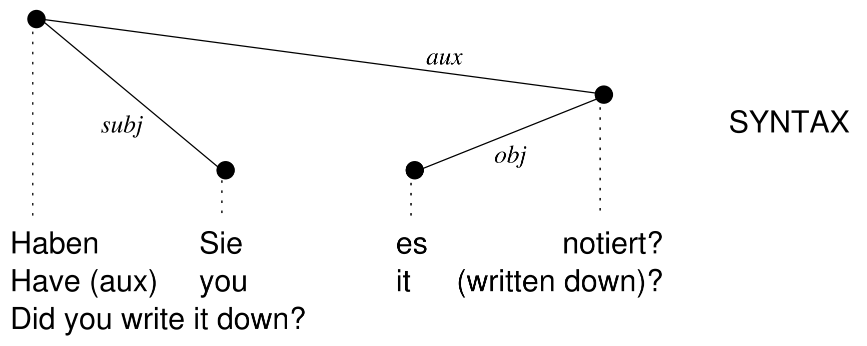


Figure 3.3: A syntactic dependency tree.

The labels in this example describe syntactic functions, e.g., `subj` marks the subject and `aux` connects the finite auxiliary verb with the past participle verb form. The finite verb does not depend on any other word form and is the root of the tree.

Traditionally only a single tree is considered during dependency analysis, namely the syntactic structure. WCDGs, in contrast, are not limited to one structural level or, more vividly, WCDGs can be used to build any number of dependency structures in parallel, not just a single syntactic one. Figure 3.4, for instance, describes some functor-argument relations for the previous example where the dependent is considered to be the argument.

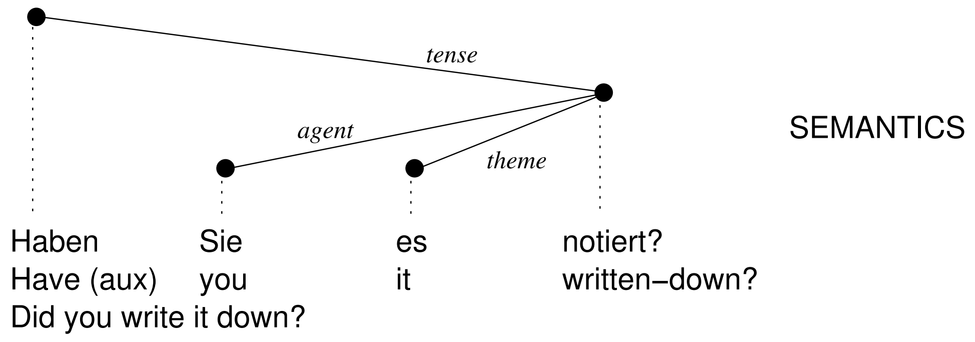


Figure 3.4: A semantic analysis belonging to the syntactic analysis of Figure 3.3.

Within the framework of WCDG, the two levels are independent in the sense that each can be constructed without recourse to the other one. However, they can be interrelated by giving appropriate constraints that couple them more or less tightly. During parsing, the structures on all levels are actually built simultaneously so that syntactic evidence not only influences the semantic structure but also vice versa. This architecture is compatible with the assumption that the human language system is organized as independent modules which interact with each other [Trueswell, Tanenhaus & Garnsey 1994].

As we shall see later, the ability to build multiple levels of description in parallel can be used for quite different purposes. Besides maintaining independent representations on a number of primary levels such as semantics (cf. for example Section 3.4.6), auxiliary ones can be used (cf. for example Section 3.4.1) to further constrain the structures on a main level.

Another generalization of WCDGs, if compared to traditional dependency grammars, is that a WCDG is quite flexible with regard to the target structures. While usually all dependency edges on each level should form a single tree, this is not enforced by WCDGs. If no single tree can be found, say, for the syntax level, an analysis may consist of disconnected subtrees each representing a partial parse. Section 3.5.4 addresses this issue in greater depth.

3.3.2 Constraints

A formal definition of constraints can be found in Section 2.5.

Constraints are propositions about the well-formedness of local configurations of edges in a dependency tree. They are used to encode grammatical restrictions that should hold for a natural language utterance.

```
C-7 {X} : SubjectVerbNoun : Subject : 0.0 :
      X.level=SYNTAX & X.label=subj -> X@cat=NOUN & X^cat=FINVERB
```

The subject is a noun and depends on a finite verb.

The most important part of a constraint is a logical formula expressing the condition imposed by the constraint. This formula is given in the second line of Constraint C-7.[8,9] When testing a dependency edge for compliance with a constraint, it is bound to the variable `X` and the formula is evaluated following the rules of an extended version of propositional logic. If it evaluates to false, the constraint is said to be violated, not fulfilled or not satisfied.

The formula of Constraint C-7 can be paraphrased as follows: For all dependency edges, it must hold that if the considered edge is on level `SYNTAX` and the label of the edge is `subj`, then the feature `cat` of the dependent word form of the edge must have a value `NOUN` and the feature `cat` of the governing word form must have a value `FINVERB`. Or following the literal formula less strictly: Subjects are nouns and modify finite verbs.

Different types of information about the edge can be accessed in constraints. Figure 3.5 on the following page shows the available information of one particular example edge that fulfills Constraint C-7 because both the premise (subject edge on syntax level) and the conclusion (noun depending on finite verb) of the formula are satisfied.

```
C-8 {X, Y} : SubjectBeforeObject : WordOrder : 0.1 :
      X.level=SYNTAX & Y.level=SYNTAX &
      X^id=Y^id &
      X.label=subj & Y.label=obj ->
        X@pos<Y@pos
```

[8]The first line in Constraint C-7 serves more technical purposes by introducing a variable that is a placeholder for a dependency edge (`X`), assigning a name to the constraint (`SubjectVerbNoun`), defining a group the constraint belongs to (`Subject`) and specifying the constraint weight (0.0). Finally, an optional comment is given underneath the constraint.

[9]In order not to complicate the representation, we will not quote too complicated constraints here. If the context is clear, we will sometimes leave out extra information like level restrictions.

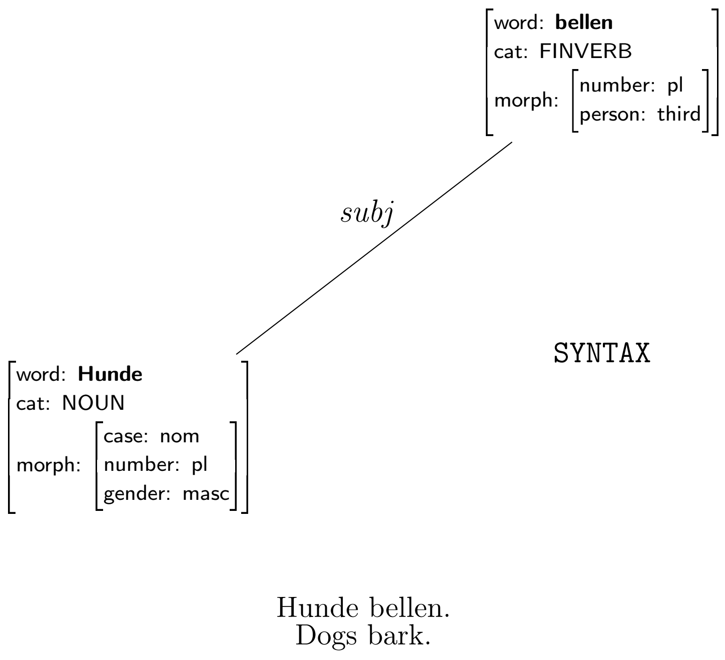


Figure 3.5: A dependency edge which satisfies Constraint C-7 on the page before.

The subject precedes the object.

Constraint C-8 on the preceding page looks at two dependency edges: the subject and the object edge (bound to variables `X` and `Y` respectively) of one and the same finite verb. It is violated in configurations where the subject follows the object as in Example E-41 (repeated from Example E-15 on page 4).[10]

E-41 Nice things you've been telling me!

Note that constraints only carry out a passive check of compatibility. In contrast, operations like unification create new objects and insert them into the space of possible structures. However, [W]CDGs depend on a selection from a uniform set of subordination possibilities which is pre-defined independently of the constraints themselves. Therefore, no operations such as value assignment or update are available. This restriction means that the compatibility of two substructures, i.e., their *potential* for unification or unifiability, can indeed be checked while the actual unification is not possible (cf. Section 3.4.2).

3.3.3 Critical appraisal of WCDG constraints

This section discusses several aspects that arise from how we defined constraints.

[10]This constraint may be part of a constraint cluster that is used to enforce the SVO order of languages like English. However, exceptions have to be included (probably as soft constraints) in order to facilitate an analysis of Example E-41.

Restricted constraints

In Section 2.6 we motivated the limitation – from a computational point of view – that any one constraint must not relate more than two dependency edges. This restriction to unary and binary constraints together with the lack of an assignment operation is a severe limitation since usually one wants to be able to write constraints about an arbitrary (and possibly unlimited) number of connected dependency edges in order to cope with some natural language phenomena such as non-local dependencies. Figure 3.6 gives a German example where certain dependencies can only be dealt with by long chains of edges. In order to constrain the agent of the infinite verb *fahren*, at least three edges have to be inspected. Possible solutions which allow one to cope with such situations are based on a combination of techniques discussed in Sections 3.4.6 and 3.5.5. Despite these mitigating techniques the restriction to binary constraints turns out to be a problematic limitation when modeling certain phenomena.

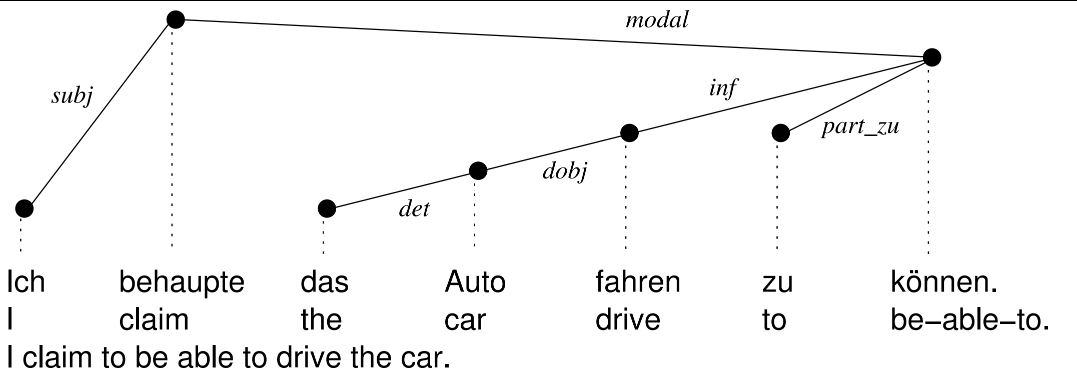

modal
subj
inf
part_zu
dobj
det
Ich behaupte das Auto fahren zu können.
I claim the car drive to be–able–to.
I claim to be able to drive the car.

Figure 3.6: A dependency tree for a German control verb construction.

As stated earlier, constraints are implicitly universally quantified. Not allowing direct existential quantification and limiting the arity of constraints are closely related. Intuitively, it is clear that universal constraints can easily be checked locally while an existence condition requires looking at the intermediate result from a global point of view. The same holds for the arity of constraints. The larger the arity of a constraint, the larger the substructures that have to be checked. In general, it is possible to express existence conditions with universally quantified constraints when the arity of the constraint equals or exceeds the number of edges in the parsing problem. Such constraints look at all edges at the same time and therefore can enforce the existence of arbitrary features. The following equivalence demonstrates this for the simplified case of only two word nodes (a and b):

$$\bigvee_{x \in \{a,b\}} f(x) \quad \equiv \quad \bigwedge_{x \in \{a,b\}} \bigwedge_{y \in \{a,b\}} x \neq y \rightarrow f(x) \vee f(y)$$

Although the grammatical framework neither allows an arbitrary arity nor existential conditions, some grammatical restrictions nevertheless need an existence quantifier:

- There must be a subject for a finite verb.

- Some nominal phrases need a determiner.
- The main part of a discontinuous word form (such as the French negation *ne … pas* or the German circumposition *um … willen*) must be accompanied by the other part.

Fortunately, it is possible to work around these limitations. Additional levels allow one to cope with situations where constraints of a higher order would be needed, and to indirectly (but not inelegantly) express existential requirements (cf. Section 3.4.1).

Domain of locality

The domain over which various dependencies can be expressed in a grammar formalism is usually called the *domain of locality* [Joshi & Schabes 1997]. In context-free grammars (CFG), it spreads over a local tree, i.e., a single rule.[11] Tree-adjoining grammars [Joshi & Schabes 1997] feature an extended domain of locality because more complex trees are the basic elements of the formalism. As always, an extended expressiveness has its price: The larger the domain of locality, the more basic elements exist and the harder it is to enumerate them all. In the extreme case of data-oriented parsing [Bod 1995], this results in an exponential number of subtrees that – at least in theory – have to be considered when parsing a sentence.

The domain of locality in the WCDG framework is both larger and smaller than that for CFG. On the one hand, the scope is rather limited because at most two dependency edges can be put in relation by any one constraint. On the other hand, constraints can relate *any* two edges, even if they are structurally and linearly distant from each other which is not directly possible in general even for tree-adjoining grammars. It is, for instance, no problem in WCDG to look at a node and the governor of its governor at the same time. Furthermore, WCDGs allow the grammar to define additional auxiliary levels which can serve as 'short-cuts' to transport and concentrate information about the sentence at certain points (cf. Section 3.4.7).

Separation of structure building and assessment

In WCDG parsing, the component that builds up structural hypotheses and the knowledge base that assesses those hypotheses are strictly separated. This property is shared with optimality theory [Prince & Smolensky 1993] but contrasts with the common practice in probabilistic grammars to associate judgments directly with grammar rules, for instance, in probabilistic context-free grammars [Charniak 1993; Krenn & Samuelsson 1996] which define probability distributions over all sets of rules which share a common left-hand symbol. Actually there is no device in WCDG that proposes feasible structures. It is simply assumed that *all* theoretically possible dependency trees have to be considered (which is equivalent to the assumption of a completely uninformed structural component).

Separating these two aspects has a number of advantages:

- The independence of the assessment from constructive rules allows for arbitrarily fine-grained formulation of well-formedness conditions. It is even possible to postulate (partially) contradicting preferences for one and the same linguistic phenomenon.

[11]We do not consider the fact that CFG that are enriched with feature-unification can freely percolate arbitrary information in the tree.

- As a related issue, the combinatorial explosion of structural rules that becomes necessary when combining different judgments with a single rule can be avoided [Menzel 2000, p. 7].
- Most importantly, parsing is not a constructive process any longer but a procedure of selection of the best among a set of alternatives. This eliminative nature can be used to guarantee that an analysis always comes up with a solution, even if it is heavily penalized.

3.4 General techniques for selected natural language phenomena

We will now describe how WCDG constraints can be used to model various grammatical phenomena. Fortunately, a great number of well-formedness conditions can easily be captured in a natural way, despite the imposed limitations. Other cases may require more complicated constraints or additional auxiliary structures that have no intuitive counterpart. Some phenomena can only be described approximately, i.e., a constraint grammar will cover most but not all instances of a certain construction. Finally, some kinds of phenomena cannot be modeled by restricted constraints at all unless an inordinate number of additional variables is introduced.

This section features mainly crisp constraints because we feel that the underlying principles can be better imparted if the additional complexity of exceptional cases and soft weights is omitted at first. However, we will point out candidates for soft constraints from time to time.

3.4.1 Valence

The important concept of *valence* models the observation that certain words tend to govern specific other words. To properly describe the valence of a lexeme, three conditions have to be considered:

- Valence possibility: A lexeme selects its argument, i.e., it determines what properties another lexeme must have to be a suitable dependent.
- Valence uniqueness: An argument position (or *slot*) must be uniquely specified, i.e., no two lexemes can fill the same slot of one lexeme.
- Valence necessity: While some arguments are optional, in many cases an argument is obligatory, i.e., there must be exactly one lexeme in an analysis that fills a specific argument position.

All three conditions can be ensured by WCDG constraints.

Valence possibility

In the following Example E-42 on the next page the verb *verschieben* opens three slots to be filled by appropriate arguments.

E-42	Wir	verschieben	das	Treffen	[auf	die	nächste	Woche].
	We	postpone	the	meeting	[until	the	next	week].

‘We postpone the meeting until next week.’

Figure 3.7 on the facing page shows the desired analysis of Example E-42 including the correct lexical entries for all word forms.

Modeling the valence possibility of a verb[12] like this means to describe what kind of words can fill the government requirements of the verb. Lexically, the subcategorization frame is encoded in the `args` feature. In this case, the first two argument positions model the obligatory nominal constituents while the last one refers to an optional prepositional phrase.

Since WCDGs have no notion of unification (which would come in handy for checking the compatibility of the verb with its arguments), constraints have to be formulated to guarantee that the arguments fulfill all of the individual requirements of the verb.

```
C-9 {X} : ValenceFirstArg : Valence :
      X.label=subj ->
        X^args:1:cat=X@cat &
        X^args:1:morph:case=X@morph:case &
        X^args:1:morph:number=X@morph:number &
        X^args:1:morph:person=X@morph:person
```

The first argument has to fulfill the valence requirements.

Constraint C-9, for example, checks the compatibility of the first argument with its governor. Note that the constraint is quite generic because it is almost completely lexically driven. Though the constraint writer has to explicitly specify all features that should (potentially) agree, the individual characteristics of the argument, e.g., its category, and the specific claims of the governor are completely lexicalized. Assigning a soft score to the constraint would allow the violation of the valence requirements by a deviant utterance.

Uniqueness of valence fillers

In the previous section, Constraint C-9 was used to enforce conditions that an argument must fulfill, such as agreement. However, the constraint cannot ensure the uniqueness of certain arguments. Nothing so far prevents an analysis with, e.g., the syntactic function `subj` assigned to two different lexemes if they both carry the required case, i.e., nominative, as in Example E-43.

E-43	Wir	treffen	die	Kollegen
	Pro,nom	Vfin	Det,{nom,acc}	Noun,{nom,gen,acc,dat}
	We	meet	the	colleagues.

Only constraint C-10 on page 56 ensures that any two dependency edges do not assign the same function `subj` to different lexemes.

[12]For ease of presentation, we restrict ourselves to the valence of verbs; the same pattern applies to different categories with valences as well, e.g., nouns.

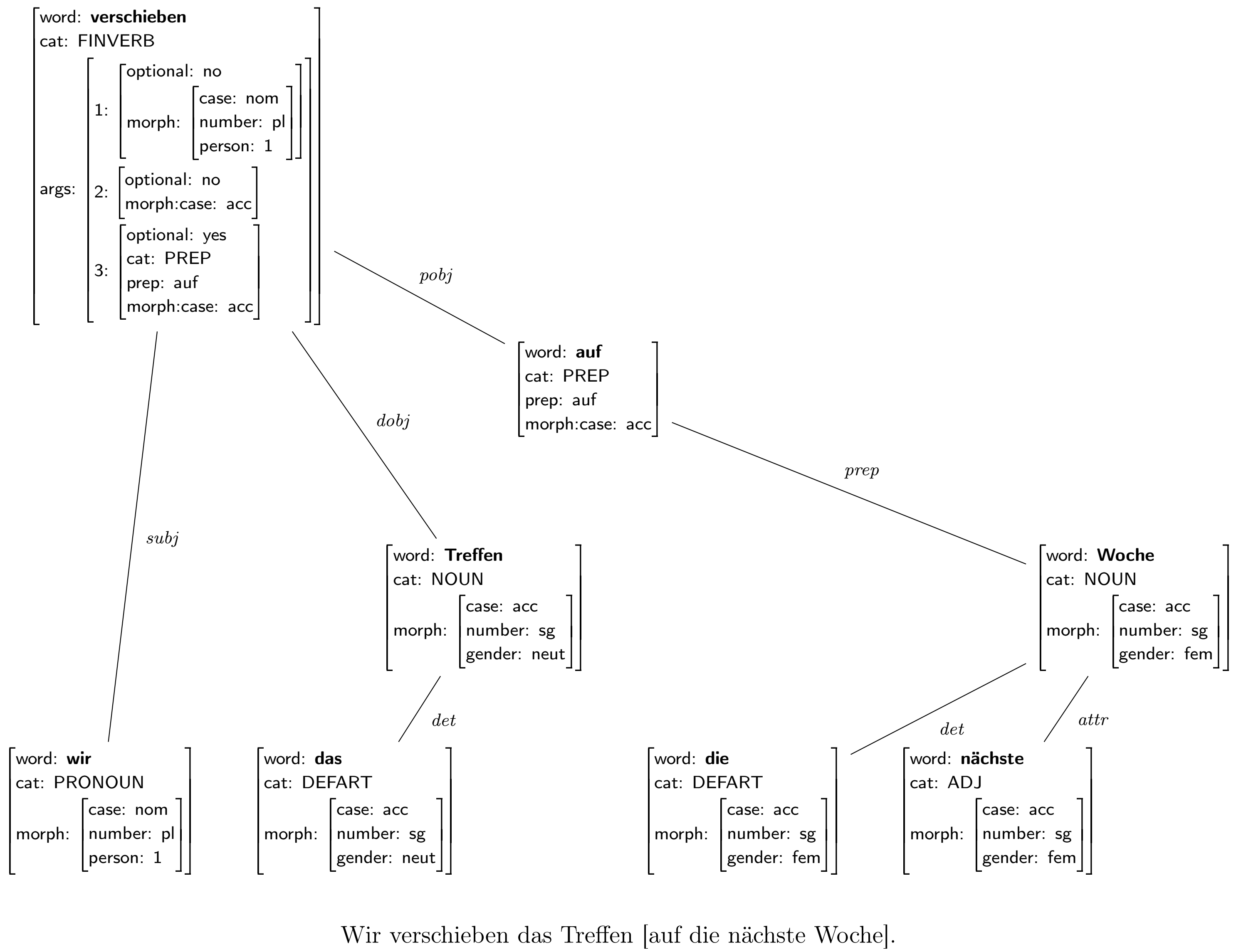


Figure 3.7: Dependency analysis with correctly chosen lexical entries.

```
C-10 {X, Y} : SubjUnique : Valence :
       X.label=subj & X^id=Y^id -> Y.label!=subj
```

The subject for a given word form is unique.

Note that while it was sufficient to examine a single edge in order to check, for example, agreement, one has to constrain *pairs* of dependency edges for the uniqueness property.

Valence necessity

Modeling valence possibilities and the uniqueness of valence fillers is not sufficient. What we cannot express up to now is the condition that there is *at least* one instance of a specific argument type. Only by adding this kind of restriction to the grammar can we make sure that *exactly* one word form exists that fills a specific valence slot.

It is not possible to write a constraint that directly expresses this condition in the restricted formalism: The presence of an edge with specific properties could only be enforced by examining all edges at once or by employing an existential quantifier, whereas our restricted constraints may only relate two edges at most and must be universally quantified. One can overcome this problem by introducing an auxiliary level. Its purpose is to establish the inverse edge between the two lexemes, i.e., whenever a governor dominates its argument on the syntax level, the argument will dominate the governor on the auxiliary level [Maruyama 1990a]. A constraint can now check whether the valence slot of a lexeme is filled by testing whether the lexeme depends on another form on the auxiliary level. Note that as many of these auxiliary levels are required as there are different valence requirements of a single lexeme. These additional levels are just a technical means to compensate for the lack of an existence quantifier. Generally, they do not have a linguistic interpretation. Figure 3.8 on the facing page shows how the syntactic level and an auxiliary level are used in combination.

Since the edges of the auxiliary level do not build complex tree structures, it is possible to represent them as simple arcs drawn below the syntactic tree. Figure 3.9 on page 58, therefore, contains the same information as Figure 3.8 on the facing page.

Finally, the syntactic dependency edge and its inverse edge on the auxiliary level need to be coupled to ensure that they can only occur together. This is achieved by Constraint C-11: A lexeme selects another lexeme as its first argument on the auxiliary level if and only if there is an inverse dependency edge on the syntactic level with the label `subj`.

```
C-11 {X, Y} : Aux1SyntaxMapping : ValenceExistence :
       X.level=SYNTAX & Y.level=AUX1 ->
       ( X^id=Y@id & X.label=subj <-> X@id=Y^id )
```

The first syntactic argument (subject) is mirrored on the first auxiliary level.

Constraint C-12 restricts the scope of the auxiliary level to those lexical items that are specified for a first argument by forcing the link to the root of the tree if no argument is possible.

```
C-12 {X} : Aux1Root : ValenceExistence :
       X.level=AUX1 & ~exists(X@args:1) -> root(X^id)
```

Words without a first complement point to the root on the `AUX1` level.

Constraint C-13 on the next page ensures that the subject slot of a finite verb is filled.

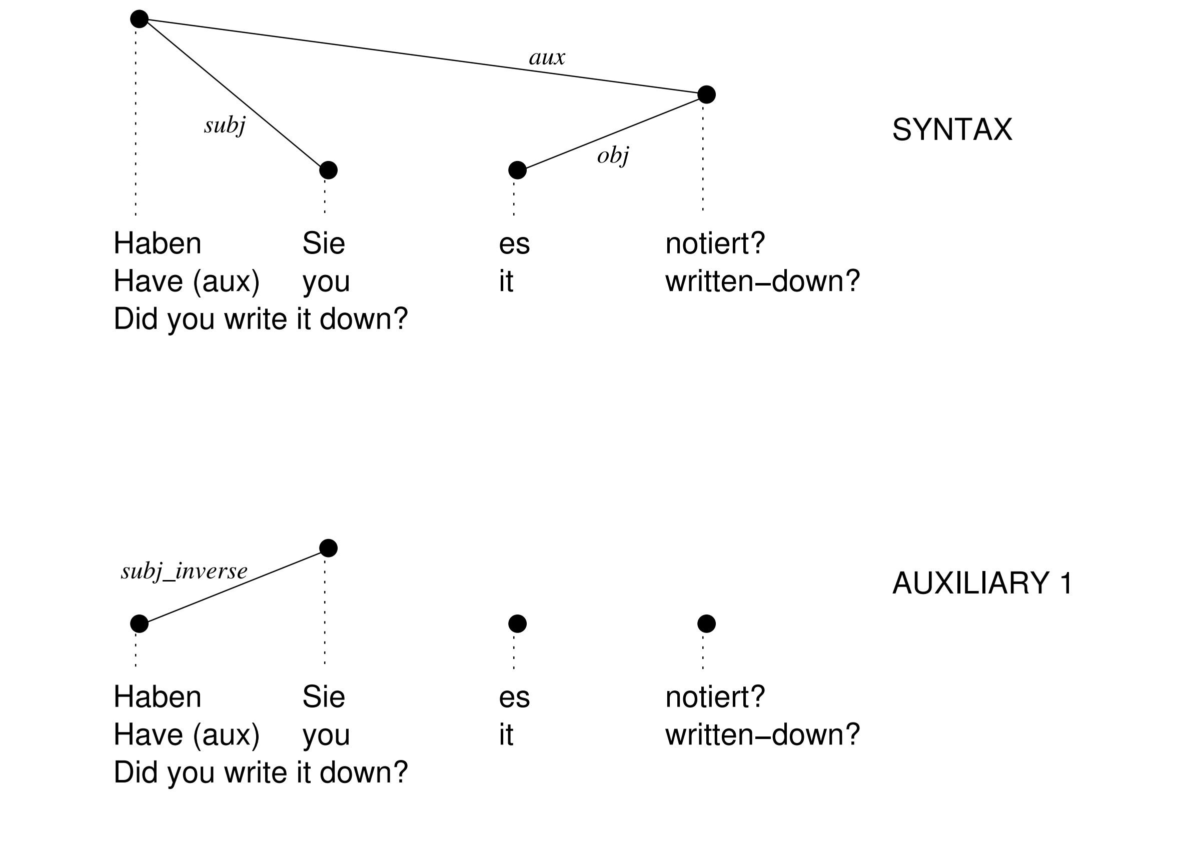


Figure 3.8: Dependency analysis for the main level SYNTAX and an auxiliary level for the first argument.

```
C-13 {X} : Aux1Existence : ValenceExistence :
       X.level=AUX1 & X@args:1:optional=no ->
         ~root(X^id)
```

Words with obligatory first arguments must modify the latter on the first auxiliary level.

Although these constraints jointly express an existence condition[13], they fulfill the conditions stipulated above: At most two edges are considered, and no existential quantifier is used. Again introducing soft weights for certain constraints would facilitate the analysis of sentences with missing arguments.

3.4.2 Feature percolation

Feature transport refers to the process of carrying some kind of information along dependency chains of possibly arbitrary length. This mechanism is required to describe natural language phenomena where constraining information is applied at a particular node but originates from a structurally distant one.

[13]As a side effect, Constraints C-11 on the facing page and C-13 also guarantee the uniqueness of the argument, which has already been enforced by the binary Constraint C-10 on the facing page.

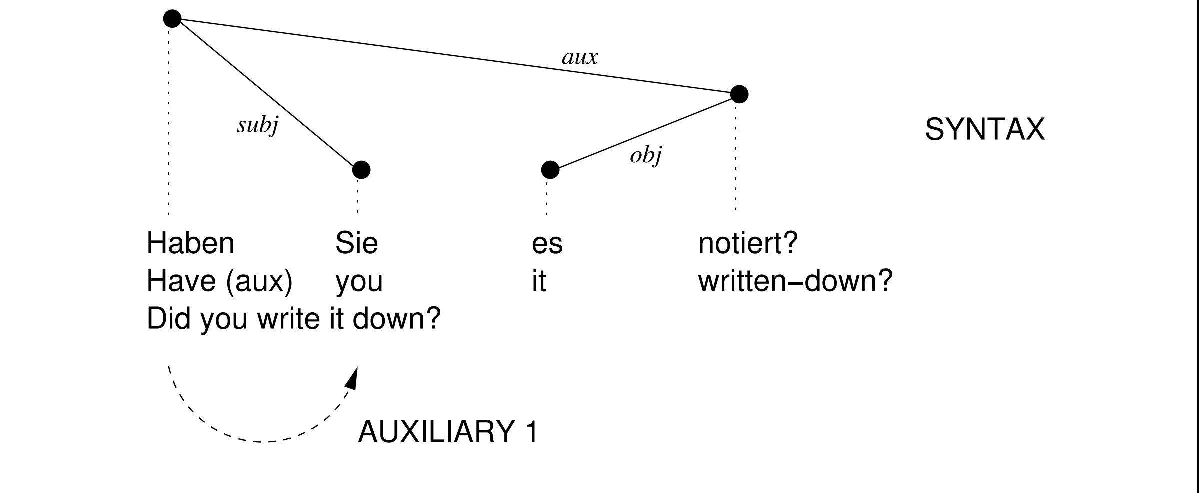


Figure 3.9: Dependency analysis with an alternative presentation of the first auxiliary level.

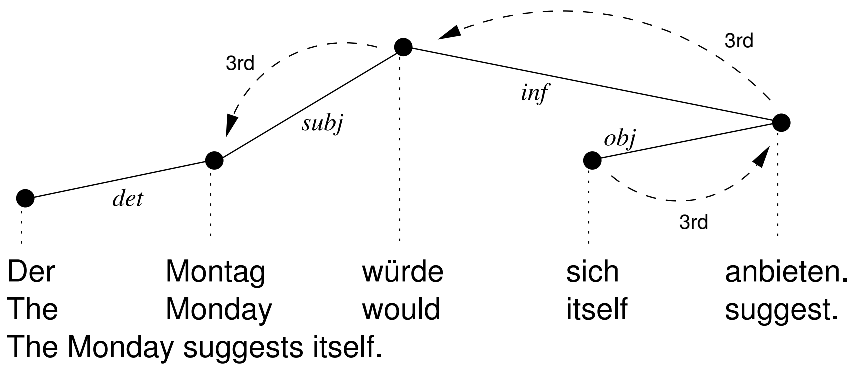

3rd
3rd
inf
subj
obj
det
3rd
Der Montag würde sich anbieten.
The Monday would itself suggest.
The Monday suggests itself.

Figure 3.10: Feature percolation in a dependency tree.

In Figure 3.10, it is necessary to have access to the `person` feature of the reflexive pronoun *sich* at a node where its antecedent can be identified in order to establish the required agreement. The information has to percolate all the way up the verb chain.

Because WCDG is based solely on passive feature checking, it provides no direct mechanism for feature transportation. Therefore it is not possible for a node to 'inherit' properties from another node (which could only be achieved through some kind of assignment or unification). The dashed arrows indicate the way that the `person` feature of the word *sich* would have to travel to be matched against the word *Montag* but no transport of this information can take place.

One possibility to overcome this deficiency is to introduce labels for the dependency edges that encode more information than just the syntactic function. The labels `obj`, `inf` and `subj` could be augmented to the labels `obj_3rd`, `inf_3rd` and `subj_3rd` so that the person agreement could be checked at both ends of the chain.

There are several disadvantages to this approach. Since labels can be accessed only as atomic values, constraints would become much more complicated, having to encode and decode the information contained in the labels several times. While this is more of a technical complication, the considerable increase in the number of possible labels is highly problematic since $n \cdot m$ different labels are necessary, where n is the number of syntactic functions and m is the number of different values the percolating feature may take.

Since the computational complexity grows polynomially in the number of labels, this method is rejected and Sections 3.4.7 (additional levels) and 3.5.5 (approximation of high-order constraints) introduce mechanisms that are better suited to solve the problem of feature transportation.

3.4.3 Word order

Word order is a complex phenomenon and some of its aspects are (partly implicitly) dealt with in other sections in this chapter, e.g., *vorfeld* realization in Section 3.4.4, linear ordering of complements in Sections 3.3.2, 3.5.1 and 3.5.2 and projectivity in Section 3.4.5. Here, we restrict ourselves to simple aspects of word order.

The standard linear ordering of an adjective and its governing noun, for example, differs for German and French in that in German the adjective precedes the noun while it often is the other way around in French (cf. Examples E-44 and E-45).

E-44	Ein	großer	Hund	...
	A	big	dog	...

E-45	Le	traitement	automatique	...
	The	processing	automatic	...
	‘The automatic processing...’			

Since the restriction on word order is directly connected to the dependency between adjective and noun, it is easy to formulate Constraint C-14 that excludes constructions with a wrong word order from the set of well-formed German sentences. Section 3.5.1 extends this example to soft constraints.

```
C-14 {X} : AdjNounWordOrder : WordOrder :
       X.level=SYNTAX & X@cat=ADJ & X^cat=NOUN -> X@pos<X^pos
```

An adjective precedes its noun.

As long as only up to two dependency edges are needed to constrain the word order, it is quite easy to write the corresponding constraints. Note, however, that some word order phenomena depend on deeply embedded structures and cannot easily be described by a single constraint. The next section gives an example of how a grammar writer can deal with such complex constructions for word order.

3.4.4 Vorfeld realization

The word order at sentence level in German is characterized by so-called topological fields (*topologische Felder* or *Satzfelder*) [Grewendorf, Hamm & Sternefeld 1987] which correspond roughly to positions in a sentence which again may be filled by no, one or more constituents.

Vorfeld	Left Sentence Bracket	Mittelfeld	Right Sentence Bracket	Nachfeld
Ich	habe	den Mann	gesehen.	
Den Mann	habe	ich	gesehen.	
Wen	habe	ich	gesehen?	
Ich	habe		gesagt	daß ich nach Berlin komme.
	daß	ich nach Berlin	komme.	

Figure 3.11: Topological fields in German sentences.

Figure 3.11 contains several examples of the assignments of constituents to the topological fields.

The *vorfeld* rule of German states that in a declarative main clause, the finite verb is preceded by exactly one constituent. This condition can be formalized by Constraints C-15 through C-17. These constraints ensure that a finite verb directly governs exactly one node on its left. Note how an auxiliary level VORFELD is used to ensure that the *vorfeld* slot is filled in much the same way as in Section 3.4.1. First the Constraint C-15 forces each finite verb to select a word to its left as its *vorfeld.* Then constraints C-16 and C-17 make sure that the main level `SYNTAX` is properly coupled with the auxiliary level VORFELD.

```
C-15 {X} : VorfeldInit : Vorfeld :
       X.level=VORFELD ->
         (X@cat=FINVERB -> ~root(X^id) & X^pos<X@pos)
         &
         (X@cat!=FINVERB -> root(X^id))
```

Only finite verb must point to the vorfeld to the left.

```
C-16 {X, Y} : UnderVerbVorfeld : Vorfeld :
       X.level=SYNTAX & Y.level=VORFELD &
       X^id=Y@id & X@pos<X^pos & Y@cat=FINVERB ->
         Y^id=X@id
```

A constituent depending on the verb from the left must occupy the *vorfeld.*

```
C-17 {X, Y} : VorfeldUnderVerb : Vorfeld :
       X.level=SYNTAX & Y.level=VORFELD & Y^id=X@id ->
         X^id=Y@id
```

The *vorfeld* constituent depends on the finite verb.

Note that a complete vorfeld treatment with subordinate clauses and non-projective constructions requires more complicated constraints with exceptions and possibly soft constraints.

Separating word order phenomena from dependency structures has first been suggested by Bröker [1998] by postulating an additional so-called word order domain structure. An extension of the Stellingen grammar to an entire model of topological fields is subject to further research. In particular, a treatment along the lines of Duchier & Debusmann [2001] who propose a complete topological dependency tree in addition to the tree of syntactic dependencies is compatible with our approach.[14] An alternative suggestion is due to Gerdes & Kahane [2001] who postulate an

[14]A formalization is given in Duchier [2001].

intermediate phrase structure to model topological fields. Assuming such a model of topological fields is feasible within the framework of WCDG, the integration of supplementary preferences about the word order within the middle field (*Mittelfeld*) [Uszkoreit 1987] is straight-forward.

3.4.5 Projectivity

A general rule about constituents is that they are nested structures of adjacent words, i.e., a constituent may be contained within another, but two constituents may not partially overlap. This rule can be formulated in terms of *projectivity*: A dependency tree is projective if every word node can be projected to the base of the tree without crossing a dependency edge. An analysis is projective if and only if a projective dependency tree can be drawn for that analysis. Throughout this chapter, we present projections as dotted lines and dependency edges as solid lines in the figures. For example, Figure 3.12 shows a non-projective dependency tree for a non-projective analysis, with the instances of projectivity violations marked by circles.

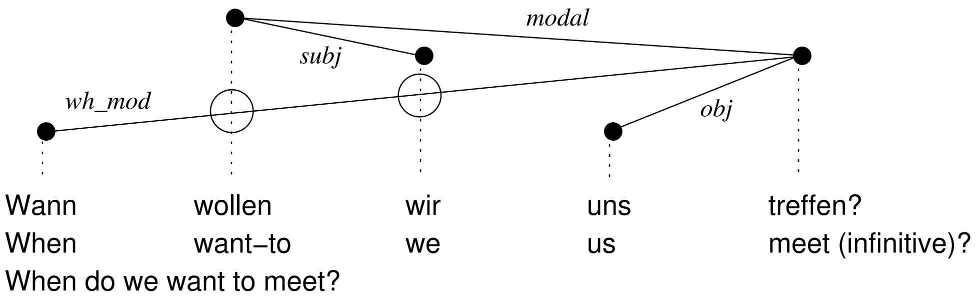

modal
subj
wh_mod
obj
Wann wollen wir uns treffen?
When want–to we us meet (infinitive)?
When do we want to meet?

Figure 3.12: A non-projective dependency tree.

For efficiency reasons, one often wants to restrict the syntactic analyses to projective ones [Kahane, Nasr & Rambow 1998; Sleator & Temperley 1991; Courtin & Genthial 1998]. Nevertheless, there are some natural language phenomena for which it is difficult to establish a projective analysis, such as the surface strings resulting from wh-movement.

In general we demand projectivity, but allow non-projective structures for some exceptions (such as modal verb constructions like the one in Figure 3.12).

For connected dependency trees, projectivity can be enforced by the following constraints C-19 and C-20 on the following page.[15] For reasons of simplicity, we have eliminated those parts of the constraints that deal with exceptional cases for which we explicitly allow non-projectivity.

```
C-19 {X, Y} : ProjectivityDirect  : Projectivity :
       X.level=SYNTAX & Y.level=SYNTAX & X^id = Y@id ->
         Y^pos <= min(X@pos, Y@pos) | Y^pos>=max(X@pos, Y@pos)
```

A dependency cannot cover an ancestor.

[15]Maruyama [1990a] used a similar Constraint C-18 for the same purpose:

```
C-18 {X, Y} :  Maruyama :  Projectivity :
     X^pos<Y@pos & Y@pos<X@pos ->
        X^pos<=Y^pos & Y^pos<=X@pos
```

Hudson [2000, p. 30] uses a combination of no-tangling, no-dangling and sentence-root principles and restricts their applicability to the surface structure.

```
C-20 {X, Y} : ProjectivityNoCrossing  : Projectivity :
       X.level=SYNTAX & Y.level=SYNTAX & min(X@pos, X^pos) < min(Y@pos, Y^pos) ->
         max(X@pos, X^pos) >= max(Y@pos, Y^pos)
         |
         max(X@pos, X^pos) <= min(Y@pos, Y^pos)
```

Dependency edges do not cross.

Figure 3.13 on the facing page proves that these constraints forbid all cases of non-projectivity.

Note that constraint violations for non-projective structures are not always visible at the same location in the tree where we mark them with circles. The non-projectivities in Figure 3.12 on the page before, for instance, are both covered by a single violation of Constraint C-19 on the preceding page.[16] When the edge from *wann* to *treffen* labeled `wh_mod` is bound to variable `X` and the edge from *treffen* to *wollen* labeled `modal` is bound to `Y` the term `Y^pos` (position of *wollen*) is neither smaller than the minimum nor larger than the maximum of `X@pos` (position of *wann*) and `Y@pos` (position of *treffen*). This corresponds to case (Ia) in Figure 3.13 on the facing page with A representing *wann*, B *wollen* and C *treffen*.

Relative clauses with inverted subject-object word order as in Figure 3.14 on page 64 are instances where the non-projectivity is detected by Constraint C-20. In this case the dependency edges that cause the violation are not connected (case (Ib) in Figure 3.13 on the facing page).

3.4.6 Thematic roles

The representation of semantic information in CDG is hampered by the fact that only direct relations between word forms can be established. A grammar that employs only binary constraints cannot always investigate an entire constituent, but only its head word. Since features from subordinated words are not available at the head word, no compositional semantic structure can be built. Nevertheless, it is possible to model a number of selection and subcategorization phenomena. Furthermore, special problems of semantics, e.g., reference resolution, can often be solved by postulating additional levels of analysis (cf. Section 3.4.7).

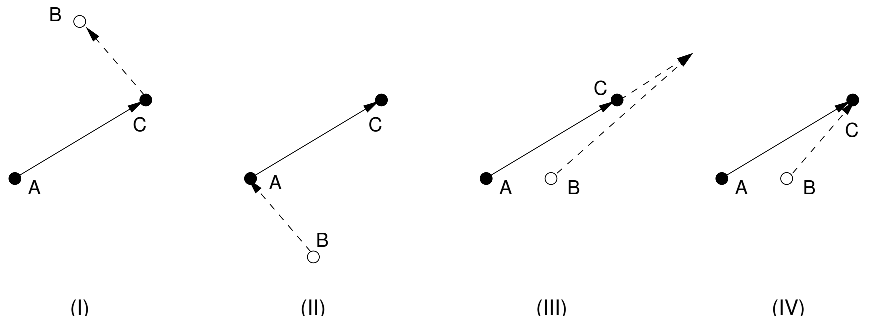
In principle, the very same mechanisms that enable the analysis of syntactic head-complement structures can be used to establish a kind of functor-argument structure. There are, however, a number of different possibilities for representing such a structure by means of dependency relations.

Since in many cases, especially with complex verb groups, the syntactic and semantic structures differ considerably (cf. Figures 3.3 on page 48 and 3.4 on page 48), an obvious solution would be to create an autonomous semantic level in analogy to the syntactic one. Semantic relations between word forms are represented as dependency edges on that level, and the kind of relation (e.g., the thematic role) can be notated as the label of the edge.[17] This approach is conceptually simple, but introduces an additional problem. While syntactic analyses are traditionally trees, i.e., each word form has a unique governor, such an assumption is not necessarily justified for

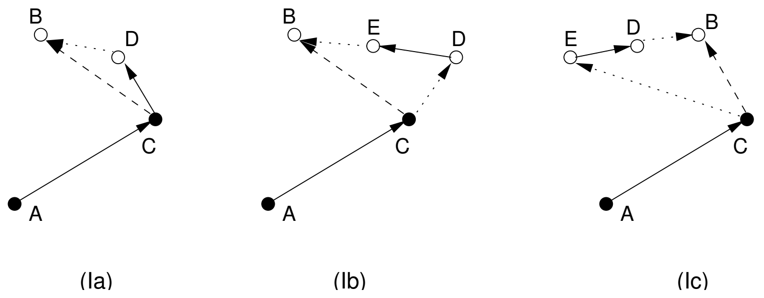
[16]It is questionable whether it is a shortcoming to detect only a single violation although two or possibly more intersections of projection lines and dependency edges exist (circles in Figure 3.12 on the page before). Arguably one can consider this a single non-projectivity caused by the subordination of the first under the last word.

[17] Word grammar [Hudson 1990; Hudson 2000] uses a similar approach but allows extra edges in the syntactic structure itself, rather than introducing additional levels.

Let V be the set of word nodes in the tree. Let $\rightarrow \subset V \times V$ be the direct subordination relation and $\rightarrow^*$ the transitive closure thereof, the partial order of subsumption. Let further $< \subset V \times V$ be the total order of linear order, i.e., $A < B$ means A appears to the left of B.
The following figure shows all possible configurations for three nodes $A < B < C$ with $A \rightarrow C$. Only configuration (I) is a non-projective one.



The next figure looks at this configuration in greater detail. There are only three possible expansions for the path $C \rightarrow^* B$:



(Ia) No edge on the path ends to the right of node C. The first edge $C \rightarrow D$ on the path $C \rightarrow^* B$ in combination with edge $A \rightarrow C$ is forbidden by Constraint C-19 on page 61.

(Ib) At least one edge on the path ends between node A and node C and starts to the right of node C. Edges $A \rightarrow C$ and $D \rightarrow E$ are forbidden by Constraint C-20 on the facing page.

(Ic) symmetric with configuration (Ib)

Figure 3.13: Proof (by case differentiation) that Constraints C-19 on page 61 and C-20 on the facing page ensure projectivity of dependency trees.

semantics. In contrast to syntactic subordination, it is quite common for a word form to fill more than just one semantic slot.

In Example E-46 the word form *Mann* fills at least two roles: It is the agent both for the 'telling' and the 'laughing' event. All attempts to represent both relations in a single tree are artificial and lead to additional problems, e.g., by introducing extra labels (cf. Section 3.4.2).

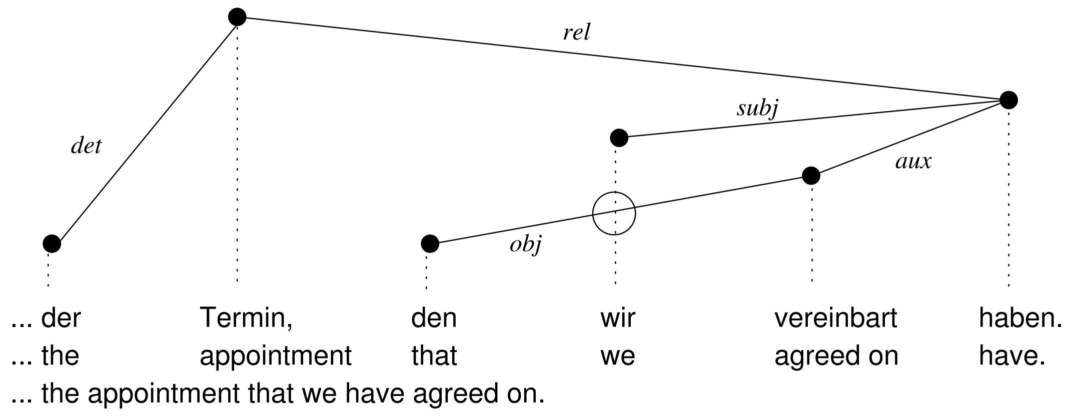

rel
det
subj
aux
obj
... der Termin, den wir vereinbart haben.
... the appointment that we agreed on have.
... the appointment that we have agreed on.

Figure 3.14: A non-projective dependency tree violating Constraint C-20 on page 62.

E-46	Der	Geschichten	erzählende	Mann	lacht.
	The	stories	telling	man	laughs.
	‘The man telling stories laughs.’				

Heinecke & Nolda [1998], for instance, keep the concept of a single semantic tree but allow *reversed* semantic edges with appropriate labels in cases where a word fills more than one semantic slot.

An alternative solution is to distribute the semantic representation across a number of additional levels (one for each type of semantic role) on which the functor is the dependant, i.e., the relation is reversed if compared to the analysis in Figure 3.4 on page 48. Now the unique specification of the governor can be maintained. The disadvantage of requiring an entire set of new levels is compensated for by the natural and flexible representation of semantic relations. On each level, word forms that have a corresponding semantic slot to fill, ‘depend on’ or select the slot filling word forms. Figure 3.15 on the next page shows three semantic edges originating from two semantic levels below the syntax tree. The use of arrows in Figure 3.15 on the facing page also emphasizes that word forms with open semantic slots select or point to the word forms that fill the slots.

3.4.7 Reference resolution

As already mentioned in Section 3.4.6, many different problems of natural language analysis can be addressed by the introduction of additional levels in WCDG. One such problem is feature percolation (cf. Section 3.4.2) that can be addressed by directly connecting the source and the destination of a percolating feature on a separate level.

Another example is reference resolution, i.e., identification of those word forms that anaphoric pronouns refer to.

On a special level, say REF, referring pronouns select their antecedents (cf. Section 3.4.6 for an in-depth explanation of the general technique of introducing an additional level on which specific

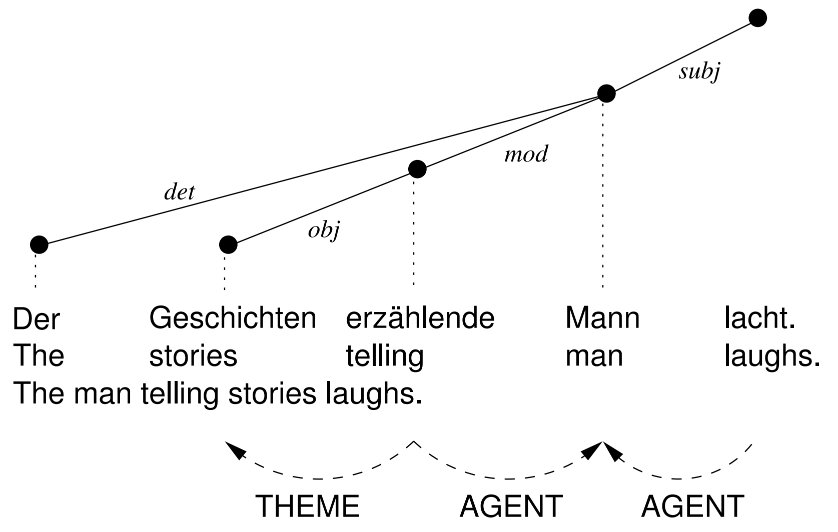

subj
mod
det
obj
Der Geschichten erzählende Mann lacht.
The stories telling man laughs.
The man telling stories laughs.
THEME
AGENT
AGENT

Figure 3.15: A syntactic dependency tree with two additional levels for thematic roles.

selectional processes are carried out). Constraints dealing with properties of relations between referent and antecedent can then easily be formulated for dependency edges on that level.

In Example E-47 the relative pronoun *der* has two possible antecedents: *Ehemann* and *Frau*.

E-47	Ich	sehe	den	Ehemann	der	Frau,	der	schläft
			Det,acc,masc	Noun,acc,masc	Det,gen,fem	Noun,gen,fem	Rel,nom,masc	
	I	see	the	husband	(of the)	woman	who	sleeps.

‘I see the sleeping husband of the woman.’

Constraints like C-21 have access to dependency edges on the level REF and constrain them to those where, for example, the pronoun and the noun agree with respect to gender and number.

```
C-21 {X} : RelativeGenderNumber  : RelativeSentence :
       X.level=REF & X@cat=RELPRONOUN ->
         X@gender=X^gender & X@number=X^number
```

Relative pronoun and antecedent agree with respect to gender and number.

Figure 3.16 on the next page presents a possible analysis for Example E-47. Note that this structure departs form the classical analysis of relative clauses [Tesnière 1959] in so far as it does not have to assume a double position for the relative pronoun, both as a governor of the whole relative clause and as a regular dependent pronoun [Gerdes & Kahane 2001, Section 2.9].

Other restrictions, such as those involving quantifier scope or discourse [Kamp & Reyle 1993], cannot easily be modeled because firstly the domain of a WCDG analysis is usually a single sentence and secondly the logical inventory is generally too weak.

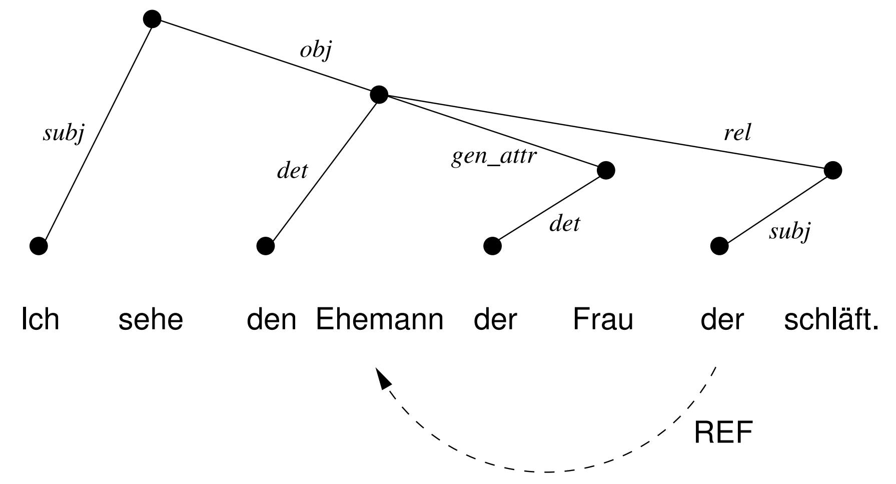


Figure 3.16: An analysis of a sentence with a relative clause including reference resolution.

3.5 Modeling gradation using weighted constraints

In order to emphasize certain techniques of WCDG grammar modeling in the previous section, we only used constraints as strict conditions, i.e., a valid dependency structure is not allowed to violate any of these constraint. We now come back to gradation in natural language which can only be (or is at least much better) described in terms of soft constraints that encode preferences rather than absolute conditions (cf. Section 1.1).

Grammatical conditions can then be subdivided into a number of classes, partially along the dimension of constraint strength, partially depending on their interaction with related constraints.

Certainty: Conditions that must hold in any case in a valid solution are modeled as hard constraints. Even if one allows almost all grammatical rules to be violable, hard conditions are used to define the 'fall-back' cases; otherwise too many alternative equivalent structures would be possible in case of an erroneous utterance.

Well-formedness: Grammatical rules are formulated as soft constraints that receive a small weight (near zero). This ensures that language errors in utterances do not lead to a breakdown of the system, but also takes into consideration that a violation is a severe fault in language use.

Preference: Preferential information enters the system in the form of constraints with high weights (near one). They allow the distinction between otherwise equivalent candidates but (usually) cannot overrule a well-formedness constraint.

Spurious ambiguity: Some constraints with weights very close to one have the sole purpose of avoiding spurious ambiguities that do not correspond to any real linguistic ambiguity in the sentences.

Markers: Constraint with a weight of exactly one do not influence the score of a structure. Their only purpose is that their violation indicates that a certain condition is fulfilled, such as the existence of a relative clause in the sentence. This completes the hierarchy of constraints that is defined by an increasing weight, from zero (certainty) to one (markers).

Information fusion: Constraints that integrate external judgments, such as acoustic scores from a speech recognizer or part-of-speech tag probabilities, are special because their weights must be carefully balanced with the rest of the grammatical constraints. No general rule about the score range can be given for them. See Section 5.6 for an example.

Defaults: Default values are a mixture of previous types: The space of possibilities is described with certainty conditions and all values except the default are dispreferred using preference constraints.

Approximation of higher order constraints: Soft constraints are well suited for encoding complex conditions that could otherwise not be modeled with the restricted constraints.

Dynamic constraints: Dynamic constraints do not have a fixed weight, but receive different scores depending on the context in which they are evaluated (cf. Section 2.5.1).

A WCDG with soft constraints defines a hierarchy of possible parses, each one annotated with a score and the list of soft constraints that are violated by the particular structure. Soft constraints have the additional advantage that they allow one to integrate all kinds of preferences into the analysis, so that even if the utterance is really ambiguous the parsing system can rank its parses based on the preferences of the grammar. A practical system may then choose to either use only the best solution or more than one parse; whichever seems more appropriate depending on the particular application and on the scores of the structures.

Note that the introduction of weighted constraints can have two almost contradictory consequences:

- If hard constraints are changed so that they may be violated, then those utterances that violate these constraints become parsable. Of course, the search space for the parsing problem increases as more dependency edges remain acceptable.
- On the other hand, ambiguous analyses for an utterance can be eliminated by introducing additional constraints with high scores. Such constraints can assign slightly different scores to competing analyses so that only one of them is selected as the solution. These preferences also guide the analysis process so that the most promising hypotheses are tried first. Thus, the expansion of the space of possible solutions (see above) can be counterbalanced by a more goal-oriented disambiguation.

3.5.1 Well-formedness

Although speakers most often follow the rules of a language, there are always exceptions: Agreement requirements are ignored, words are omitted, sentences are truncated, word order regularities are not satisfied etc.

Constraints in the previous section encode crisp 'grammatical' rules. In order to be able to parse deviant utterances these constraints have to be extended by exceptions and non-zero weights.

For instance, Section 3.4.3 featured Constraint C-22 (repeated from Constraint C-14 on page 59) that forces an adjective to precede its governing noun in a German noun group.

```
C-22 {X} : AdjNounWordOrder : WordOrder :
       X.level=SYNTAX & X@cat=ADJ & X^cat=NOUN -> X@pos<X^pos
```

An adjective precedes its noun.

While this constraint is correct also for English from a text book point of view (cf. Example E-44 on page 59), there are exceptions to the rule where it makes sense to allow for post-modifying adjectives as well. Examples E-48 [Cowie 1989] through E-51 are such instances of different types.

E-48 (a) Students have to do a year's preparation before they start the degree course proper.

(b) She hadn't had a proper holiday for years.

(c) A hammer is the proper tool for the job.

E-49 The task could not be finished in the time available.

E-50

Forelle	blau
Trout	blue

'Truite bleu'

E-51

Es	gibt	hier	keine	Bücher...	alte.
It	exists	here	no	books...	old.

'There are no books here, old ones.'

Of course, it is always possible to argue for alternative (more appropriate) analyses of these examples that differ from that of an adjective modifying a noun: One might want to provide multiple lexical entries for the word *proper* because it has a different meaning in Example E-48a than in Examples E-48b and E-48c; an analysis of Example E-49 as a reduced relative clause might seem more appropriate;[18] Example E-50 could be considered a fixed term originating from the French translation[19]; the adjective in the spontaneous utterance of Example E-51 could be analyzed as extraposed. However, offering a special treatment does not solve the problem completely. Alternative analyses can only be provided for anticipated constructions. But even for foreseen deviations from the standard, a grammar writer may abstain from arranging a special treatment for them, for instance, because the construction is computationally expensive to parse or difficult to express in the formalism. The point here is that a simple standard analysis can easily be extended to deviating cases so that even unforeseen constructions can be parsed (although maybe in a linguistically suboptimal way).

Constraint C-23 on the next page demonstrates two major methods to extend the coverage of the grammar to non-standard utterances. Firstly, and most importantly, the constraint gets a soft weight of 0.1, making post-modifying adjectives possible but penalized. Secondly, an exception has been coded into the constraint. Adjectives that are explicitly marked as post-modifying in the lexicon (such as *proper*) are not penalized by the constraint.

[18]There are also cases that do not allow an adjective to be placed before the noun. The adjective *asleep* in Example E-52 only allows a predicative use, thus supporting an analysis as a reduced relative clause.

E-52 The fire alarm saved a man asleep.

[19]The word order of the adjective *proper* when placed after the noun is probably also influenced by the word order of the French adjective *propre*.

```
C-23 {X} : AdjBeforeNoun : WordOrder : 0.1 :
       X.level=SYNTAX & X@cat=ADJ & X^cat=NOUN & X@type!=postmodifier ->
         X@pos<X^pos
```

Non-postmodifying adjectives (usually) precede the noun.

Note that Example E-51 on the facing page shows an additional difficulty. The adjective *alte* does not agree with the noun, neither with regard to case as in *'Es gibt hier keine alten Bücher.'* nor with regard to adjective type (*'Es gibt hier alte Bücher.'*). The next paragraph shows how the sentence can nevertheless be analyzed in the desired way.

In Section 3.4.1, the hard Constraint C-9 on page 54 was suggested to enforce the complete set of lexical valence requirements (category & morpho-syntactic agreement) in a head-argument structure at once. Assigning a non-zero weight facilitates the successful analysis of utterances with agreement errors. A much better idea, however, would be to split it into a set of smaller soft constraints which only constrain a single feature respectively, e.g., as in Constraints C-24 through C-27.

```
C-24 {X} : ValenceFirstArgCat : Valence : 0.1 :
       X.label=subj -> X^args:1:cat=X@cat

C-25 {X} : ValenceFirstArgCase : Valence : 0.2 :
       X.label=subj -> X^args:1:morph:case=X@morph:case

C-26 {X} : ValenceFirstArgNumber : Valence : 0.2 :
       X.label=subj -> X^args:1:morph:number=X@morph:number

C-27 {X} : ValenceFirstArgPerson : Valence : 0.2 :
       X.label=subj -> X^args:1:morph:person=X@morph:person
```

The smaller constraints have the advantage that the weights can be assigned in a more fine-grained fashion and that a sentence with a single error, e.g., wrong case only, can be treated differently from utterances with multiple faults. Additionally, errors can be detected individually, which is crucial for diagnosis of language faults (cf. Section 5.4).

3.5.2 Preference constraints

Besides contributing to a robust analysis, soft constraints allow the integration of preferential knowledge into the parsing system. By preferences, we mean information about the structure of natural language utterances that are often, but not always true and that help to find the most plausible analysis of an ambiguous sentence.

In German, for example, subjects *usually* precede the object, but this is not as strict as it is in English, for instance. If the speaker wants to focus on the object or just get the attention of the hearer, it is perfectly grammatical to move the object to the beginning of the sentence as in Example E-53 (cf. Example E-15 on page 4).

E-53	*Diesen*	Termin	mag	ich	nicht.
	This	appointment	like	I	not.

'This appointment, I do not like.'

Thus, Constraint C-28 (repeated from Constraint C-8 on page 49) is too strict for German. Softening the constraint with a weight of, say, 0.9, preserves the preferential characteristic of the condition without leading to a system failure in the topicalized case.

```
C-28 {X, Y} : SubjectBeforeObject : WordOrder : 0.1 :
       X.level=SYNTAX & Y.level=SYNTAX &
       X^id=Y^id &
       X.label=subj & Y.label=obj ->
         X@pos<Y@pos
```

The subject precedes the object.

This kind of preferential information blends well with the lexicalized constraints shown in Section 3.4. For instance, the word form *paßt* (‘suit$_{3sg}$’) expresses approval. It takes an indirect object that indicates who gives the approval. In the Verbmobil domain this will most often be the speaker and sometimes the hearer. The lexical entry for the word form may include this domain-specific knowledge as a list of preferred types for its second argument (cf. Figure 3.17). Note that the least preferred sort is `anything` that subsumes every other sort so that a second argument that is not a human being only leads to a penalization with a weight of 0.5.

$$\begin{bmatrix} \text{word: } \textbf{paßt} \\ \text{cat: finite} \\ \text{args: } \begin{bmatrix} 1\text{: } \begin{bmatrix} \text{cat: nominal} \\ \text{case: nom} \\ \text{number: sg} \end{bmatrix} \\ 2\text{: } \begin{bmatrix} \text{cat: nominal} \\ \text{case: dat} \\ \text{sort: } \langle \text{speaker 1.0 human 0.8 anything 0.5} \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3.17: A lexical entry with preferences for the type of its second argument.

3.5.3 Spurious ambiguity

The weakest kind of constraints is used to avoid spurious ambiguities in natural language analyses that can be traced back to the failure of the formalism to cope with underspecification.

E-54	Ich	dachte	noch	in	der	nächste	Woche.
						Adj,{nom,acc}	Noun,dat
	I	thought	still	in	the	next	week.

‘I thought still next week.’

Example E-54 (n001k001-1 from the Stellingen corpus) violates an agreement constraint because the dative noun *Woche* does not agree with its adjective with respect to case. But *nächste* is ambiguous in that it can be nominative or accusative case. In order to avoid an enumeration of these readings, a constraint very slightly prefers nominative case to the other cases. Thus, we get an analysis with nominative assigned to *nächste* (and, of course, identifying that a case violation occurred). This kind of preference corresponds to the linguistic preference for the unmarked nominative case. Nevertheless a speaker of a language would not assign nominative case to *nächste* but instead note that *nächste* should be replaced by the correct form *nächsten.* The preference is here used as a technical means to overcome a technical problem that humans do not even have.

3.5.4 Default subordination

Usually, the finite verb is considered the root of the dependency tree. However, in cases where no consistent structure can be found, it may be desirable that word forms of other categories form the root. This may be because no adequate governor is available for them or because the finite verb is missing from the utterance. Allowing word forms of all categories to form the root of a tree considerably contributes to the robustness because unknown constructions and fragmentary input can be at least partly analyzed.

Mitjushin [1992] identifies the general underlying cause why dependency analysis is well-suited for partial parsing:

> "In our opinion, dependency structures are better adapted to partial parsing than constituent structures. The reason is that the dependency structure of a segment is the same both when the segment is considered as isolated and when it is considered as a part of some sentence (by "segment" we understand any sequence of words)."
>
> —— Mitjushin [1992, p. 1]

The Constraints C-29 and C-30, for example, demonstrate the method of an isolated analysis for adverbs.

C-29
```
{X} : AdverbInit : Adverb : 0.0 :
  X@cat=ADVERB ->
    X^cat=VERB | X^cat=ADJ | X^cat=ADVERB | root(X^id)
```
An adverb is subordinated under a verb, an adjective or another adverb or may be the root of the tree.

C-30
```
{X} : AdverbNoRoot : Adverb : 0.1 :
  X@cat=ADVERB -> ~root(X^id)
```
Most often, an adverb is not the root of the tree.

Constraint C-29 generally allows the subordination of adverbs under verbs, adjectives and adverbs, as well as an analysis of adverbs as the root of the tree. Constraint C-30 penalizes the root analysis with a weight of 0.1. Therefore, the subordination under verbs, adjectives and adverbs is highly preferred, but in case that no such dependency can be found, an analysis as root is acceptable as well.

E-55 (Wie hat Schalke die Bayern geschlagen?)
(How has Schalke defeated the Bavarians?)

Haushoch$_{ADV}$!
Resoundingly!

In Example E-55 on the preceding page, the single adverb *haushoch* can be analyzed even though no complete sentence, especially no finite verb, can be found. While this example is trivial the very same mechanism allows an analysis of fragmentary input (cf. Example E-56).

E-56	Ich	...	wie	ist	es	am	Donnerstag?
	I	...	how	is	it	on	Thursday?
	'How about Thursday?'						

Obviously (at least for human beings), the speaker started a sentence with *ich*, but changed his mind, aborted the sentence and uttered a different question. This kind of self-correction is found quite often in spontaneous speech and practical systems have to deal with it. Figure 3.18 shows a possible dependency analysis for this utterance.

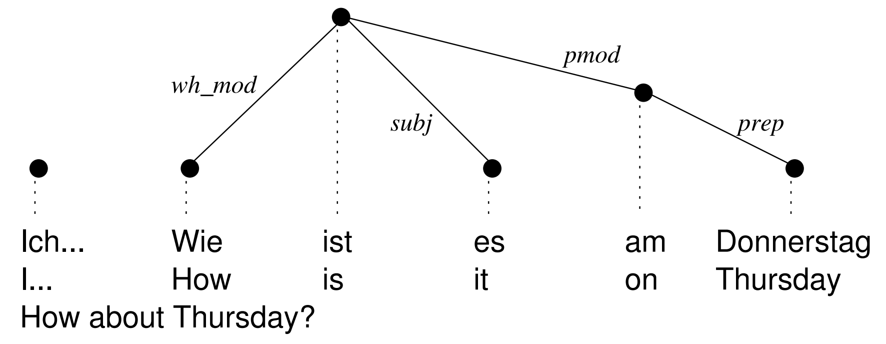


Figure 3.18: An analysis of an utterance with self-correction.

The personal pronoun *ich* is subordinated under the root of the tree because it does not fit very well into the dependency tree for the rest of the utterance. Such a structure cannot only be used by a subsequent processing stage to identify the structure of the main sentence but also (in conjunction with a corresponding constraint violation) to determine that there was some kind of self-repair. However, some additional component has to detect that the overall low score of the analysis can solely be attributed to the sentence restart; the structure of the sentence proper (without the leading superfluous words) probably receives a high score.

Hudson [2000, p. 24] suggests the similar sentence-root principle in the context of word grammar to allow other categories but finite verbs to be analyzed as the root element. However, this principle does not solve the problem of fragmentary input because it still limits the number of root elements to one.

The additional subordinations under root allow for a fall-back in case nothing better can be found. This applies not only to anticipated cases like the missing finite verb but leads to a general fall-back strategy by partial parsing. Although this suffices to cover simple cases of self-repair such as the one above, partial parsing cannot handle all problems that arise from self-corrections, especially when later parts of the utterance refer to the aborted sentence or when the correction takes place in the middle of the utterance.

A similar pattern can be used to cope with cases of category conversion such as the one in Example E-57 where the preposition *ohne* depends on another preposition *inklusive* although the latter usually only takes nouns as complements.[20]

E-57	Inklusive	ohne	drei	Monate	Grundgebühr!
	Including	without	three	months	basic fee!
	'Including free basic fee for three months!'				

Soft constraints which strongly prefer nouns as dependants of prepositions but also allow other categories such as prepositions can be used to find a connected analysis for the example. Of course, a partial analysis analogous to the one in Figure 3.18 on the facing page with both prepositions subordinated under the root is possible as well.

3.5.5 Approximation of higher order constraints

As some examples in previous sections have shown, it is sometimes desirable to relate more than only two dependency edges using a single constraint. Different methods to circumvent this problem have been introduced so far, including feature percolation (cf. Section 3.4.2) and the addition of specialized levels (cf. Section 3.4.7).

For some phenomena, however, it is not necessary to make the constraints bullet-proof because the alternative solutions are so rare that they do not merit the effort. In these cases an approximation of higher order constraints comes in handy.

In order to correctly model the restrictions of the reflexive pronoun in Figure 3.10 on page 58, a constraint must access at least three dependency edges simultaneously. Constraint C-31 is especially tailored for this problem, but unfortunately it is a ternary one and therefore fails to meet the requirements for constraints we set up in Section 2.6 (cf. Section 3.3.3).

```
C-31 {X, Y, Z} : ReflexivePronounAntecedent : Ternary : 0.0 :
       X.level=SYNTAX & Y.level=SYNTAX & Z.level=SYNTAX &
       X^id=Y@id & Y^id=Z^id &
       X.label=obj & Y.label=inf & Z.label=subj &
       X@cat=REFLEXIVE ->
         X@morph:person=Z@morph:person
```

A reflexive pronouns agrees with its antecedent with respect to the person feature.

Approximations of higher order constraints can take two forms:

- Only a subset of the configurations that are not well-formed is forbidden by the approximation. The rest may actually occur so seldom that it is not worthwhile writing rules to forbid them.
- A superset of the invalid configurations is penalized by the approximating constraints, i.e., there are some well-formed structures that spuriously make constraints fail. Most often, a grammar writer wants to use soft constraints in this case because, then, only the score of a structure decreases, but it is still possible to analyze the utterance correctly.

[20]The example has been taken from an advert for 'Genion'.

Constraint C-32 is of the second kind. The rationale behind it is that, if there is a subject to an auxiliary verb and a reflexive pronoun depends on an infinitive verb to the right of the first verb, we assume that the infinitive verb directly or indirectly depends on the auxiliary verb.

```
C-32 {X, Y} : ReflexivePronounAntecedent : Approximation : 0.5 :
       X.level=SYNTAX & Y.level=SYNTAX &
       X.label=subj & Y.label=obj & X^pos<Y^pos &
       Y@cat=REFLEXIVE & X^cat=AUX & Y^cat=INFINITIVE ->
         X@morph:person=Y@morph:person
```

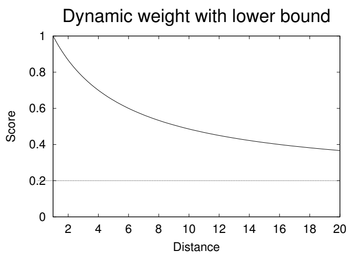
3.5.6 Dynamic constraints

Dynamic constraints do not have a weight that is fixed by the grammar writer. Instead they depend on the sentence context in which they are evaluated.

Dynamic constraints can, of course, be used for all purposes listed in the previous sections. Typical examples are constraints that handle the preferred maximum length of certain dependency links, e.g., article-noun dependencies such as in Constraint C-33.

```
C-33 {X} : LengthArticleNoun : Length :
       [ .2 + .8 * 5/[4 + abs(distance(X@id, X^id))] ] :
         X.level=SYNTAX & X.label=det -> false
```

An article modifies a nearby noun.



Usually short distances are favored and the longer the distance, the higher the penalty. The particular form of the penalty term in Constraint C-33 ensures a lower bound of 0.2 since grammatical sentences exist with arbitrarily long distances between article and governing noun (cf. Example E-1 on page 2).

Another type of dynamic constraint is one with a penalty term with reference to the lexicon, e.g., the penalty in Constraint C-34 which enforces a dynamic penalty for missing arguments (cf. Constraint C-13 on page 57).

```
C-34 {X} : ObligatoryArgument : Argument : [ X@args:2:obl ] :
       X.level=AUX2 & exists(X@args:2:obl) ->
         ~root(X^id)
```

A missing argument is penalized as specified in the lexicon.

Constraints like Constraint C-34 are supplemented by lexical entries with specifications as to how a missing argument should be penalized (cf. Figure 3.19 on the facing page). Completely optional arguments can be described by a very high value for the corresponding lexical feature, for instance 0.9 for the feature `args:3:obl` in Figure 3.19 on the next page.

Strictly speaking, dynamic constraints do not have to be soft constraints since it is also possible to specify the value zero in the lexicon. Our grammar, for instance, contains hard dynamic constraints for the obligatoriness of the coordinated components for particular coordination particles. Most of the dynamic constraints nevertheless have soft weights.

$$\begin{bmatrix}
\text{word: } \textbf{verschiebst} \\
\text{cat: FINVERB} \\
\text{morph: } \begin{bmatrix}\text{number: sg}\\ \text{person: second}\end{bmatrix} \\
\text{args: } \begin{bmatrix}
1\text{: } \begin{bmatrix}\text{realized: yes}\\ \text{obl: 0.1}\\ \text{cat: Nominal}\\ \text{morph: } \begin{bmatrix}\text{number: sg}\\ \text{person: second}\\ \text{case: nom}\end{bmatrix}\\ \text{role: AGENT}\\ \text{sort: human}\end{bmatrix} \\
2\text{: } \begin{bmatrix}\text{realized: yes}\\ \text{obl: 0.1}\\ \text{cat: Nominal}\\ \text{morph:case: acc}\\ \text{role: THEME}\\ \text{sort: appointment}\end{bmatrix} \\
3\text{: } \begin{bmatrix}\text{realized: yes}\\ \text{obl: 0.9}\\ \text{cat: Adposition}\\ \text{loctype: lative}\\ \text{prepsort: temporal}\end{bmatrix}
\end{bmatrix}
\end{bmatrix}$$

Figure 3.19: A lexical entry with penalties for missing arguments specified in the lexicon.

3.5.7 Outvoting

Multiple violated soft constraints induce a more severe fault than any violation of the individual constraints alone. Furthermore, multiple less important constraints can even ally against a single more important constraint as long as the single constraint is not weighted with zero in which case no solution at all can violate this constraint.

The reason for this behavior is due to the employed aggregation function (cf. Sections 2.2.4 and 2.3.3). Penalties of violated constraints are multiplied and, hence, the combined power of multiple soft constraints can outvote a different single constraint which has a soft score which is lower (is more important) than any of the other weights alone.

This behavior is in conflict with the strict dominance hypothesis of optimality theory [Prince & Smolensky 1993, cf. Section 1.4] which postulates the lexicographic aggregation of Section 2.2.4.

The following artificial variants of the impeccable Example E-58 can help to illustrate both effects. In all sentences, the word *Mann* is supposed to be analyzed as the subject.

E-58	Der	Mann$_{sg}$	kauft$_{sg}$	Autos$_{pl}$.
	The	man	buys	cars.

E-59 Autos kauft der Mann.

E-60 Mann kauft Autos.

E-61 Der Mann kaufen Autos.

E-62 Autos kauft Mann.

E-63 Autos kaufen Mann.

In Example E-59, the word order preference subject-before-object is not satisfied. Example E-60 violates the weak constraint that countable singular nouns must be accompanied by an article. Example E-61 contains a number agreement violation. Example E-62 violates both constraints of E-59 and E-60. Example E-63 finally combines all three constraint violations.

Given the results of an uncontrolled experiment to rank the six sentences[21] it can be assumed that each example can be judged worse than its predecessor expect that the result for E-61 $\succ$ E-62 is not significant.

The assumption that Example E-62 is worse than both E-59 and E-60 supports the claim that constraint violations are culminative. The placement of E-61 after both E-59 and E-60 but (at least) en par with E-62 indicates that weak constraints can gang up against a stronger one and finally overrule the latter.

This example is by no means meant as a proof but only as an illustration. Keller [2000b], however, gives extensive experimental psycholinguistic evidence for outvoting linguistic constraints. Note that his notion of hard constraints differs from ours; his hard constraints compare more closely to low weighted constraints in our framework.

> "Furthermore, we found evidence against strict domination among hard constraints. The constraints Res is ranked higher than both Inv and Agr. However, the combined violations of Inv and Agr are as unacceptable as a single violation of Res (cf. Figure 3.8). Such a ganging up of constraint violations should be impossible under strict domination; the combination of two lower ranked violations should not compensate for a single violation of a higher ranked constraint."
>
> —— Keller [2000b, p. 95]

[21]Fifteen native speakers of German were requested to intuitively rank the variants with regard to quality or acceptability under the condition that the man (*Mann*) is the one who buys cars. The following results were obtained ($\succ$ means 'is ranked higher'):

word order vs. unmarked	E-58 $\succ$ E-59	12	80%
article vs. unmarked	E-58 $\succ$ E-60	13	87%
word order vs. word order+article	E-59 $\succ$ E-62	15	100%
article vs. word order+article	E-60 $\succ$ E-62	13	87%
word order vs. number	E-59 $\succ$ E-61	15	100%
article vs. number	E-60 $\succ$ E-61	14	93%
number vs. word order+article	E-61 $\succ$ E-62	8	53%

His findings also support the property of the multiplicative aggregation function that the combined score of two violated constraints is lower than any of the individual scores.[22]

"We found evidence for the hypothesis that constraint violations are cumulative: the more constraints a structure violates, the higher its degree of unacceptability."
—— Keller [2000b, p. 95]

3.6 Related work

Dependency grammar as a grammatical theory has been introduced to modern linguistics by Tesnière [1959] though he did not provide a formalization. Subsequent dependency grammar approaches include a grammatical account for German [Kunze 1972; Kunze 1975], *word grammar* [Hudson 1990; Hudson 2000], *meaning text theory* [Mel'čuk 1988; Kahane forthcoming] and *dependency unification grammar* [Hellwig 1988]. Weber [1992] presents a textbook for dependency grammars.

A systematic approach to German word order in dependency grammar was suggested by Bröker [1998] and has thereafter been modified within various frameworks [Gerdes & Kahane 2001; Duchier & Debusmann 2001; Duchier 2001]. An English version [Järvinen & Tapanainen 1997, EngCG-2] of *constraint grammar* [Karlsson 1990; Karlsson et al. 1995] which aims at dependency structures, features a large coverage and robust analysis.[23] The *ParseTalk* model [Bröker, Hahn & Schacht 1994; Bröker et al. 1997; Bröker et al. 1996] is a cognitively motivated parsing system that draws on a lexicalized performance grammar aiming at dependency structures. Here, parsing is understood as parallel communication (based on message passing) between linguistic objects.

In the context of speech parsing as word graph filtering Harper & Helzerman [1995, p. 47] developed a CDG for English which uses hard constraints only and is still under further development [Harper et al. 1999a, Section 3.1].

Our WCDG (Stellingen grammar, cf. Section 5.2.1) that employs most of this section's techniques in one or the other way, has been inspired by an earlier grammar [Heinecke & Schröder 1998; Heinecke & Nolda 1998; Nolda & Heinecke 1999] but knowingly departs from the latter in some fundamental aspects. In particular, it puts the focus on distributing structure over several levels, on weighted constraints and therefore on graded assessment and robust analysis.

3.7 Summary

This section demonstrated major techniques for [W]CDG modeling that have been employed in a grammar for German (cf. Section 5.2.1) but are more generally applicable.

[22]See also Keller [2000a] and Keller [2001]. Fanselow & Féry [forthcominga, Section 6], in contrast, argue that multiple constraints are conjoined into a new constraint which is more important than each of the original constraints. Of course, cumulative acceptability ratings can always be explained even without cumulative aggregation functions by postulating aggregated constraints, in the extreme case the power set of the basic constraints.

[23]Conexor *functional dependency grammar*, the successor of EngCG-2, is now a commercial product. An online demo is available at `http://www.conexoroy.com/products.htm`.

The first part emphasized techniques for crisp constraints. Linguistic phenomena such as agreement, uniqueness and necessity of arguments, feature percolation, word order and projectivity as well as semantic aspects like thematic roles and reference resolution have been addressed. To do so a rich inventory of methods such as feature agreement, uniqueness conditions for certain types of dependency edges, introduction of auxiliary levels mirroring edges on a main level, enrichment of edge labels with additional information, access to positional variables and introduction of specialized auxiliary levels have been employed.

Subsequently, we focused on weighted constraints. Various types of constraints depending on their strength as well as on their purpose have been identified. Different aspects such as word order errors, agreement violations, preferential knowledge, spurious ambiguities, partial parsing and the approximation of complex constraints can be modeled with soft constraints such that the robustness of the system improves while at the same time the disambiguation power of the grammar increases. The cumulative nature of constraint violations and the possibility of constraint outvoting have been linguistically motivated.

It was argued that hard and soft constraints together can handle a wide variety of linguistic phenomena, originating from both the competence system and the performance system of a speaker. However, we also identified limitations of the expressiveness of the formalism. While the lack of existential quantifiers can be elegantly (if not also automatically) overcome, the restriction to at most binary constraints which originates from procedural considerations is a more severe one. The eliminative nature of [W]CDG turned out to be the reason for an unfamiliar style of grammar development where old constraints have to be revised when extending the coverage of the grammar (for instance, making hard constraints soft under certain conditions such as exceptions to the projectivity condition). But exactly this selection mechanism from a predefined set of possibilities also enables a fall-back strategy that increases the robustness of a grammar and leads to a fail-soft behavior.

The introduction of soft constraints in all their different manifestations has been identified as the main cause for increased robustness and coverage. This claim will be verified quantitatively in Section 5.3.

Chapter 4

Algorithms

4.1 Overview

In Chapter 2, weighted constraint dependency grammars (WCDG) have been presented including a mapping from the WCDG parsing problem to constraint satisfaction problems (CSP). This chapter presents general algorithms for CSP, classifies them according to specific properties and describes and evaluates special variants thereof for WCDG parsing.

Section 4.2 looks at solving CSPs from an informal point of view. Human heuristics are mentioned, and the importance of a good problem model and effective heuristics is emphasized.

The subsequent Section 4.3 introduces algorithms for general CSP and CSOP solving.

Criteria with regard to soundness, completeness, temporal behavior (efficiency, incrementality, anytime) and parallelism that are developed in Section 4.4 are applied to the algorithmic schemes in Section 4.5 and advantages and disadvantages of the algorithms are discussed.

The Sections 4.6 through 4.9 then go into detail regarding variants of these algorithms for WCDG parsing. An implementation is mentioned in Section 4.10 and results from experiments with these methods are described in Section 4.11.

An incremental mode of the WCDG parsing algorithms is briefly discussed in Section 4.12, and program optimizations are presented in Section 4.13.

Related work is reviewed in Section 4.14 and a summary is given in Section 4.15.

4.2 Solving a constraint satisfaction problem

There exist sound and complete algorithms to solve a CSP as defined in definition 2.1, i.e., as the problem of consistently assigning values to variables. Since the number of possible combinations is finite,[1] all of them can be generated and tested for consistency. This naïve approach – known as generate-and-test – of course suffers from combinatorial explosion: The number of possible

[1]More precisely, the number of solution candidates for a CSP as defined in definition 2.1 is the product of the sizes of all domains $|D_1 \times \ldots \times D_n| = \prod_{1 \leq i \leq n} |D_i|$.

$$V = \{\underbrace{1, ..., n}_{queens}\}$$

$$\bigvee_i : D_i = \{\underbrace{1, ..., n^2}_{positions}\}$$

$\rightarrow n^{2n}$ candidates

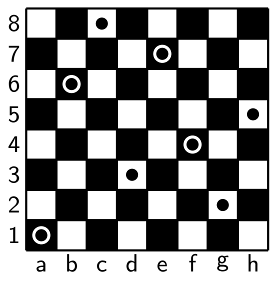
(a)

$$V = \{\underbrace{1, ..., n}_{rows}\}$$

$$\bigvee_i : D_i = \{\underbrace{1, ..., n}_{columns}\}$$

$\rightarrow n^{n}$ candidates

(b)



(c)

Figure 4.1: Two CSP models for the n-queens problem. (a) naïve model with the field numbers (from 1 to n^2) used as values for the variables, (b) modeling that exploits problem specific information and uses column numbers as values, (c) example solution for $n = 8$.

combinations grows exponentially in the number of constraint variables and renders the problem intractable even for moderate problem sizes.

Although the human capabilities to solve combinatorial problems are limited because usually a large number of possibilities have to be evaluated (in parallel),[2] human beings manage to decide on a solution by applying a number of techniques:

- **Problem specific information:** Often domain specific knowledge can be used to dramatically reduce the set of solution candidates. For instance, when solving the n-queens problem, i.e., the task to place n queens on a $n \times n$-checkerboard so that no queen threatens another one, it is a good idea to assume (without loss of generality) that queen number n is in column n (cf. Figure 4.1).
- **Early constraint checking:** The naïve generate-and-test procedure always generates complete solution candidates before checking the first constraints. Obviously, a lot of work is wasted[3] if an inconsistency is due to values of variables that are assigned early. A better strategy is to apply constraints as soon as possible, i.e., already while building a solution candidate. Two kinds of interleaved constraint checking can be distinguished.
  - **Prospective checks:** Whenever a new value is assigned, it is checked whether all future variable domains contain values that are consistent with the newly assigned value. If all values from at least one domain are inconsistent, another value is tried. One algorithm based on this method is forward checking (cf. Section 4.3.1).
  - **Retrospective checks:** New values are checked for consistency with values that have already been assigned. If an inconsistency is detected, a different value is tried. The most basic algorithm using this technique is known as backtracking (cf. Section 4.3.2).
- **Heuristic information:** Heuristic knowledge is different from problem specific information because it is not used to reduce the problem before actually solving it. Instead, the

[2]Try to solve the cryptarithmetic puzzle from Section 2.2.2 without using a pen and paper!

[3]However, there exist successful algorithms that are based on the random generation of candidates [Minton et al. 1992, p. 3, Las Vegas algorithm for n-queens problem]. Problems where most of the candidates are actually solutions, i.e., only a small portion is not allowed by constraints, can easily be solved by randomly picking a candidate until a solution is found.

search process itself is modified. Miguel & Shen [2001a, Section 2] list some of the general heuristics. Consider again the cryptarithmetic puzzle from Section 2.2.2 and Figure 2.1 on page 17 (repeated as Figure 4.2). People tend to start with variables for which there is a lot of constraining information available in order to narrow down the set of possible values for these variables. From these islands of relative certainty, they then try to proceed to the next variable. The general principle applied is to re-order the constraint variables so that problem solving becomes easier. While this rearrangement of variables can have a great impact on the efficiency, solutions cannot be cut off nor does the model have to change.

- **Heuristic search:** Sometimes the set of possibilities is so large that it is prohibitive to look at it systematically. However, a problem solver might be willing to apply a rule of thumb and use his experience to make local decisions without being sure that the globally correct branch is taken. A traveling sales person, for example, who wishes to minimize the length of a round trip through a number of cities, might assume that he or she has to visit two nearby cities directly one after the other because experience shows that it is a good strategy (or heuristic) in most cases. Note however, that the really shortest path may include an arbitrary number of intermediate cities between the two nearby. In contrast to heuristic information which simply changes the way the search space is traversed, heuristic search really cuts away branches which might contain the solution. Therefore, heuristic search is neither sound nor complete. Real-world problems are nevertheless often solved this way. Reasons why this is acceptable include absence of complete information and the readiness to live with outcomes that are close to the overall best solution.

```
    S E N D           S   E   N   D          9 5 6 7
  + M O R E         + M   O   R   E        + 1 0 8 5
                      C3  C2  C1               0 1 1
  ---------         -----------------      ---------
  M O N E Y         M O   N   E   Y        1 0 6 5 2

     (a)                  (b)                 (c)
```

Figure 4.2: A cryptarithmetic puzzle is a typical CSP problem (repeated from Figure 2.1 on page 17). Humans typically start by narrowing down the set of possible values for the letter `M` because it seems to be highly constrained in a direct way. After proving that only one or a small number of values are possible for the letter `M`, this additional (formerly implicit but now unveiled) information is used to procede to the next variable.

The importance of, e.g., interleaved constraint checks and application of heuristic information is demonstrated by two Prolog [Clocksin & Mellish 1987] programs in Figure 4.3 on the next page which both solve the cryptarithmetic puzzle of Figure 4.3 on the following page. Prolog generally employs a top-down left-to-right systematic depth-first search (cf. Figure 4.5 on page 85 in Section 4.3.2). In order to activate the retrospective checks, the corresponding clauses have to be textually interleaved with clauses which generate choice points. Heuristic information is applied by means of variable re-ordering: The variables `S` and `M` which seem to be constrained in the most direct way are tried first. Table 4.1 compares the two procedures with respect to the required computational resources.

```
send_more_money_naive(S,E,N,D,M,O,R,Y) :-
   ; First generate a complete solution candidate...
   member(S, [0,1,2,3,4,5,6,7,8,9]),
   member(E, [0,1,2,3,4,5,6,7,8,9]),
   member(N, [0,1,2,3,4,5,6,7,8,9]),
   member(D, [0,1,2,3,4,5,6,7,8,9]),
   member(M, [0,1,2,3,4,5,6,7,8,9]),
   member(O, [0,1,2,3,4,5,6,7,8,9]),
   member(R, [0,1,2,3,4,5,6,7,8,9]),
   member(Y, [0,1,2,3,4,5,6,7,8,9]),
   ; ... then apply the general uniqueness constraint
   no_duplicates([S,E,N,D,M,O,R,Y]),
   ; ... and finally the problem specific constraint.
   0 is          S*1000+E*100+N*10+D
                + M*1000+O*100+R*10+E
        - M*10000+O*1000+N*100+E*10+Y.
```

```
no_duplicates([E|Es]) :-
    not(member(E, Es)),
    no_duplicates(Es).
no_duplicates([]).
```

(a) Naïve generate-and-test procedure

```
send_more_money_interleaved_heuristic(S,E,N,D,M,O,R,Y) :-
   ; Start with directly constrained variables,
   member(S, [0,1,2,3,4,5,6,7,8,9]),
   ; ... apply the uniqueness constraint as soon as possible
   member(M, [0,1,2,3,4,5,6,7,8,9]), not(member(M, [S])),
   member(C3, [0,1]),
   ; ... and also utilize specific constraints early.
   M is (S+M+C3) // 10,
   O is (S+M+C3) mod 10,
   member(O, [0,1,2,3,4,5,6,7,8,9]), not(member(O, [M,S])),
   member(E, [0,1,2,3,4,5,6,7,8,9]), not(member(E, [M,S,O])),
   member(C2, [0,1]),
   C3 is (E+O+C2) // 10,
   N is (E+O+C2) mod 10,
   member(N, [0,1,2,3,4,5,6,7,8,9]), not(member(N, [M,S,O,E])),
   member(R, [0,1,2,3,4,5,6,7,8,9]), not(member(R, [M,S,O,E,N])),
   member(C1, [0,1]),
   C2 is (N+R+C1) // 10,
   E is (N+R+C1) mod 10,
   member(D, [0,1,2,3,4,5,6,7,8,9]), not(member(D, [M,S,O,E,N,R])),
   C1 is (D+E) // 10,
   Y is (D+E) mod 10,
   member(Y, [0,1,2,3,4,5,6,7,8,9]), not(member(Y, [M,S,O,E,N,R,D])).
```

(b) Interleaved constraint checking and variable re-ordering

Figure 4.3: Two Prolog programs for the example cryptarithmetic puzzle.

The following section looks at different solution methods in greater detail and discusses how the general intuitive techniques described in this section can be realized in precise algorithms.

	CPU Seconds	Inferences	# Solutions
naïve	3,332.33	1,927,424,291	25
interleaved	0.03	14,322	25

Table 4.1: Interleaving constraint checks with variable assignments as well as re-ordering of variables can have a great impact on the efficiency of the algorithms. The table shows CPU time (on a SUN Ultra5 360 Mhz workstation) and number of logical inferences needed by the SWI Prolog system [Wielemaker 2000] as well as the number of solutions produced for the example programs from Figure 4.3 on the preceding page.

4.3 Algorithms for (partial) constraint satisfaction problems and constraint optimization

While some aspects from the previous section are problem specific so that one cannot deal with them in general, various algorithms have been developed based on general insights in CSP solving.

4.3.1 Consistency techniques

Consistency or filtering algorithms achieve a state of consistency by removing those values from the domains of variables that are incompatible with some other 'neighboring' variables. Generally, the problem becomes smaller and easier to solve when less values have to be considered because they were previously removed. By proving local inconsistency for the removed values, it is guaranteed that the reduced problem is equivalent to the original problem with regard to the solutions. However, a *consistent* problem may still have no, one or more solutions so that other techniques must finally be used to solve the now smaller problem (see below for other methods).

The notion of local consistency was first introduced by Waltz [1975] who was looking for a way to reconstruct the three-dimensional composition of a blocks' world scene from a simple line drawing (see Figure 4.4 on the following page for an example). Each line (variable) should be tagged with a label that describes its type (value), e.g., whether an edge divides an object from the background or whether the angle at that edge is concave or convex. There exist only a limited number of junctions so that not all combinations of edge types can actually occur (constraints).

Local consistency has been described in more general frameworks, and algorithms to achieve local consistency of different degrees have been designed [Mackworth 1977]. The following definition describes local consistency formally.

Definition 4.1 *A set of constraint variables is* k-consistent *if for each set of* $k-1$ *variables that have been assigned values that satisfy all constraints among them, it is possible to find a value for an arbitrary* k*th variable such that all constraints among these* k *variables are satisfied. A set of variables can be* k*-consistent without being* $k-1$*-consistent [Freuder 1988, p. 349]. If the set is* k'*-consistent for all* $k' \leq k$ *then it is* strongly k-consistent.

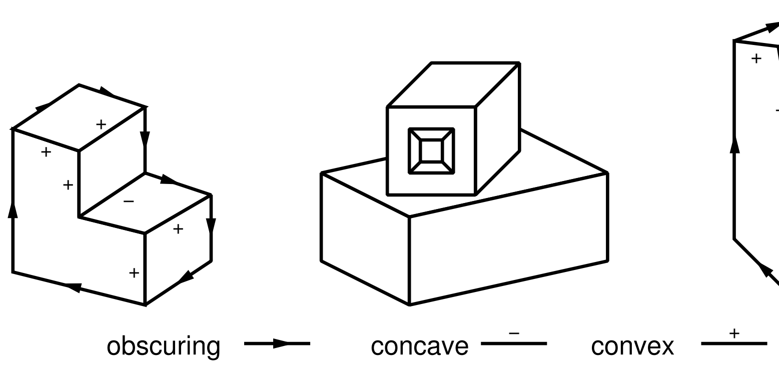


Figure 4.4: Reconstruction of three-dimensional information from line drawings by constraint satisfaction. The edges of two objects have already been classified; the remaining one in the middle is truly ambiguous.

Strong 1-consistency is known as *node consistency*, strong 2-consistency as *arc consistency* and strong 3-consistency as *path consistency.*

There are algorithms that make arbitrary CSP k-consistent by removing inconsistent values from the domains [Freuder 1985]. For instance, the temporal and spatial complexity of the optimum arc consistency algorithm AC-4 is $O(|C| \cdot \max_i |D_i|^2)$ while PC-4 achieves path consistency with a temporal and spatial complexity of $O(n^3 \cdot \max_i |D_i|^3)$ [Mohr & Henderson 1986]. Generally the consistency methods for path consistency and above are often too expensive in practice [Miguel & Shen 2001b, p. 261]. Freuder [1985] gives a general algorithm for k-consistency but it has exponential complexity.

Definition 4.2 *An* ordered CSP *is a CSP with a total order on the constraint variables. The* width *of a variable in an ordered CSP is the number of constraints between that variable and variables that precede it in the ordering. The* width of an ordering *is the maximum width of the variables and the* width of a CSP *is the minimum width of all orderings.*

A theorem by Freuder [1982] states that a strongly k-consistent problem that has a width smaller than k can always be solved without backtracking.[4] The problem with this approach is that achieving k-consistency is, for larger k, very expensive, even more expensive than simple backtracking. That means that while it is generally infeasible to solve larger CSP through consistency techniques alone, they are well suited for reducing the problem size before actually applying a search algorithm (cf. Section 4.3.3).

One particular problem with consistency-based solution methods and PCSP is that it is often not obvious what incompatible means in this context. Since occasional constraint violations have to be tolerated in this context, a single conflict is not a sufficient condition for eliminating a possible value (cf. Section 4.6).[5]

[4]The original algorithm of Waltz [1975] achieves strong 2-consistency while the problem of identifying the types of lines in line drawings (cf. Figure 4.4) has a width of 2 even for moderately complicated scenes. Therefore at least strong 3-consistency is required to guarantee a backtrack-free extraction of solutions [Freuder 1988, p. 352].

[5]Compare the notion of arc consistency counts in Miguel & Shen [2001a, 5.2.1].

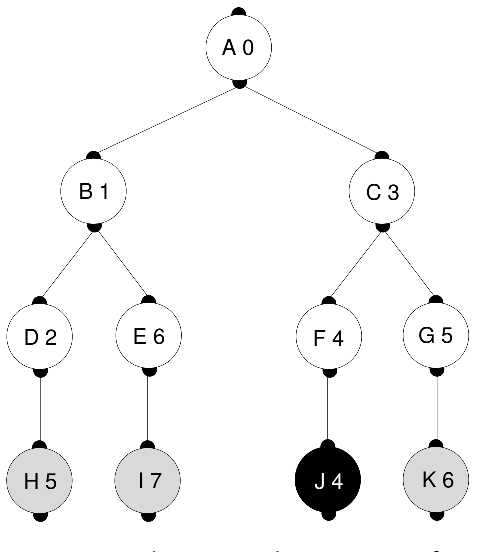
4.3.2 Retrospective techniques

Retrospective search algorithms successively assign values to constraint variables and maintain a consistency measure for the values assigned so far. Their key characteristic is that they always look at the assignments made so far (and not at those that have to be made in the future) and try to avoid unnecessary work when it becomes clear that the partial assignment under consideration cannot lead to a global solution.

Backtracking is the simplest such technique. The order in which the variables are assigned is fixed and whenever a freshly chosen value emerges as inconsistent with the previous bindings, the latest assignment is reverted and an alternative value is tried. If no more alternatives for a given variable exist, the value assignment for the preceding variable is questioned. The time complexity is exponential, of course, but the algorithm only needs space linear in the number of variables because only a single path is considered at any one moment. This simplest variant is known as chronological backtracking because the assignments are reverted in the exact reverse chronological order.

Branch-and-bound is the analogue to backtracking for the PCSP case. While a search path in a crisp CSP can be truncated when the first constraint violation is found, a certain degree of inconsistency has to be accepted in the PCSP case. Only if the path falls short of a lower bound of consistency can it be ignored. A lower bound may be defined externally and can be updated, i.e., increased, whenever a new complete solution has been reached.

Backtracking and branch-and-bound in their simplest forms both traverse the search tree in a depth-first manner which is the reason why their spatial complexity is linear. However, both techniques can also be used with a breadth-first or best-first agenda strategy.



One and the same search tree is traversed in different orders depending on the agenda strategy of the search procedure. The numbers next to the node identifiers denote the number of violated constraints up to that point.

Breadth-first:	A B C D E F G H* I J* K
Depth-first:	A B D H* !! E# !! C F J* !! G#
Best-first:	A B D C F J * H G# E#

An asterisk '*' marks a new solution state, a exclamation mark '!' indicates backtracking and a hash sign '#' signals that the search space has been pruned, i.e., a whole branch of the search tree has been cut off.

Figure 4.5: Three agenda strategies for systematic tree search: breadth-first, depth-first and best-first.

Figure 4.5 gives examples of the sequences of visited nodes while traversing the search tree with the three named strategies. Obviously, the linear space bound is sacrificed when using a breadth-first or best-first strategy since the agenda must hold an exponential number of nodes in the worst case. The breadth-first strategy guarantees that the 'smallest' solution is found first, which is important when search trees with varying depths for the leaf nodes or even cyclic search graphs are considered. This is not the case in standard constraint satisfaction problems since the number of variables determine the uniform depth of the search tree. A best-first strategy always selects the most promising internal nodes for the next node expansion; therefore detours can be avoided when the search space is 'well-behaved' (compare the number of nodes visited in Figure 4.5 on the page before).

These two basic algorithms have been extended by Freuder & Wallace [1992]. *Backjumping*, for instance, tries to avoid duplicate work when a failure somewhere in the search tree has its ultimate cause in an assignment much higher in the search tree. The algorithm then jumps directly to that variable instead of the preceding one. *Backmarking*, on the other hand, tries to minimize the number of constraint checks by maintaining additional data about the level of backtracking and re-using results from previous constraint checks (cf. Section 4.8).

4.3.3 Mixed techniques

The two approaches to solving a CSP that have been introduced so far can be combined and, in fact, a whole spectrum of algorithms have been devised, each of which combines a systematic tree search algorithm with intermediate filtering steps which ensure a certain degree of local consistency. Nadel [1988], for instance, compares the performance of six algorithms that successively include a higher degree of consistency: generate-and-test, simple backtracking, forward checking, partial lookahead, full lookahead and really full lookahead. Nudel [1983] investigates the expected complexity of such algorithms and derives specific heuristics for different classes of CSP. Forward checking [Haralicj & Elliot 1980] is particulary simple and successful at the same time: Each time a value is assigned to a variable, the domains of all future, i.e., yet unassigned, variables are checked for compatibility with the current value. If at least one domain contains no consistent value, the current assignment is reverted and backtracking starts.

Freuder & Wallace [1992] compare the performance of a set of mixed-techniques algorithms based on a number of artificially designed partial constraint satisfaction problems.

4.3.4 Heuristic search

Heuristic search [Selman 1999] is fundamentally different from systematic search in that it does not explore the search space systematically. The main reason is that the search space is not a tree as in tree search (hence, the name) but a graph which leads to greatly increased costs for keeping records about which parts of the search space have been visited. Mostly it is even intractable to store complete information about the seen parts so that no systematic and complete exploration is feasible.

The search states in heuristic search are complete assignments or solution candidates. This is in contrast to tree search where search states consist of *partial* assignments. Heuristic search methods move from the current solution candidate to that solution candidate among all solution candidates in the neighborhood which seems most promising with regard to specific measure or heuristics.

Since the next search state is always a neighbor of the current search state, the general search scheme is also called *local search.* The decision of which neighbor to move to depends on the variant of local search [Hromkovič 1998, p. 177]: *steepest ascent* selects the neighbor that improves the assessment in some maximum way while *greedy search* takes the first improving step that can be found.

Obviously local search can get stuck in local optima when none of the neighboring states improves on the current state. A so-called meta-heuristic is then applied in order to escape from the local optimum. The heterogeneity of these meta-heuristics accounts for a large variety of different types of heuristic search. Simple examples for meta-heuristics include the *multistart approach* which re-initializes the start assignment, the *multilevel approach* which uses variable definitions for the neighborhood and the *variable-depth local search* which iteratively increases the neighborhood [Aarts & Lenstra 1997a]. Advanced techniques are, for instance, *simulated annealing* [Kirkpatrick, Gelatt & Vecchi 1983; Aarts, Korst & van Laarhoven 1997], *tabu search* [Glover & Laguna 1997] and *guided local search* [Voudouris & Tsang 1995].

One of the first meta-heuristics in the heuristic search paradigm is *min-conflicts* [Minton et al. 1992] which starts with a complete but random assignment and repeatedly selects a variable (or a set of variables) with a conflict and chooses a better value for that variable.

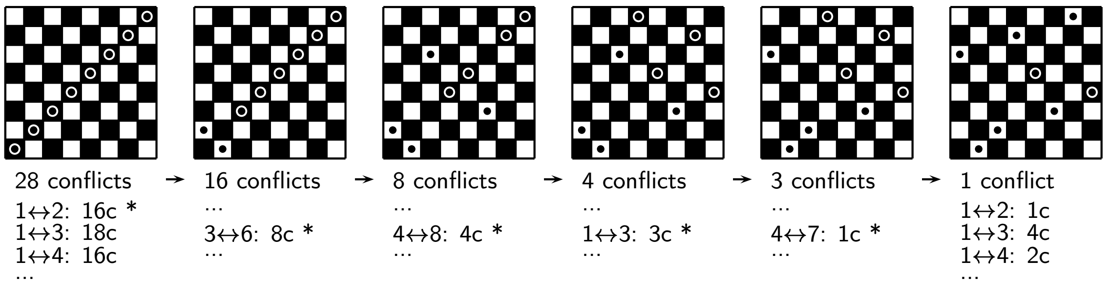


Figure 4.6: Example of how *min-conflicts* tries to solve the 8-queens problem: Starting from an arbitrary (here regular) assignment, the best improving swap of two columns (marked by an asterisk) is chosen at each time. However, after swapping two columns five times the process is stuck in a local optimum and no progress can be made although the last assignment is not a valid solution.

Figure 4.6 illustrates the progress of the algorithm for the n-queens problems. Since the method always greedily selects the locally most attractive move and only allows improving steps, the process can still be caught in local optima.

Some variations of the basic algorithm have been proposed. While *walk-minconflicts* adds random (possibly degrading) steps, *retry-minconflicts* occasionally restarts the process from a random assignment.

The search space is explored unsystematically by heuristic search procedures and therefore it is hard to provide a termination criterion except when a perfect solution has been found. Usually an external termination criterion must be met. Hence no generally valid analysis of the worst-case complexity can be made without reference to the employed termination criterion. The time complexity of the local search procedure itself of course depends on the definition of the neighborhood and the nature of the scoring function. Therefore, even the worst-case complexity of local search alone is exponential in general. The art of designing a successful heuristic algorithm includes the definition of a neighborhood that is small enough to be searched in polynomial time and that is large enough so that the global optimum can be reached from arbitrary points in the search space in a reasonable amount of time (cf. Hromkovič [1998, Section 3.6.3]). Since only a single candidate is kept during local search the space complexity is only linear in the number of variables.[6]

4.4 Evaluation criteria for CSP solution methods

This section develops a set of criteria that can be used to classify the different algorithms for CSP solving with regard to specific application requirements.

Besides the classical features of soundness, completeness, termination and complexity we discuss four more properties that determine aspects of the temporal behavior of an algorithm.

- *Efficiency* refers to practical runtime in contrast to the theoretical worst case considerations that are examined under the keyword complexity.
- *Anytime algorithms* have the ability to dynamically improve the quality of a solution during processing, thus delivering an approximate solution quickly and better alternatives if more time is available.
- *Incrementality* is a mode of operation where an algorithm starts while the problem still changes, e.g., before the complete input is available.
- *Parallelism* refers to the potential of an algorithm to be split into a number of subtasks that can be executed simultaneously and (relatively) independently on separate processors.

4.4.1 Soundness, completeness and termination

The properties of soundness and correctness are fundamental aspects of algorithmic analysis but have their roots in logical calculus [Schöning 1992a; Schöning 1992b].[7]

Definition 4.3 *A (non deterministic) algorithm is called* sound *if it yields only correct results.*

Definition 4.4 *A (non deterministic) algorithm is called* complete *if it yields (at least) every correct result.*

Given the above definitions, one might be tempted to demand that *all* algorithms fulfill these conditions. While this is justified for 'standard' algorithms, there are exceptions:

- *Approximations* per se only return approximate results, often within some guaranteed bounds. Strictly speaking, the results differ from the ideal, thus preventing soundness. Example: Newton's approximation for finding a null of a function [Kiyek & Schwarz 1989, p. 237].

[6]Of course, different meta-heuristics may need additional space, for instance, for a tabu list in tabu search.

[7]Compare the terms soundness and correctness to the terms of accuracy and recall in Figure 5.1.

- *Probabilistic algorithms* can guarantee that the returned result is correct with a certain probability. Nevertheless not all output is correct. Example: Rabin primality test [Rabin 1980].
- *Heuristic search* usually cannot guarantee that the returned result is correct and has even more difficulties to enumerate all correct results. Example: cf. Figure 4.6 on page 87.

Definition 4.5 *An algorithm is* terminating *if it finishes its computation on arbitrary input in finite time.*

We refrain from requiring an algorithm to be terminating (as is sometimes done) since there exist algorithms that are designed to run forever, e.g., operating system procedures or monitoring software. Algorithms that are not terminating can be controlled by a kind of meta-control mechanism, e.g., a timeout.

4.4.2 Complexity and efficiency

Complexity theory [Schöning 1992b, for instance] determines upper and lower bounds for the amount of computational resources (depending on input length n) that are required for specific problems or algorithms. Standard computational resources are processing time or processing steps and memory.

Since the general CSP is an $\mathcal{NP}$-complete problem (cf. Section 2.8.2) it can be assumed that *every* sound and complete algorithm has an exponential worst time complexity of $O(2^n)$. Of course, this does not mean that algorithms that fail to meet the soundness or completeness criteria also have an exponential time complexity.

For practical systems, it is often more important how an implemented algorithm behaves on real input data. We will use the term *efficiency* for the practical runtime of alternative algorithms. Obviously, runtime can only be used to compare solution procedures if the input data and the experimental settings are kept constant.

4.4.3 Anytime property

The term *anytime algorithm*[8] [Hertzberg 1995, for instance] was coined by Dean & Boddy [1988] for application in the domain of time-dependent planning. They identify three characteristic properties for this class of algorithms:

> "The most important characteristics of these algorithms are that (i) they lend themselves to preemptive scheduling techniques (i.e., they can be suspended and resumed with negligible overhead), (ii) they can be terminated at any time and will return some answer, and (iii) the answers returned improve in some well-behaved manner as a function of time." —— Dean & Boddy [1988, p. 52]

[8]Horvitz [1987] uses the term *flexible computations* for a similar concept.

Obviously, the first point is less useful for purely computational processes because modern operating systems already interrupt processes preemptively with negligible overhead on a very frequent basis.

The authors [Dean & Boddy 1988] distinguish different kinds of compromises between complexity and the standard measure for algorithmic quality, i.e., correctness:

- *Monte Carlo simulations* randomly generate data samples using the assumingly 'true' probability distribution and derive the results by generalization from these samples. They are likely correct and guaranteed polynomial. Example: Data-oriented parsing [Bod 1995; Bod 1998].
- *Las Vegas algorithms* are randomized algorithms that are guaranteed correct and likely polynomial but exponential in the worst case, i.e., randomization concerns runtime not correctness. Example: Las Vegas algorithm for n-queens problem [Minton et al. 1992, p. 3].
- An *approximation* is guaranteed optimal within some error margin and guaranteed polynomial. Example: Newton's approximation for finding a null of a function [Kiyek & Schwarz 1989, p. 237].
- An *anytime algorithm* can guarantee to return a (not necessarily correct) solution in the time available.

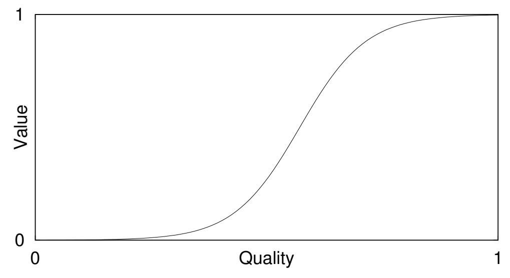
Two prerequisites for anytime algorithms can be identified (cf. Figure 4.7 on the facing page).

1. The (time independent) utility or value of a computation result must gradually increase with increasing quality (cf. Figure 4.7a on the next page).
2. The value of a solution decreases or is discounted with increasing computation time (cf. Figure 4.7b on the facing page).

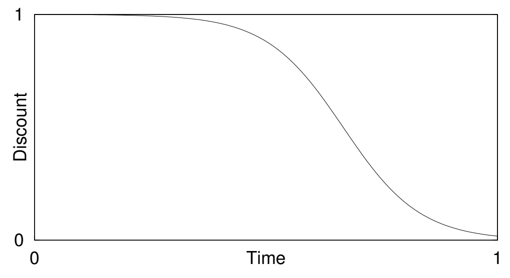
Under these circumstances, the utility reaches an optimum at a certain point in time at which the computation should be stopped at the latest (cf. Figure 4.7d on the next page).

Deliberation scheduling [Boddy & Dean 1994], i.e., meta-reasoning about how much of the resources one should spend for computation and how much for action in dynamic environments, concerns the distribution of available computational resources to reasoning procedures depending on what kind of impact on the system's performance can be expected from these computations. In contrast to their original presentation [Dean & Boddy 1988], the authors make clear that they focus on probabilistic performance profiles:

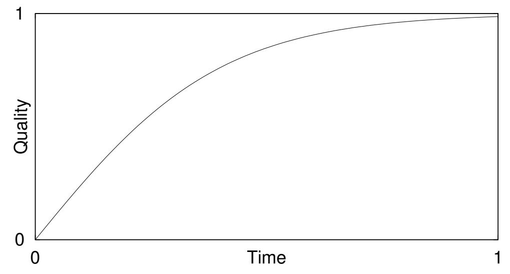
> "Instead, we employ algorithms for which the *expected* value increases monotonically. For some algorithms, there is no difference: a probabilistic algorithm for identifying large prime numbers has a "value" that is strictly increasing, in the sense that the probability of a correct identification is guaranteed to improve by a known amount with increasing computational effort. [...] The statistics are summarized in what are called *performance profiles*... [...] Knowing the expected value is in general insufficient; there are situations where what is needed is a complete probability distribution." —— Boddy & Dean [1994, pp. 254-255]



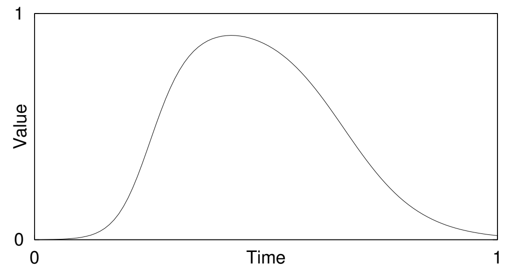
(a) Value of quality



(b) Discount over time



(c) Quality over time



(d) Value over time

Figure 4.7: A utility that grows with the quality of the solution (a) and shrinks over time (b) are prerequisites for anytime algorithms. An anytime performance profile (c) relates the solution quality to runtime so that an optimal point in time for the termination of the computation can be determined (d).

The latter point is illustrated by Figure 4.8 on the following page. Although both probability distributions exhibit the same mean $\mu = 3$, they cannot be considered equivalent in the presence of a deadline τ. Solutions after the deadline have no value and the distribution with greater variance assigns a greater probability mass to those events (shown as the grey area in Figure 4.8 on the next page).

A related point is made by Menzel [1995] by distinguishing between *weak* and *strong* anytime algorithms. For weak anytime algorithms, performance profiles are instance specific, i.e., a probabilistic profile for them can be given only on the basis of a given set of input instances (which might be more or less representative). Strong anytime algorithms, on the other hand, have performance profiles that are generally valid for all kinds of input.

Figure 4.9 on the following page shows why caution must be exercised when dealing with probabilistic performance profiles. Probabilistic performance profiles only describe the *expected* quality after a certain time span.

Russell & Wefald [1991] give a cognitive motivation for limited rationality under bounded resources:

> "For us, the promise of AI lies in explaining how it might be possible for humans, computers, and other animals, with their slow and tiny calculating equipment, to

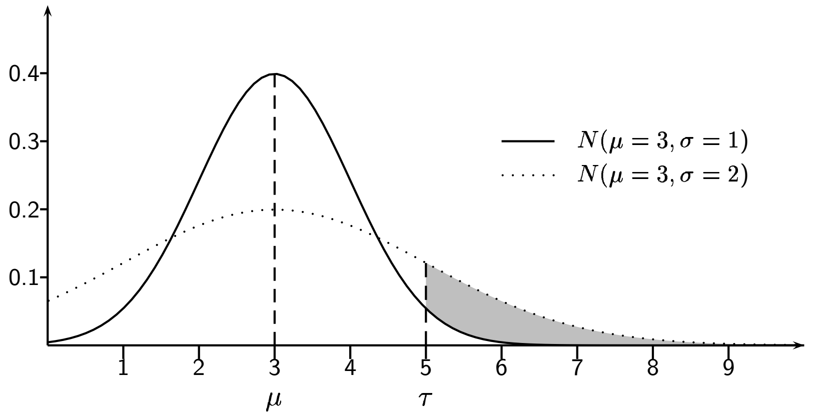


Figure 4.8: Two Gaussian distributions with the same mean but different variances illustrate that the mean as a criterion for performance profiles is not enough in general if one considers absolute deadlines (adapted from Boddy & Dean [1994]).

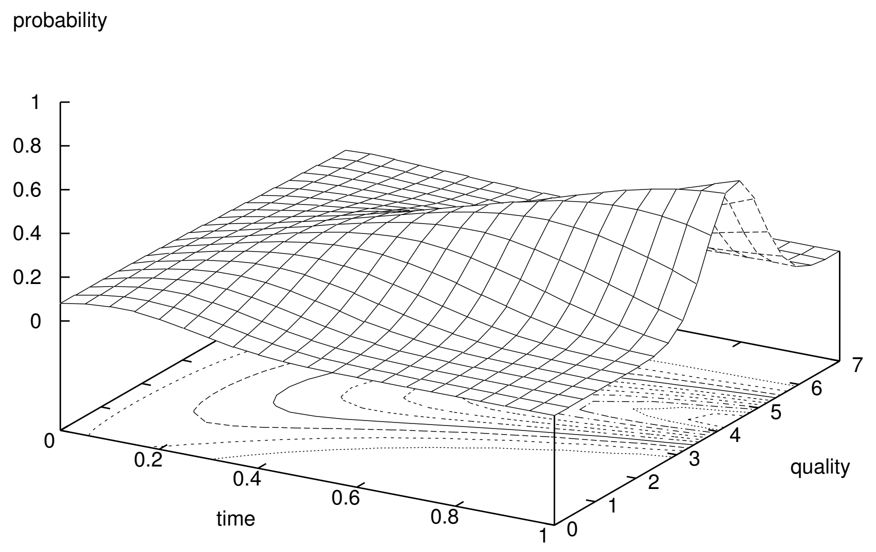


Figure 4.9: Probabilistic performance profiles: The expected quality increases monotonically over time (visible as the contour lines in the xy-plane). Better or worse results are nevertheless possible, although with lower probabilities.

cope successfully with a relatively huge world of blooming, buzzing and often urgent confusion. Rather than achieving some absolute standard of performance with unlimited amounts of resources and simple algorithms, intelligence seems linked with doing as well as possible given what resources one has; good design, elegance and efficiency ought to get in somewhere." —— Russell & Wefald [1991, p. 11-12] and

> "The original formulation of perfect rationality allowed the rational agent to make 'mistakes' – that is, choices with undesired actual outcomes – through ignorance. A bounded optimal agent can also make mistakes through stupidity, because it can't calculate the rational action fast enough." —— Russell & Wefald [1991, p. 28]

In this chapter, we focus on the question of whether the algorithms per se exhibit the anytime property. Section 5.5 calls this kind of anytime behavior *intrinsic* in order to distinguish it from *extrinsic* anytime behavior which is based on meta-control mechanisms or modifications of the basic algorithms.

4.4.4 Incrementality and parallelism

Incrementality is a mode of operation which starts the computation on the partial problem before the problem is completely available. Although often used that way, it is not strictly required to output (partial) results before the input is complete. Typical examples are filter programs, e.g., compressors or word counters, which do not need all of the data to start processing the first items. Incrementality plays an important role in natural language analysis and we will discuss it more thoroughly in Section 4.12.

Parallelism of an algorithm refers to the ability to execute subtasks independently and (potentially) simultaneously. A parallel program can theoretically run n times faster on n processors than on a single one. Practically, this speedup is often limited by additional factors, such as communication overhead and latency. We will only look at the *potential* of algorithms to be executed in parallel environments. Further experiments are subject to future research.

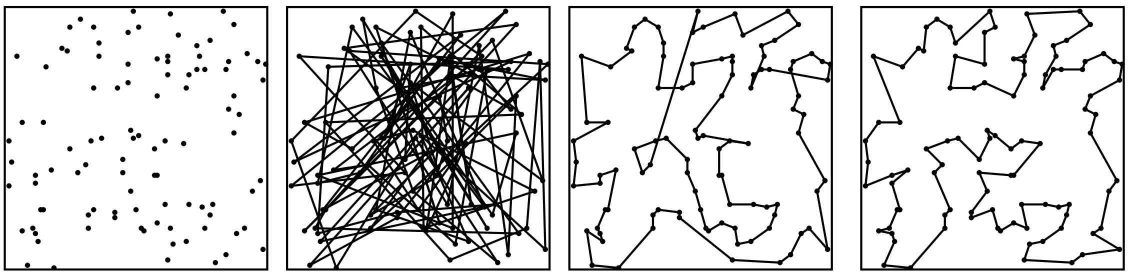
4.5 Classification of CSP algorithms

This section evaluates the algorithms presented so far with regard to the criteria of the previous section: soundness, completeness, termination, complexity, efficiency, anytime behavior, incrementality and parallelism.

We will use a standard academic problem, the traveling sales person (TSP) problem, as an example throughout the section. The generic TSP problem is defined as the task to find the shortest cyclic path through a weighted graph so that every node is visited exactly once. The euclidic TSP problem assumes that the nodes are points in an euclidic space and that the weight of an arc between two nodes is their euclidic distance. The TSP problem is a well-known $\mathcal{NP}$-complete optimization problem; the problem of finding a Hamilton circuit is a special case thereof [Hopcroft & Ullman 1979].

The TSP is a typical constraint satisfaction optimization problem but only has a single soft constraint which judges a complete assignment. Although this does not match the WCDG parsing case very well, which uses grammars with hundreds of soft constraints, it is nevertheless well suited for our discussion because, in this section, we are not interested in exactly how an assessment is determined but only in the algorithms which use a generic evaluation function.

This well-understood example has been chosen for its simplicity and clarity of presentation. Sections 4.6 through 4.9 will give details about WCDG parsing procedures which are experimentally evaluated in Section 4.11.

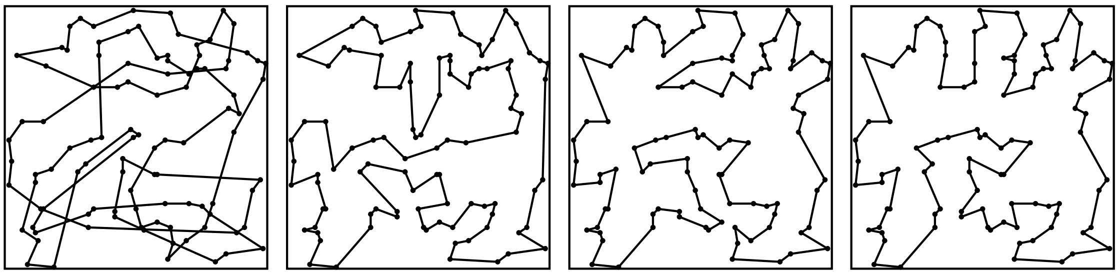


(a) A TSP problem instance: 100 points in a 100×100 euclidic space

(b) A random sequence of edges. Distance 5,558.8.

(c) A path resulting from always visiting the nearest yet unvisited city next. Distance 859.3.

(d) A route resulting from successive extension. Distance 790.5.



(e) A path found by repeated local search with node exchange. Distance 1,184.3.

(f) A round trip found by a single local search with edge exchange. Distance 843.2.

(g) A path found by repeated local search with edge exchange. Distance 779.2.

(h) The (probably) shortest round trip. Distance 764.9.

Figure 4.10: A sample euclidic traveling sales person problem and some solution candidates obtained with different methods. Our still naïve implementation of branch-and-bound was not able to prove optimality for the example TSP problem in reasonable time. There are, however, specialized algorithms that can compute the optimal path even for problems consisting of thousands of cities; see, for instance, the CONCORDE project (`http://www.math.princeton.edu/tsp/concorde.html`).

Figure 4.10 shows a sample problem instance of the euclidic TSP and some solution candidates found by different solution methods. In particular, we look at the following algorithms:[9]

Random sequence: Start with an arbitrary node; choose a random next node; finally go back to first node.

[9]The software used in this demonstration can be downloaded from `http://nats-www.informatik.uni-hamburg.de/~ingo/prg/`.

Nearest neighbor: Start with a random node; always choose the nearest yet unvisited node as the next node.

Successive extension: Start with a random edge; choose next node randomly and insert it into the intermediate path so that the resulting path is minimal.

Node exchange: Start with random path; local search: repeatedly exchange two nodes so that the path is minimal (among the reachable paths) after the exchange; re-start with new random path.

Edge exchange: Start with random path; local search: repeatedly exchange two edges so that the path is minimal (among the reachable paths) after the exchange; re-start with new random path.

Biased depth-first search: Conduct a depth-first search as explained in Section 4.3.2 but choose nearby neighbors first at each choice point.

Random sequence is used to find a (weak) lower bound. Nearest neighbor and successive extension are randomized constructive methods that extend partial solutions candidates using a heuristic approach. Node exchange and edge exchange are heuristic search variants with different definitions of neighborhood. Biased depth-first search is a systematic tree search method with heuristic value ordering.

All methods except the last one are not deterministic in that they use one or the other random component. Therefore, it makes sense to restart the process, in particular the local search variants, from time to time in order to reach different positions in the search space (multistart approach).

Of course, the systematic search is the only method that is sound and complete but it also has an exponential time complexity. Figure 4.11 on the following page shows experimental results of the relation between problem size and computational resources needed (time and recursion steps) for the biased depth-first solution procedure. Clearly, the requirements increase exponentially (note the logarithmic scales) which must be expected since the procedure must potentially explore all $(n-1)!$ paths for a problem of size n. In practice, some branches are cut early during the search whenever it becomes clear that they cannot lead to a solution candidate better than the one already found (cf. branch-and-bound in Section 4.3.2).

All other solution methods are incomplete and not sound since they lack a systematic exploration of the search space.

Random sequence, nearest neighbor and depth-first search are terminating or can easily be modified to do so. Successive extension, node exchange and edge exchange rely to a higher degree on randomness and may require an external control that terminates the algorithm after a certain period of computation or when a satisfactory solution has been found.

4.5.1 Efficiency and anytime behavior

It is practically impossible to prove optimality of a path using depth-first search for a TSP problem size larger than 20. Therefore, one might be willing to accept suboptimal solution

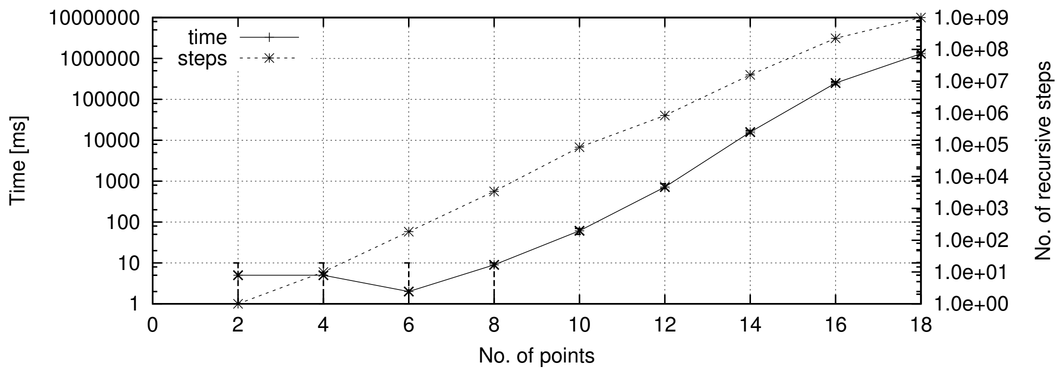


Figure 4.11: Computational resources (number of recursive steps and time needed) vs. problem size during depth-first search. The combinatorial explosion of the search space clearly turns up in the experimental setting. For problem sizes smaller than 8, the accuracy of the time measurement prevents the practical curve from following the theoretical one.

candidates if they can be found fast. Figure 4.12 on the next page relates the problem size to the quality, i.e., the length of the path in this example, that can be achieved by different methods in a limited amount of time (here 30 seconds on a 450 MHz Intel Pentium III).

While the random and the node exchange methods perform much worse than the rest, depth-first search and nearest neighbor are intermediate, and successive extension and edge exchange are by far the most efficient methods. It can be seen that depth-first search fails to find optimal solutions in the limited time starting at a problem size of 32 nodes.

Figure 4.13 on page 98 shows how the solutions improve over a period of 60 seconds for the methods successive extension, edge exchange and depth-first search and problem sizes of 64 and 256.[10]

It can be observed that not only the (intermediate) results are better for non-systematic methods, but that the heuristic variants also improve their results at a much higher rate.

Freuder & Wallace [1992] claim that branch-and-bound is well-suited for anytime applications, and this can be confirmed in a limited way, although the other methods seem much better with regard to their anytime behavior (early first solution, steady improvement, good final results).

> "Circumstances may also impose resource bounds. In particular, real time processing may require immediate answers, that can be refined later if time allows. The branch-and-bound process is well suited to providing resource-bounded solutions. We can simply report the best solution available when, for example, a time bound is exceeded. The branch-and-bound process is also clearly well-suited to support an anytime algorithm, which can repeatedly provide a "best-so-far" answer when queried. It can quickly provide *some* answer, with a better one perhaps to follow as time allows." —— Freuder & Wallace [1992, p. 29]

[10]The absolute values for the distances in Figures 4.12 on the next page and Figure 4.13 on page 98 differ because the underlying TSP problems have been freshly generated. Of course, the problems have been kept constant within the experiments themselves.

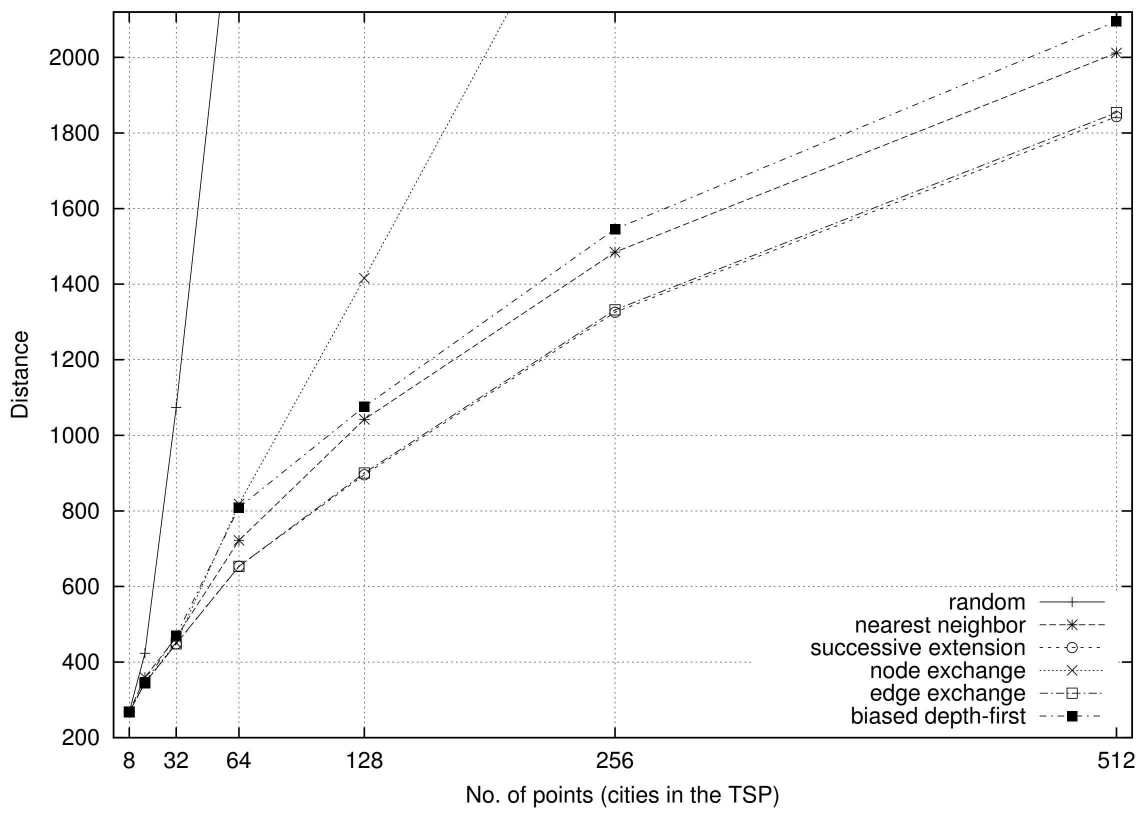


Figure 4.12: Quality (length of path; shorter paths are better) of TSP solutions obtained within 30 seconds for various problem sizes and various solution methods. Figures are averaged over multiple runs.

We conclude that for large problems the heuristic search methods can be much more efficient when appropriate neighborhoods are chosen (compare node exchange and edge exchange) and either computation time is a concern or optimality cannot be proven in reasonable time. Not only do local search algorithms return approximate solutions earlier, but the quality of the delivered results is often much better than those of systematic methods. This favorable behavior, according to Minton et al. [1992], is due to the useful information contained in suboptimal complete solution candidates as opposed to partial solutions in systematic tree search.

Although our findings for a single example cannot easily be generalized, the above observations carry over quite well to the application of WCDG parsing (cf. Section 4.11).

4.5.2 Incrementality and parallelism

In principle, constraint satisfaction problems are well-suited for incremental solution methods because the set of consistent assignments for a subset of variables is always a superset of the corresponding set in all complete solutions. An intuitive approach to constraint solving that starts by looking at the set of consistent values for a small subset of variables and proceeds by extending this set of variables step by step (Section 4.2) can already be considered incremental because the complete set of variables does not have to be known in advance. A systematic

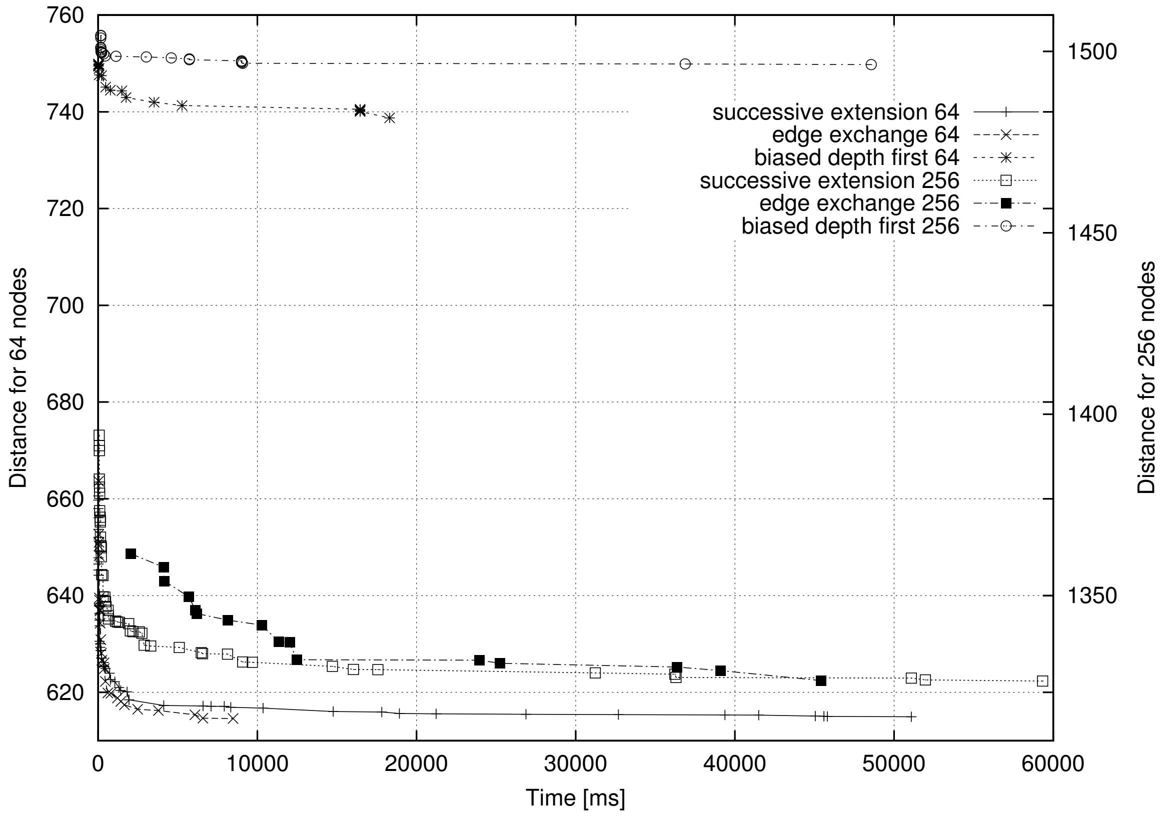


Figure 4.13: Improvements of best round trip lengths so far depending on computation time and search method for traveling sales person problems with 64 and 256 nodes. Only improving steps are shown, i.e., the biased depth-first method, for instance, has not been able to improve the result for the 64 nodes problem after 18,310 ms. Figures have been averaged over five runs.

depth-first tree search can also start to process the first variables even if the complete problem is not yet available. In Section 4.12 we will explain why the constraint satisfaction problems that correspond to WCDG parsing problems are more difficult to solve incrementally. Two alternative approaches will be discussed.

Consistency-based methods have already been parallelized [Helzerman & Harper 1992; Harper et al. 1995]. Furthermore, both search variants, systematic and local, are well-suited for parallel processing [Hromkovič 1998, Section 7.5]. For tree search it is either imaginable to keep the agenda from which subtasks receive nodes for further processing centralized or to divide the search tree into disjoint partitions early and let each subtask work on one of them. Local search methods can start on different initial states or diverge from each other by means of different parameterization [Yannakalis 1997].

4.5.3 Summary

Table 4.2 on the next page summarizes the properties of the three classes of solution methods: consistency-based algorithms, systematic tree search and heuristic search. Note that the consistency-based algorithms do not completely solve the CSP problem but only reduce its size (cf. Sections 4.3.1 and 4.6). Furthermore, they cannot handle soft constraints very well (cf. Section 4.6).

Criterion	**Consistency**	**Systematic search**	**Local search**
Soundness	yes	yes, as long as an unlimited agenda is used; often this is not the case for efficiency reasons	no
Completeness	yes	yes, same restriction as for soundness	no
Termination	yes	yes	external
Complexity	polynomial, $O(n^2)$ for arc consistency in general and $O(n^4)$ for CDG (incl. SLCN)	exponential $O(2^n)$	depends on neighborhood definition and commitment strategy; largely externally controlled
Efficiency	very fast	varying; slow for very large problems; partially shows exponential worst case complexity in practical cases	moderate; termination criteria not clear; quickly finds first solution
Anytime property	not relevant	theoretically yes, but depends on agenda strategy and sparseness of soft solutions	yes
Incrementality	not relevant	well-suited in general, but difficult for parsing (cf. Section 4.12	not directly
Parallelism	yes	various realizations possible	various realizations possible

Table 4.2: Summary of evaluation of algorithmic classes for CSP solving.

4.6 Consistency-based parsing

Consistency-based methods are the algorithms of choice for both Maruyama [1990b], the 'founder' of CDG, and the researchers at Purdue University [Harper et al. 1994; Harper & Helzerman 1994; Harper & Helzerman 1995; Harper et al. 1999b]. Both restrict themselves to crisp constraints and propose a limited number of representational levels besides the syntactic one, mainly for modeling existential conditions. Their motivation, of course, is to keep the CSP relatively small and hence the computation efficient.

While the suggestion of Maruyama [1990a] only covered the analysis of sequences of words where all words have unique unambiguous lexical entries, Helzerman & Harper [1996] extend the algorithms to so called spoken language constraint networks (SLCN) which can handle word graphs and lexical ambiguity [Zoltowski et al. 1992; Harper et al. 1999b]. The proposed algorithm achieves arc consistency for constraint networks where only a subset of variables are part of the final solution, thus facilitating to ignore superfluous words in the input. The worst case complexity is $O(n^4)$[11] where n is the number of words in the input. This seemingly increased complexity when compared to that of arc consistency ($O(n^2)$ where n is the number of constraint variables) is due to the fact that words are needed as both variables and values in CDG.

Harper et al. [1999a] use SLCN as a post processing component for a speech recognizer in order to filter out ungrammatical sentences from word graphs. For this application, the use of crisp constraints is appropriate and the system works satisfactorily well although the consistency-based methods are not always able to reduce the remaining ambiguity enough [Harper et al. 1995]. Helzerman & Harper [1992] developed a massively parallel version of their algorithms.

As soon as one looks at soft constraints, however, a fundamental difficulty emerges. Consistency methods heavily rely on the notion of local consistency, a concept which refers to the isolated compatibility of the domains of a single constraint variable, a pair or more variables (cf. Section 4.3.1). For hard constraints, it is easy to define that domains are consistent when for each value at least one combination exists so that no constraints are violated. For soft constraints, however, such a definition is much more problematic. Value combinations are now assessed with a numeric score between zero and one. In general, only a score of zero justifies the deletion of values from domains, which in turn leads to an insufficient reduction of the domains because almost all values receive a score greater than zero when a lot of soft constraints are used.[12]

Alternative definitions of inconsistency have been tried for the treatment of soft constraints, but with limited success [Menzel & Schröder 1998c].

- The simplest form uses an external absolute threshold for the assessment of values and value tuples. Of course, this is only a very coarse approximation since firstly, the absolute score is nearly irrelevant when looking for the best alternative, and secondly, even a value that has a low score locally can be part of the overall solution. Additionally, it is difficult for a user to determine reasonable thresholds.
- A slightly better strategy is to remove a certain percentage of the values from the domain of a variable since it at least considers which direct alternative values are available and does not blindly assume an absolute bound.
- A similar strategy for value pairs compares the assessment of a value pair to the assessments of pairs with one value replaced by an alternative one. The same objections as before hold.
- A variant of the previous strategy computes the sum of squares of the assessments of similar value pairs [Schröder 1996].

No attempted strategy is satisfactory for soft constraints, i.e., either the heuristics deletes too many values it should not remove, or the ambiguity remains too high.

[11] More precisely, the complexity of achieving arc consistency for a SLCN is $O(c \cdot n^4)$ where c is the number of constraints.

[12] de Givry & Verfaillie [1997] identified the same problem for the general VCSP framework (cf. Section 2.2.4):

> "The *filtering* methods, also called *constraint propagation* or *consistency enforcing* methods (*arc, path, i, i-j-consistency* (Freuder 1978)) can be extended to the *VCSP* framework, but only when the aggregation operator is idempotent." —— de Givry & Verfaillie [1997, p. 3]

It can be concluded that consistency-based methods are not appropriate for weighted CDG parsing since no reliable local consistency measure can be defined.

4.7 Heuristic consistency: Pruning

The procedure *Pruning* [Schröder 1997c] is similar to consistency-based methods in that it uses the support a specific value gets from other variables as a criterion for the decision of which values to discard and which to keep. It also runs in a repeated loop where information about deleted values is propagated. However, exactly one value is removed in each iteration and the selection scheme is more flexible so that fine tuned procedures can be devised that select the next 'doomed' value.

Figure 4.14 gives the simple procedure for pruning in pseudo code notation.

```
procedure Pruning
while noOfPathsThroughWordgraph > 1 or $\exists(v_i \in V) : |D_i| > 1$ do
  select $\langle v', d' \rangle$ to be deleted
  delete $d'$ from domain of variable $v'$
done
```

Figure 4.14: Procedure Pruning in pseudo code. Values are successively deleted from the domains of variables in a loop based on a selection heuristics.

Obviously, the selection heuristics is the crucial part of the procedure.

Simple selection functions only consider the minimum support a value gets from another variable [Menzel 1994]. They combine the mutual compatibilities of the value under consideration and all possible values for another variable. Then the minimum support for each value is determined and finally the value with the least support is selected for deletion. In other words, the value for which at least one variable's values have only low or no support is ruled out as a possible solution.

However, these selection functions suffer from the same problem as the consistency-based methods. The context considered is too small for global correctness and the absolute numeric value of an assessment is an unreliable measure.

More complicated selection methods take into account at least the difference between lack of support due to grammatical incompatibilities and lack of support due to temporal alternatives in word graphs.

But even these fail to analyze word graphs with overlapping edges (cf. Figure 4.15) correctly because they have to approximate the path criterion for efficiency reasons.[13] Menzel & Schröder [1998c] describe the issue in greater detail.

The selection heuristics generally lie between two extreme points. Either they only consider local information and can thus operate very fast, or more global aspects are regarded at the cost of prolonged computation time.

[13]It seems feasible however to extend the pruning method in a similar way the basic arc consistency algorithm has been extended to word graphs [Helzerman & Harper 1996].

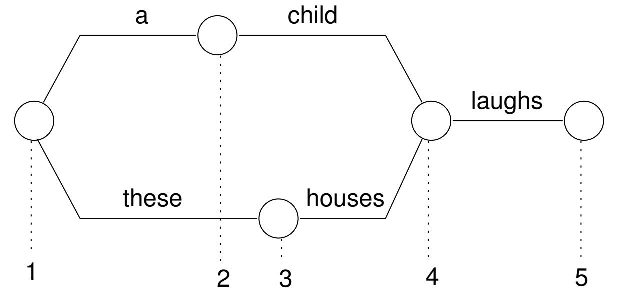


Figure 4.15: A word graph with overlapping edges.

We can conclude that although the pruning procedure leaves more room for powerful selection schemes, it cannot fulfill the overall requirements of an efficient and reliable solution method. Similar remarks as for the consistency-based methods hold here as well.

4.8 Systematic tree search: Netsearch

The procedure *Netsearch* is a variant of the systematic tree search algorithm branch-and-bound with best-first agenda strategy and node as well as value ordering for WCDG parsing. Additional checks ensure acyclicity and compliance with the path criterion for words graphs.

Figure 4.16 on the next page lists the procedure in pseudo code notation. It uses the following variables:

b	best score found so far; lower bound
r	set of (currently) best solution candidates
a	agenda; contains search states $\langle B, V, s\rangle$
B	set of bindings $\langle v, d\rangle$ made so far (partial assignment)
V	set of yet unassigned variables
s	score of partial assignment
$\langle v, d\rangle$	binding of value d to variable v

The procedure solves a CSOP which corresponds to a WCDG parsing problem (cf. Section 2.4). It selects the correct path in a word graph by assigning the dummy value $\square$ to variables corresponding to unused arcs and assigns a structure to the words in the unambiguous path by binding appropriate values to the variables.

The procedure is sound, complete and fast for small to medium-sized parsing problems. For larger problems, however, it often shows the exponential worst-case complexity. In these cases it is indispensable to limit the size of the agenda so that the search space is pruned based on intermediate heuristic information. As a consequence, the procedure loses its soundness and completeness, not only theoretically but also for real parsing problems.

Fünning [2001] extends the procedure to backmarking which does not, however, improve the runtime because the additional administration effort on the meta-level is larger than the time saved through fewer constraint checks (on the basic level).

```
procedure Netsearch
set $b := 0.0$                                  ; BEST SCORE SO FAR
set $r := \emptyset$                            ; SET OF SOLUTIONS
set $a := \{\langle\emptyset, V, 1.0\rangle\}$  ; INITIALIZE AGENDA
while $a \neq \emptyset$ do                     ; PROCESS AGENDA
  get best item $\langle B, V, s\rangle$ from agenda $a$   ; BEST FIRST
  if $s < b$ then continue fi                   ; BOUND
  if $V = \emptyset$ then                       ; COMPLETE ASSIGNMENT?
    if $s = b$ then                             ; BEST SO FAR?
      add $\langle B, V, s\rangle$ to $r$       ; EQUALLY GOOD
    else
      set $r := \{\langle B, V, s\rangle\}$     ; BETTER
      set $b := s$
    fi
  else
                                                ; NODE ORDERING
    select $v_i \in V$                          ; TRY NEXT FREE VARIABLE
    set $V' := V \backslash \{v_i\}$
                                                ; VALUE ORDERING
    for all $d \in D_i$ do                      ; TRY ALL VALUES INCLUDING $\square$
      if incompatible($\langle v, d\rangle, B$) then
        continue                                ; ACYCLICITY & PATH CRITERION
      fi
      set $B' := B \cup \{\langle v, d\rangle\}$
      compute new score $s'$ for $B'$
      if $s' \geq b$ then                       ; ALREADY WORSE?
        add $\langle B', V', s'\rangle$ to agenda
      fi
    done
    truncate agenda $a$ (if desired)
  fi
done
return $r$
```

Figure 4.16: Systematic tree search procedure Netsearch in pseudo code.

4.9 Heuristic search: Frobbing and Gls

The search states in heuristic search are complete solution candidates, i.e., complete analyses in the parsing case. The process moves from one such candidate to a better one in the neighborhood until no further improvement can be made. Different variants of the basic algorithm can be distinguished based on their definitions of neighborhood and on the means that are employed to escape local optima. The latter techniques are called meta-heuristics.

For WCDG parsing, admissible operations that transform one candidate into a neighboring one include at least the following:

- Exchange a lexical reading of a word with an alternative one.
- Change the label of a dependency edge.
- Change the governor of a word.
- Choose an alternative edge from the word graph which is not part of the current intermediate candidate.

Note that some operations entail subsequent modification steps in order to arrive at a feasible candidate again. Choosing an alternative edge, for instance, requires not only changing all assignments that use the old edge but also ensuring that the new edge is part of a complete path through the word graph. In extreme cases this necessitates a complete change of the assignments.

Figure 4.17 presents two steps from an intermediate candidate (a) to the final solution (c) for Example E-64 (a typical German garden path sentence).

E-64	Er	gibt	das	Buch	der	Frau	dem	Mann.
						{gen,dat}		{dat}
	He	gives	the	book	the	woman	the	man.
	'He gives the book of the woman to the man.'							

In the first step, the governor of the word *Mann* and the label of the edge are changed. The second step involves the exchange of the lexical entry for the word *Frau* (from dative to genitive) and again a switch of an edge label.

It can be seen that the intermediate candidate (b) violates at least one hard constraint so that its score falls to zero. It is sometimes said that the local search procedure has to 'walk through a valley' in order to arrive at a new better candidate.[14]

Different variants of the pure local search algorithm use different means to deal with these frequent local optima in the search space. Two have been realized for WCDG parsing. They employ either a sophisticated neighborhood definition (cf. Section 4.9.1) or a modification scheme for the scoring function itself (cf. Section 4.9.2).

In order to be able to reasonably compare two solution candidates which both violate hard constraints, we have to generalize the candidate scoring function. Given a weighted constraint dependency grammar WCDG = $\langle \mathrm{LEV}, \mathrm{LAB}, \alpha, C, \phi \rangle$ and a solution candidate SC = $\langle P, \pi, \delta \rangle$, the extended candidate scoring function $\phi_{\otimes}^{-}(\mathrm{SC})$ is defined as follows.

$$\phi_{\otimes}^{-}(\mathrm{SC}) = \prod_{\substack{c \in \Xi_0(\mathrm{SC}) \\ \phi(c) > 0}} \phi(c) - |\{c \in \Xi_0(\mathrm{SC}) | \phi(c) = 0\}|$$

The idea behind this definition is simple; all soft constraints together account for a range of $(0, 1]$ and for each violated hard constraint the score is decremented by one. The resulting range is $[-q, 1]$ where q is the maximum number of hard constraint violations that depends on the

[14]This is even more important for grammars with interdependent levels and for word graphs as input where one change can trigger a whole set of necessary follow-up changes.

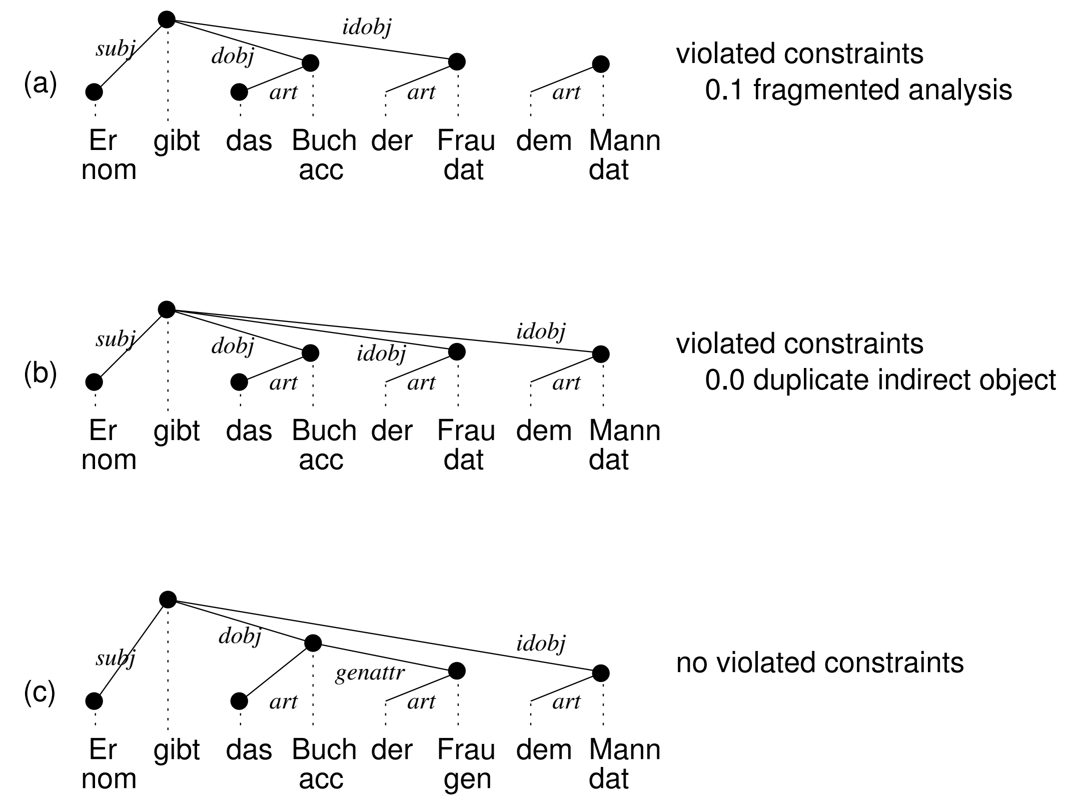


Figure 4.17: Heuristic search for CDG parsing: Transforming one analysis into another one.

grammar and the utterance. The score of a solution candidate that violates at least one hard constraint is always lower than the score of an alternative solution candidate that violates only soft constraints, i.e., a number of weak soft constraints can overrule a single not so weak soft constraint, but a hard constraint can never be overruled by soft constraints.

Figure 4.18 on the next page lists the general heuristic search procedure in pseudo code notation.

4.9.1 Min-conflicts with tabu list

The procedure *Frobbing* [Foth, Menzel & Schröder 2000] is a variant of the local search algorithm for WCDG parsing with a sophisticated neighborhood function. It is based on the min-conflict algorithm [Minton et al. 1992, see above], i.e., it primarily considers those solution candidates as neighbors to the current one that manage to remove at least one of the constraint violations.

Additionally it maintains a list of recently removed conflicts that are not allowed to be reintroduced in neighboring states, thus avoiding cycles in traversal of the search space. This technique is borrowed from *tabu search* [Glover & Laguna 1997; Glover & Laguna 2000; Hertz, Taillard & de Werra 1997; Domschke, Klein & Scholl 1996], another variant in the local search framework that uses a dynamic, adaptive memory, a so called tabu list, to store features of search states that are prohibited (or *taboo*). Of course, depending on the size of the tabu list the linear space complexity of the algorithm may increase.

```
procedure HeuristicSearch
set SC := random initial solution candidate            ; START RANDOMLY
set BSF := SC                                          ; REMEMBER BEST SO FAR
while not finished do                                  ; EXTERNAL TERMINATION CRITERIA
  repeat                                               ; LOCAL SEARCH
    set IMPROVED := false
    for all SC′ neighboring SC do                      ; STEEPEST ASCENT
      if $\phi^{-}_{\otimes}(SC') \succ \phi^{-}_{\otimes}(SC)$ then     ; COMPARES FAVORABLE
        set NEW := SC’                                 ; FOUND BETTER STATE
        set IMPROVED := true
      fi
    done
    if IMPROVED = true then                            ; BETTER NEIGHBOR
      set SC := NEW                                    ; STORE NEW STATE
    fi
  until IMPROVED = false
  if $\phi^{-}_{\otimes}(SC) \succ \phi^{-}_{\otimes}(BSF)$ then   ; COMPARE LOCAL OPT. TO BEST SO FAR
    set BSF := SC                                      ; FOUND NEW BEST STATE SO FAR
  fi
                                                       ; ESCAPE LOCAL OPTIMUM
  apply meta-heuristics (new SC, modified neighborhood or $\phi^{-}_{\otimes}$)
done
return BSF
```

Figure 4.18: Heuristic search in pseudo code: local search + meta-heuristics.

The third extension to pure local search is a careful ranking of neighboring states according to the limits (similar to but more powerful than support in consistency-based algorithms) that are attached to the basic blocks of the solutions candidates.

Foth [1999] is the authoritative reference to the Frobbing procedure as a WCDG parsing method.

4.9.2 Guided local search

Guided local search [Voudouris & Tsang 1995] takes a different approach to the problem of local optima in heuristic search. Instead of ‘walking through the valley’, i.e., allowing occasional deteriorating steps, it uses an auxiliary scoring function as the objective function which is initially identical to the candidate scoring function $\phi^{-}_{\otimes}$ [Voudouris & Tsang 1995, p. 9]. While monitoring the progress, the auxiliary scoring function is modified such that local optima simply disappear for the adapted versions of the auxiliary function. Where Frobbing uses recently resolved conflicts, i.e., formerly violated constraints at a specific position on a tabu list, here violated constraints are used as solution features that characterize certain properties of search states [Voudouris & Tsang 1995, Section 6.1]. The scoring function is augmented such that states with solution features that have often been observed in local optima are downgraded.

Figure 4.19 on the facing page illustrates how a local minimum is ‘filled up’ for revised versions of the augmented scoring function $h_i()$ so that the local search can finally escape from the local minimum and find a better candidate.

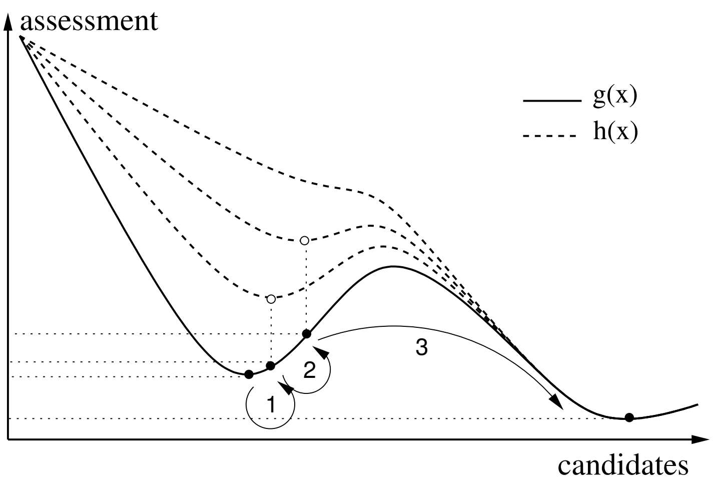


Figure 4.19: Guided local search: The augmented scoring function $h()$ is modified in local optima so that the optimum vanishes for the updated function. Figure adapted from Schulz & Menzel [submitted].

For the final decision of which candidate to return, the original scoring function is of course still available.

The procedure *Gls* employs a guided local search strategy for WCDG parsing but instead of the multiplicative scoring function it uses an additive cost measure as the auxiliary function in order to allow multiple soft constraints to overrule a hard constraint when trying to escape a local optimum. Schulz [2000] is the authoritative reference.

Gls cannot analyze word graphs with overlapping edges (such as the one in Figure 4.15 on page 102) at the time of this writing; it is therefore limited to a subset of simpler parsing problems compared to the other methods.

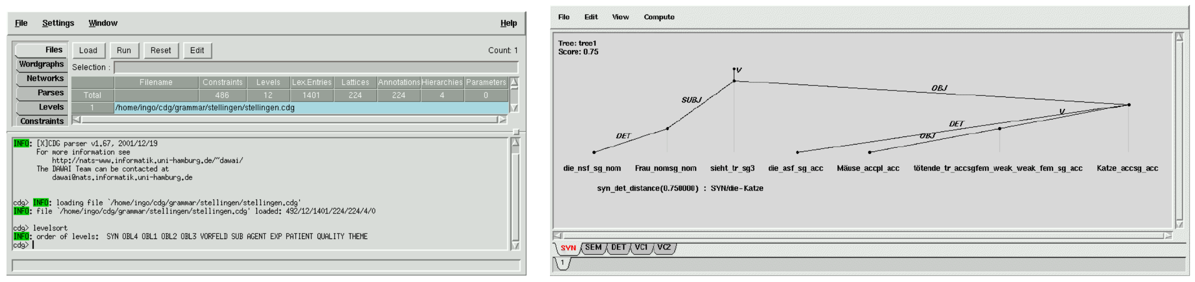
4.10 Implementation

A computer system [Schröder 1997d; Schröder 1997a; Foth, Schröder & Schulz 1998; Foth et al. 1999] for experiments with weighted constraint dependency grammars has been implemented[15] that includes the parsing methods described in Sections 4.6 to 4.9.

Besides this core functionality it provides facilities to develop and maintain grammars, i.e., sets of weighted constraints, lexica, ontologies and annotated corpora. A user can interact with the system through an integrated command-line tool or a graphical user interface (cf. Figure 4.20 on the following page).[16] The core functionality can also be accessed through an application programming interface and a library [Foth 1998a].

[15]The software, documentation and example grammars can be downloaded from `http://nats-www.informatik.uni-hamburg.de/~ingo/dawai/`.

[16]I am grateful to various people who have contributed to the WCDG parsing system. In particular, I would like to mention Kilian Foth and Michael Schulz. The latter also realized the graphical user interface XCDG.



	Filename	Constraints	Levels	Lex.Entries	Lattices	Annotations	Hierarchies	Parameters
Total		486	12	1401	224	224	4	0
1	/home/ingo/cdg/grammar/stellingen/stellingen.cdg							

```
INFO: [X]CDG parser v1.67, 2001/12/19
      For more information see
          http://nats-www.informatik.uni-hamburg.de/~dawai/
      The DAWAI Team can be contacted at
          dawai@nats.informatik.uni-hamburg.de


cdg> INFO: loading file `/home/ingo/cdg/grammar/stellingen/stellingen.cdg'
INFO: file `/home/ingo/cdg/grammar/stellingen/stellingen.cdg' loaded: 492/12/1401/224/224/4/0

cdg> levelsort
INFO: order of levels:  SYN OBL4 OBL1 OBL2 OBL3 VORFELD SUB AGENT EXP PATIENT QUALITY THEME
cdg> |
```

```
syn_det_distance(0.750000) : SYN/die-Katze
```

Figure 4.20: Graphical user interface XCDG. The left figure shows the main view with subsections for word graphs, constraint networks, dependency trees, databases of stores analyses and grammar information. A syntactic analysis is pictured in the right figure.

The system has been extensively used for experiments which are partly described in this thesis (cf. Section 4.11 and Chapter 5). It has also been embedded in a prototypical system for computer aided language learning [Menzel & Schröder 1998f; Menzel & Schröder 1999, cf. Section 5.4].

4.11 Experiments

A number of experiments have been run in order to empirically examine the properties of the algorithms for WCDG parsing.

The Stellingen corpus of approximately 220 utterances has been analyzed with a grammar which makes extensive use of soft constraints (cf. Section 5.2.1 for a description of the grammar and the corpus). No word graphs were used but instead linear sequences of possibly ambiguous words.[17] For all runs the computation time has been limited per problem instance. All experiments reported in this section were run on a PC with a 1.1GHz Intel Pentium III processor under the Linux operating system. Most results are given separately for varying problem sizes. To do so we split the corpus into partitions with different numbers of possible constraint values (after unary constraints); Table 4.3 on the next page lists the ranges and the number of instances per class.[18] The mean number of values per class is used as an x-value in the figures of this section.

4.11.1 Tree search

Figures 4.21 on the facing page through 4.23 on page 111 give experimental data for the systematic tree search procedure Netsearch.

Figure 4.21 on the facing page presents the influence of the agenda size on accuracy and efficiency. The agenda is restricted in size in order to limit the memory requirement of the algorithm.

[17]See remark in the previous section about the incapability of Gls to handle 'real' word graphs. Note that a procedure that is prepared to handle word graphs can possibly do worse on linear sequences of words than a similar procedure that cannot handle word graphs at all. However, no evaluation of the increase of the parsing difficulty due to word graphs has been conducted so far.

[18]A parsing problem which consists of more admissible values is not necessarily more difficult to solve than one with less. In practice, the figure is nevertheless a good approximation for parsing difficulty.

Range #val.	No.	Avg. #nodes	Min. #val.	Max. #val.	Avg. #val.
$0 \leq \# < 100$	9	55.000	49	95	77.556
$100 \leq \# < 300$	73	91.301	100	299	205.301
$300 \leq \# < 500$	47	130.702	304	492	386.723
$500 \leq \# < 700$	22	156.636	500	699	596.273
$700 \leq \# < 900$	17	178.529	701	876	769.059
$900 \leq \# < 12,00$	15	193.533	905	1,144	1,018.400
$1,200 \leq \# < 1,600$	13	231.154	1,226	1,591	1,445.538
$1,600 \leq \# < 2,500$	14	240.643	1,612	2,456	1,880.714
$2,500 \leq \# < \infty$	14	338.000	2,810	10,047	4,275.857

Table 4.3: Problem classes based on number of constraint values: ranges for the number of constraint values; number of problem instances per range; average number of nodes in these ranges; average, minimum and maximum number of values in these ranges

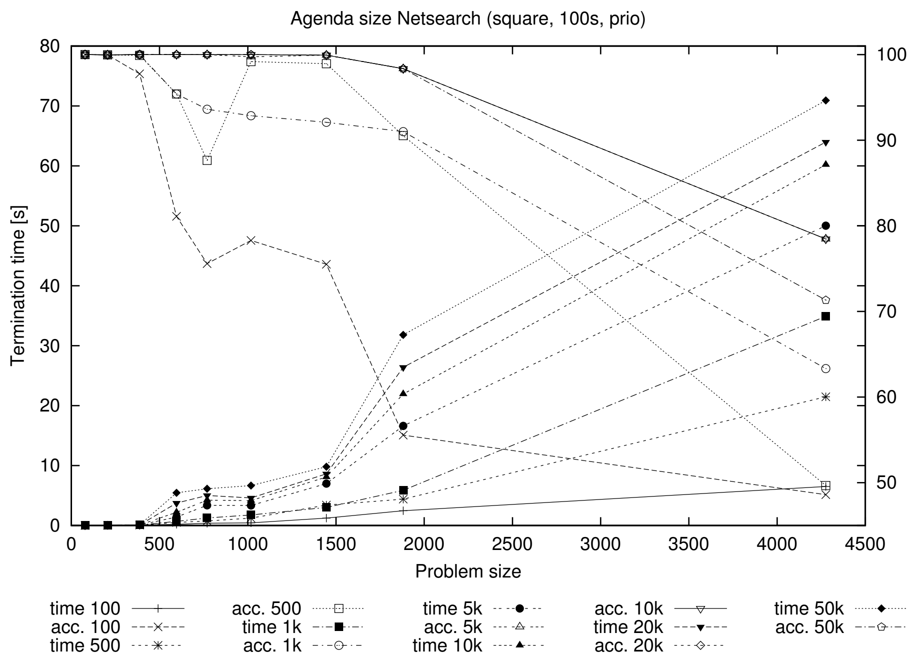


Figure 4.21: Accuracy and efficiency depending on agenda size for the solution method Netsearch.

Although it is hoped for of course, nothing guarantees that only those parts of the search space are pruned that do not contain solutions. A clear dependency of the needed processing time on the agenda size can be observed: The smaller the agenda, the faster the search. The general principle that larger agenda sizes lead to better accuracies because fewer solutions are pruned by mistake can also be confirmed, at least for agenda sizes smaller than 10,000. The larger agenda sizes 20,000 and 50,000 do not differ significantly in efficiency and accuracy, which can be explained by the fact that the limited processing time becomes the primary restriction.

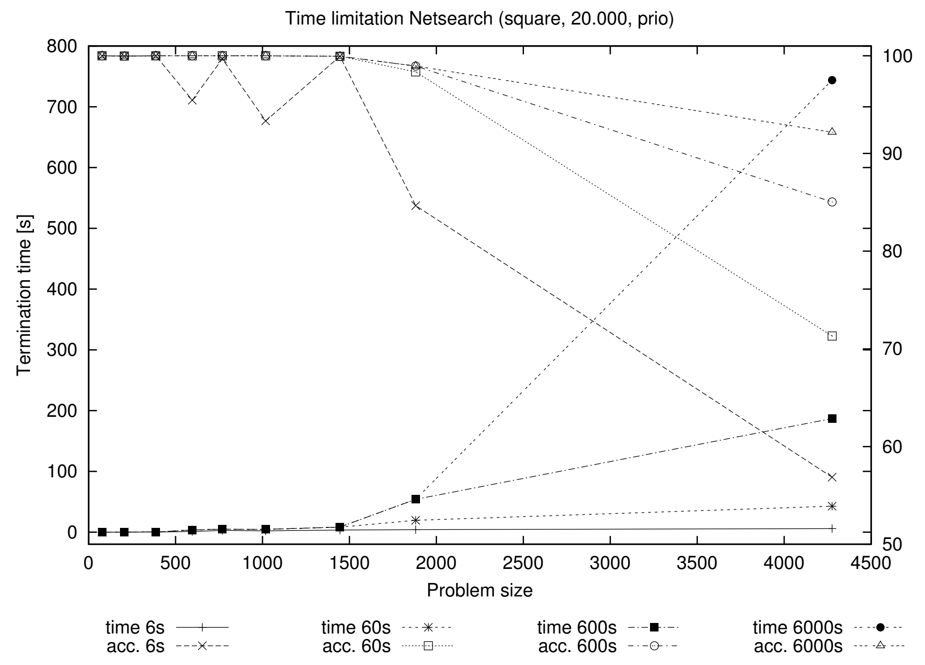


Figure 4.22: Accuracy and efficiency depending on the pre-set time limitation for the solution method Netsearch.

Figure 4.22 presents results from an experiment about the impact of the pre-set time limit on accuracy and efficiency. The figure suggests the possibility that the procedure Netsearch in fact suffers from the exponential explosion that is typical for combinatorial processes. While nearly no differences among the various runs are visible for the small problem classes, the large problems take much longer for those runs where a high time limit has been set. However, the additional time spent is not wasted; long lasting runs score better accuracy than runs that are prematurely stopped.

Figure 4.23 on the facing page relates three types of time measures and two types of accuracy.

Soft time is the amount of time needed to find a first solution candidate that does not violate a hard constraint. Solution time is the time until the final solution has been found, and termination time is the total time until the process terminates. It can be seen that the final solution is found shortly after the first soft solution and that the majority of time is spent between the final solution and the termination. This confirms the claim [Freuder & Wallace 1992] that branch-and-bound can supply a solution relatively early; nevertheless no anytime property can be diagnosed because the solution does not steadily improve over time but stays the same for long periods of time (cf. Section 5.5).

Accuracy can be given as the mean over all levels or for a particularly important level such as syntax. It can be seen that the accuracy for syntax is slightly worse than the overall one. This can be explained by the fact that all words must have a meaningful governor on the syntax level; in contrast, auxiliary levels such as semantic ones (cf. Section 3.4.6) often only contain meaningful links for a subset of the words.

Finally, it should be mentioned that the decreased accuracy for large problems is mainly due to failures to find a solution at all. For all classes but the largest problem size the procedure

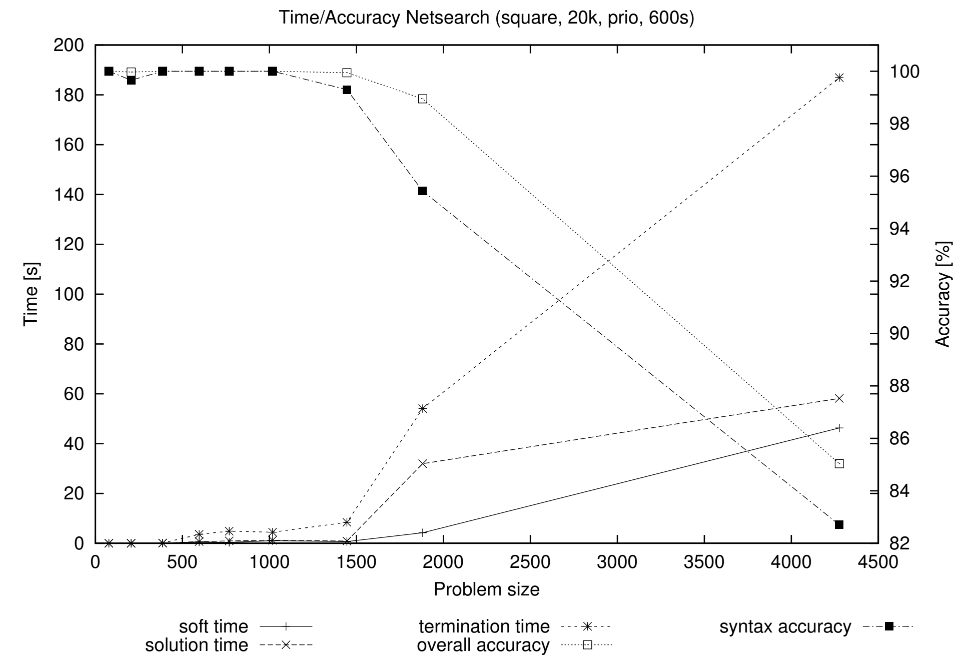


Figure 4.23: Three types of time measures and two types of accuracy for Netsearch: Soft time, solution time and termination time as well as overall accuracy and accuracy for the syntax level alone.

was able to find solutions for all problems. However, 14.3% of the problems in the last class could not be solved at all by the procedure which corresponds quite well to an overall accuracy of 84.5%.

4.11.2 Heuristic search with tabu list

Figure 4.24 on the next page gives the analogous experimental data for the heuristic search procedure Frobbing as does the last Figure 4.23 for Netsearch.

It stands out that the quality of the procedure is very high, even for large problems.[19] The syntax accuracy is again worse than the overall accuracy but this time a difference of four percent can be found which once more can be explained by the different types of failures. While Netsearch fails to find some solutions for the largest class of problems, the heuristic search methods always yield a result. The lack of a solution candidate results in a smoothing of the differences in parsing difficulty among the levels. Therefore, the gap between syntax and overall accuracy is larger for the heuristic search methods. The percentage of time spent after the final solution is found is smaller than that for the Netsearch procedure.

4.11.3 Heuristic search as guided local search

Figure 4.25 on the following page gives the corresponding experimental data for the heuristic search method Gls.

[19]Note the different scales in Figures 4.23, 4.24 on the following page and 4.25 on the next page.

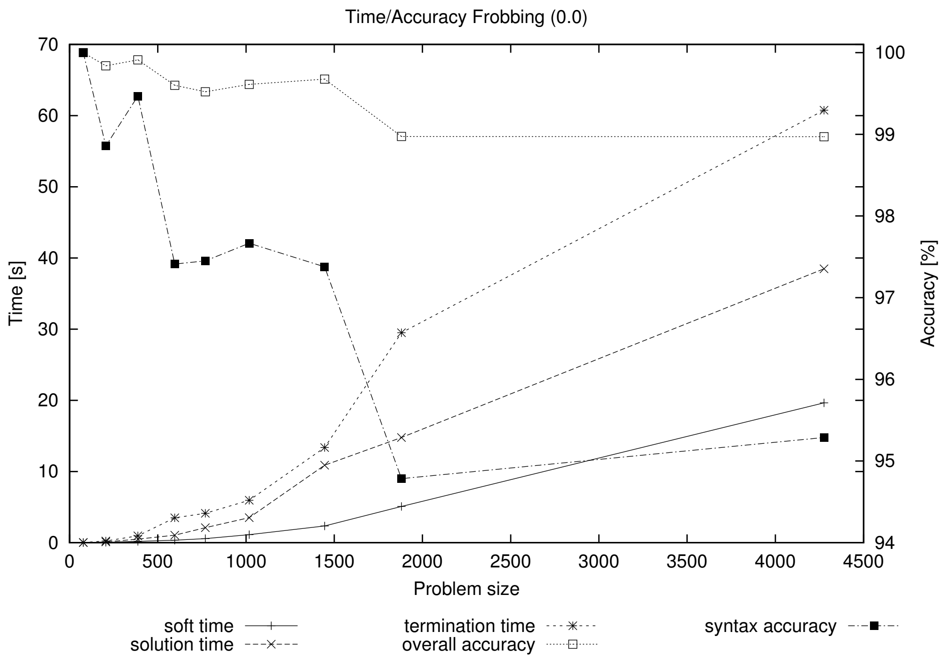


Figure 4.24: Three types of time measures and two types of accuracy for Frobbing: Soft time, solution time and termination time as well as overall accuracy and accuracy for the syntax level alone.

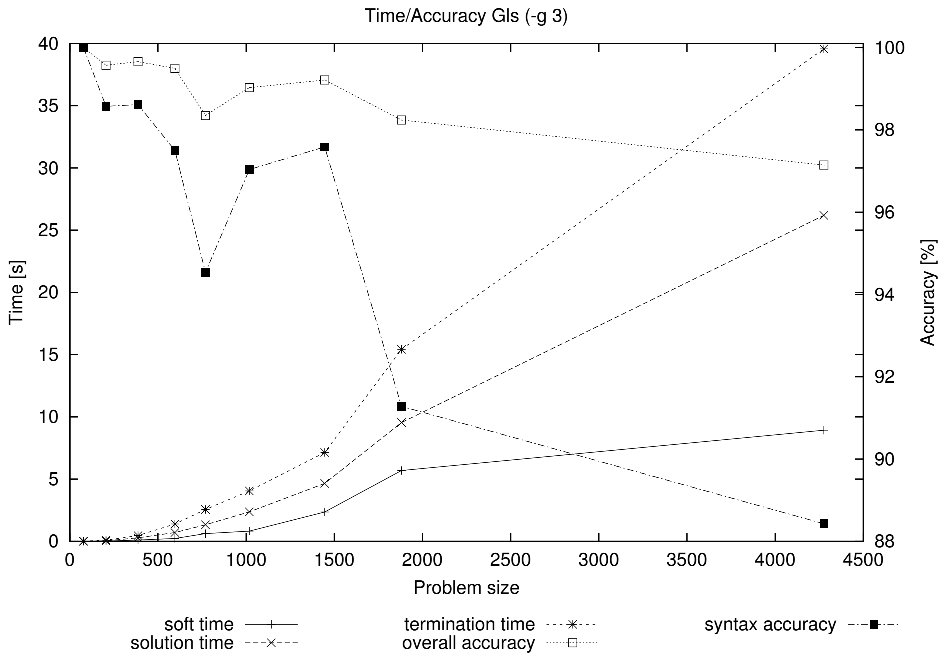


Figure 4.25: Three types of time measures and two types of accuracy for Gls: Soft time, solution time and termination time as well as overall accuracy and accuracy for the syntax level alone.

In general, the run of the curves is similar to those for Frobbing. However, the procedure tends to run faster at the cost of delivering a slightly worse accuracy than Frobbing. Again, the discrepancy between overall accuracy and that for syntax alone increases.

4.11.4 Comparison

Figures 4.26 and 4.27 on the following page finally compare the three solution methods directly.

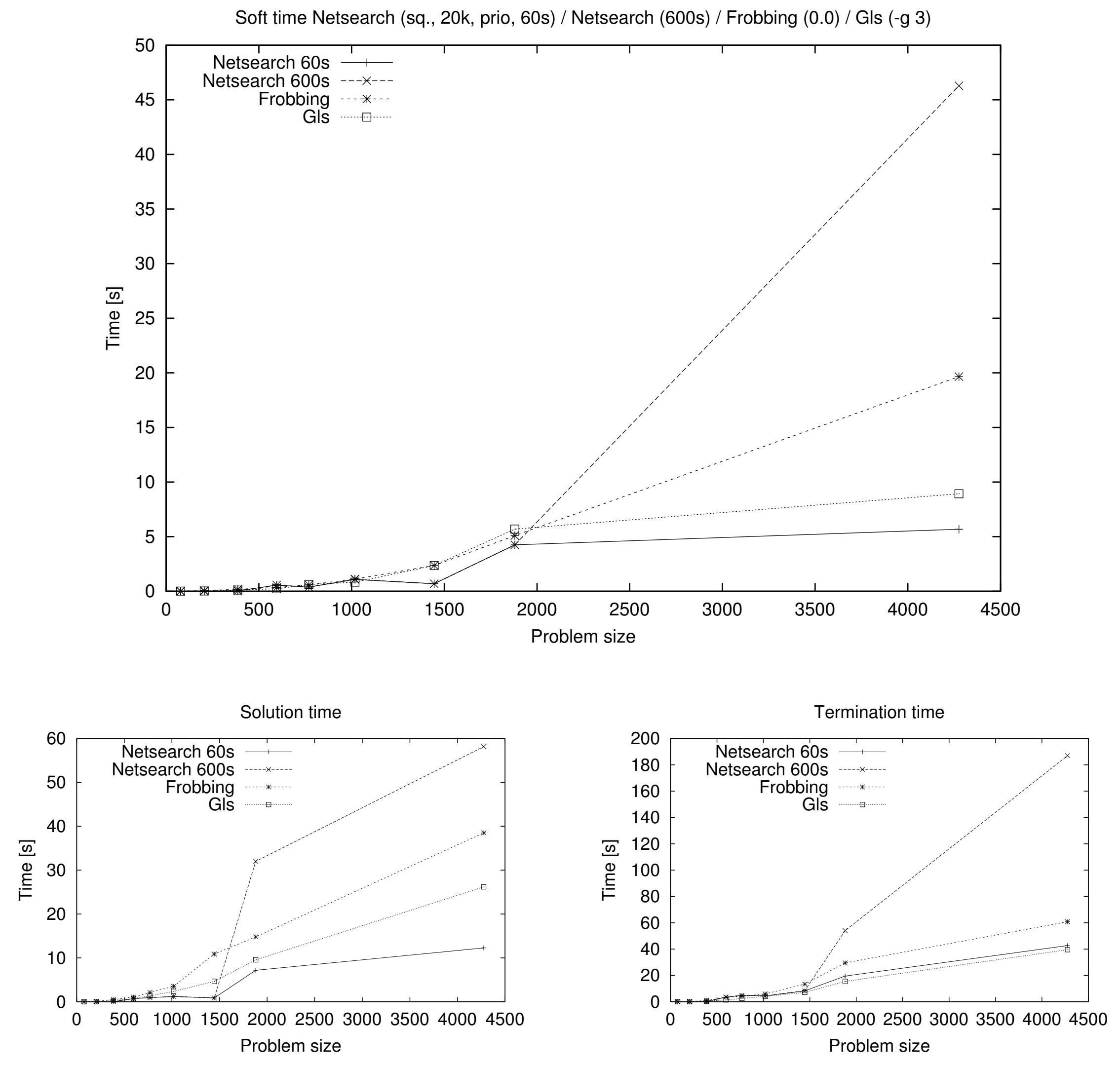
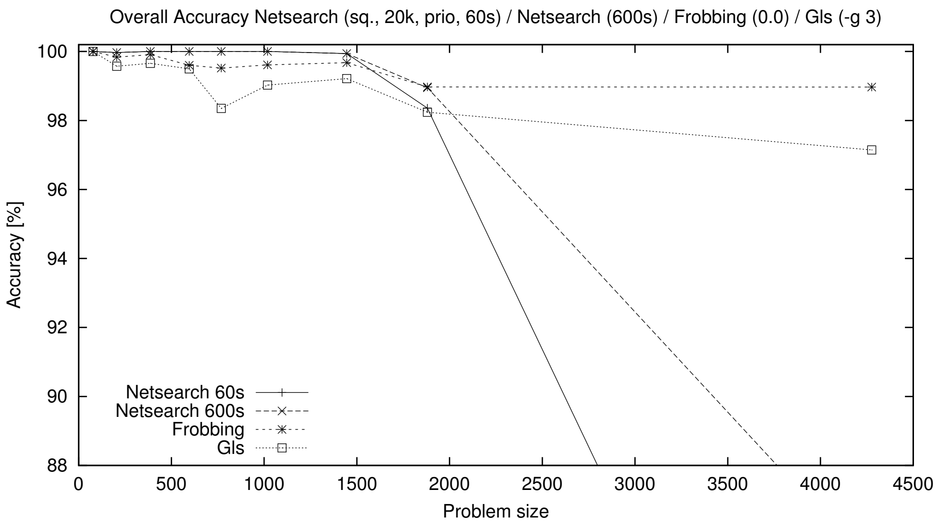


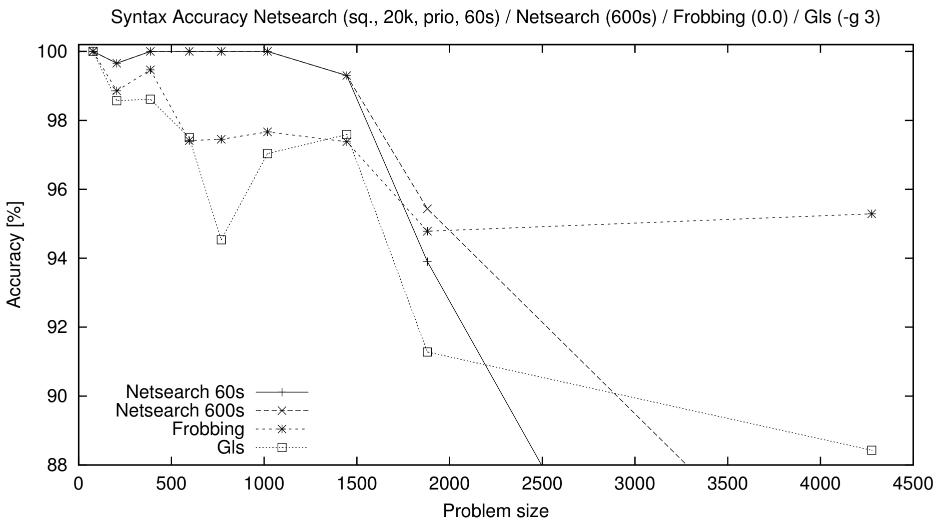
Figure 4.26: Comparison of soft, solution and termination time for various solution methods.

The runtime of the systematic tree search Netsearch grows combinatorially with the problem size. Only with a runtime limited to a reasonable amount of time (here 60s) can the procedure compare favorably to the heuristic search methods Frobbing and Gls. It achieves the best soft as well as the best solution time and is on par with Gls with regard to the termination time. Note that this is not only an artificial effect due to the failure of Netsearch to provide a solution at all. Only for the largest problems such failures exist (28.6% for Netsearch 60s, 14.3% for Netsearch

600s). If the runtime is not limited enough, the time needed grows fast with the problem size (see Netsearch 600s). The heuristic search variants Frobbing and Gls are more 'well-behaved' with respect to scalability. Even though their runtimes grow considerably when the problem size increases they do not show the combinatorial explosion typical for exponential processes. In general, Gls seems to run faster than Frobbing.



(a) Overall accuracy (Netsearch 60s/600s, out of displayed range: 71.3%/85.0%).



(b) Syntax accuracy (Netsearch 60s/600s, out of displayed range: 70.9%/82.7%).

Figure 4.27: Comparison of accuracy for various solution methods.

Figure 4.27 compares the algorithms with regard to the achieved accuracy. The systematic search procedure Netsearch scores the best accuracy possible for small and medium-sized problems (with one exception). It achieves results comparable to those of Frobbing for the second largest problem class but its accuracy breaks down for the class of the largest problems, which is due to the failure to produce a solution at all for some problems within the time allowed (see above).[20] The heuristic search variants Frobbing and Gls both show more constant results, even for the largest problems. However, neither one reaches the absolute correctness of a systematic search for small and medium-sized problems. In general, Frobbing achieves a slightly higher accuracy than Gls. For both heuristic search variants, the distance in accuracy to that of the systematic tree search even increases when moving from the overall accuracy to the syntax accuracy.

4.11.5 Summary

In conclusion, it can be confirmed that the parsing methods for WCDG show the characteristic behaviors of the algorithmic paradigms they are based on. A grammar writer and experimenter can choose the algorithm that best suits his needs. For small and medium-sized problems the systematic tree search method delivers accurate results quickly. For the largest problem class, the heuristic search procedure Frobbing manages to find very good results in a reasonable amount of time, and if time is absolutely crucial Gls returns results that are still good even faster.

4.12 Incremental mode of operation

Section 4.5.2 outlined a general incremental mode of operation for a solution method for constraint satisfaction problems. The rationale behind such a procedure is that values that are judged inconsistent based on the constraints among a subset of variables can never become consistent again, even if more variables have to be considered. This monotonic behavior of constraint satisfaction make CSPs promising candidates for incremental solution methods.

In the case of natural language parsing, however, an incremental mode not only means that not all *variables* are available when the computation starts, it also means that not even all *values* for the available variables are known. Variables roughly correspond to words in an utterance; values to dependency relations between words. A dependency link of the first word in a sentence to the last word can only be postulated and checked after the last word is available.[21] This intertwining of variables and values complicates the incremental case for WCDG parsing. The general case of dynamic CSP [Dechter & Dechter 1988] where arbitrary elements of the CSP can change over time can, for instance, be solved by a variant of the dynamic backtracking algorithm [Verfaillie & Schiex 1994a].

Mitjushin [1992] identifies the reason why dependency structures are well-suited for incremental analysis:

> "The reason is that the dependency structure of a segment is the same both when the segment is considered as isolated and when it is considered as a part of some

[20]Note that Netsearch never delivers solutions which violate any hard constraints while the heuristic search procedure does. Consequently Netsearch always scores zero percent accuracy for these problems, although it could easily get at least 47.9% which is the easily achievable baseline accuracy for this kind of measure (cf. Section 5.9.4).

[21]This double role of words both as variables and as values is also the reason why consistency-based methods based on arc consistency (cf. Section 4.6) need $O(n^4)$ time where n is the number of words, while arc consistency itself can be achieved in $O(n^2)$ where n is the number of constraint variables.

sentence [...] In our opinion, this comparison demonstrates that, in a certain sense, dependency structures reflect the incremental nature of sentence comprehension from left to right better than constituent structures do." —— Mitjushin [1992, p. 930]

Incremental natural language parsers can be divided into two classes: Either they always pursue all structural possibilities and only extend the set of hypotheses or they commit themselves relatively early to one or a small set of possibilities and remove the remaining ones. Chart parsers [Wirén 1992; Amtrup 1999] are typical examples of the first class. Deterministic parsers [Marcus 1987] and parsers with an early commitment strategy as well as repair mechanisms [Lombardo 1992] fall into the second.

In the context of WCDG, Schulz [1999] realized a parser of the first kind that utilizes an underspecified word node that is used temporarily to represent all future possibilities.

A WCDG parser of the second kind, described by Foth et al. [2000], robustly parses larger and larger prefixes of an utterance into intermediate structures. It takes advantage of the optimal structure of the previous smaller prefix to efficiently compute the structure of the next larger prefix by only allowing a limited amount of modifications and extensions of the previous structure. Verfaillie & Schiex [1994b] propose a similar re-use of intermediate solutions in the context of general dynamic constraint satisfaction problems.

Figure 4.28 presents some results of such a parser with regard to the achieved accuracy and the computation time needed.

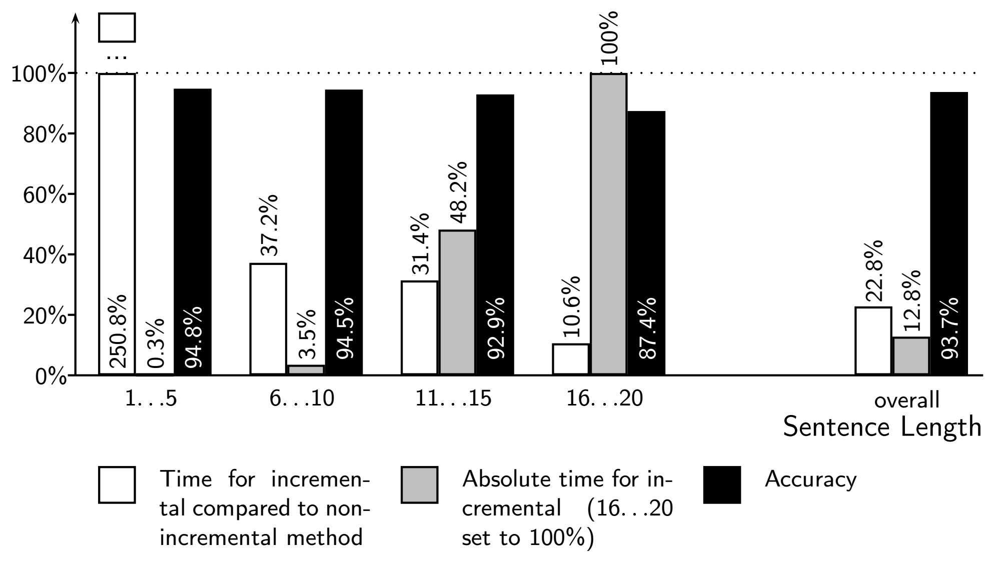


Figure 4.28: Incremenal WCDG parsing: Processing time and accuracy vs. sentence length. Numbers extracted from Foth et al. [2000].

The following observations can be made:

- The accuracy of the incremental parser is a little worse than that of the non-incremental one. The difference can be explained by the fact that only a subset of all structural possibilities are taken into account during re-analysis in incremental mode. The solution to the previous prefix and the update heuristics determine which dependency links are not even considered. Naturally, the heuristics make mistakes occasionally so that solutions are missed.

- For small problems, the incremental parser is less efficient than the non-incremental one. The reason for this behavior is, of course, that a non-incremental analysis is reasonably fast for small problems. Only for larger problems, the additional effort that is needed for re-analysis in the incremental case can be compensated for by the reduction of work that is possible because only smaller problems have to be solved at each step. In these important cases, however, a considerable amount of time can be saved: The incremental procedure is ten times faster than the non-incremental one for sentences longer than sixteen words.
- Even with the incremental parser, longer sentences take considerably longer to parse than shorter ones. Of course, no final statement can be made from only four data points, but it seems as if the growth rate of the runtime is super-linear, i.e., at least polynominal if not even exponential. This means that, although the suggested modification is a step towards the goal of a parser whose runtime grows only linearly in the number of words, this goal has not yet been achieved.

4.13 Optimizations

This section describes various modifications of the program system that improve the practical runtime. Their applicability depends on the specific application and they cannot change the theoretic order of growth of the solution methods. Nevertheless they are indispensable not only for real applications but also for systematic experimentation because only due to their speeding up the runtime can the program system be put to practical use. They allow the user to run experiments much more efficiently. We merely touch the surface of program optimizations in this section since literally hundreds of code improvements found their way into the system during the period of its still ongoing development.

Note that not all steps that seem promising at first automatically lead to an improved runtime. Only experimental evaluations can determine the actual impact and thus the practical use.

4.13.1 Sorting constraint nodes

A well known heuristic to improve the runtime efficiency listed in the constraint satisfaction literature [Meseguer 1989, for instance] is to change the order in which the constraint nodes are considered for assignment (cf. heuristic information in Section 4.2).

Only the branch-and-bound search method is influenced by these heuristics because the heuristic search methods do not have a notion of strict sequential assignment of value to nodes.

Three different orderings have been tried.

- **Natural order:** The constraint variables are assigned in the order in which their corresponding words appear in the word graph. For one and the same word, the nodes are sorted in the order in which the levels are declared in the grammar.

- **Domain size:** In this ordering, the variable with the smallest number of domain values is used first, and the more values are available the later the node is accessed. The motivation for this heuristic is that one tries to keep the search tree narrow at the top and wider at the bottom in order to reduce the number of search nodes.

- **Level priority:** As discussed in, for instance, Section 3.4.1, not all levels are equally important. Only some carry real linguistic information, and it depends on the grammar writer to decide which levels are of central importance. Therefore, the user of the system can associate priorities with levels and these priorities are used in this ordering scheme. A grammar writer might decide to include more (constraining) information on a syntactic level; in this case the syntactic nodes should be assigned first so that the effective constraints can be applied early.

Figure 4.29 presents the impact of the different node orderings on efficiency and accuracy.

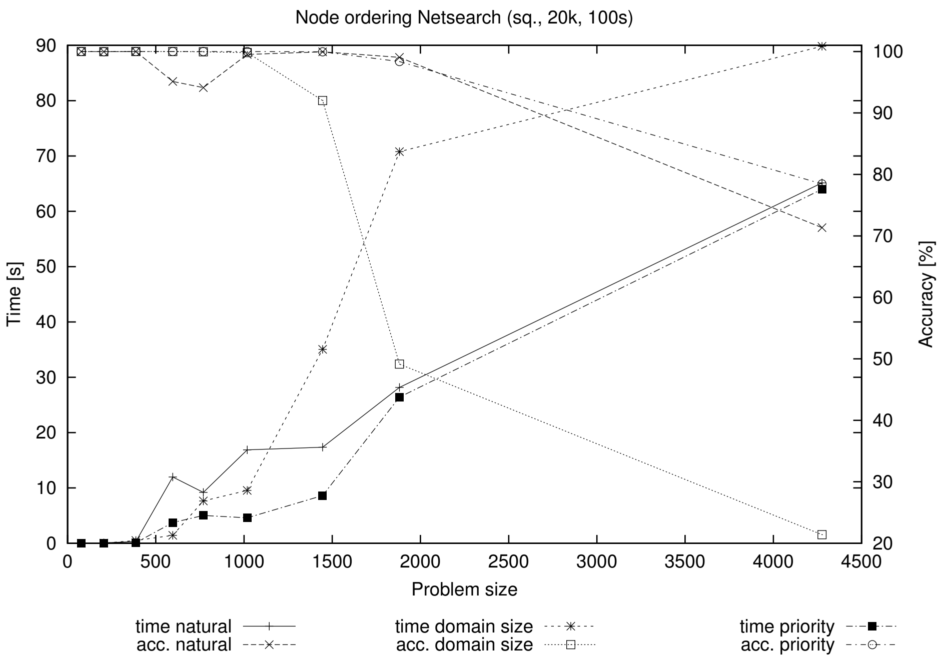


Figure 4.29: Impact of various node ordering strategies for Netsearch.

Surprisingly the general heuristic of assigning variables with small domains first is clearly the worst of all alternatives. It not only needs more computation time but it also achieves only a reduced accuracy. The node ordering that is determined by the level priority[22] yields the best results. The natural ordering is slightly worse than the best ordering for both efficiency and accuracy.

[22]Priorities have been determined by the number of binary constraints that are applicable to the corresponding level in order to approximate the constrainedness and importance of the level. For the Stellingen grammar, this resulted in the following order: `SYN` (syntax) `OBL4` `OBL1` `OBL2` `OBL3` (auxiliary levels for obligatoriness conditions) `VORFELD` (word order) `SUB` (subjunctive clauses) `AGENT` `EXP` `PATIENT` `QUALITY` `THEME` (thematic levels).

4.13.2 Agenda normalization

In Section 4.3.2 where systematic tree search was introduced we discussed three basic agenda strategies (cf. Figure 4.5 on page 85):

- A queue as the agenda leads to a breadth-first traversal of the search tree, which is not appropriate for WCDG parsing since *all* incomplete analyses have to be constructed before the first complete one is encountered, which makes the method intractable for large problems.
- Depth-first search only needs a lean stack as the agenda. However, it cannot be used for WCDG parsing because the valuable information about the score of partial solutions is almost completely ignored (except for value ordering within the constraint variables).
- An agenda that is sorted by the score of the partial assignments results in a best-first search which is generally employed in the systematic tree search procedure Netsearch.

A number of feasible variations of the naïve best-first method take into account that two partial solution candidates with equal score but different numbers of still missing assignments should be treated differently since the expected penalties resulting from future constraint violations of yet unassigned variables are different.

Three adjustment functions have been tried that map a score of a partial assignment ϕ to a score $f(\phi)$ used for agenda sorting.

- **Off:** This is the naïve best-first approach: $f(\phi) = \phi$.
- **Linear:** Since a partial assignment accumulates the violations of all assignments, the strategy multiplies the score with the number of assignments made so far: $f(\phi) = n \cdot \phi$ where n is the number of assignments made so far.
- **Square:** This approach uses the square root since penalties are aggregated multiplicatively in the WCDG framework: $f(\phi) = \sqrt[n]{\phi}$ where n is the number of assignments made so far.

Figure 4.30 on the next page summarizes the impact of the various normalizations on efficiency and accuracy.

Although the results are not as dramatic as for the node ordering schemes they nevertheless provide a clear picture. The adjustment that uses the nth square root of the assignment score as the basis for agenda ordering outperforms the alternatives in terms of both efficiency and accuracy. The linear adjustment gets medium placing.

The search algorithm obviously benefits from a strategy that mixes aspects from depth-first and best-first. The more relatively complete assignments are preferred, the more the behavior resembles depth-first. The square root method is closer to depth-first than the linear adjustment which in turn is closer than the naïve approach.

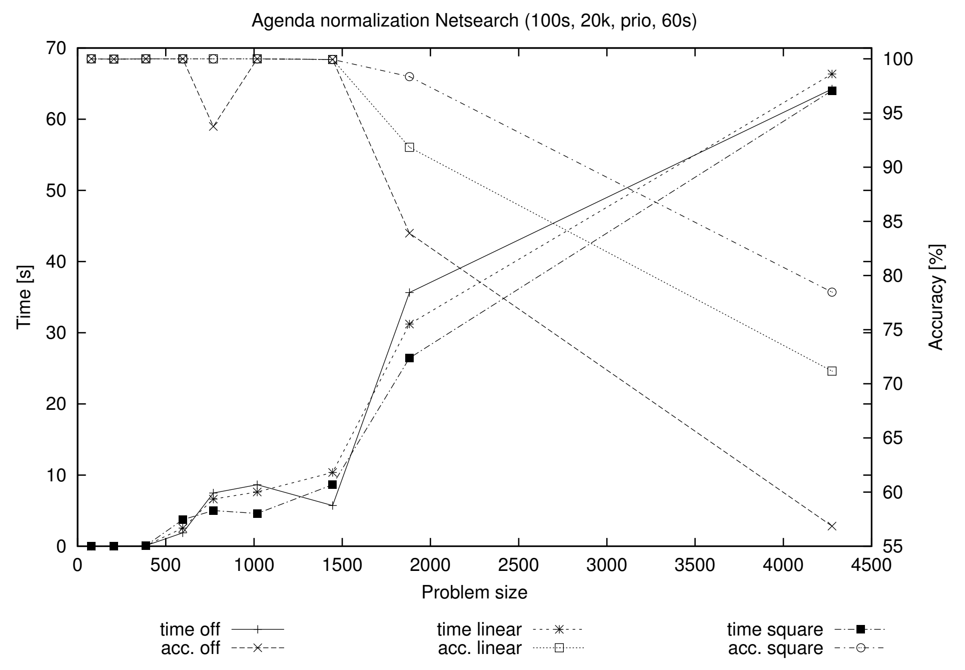


Figure 4.30: Impact of various agenda normalization schemes for Netsearch.

4.13.3 Constraint indexing

Constraints are usually tailored to specific levels. General constraints that are valid for all or at least some levels are relatively rare.[23] Therefore, constraints usually take the form of an implication where the premise ensures that the currently tested dependency edges belong to the correct level.

```
C-35 {X, Y} : UnderVerbVorfeld : Vorfeld :
       X.level=SYNTAX & Y.level=VORFELD &
       X^id=Y@id & X@pos<X^pos & Y@cat=FINVERB ->
         Y^id=X@id
```

Constraint C-16 on page 60, repeated here as Constraint C-35, is a typical example. The constraint is only relevant if the first edge is a syntactic one and the second one belongs to the level VORFELD. In all other cases the constraint is trivially satisfied because the premise evaluates to false.

Since the levels of the dependency edges are known during constraint checking, constraints can be indexed before the actual evaluation based on their levels. Then, only the *relevant* constraints are touched and only a small portion of constraint evaluations has to be carried out. This optimization cannot change the operational semantics of the system since irrelevant constraints are always satisfied and do not contribute to the assessment of a configuration.[24]

[23]One exception to this rule are constraints that deal with general properties of structures like acyclicity or projectivity (cf. Section 3.4.5).

[24]Side effects actually can lead to a modified behavior of the indexed constraints. The predicate `print()` (cf. Section 2.5.2) is the only source of side effects in the constraint language. It is mainly used while debugging a grammar.

This idea can be extended beyond an index based on levels. Often a specific spatial configuration of a dependency edge or of two dependency edges is a prerequisite for a constraint to be applicable. In addition to the level membership, this information can be used as an index key.

Extension of the constraint language

The syntax of constraints (cf. Table 2.2 on page 31) has been slightly extended to separate the information that can be used for indexing from the main formula. Conditions regarding levels or spatial configurations are moved into the signature of the constraints. Table 4.4 lists the extensions in EBNF.

```
Connexion  ::=  ‘,’ | ‘/\’ | ‘\/’ | ‘||’ | ‘==’ | ‘~=’ | ‘\’ | ‘/’
Direction  ::=  ‘:’ | ‘|’ | ‘\’ | ‘/’ | ‘!’
Formula    ::=  Variable Connexion Variable
             |  Variable Direction
VarInfo    ::=  Variable Direction String
VarList    ::=  VarInfo | VarInfo Connexion VarInfo
```

Table 4.4: Extension (in EBNF notation) of the syntax of the constraint language for constraint indexing. These rules supplement or replace those in Table 2.2 on page 31. Given this extension, the limitation to at most binary constraints (cf. Section 2.6) is anchored even in the syntax of the constraint language.

The signature of a constraint not only introduces the placeholders for the variable bindings (often `X` and `Y`), but also restricts the applicability of the constraints to certain configurations, e.g., to variables belonging to a certain level. Two kinds of restrictions can be used in the signature and both have an equivalent formulation in the formula body.

If the syntactic element VarInfo (cf. Table 4.4) in the signature of a constraint has, for instance, the form `X!FOO` and the formula is `F`, then the restriction of the signature can also be written as `(X! & X.level=FOO) -> F`, i.e., the information is a prerequisite for the constraint to fail. These limitations of the orientation of a dependency edge are called directions.

In binary constraints, the signature may contain information about the spatial configuration of the two involved edges. A restriction `X:FOO /\ Y:BAR` in a signature can be moved to the formula by using `(X.level=FOO & Y.level=BAR & X^id=Y^id) -> F` instead of the original formula `F`. These limitations of the configuration of two dependency edges are called connexions.[25]

Direction

Directions describe the spatial orientation of a single dependency edge. The following enumeration lists the possible forms and the equivalent formulations.

- `X|` (equivalent to `root(X^id)`)
  The edge has no governor, i.e., the corresponding word is the root.

[25]Note that the element ‘connexion’ of our constraint language is related but not identical to the general concept that has been introduced in Section 3.2.

- `X!` (equivalent to `~root(X^id)`)
  The edge has a real governor, i.e., it is not the root.
- `X/` (equivalent to `distance(X^id, X@id)<0`)
  The edge leads to the right, i.e., the governor is a word to the right of the dependant.
- `X\` (equivalent to `distance(X^id, X@id)>0`)
  The edge leads to the left, i.e., the governor is a word to the left of the dependant.
- `X:` (equivalent to `true`)
  No further restriction.

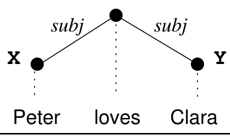
Connexion

Connexions describe the 'spatial' configuration of two dependency edges. The following enumeration lists the possible forms, equivalent formulations in parentheses, descriptions and example constraints. For some cases a violating configuration is provided as a side figure.

- `X/\Y` (equivalent to `X^id=Y^id`)
  The governors of both edges are identical.

```
C-36 {X!SYN/\Y!SYN} : subj_uniq : 0.0 :
       X.label=subj -> Y.label!=subj;
```

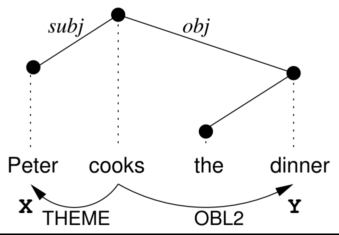
At most one subject per verb.



- `X\/Y` (equivalent to `X@id=Y@id`)
  The dependants of both edges are identical.

```
C-37 {X:THEME\/Y!OBL2} : THEME_OBL2_parallel : 0.1 :
       Y@args:2:role=THEME -> X==Y;
```

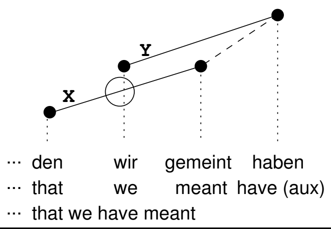
An auxiliary level edge for obligatory complements follows a thematic edge (if required by the lexical entry).



- `X||Y` (equivalent to `X^id!=Y^id & X^id!=Y@id & X@id!=Y^id & X@id!=Y@id`)
  All involved dependants and governors are disjoint.

```
C-38 {X!SYN||Y!SYN} : projectivity : 0.0 :
       min(X@from, X^from) < min(Y@from, Y^from) ->
         max(X@from, X^from) > max(Y@from, Y^from) |
         max(X@from, X^from) < min(Y@from, Y^from);
```

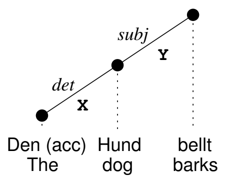
Dependency edges do not cross (cf. Constraint C-20 on page 62).



- `X\Y` (equivalent to `X^id=Y@id`)
  The governor of `X` is identical to dependant of `Y`, i.e., there exists a dependency chain `X@id` → `X^id` = `Y@id` → `Y^id`.

C-39
```
{X!SYN\Y!SYN} : subj_case_art_verb : 0.1 :
  X@cat=ART & X^cat=NOUN & Y.label=subj ->
    Y^subcat:subj:case=X@case;
```

The article of a (subject) complement noun (maybe underspecified for case) agrees with the verb with regard to case (cf. Section 3.4.2 on page 57).



- `X/Y` (equivalent to `X@id=Y^id`)
  The governor of `Y` is identical to dependant of `X`, i.e., there exists a dependency chain `Y@id` → `Y^id` = `X@id` → `Xid`.

- `X==Y` (equivalent to `X^id=Y^id & X@id=Y@id`)
  The edges are parallel edges on different levels (cf. Constraint C-37 on the preceding page).

- `X~=Y` (equivalent to `X^id=Y@id & X@id=Y^id`)
  The edges are inverse to each other.

- `X, Y` (equivalent to `true`)
  No further restriction.

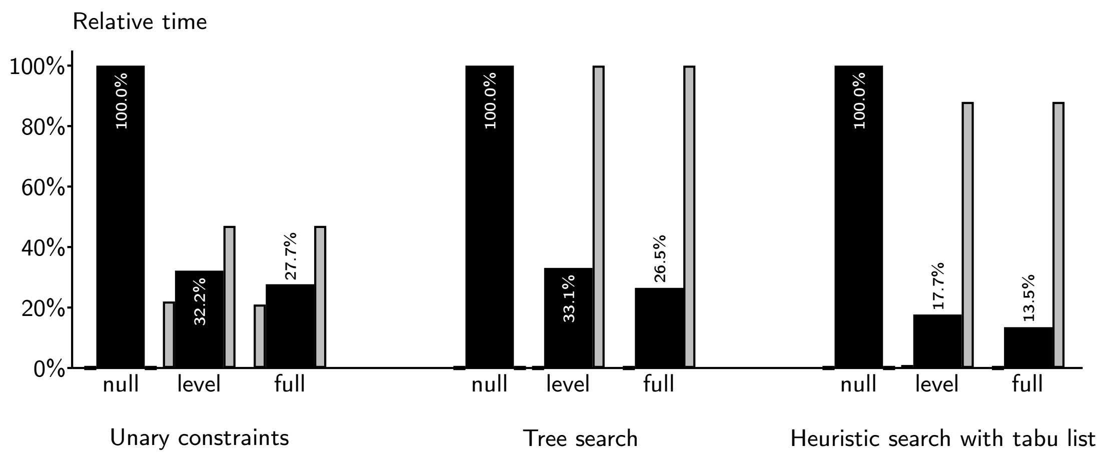
Example

Constraint C-40 is semantically equivalent to Constraint C-35 on page 120 but has some information moved from the formula body to the signature of the constraint. The complete information of level membership, spatial configuration of the edge `X` and how the two edges are arranged can then be used to compute an index so that only relevant edges have to be checked for compatibility with this constraint.

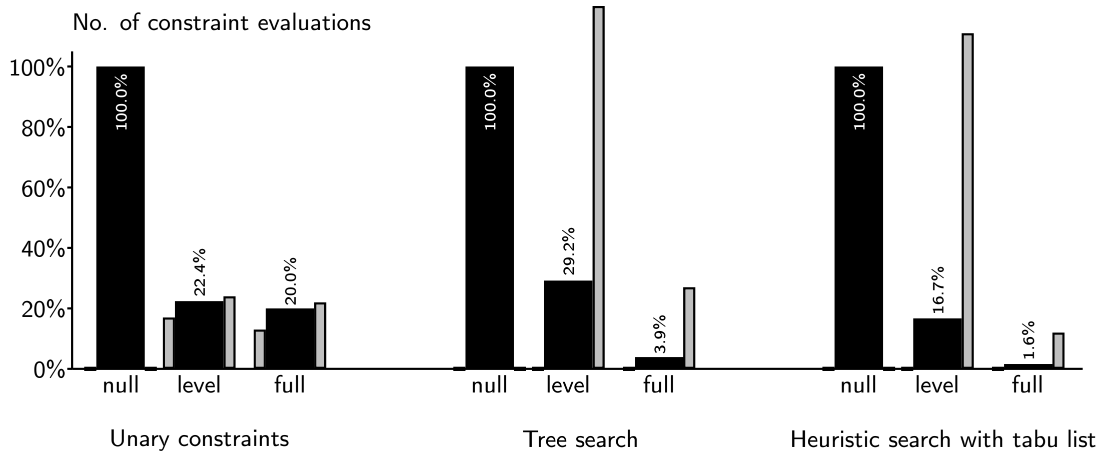
C-40
```
{X/SYNTAX\Y:VORFELD} : UnderVerbVorfeld : Vorfeld :
  Y@cat=FINVERB -> Y^id=X@id
```

Evaluation

Figure 4.31 on the following page shows the results of an experiment that deals with the impact of two different types of constraint indexing on the efficiency of the system. No indexing is used as a baseline to which a level indexing and a full indexing scheme are compared. The comparison is done for three operations (application of unary constraints only and complete analysis using tree search as well as heuristic search), and two measures (time and number of constraint evaluations) are given. Taking into account that 12 levels were used in the grammar and assuming an (over-optimistic) uniform distribution of constraints to levels, the theoretical lower bound for the level indexing method is 8.3% for unary constraints and 0.7% for binary ones (systematic tree and heuristic search). It turns out that these minima cannot be reached due to asymmetries in the grammar, but a huge gain in efficiency can still be observed. Of course, binary constraint evaluations benefit much more than unary ones because there are many more possible indices if one considers the combination of two levels and configurations instead of only a single level and configuration. The bigger part of the improvement is already achieved by the level index but the full index nevertheless facilitates an additional enhancement. The number of constraint evaluations benefits more than the time requirement because the indexing directly



(a) Relative time requirements



(b) Relative number of constraint evaluations

Figure 4.31: Relative runtime and relative number of constraint evaluations for two types of constraint indexing compared to the baseline with no indexing. The black bars represent the means over a corpus while the small gray bars on the left and the right give the minima and maxima respectively.

reduces the number of evaluations while the practical runtime is influenced by other factors as well. The heuristic search method benefits most from this optimization because it always evaluates the full set of constraints for a given configuration, even if a hard constraint is violated already. In contrast, the systematic tree search immediately abandons the branch when a hard constraint is violated.

4.13.4 Further optimizations

This section describes some additional optimizations that have been implemented in the WCDG grammar system.

- **Compact level values:** Every implementation of the solution methods must have a representation of the combination of a word form (including the lexical entry) and a level. These items are the elementary building blocks from which the dependency edges are constructed. The constraint variables of Section 2.4 on page 27 are closely related to these units, too. However, for a large number of words, lexical variations exist that only differ in a small number of features. For instance, *run* as a finite verb occurs with the morphosyntactic feature combinations `1st-sg`, `2nd-sg`, `1st-pl`, `2nd-pl` and `3rd-pl`. A similar situation can occur in word graphs where edges that cover the same time span resemble each other, such as the personal pronouns *she* and *he* that are distinguished by a single feature gender. Furthermore, only a small portion of the levels (or more precisely: constraints defined for these levels) refer to these distinguishing features. Therefore, it makes sense to fold those units consisting of a lexical entry and a level where the lexical entries are equivalent with respect to the level, i.e., no constraint exists that behaves differently for the lexical entries on this level. A good example are semantic homonyms, i.e., word forms with identical syntactic features but different semantic features like *bank* (financial institute and river side). All levels that do not deal with semantic information can safely fold the two readings, thus saving a lot of work. Foth [1998b] gives a detailed description of compact levels and how optimal partitions of lexical entries can be constructed.

- **Indexing of lexical information:** Recent years have seen a trend to lexicalization of natural language grammars [Pollard & Sag 1994, for instance] and WCDGs also offer this possibility to the grammar writer (cf. Section 3.4.1 on page 53 for an example). Lexical entries are recursive data structures in our system (cf. Section 2.3.2 on page 22) and constraint evaluations often require the extraction of certain pieces of information that are embedded in these entries. In order to avoid the expensive recursive descent during constraint evaluation, all lexical entries are flattened (cf. Figure 2.3b on page 24) while they are read into the system and all paths within lexical entries and within constraints are mapped to unambiguous indices. This technique reduces the extraction to a simple array lookup. A similar technique (but in a different context) is used for the compilation of attribute-value matrices of the grammar formalism ALE [Carpenter & Penn 1994] into first order terms of the programming language Prolog.

- **Constraint compiler:** Constraint evaluations are one of the most frequently executed operations in a constraint system. Hence, they are often used as a measure of how efficient a specific solution method is. In the WCDG system, for instance, an analysis may need several millions of these evaluations. Since the constraints (or more exactly: their formulae) are recursively defined (cf. Table 2.2 on page 31), an evaluation usually is the application of an also recursively defined interpretation function to the formula and the dependency edge(s) to be tested. This interpretation process returns whether the constraint is satisfied or violated. In order to accelerate the constraint evaluation, a compiler has been implemented which translates the constraints into C functions, then calls an appropriate C compiler and dynamically loads the object code into the address space of the WCDG parser. Thereafter all constraints are evaluated by the native functions instead of the recursive interpretation process. The integration of compiled constraints is seamless, i.e., compiling, loading and unloading are all online operations that do not require any modification of the system, in particular no re-compiling. Switching back and forth between interpreted and compiled mode as well as mixing compiled and interpreted constraints is possible. Pre-compiled object files of constraints can be re-used in later parser runs and are automatically recognized during the load operation.[26] Details about the implementation can be found in Hagenström [2001].

- **Symbol table:** Finally, as a further example of program optimization, it should be mentioned that the whole WCDG system operates on symbols (stored in a system-wide table) instead of strings. Symbols are efficient representatives of strings and, in particular, can be compared in time that is independent of the string length. Since string comparisons are very frequent, a considerable speed-up can be observed.
- **Hashing constraint evaluation:** Constraint evaluation is a frequent operation. Hashing the results and returning the pre-computed information if the information is requested a second time therefore saves some computation time. The different solution methods benefit to different degrees.

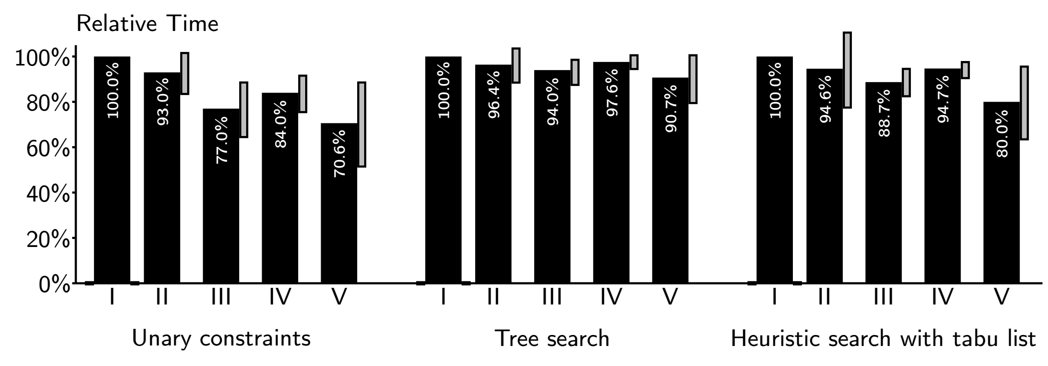
Figure 4.32 on the facing page presents comparative results for the impact that the compact level values (labeled II), the constraint compiler (labeled III) and the indexing of the lexical information (labeled IV) have on the efficiency of the system. The numbers are relative to a baseline with all optimizations disabled (labeled I). This baseline already corresponds to the best results of Figure 4.31 on page 124. The improvements are at least partially orthogonal to each other as can be seen from the results for the combined optimizations (labeled V). Two measures for the efficiency are used again: runtime in Figure 4.32a on the facing page and number of constraint evaluations in Figure 4.32b on the next page.

Of course, only the compact representation of level values actually reduces the number of constraint evaluations; the indexing of lexical information and the constraint compiler merely speed them up. Since the runtime of the parsing has been limited in the evaluations, the number of constraint evaluations sometimes actually increases for all optimizations other than the compact representation of level values. More constraints can be evaluated in the fixed amount of time since the evaluation itself is faster.

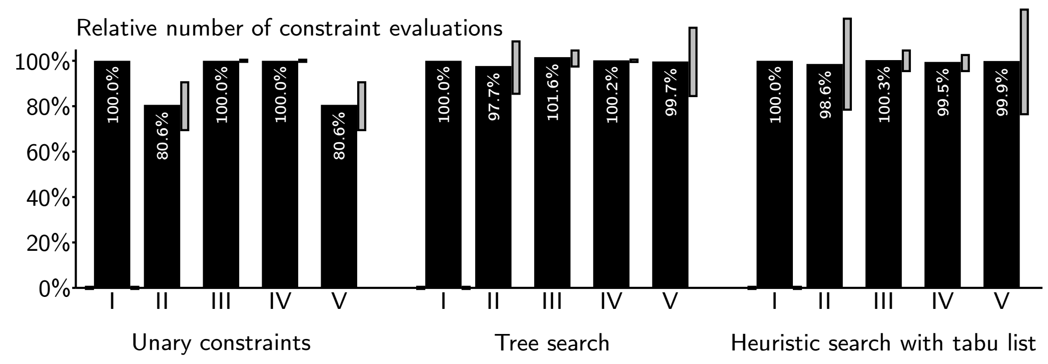
4.14 Related work

Algorithms for general constraint satisfaction problems and constraint satisfaction optimization problems have been studied extensively. There exists an abundance of literature, for instance general overview articles [Meseguer 1989; Kumar 1992; Miguel & Shen 2001b; Miguel & Shen 2001a], monographs about systematic search [Kanal & Kumar 1988] as well as about heuristic search [Aarts & Lenstra 1997b; Voß et al. 1999] and textbooks [Tsang 1993; Hromkovič 1998] all of which contains further references. Wallace & Freuder [1995a] compare heuristic (min-conflicts, walk-minconf, retry-minconf) and systematic (branch-and-bound, forward checking) methods

[26]Harper et al. [1995, p. 848] already describe a similar compiler for their CDG constraints which however seems to lack a seamless integration into the parser.



(a) Relative time requirements for different program optimizations



(b) Relative number of constraint evaluations for different program optimizations

Figure 4.32: Relative runtime and relative number of constraint evaluations for three optimizations and for the combination of them: (I) no optimization (baseline), (II) compact level values, (III) constraint compiler, (IV) indexing of lexical information and (V) combined (II)-(IV). The wide black bars are the means over a corpus and the small gray bars indicate the range $[\phi - \sigma, \phi + \sigma]$ where ϕ is the mean and σ is the standard deviation.

for CSP solving with regard to their anytime properties. Probabilistic performance profiles are given. They [Wallace & Freuder 1995b] suggest using heuristic methods first to find a good approximation and continue with systematic techniques to ensure optimality. Gomes, Selman & Kautz [1998] add randomization to complete, systematic, backtrack-style search procedures in order to break the unpredictability of running time which may seem promising for the application of WCDG parsing as well. If the heuristic function ranks n possible actions (e.g., an extension of a partial solution) to be within the H percent best assessment then the next action is selected randomly among those n possible ones. Additionally a cutoff parameter is used to restart the search whenever it seems to be stuck. Bookkeeping makes sure that no part of the search space is explored more than once. The cutoff parameter, however, does limit the size of the space that can be searched exhaustively between restarts. The authors give a formal account for so-called heavy-tail problems and for the restart strategy.

Waltz [1975] first suggested the filtering algorithm that Maruyama [1990b] proposed for CDG parsing. But only the extension to multiply-segmented constraint satisfaction problems [Zoltowski et al. 1992; Helzerman & Harper 1996; Harper et al. 1999b] made the application to lexically ambiguous sentences and word graphs feasible for the consistency-based methods (cf. Section 4.6).

4.15 Summary

Algorithms for WCDG parsing have been studied.

Standard techniques of CSP and CSOP solving have been described: consistency-based methods, systematic tree search and heuristic search. Criteria for an evaluation of algorithms have been developed and typical characteristics of these algorithmic families have been identified. The standard techniques have then been applied to and adapted for the WCDG parsing problem; experimental evaluations have been conducted.

- Consistency-based methods (cf. Section 4.6) have proven themselves problematic for WCDG parsing since no definition of consistency is available for soft constraints which on the one hand filters out a considerable amount of structural alternatives and on the other hand does not delete (too many) wrong values based on local decisions.
- Heuristic pruning (cf. Section 4.7) is a mixture between local consistency and global search in that it takes more information into account for the decision which candidate to discard next but cannot guarantee global correctness. Although different pruning schemes were tried and limited success was achieved, heuristic pruning has not been identified as a satisfactory solution method for WCDG parsing.
- Systematic tree search (cf. Section 4.8) can solve small and medium-sized WCDG problems fast and absolutely reliable. For the largest class of problems however the theoretical exponential complexity shows up in practice and either an external limitation of the processing time or of the agenda size or of both are required in practical cases.
- Heuristic search (cf. Section 4.9) solves even large WCDG parsing problems in moderate time and yields a only slightly suboptimal quality for all problem classes. However, it cannot prove optimality of a solution and depends on external termination criteria.

An application can therefore base its decision of which search method to employ on the varying characteristics that have been found and on the actual requirements.

Preliminary experiments about an incremental mode of operations showed that the early commitment strategy can lead to increased system performance without the loss of a considerable amount of accuracy. Finally, a set of necessary optimizations of the algorithms has been described and evaluated.

Chapter 5

Experimental Results

5.1 Overview

This chapter presents results of various experiments within the weighted constraint dependency grammar (WCDG) framework. It will be shown that the proposed approach to gradation in natural language is well suited to handling the soft conditions of natural language listed in the introduction, such as grammatical deviation, preference, multi-level interdependence and (external) uncertain information. Some attractive properties that are due to the concept of gradation can be confirmed for WCDG parsing.

Section 5.2 describes the experimental environment; the corpus, the grammar and evaluation criteria that were used in most of the experiments throughout this chapter are introduced.

Section 5.3 deals with robust parsing with WCDGs.

In Section 5.4, a diagnosis component capable of detecting errors in language use is discussed that is based on a robust WCDG parser.

The anytime property of algorithms that we discussed in Section 4.4.3 is revisited in Section 5.5 and experiments with both the basic methods and meta-control mechanisms are described.

The subsequent Section 5.6 presents details about what influence the integration of acoustic weights from a speech recognizer has on the parsing process. Similarly, Section 5.7 outlines the integration of another external component, namely a probabilistic part-of-speech tagger, into the parsing system.

In Section 5.8, the weights of a small number of conflicting soft constraints that model the isolated linguistic phenomenon of extraposition of German relative clauses are estimated from an annotated corpus.

Section 5.9 describes an experiment in automatic learning of a complete set of WCDG constraint weights.

5.2 Experimental environment

5.2.1 Stellingen grammar and corpus

This section introduces the grammar and the corpora we used in the experiments described in Section 4.11 and throughout this chapter. Two factors prevent the use of standard corpora, such as the English Penn Tree Bank [Marcus, Santorini & Marcinkiewicz 1993] or the German NEGRA corpus [Skut et al. 1997b; Brants, Skut & Uszkoreit 1999], as the domain of WCDG parsing in this thesis:

1. Style and depth of annotation: An analysis with weighted constraint dependency grammars parses a natural language utterance into multiple dependency analyses on different representational levels. While the problem of a dependency model being different from the phrase structure model of standard corpora can be overcome by a mapping [Collins 1996; Collins 1997], the amount of annotated information cannot easily be enlarged. In particular, information about the semantic structure needed for a WCDG annotation is not available. Even worse, nearly no assumption about the domain of general-purpose corpora can be made so that an inclusion of domain knowledge is rarely feasible.

2. Grammar development: Designing and implementing a WCDG is a labor-intensive task that essentially still has to be done manually (but see Sections 5.8 and 5.9). While very large corpora are an advantage for automatically generated grammar systems (such as probabilistic grammars) because they allow for a more precise training of the parameters, we had to limit the set of linguistic phenomena present in the corpus in order to keep manual grammar development feasible.

Therefore we developed our own grammar and corpus, named *Stellingen* grammar and corpus, for testing of and experiments with the WCDG parsing system.

The grammar is based on the general techniques described in Chapter 3. It is the successor to an earlier WCDG grammar [Heinecke & Nolda 1998; Nolda & Heinecke 1999; Heinecke & Schröder 1998] but departs from its predecessor in so far as it focuses less on the traditional notion of crisp grammaticality and is more concerned with actual language data, practical solutions and, in particular, soft constraints and preferences. It does not try to *accept* the sentences of the German language but aims at assigning the *most plausible structural description* to any input.

The expected input format is a sequence of words, i.e., unambiguous word forms (containing no acoustic alternatives) which however have been extracted from spoken language dialogues. That does not mean that the grammar or parser are limited to word sequences. In fact, the sequences have been enriched with alternatives originating from a finite-state chunker which identifies likely non-recursive phrases, mainly for date and time expression. Hence, the typical input consists of small word graphs with some overlapping arcs labeled with artificial words such as DATE.

It features twelve representational levels: syntax (`SYN`), four auxiliary levels for modeling obligatoriness (`OBL1`, `OBL2`, `OBL3`, `OBL4`), word order of the vorfeld (`VORFELD`), an auxiliary level for subordinate clauses and miscellaneous purposes (`SUB`) and five semantic levels modeling thematic roles (`AGENT`, `EXP`, `PATIENT`, `QUALITY`, `THEME`). Some 500 constraints are contained in the grammar; 44% are hard, 56% soft constraints; 14% of the soft constraints are dynamic constraints; 65% of the constraints are unary, 35% binary.

The lexicon consists of some 1,400 word forms which expand into 7,000 unambiguous lexical entries, i.e., the average lexical ambiguity on the feature level is 4.97 entries per word form.

The employed test corpus consists of approximately 220 utterances which have been taken from the Verbmobil [Wahlster 2000] corpus of appointment-scheduling dialogues. However, the Verbmobil turns of spontaneously spoken language have been hand-segmented into 'sentence-like' sections. Interjections and punctuation marks (if any were introduced by the annotators) have been removed. For instance, Examples E-66 and E-67 are extracted from the original Verbmobil annotation `g460ac` of Example E-65.

E-65 ja , Wochenende wei"s ich noch <!1 no'> nicht <!1 nich'> . ich hab' mal <!1 ma'> meinen <!1 mein'> Kalender <:<#> mitgebracht:> . schauen <!1 schau'> wir <!1 me> mal , wann Sie Zeit haben . <#>

E-66 ich habe mal meinen Kalender mitgebracht

E-67 schauen wir mal wann Sie Zeit haben

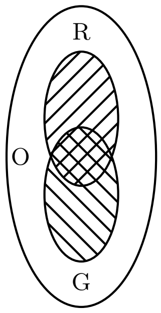
The sentences have an average length of 8.7 words (sd 3.96, min 3, max 29). There are 442 word types and 1,941 word tokens in the corpus. The corpus has a mean lexical ambiguity of 3.53 lexical entries per word form. See Section 5.7.2 for further corpus statistics.

The grammar has been designed to minimize modeling errors, i.e., it assigns the highest grammar score to the desired structure that has been included in the annotations. However, it could not be guaranteed that the scores of two arbitrary *suboptimal* structures always reflect their ranking with respect to their accuracies when compared to the annotation. All experiments (except when integrating external assessments) therefore assume that a reduced accuracy is due to search errors but not to modeling errors.

5.2.2 Criteria

This chapter mainly uses accuracy and efficiency (cf. Section 4.4) as evaluation criteria. Unless otherwise stated, average processing time per sentence on a 1.1GHz Pentium III running the Linux operating system will be reported as an efficiency measure. However, the focus will be on comparative results and care has been taken not to confuse results from different experimental settings, such as different machines or grammars. Parsing usually is limited to 60 seconds per utterance (cf. Section 4.11) unless otherwise stated. In most of the experiments, labeled accuracy on the syntax level is used as an accuracy measure. If, however, the experimental settings do not allow such a measure, we will report sentence accuracy, e.g., in Section 5.3, or f-measure (cf. Figure 5.1 on the next page), e.g., in Sections 5.6 and 5.9.

For an evaluation, guesses $G \subseteq O$ of a solution procedure for a problem set can be compared to a reference $R \subseteq O$. Both G and R are subsets of the set of all possible outcomes $O = I \times S$ which is the cross product of problem instances I and problem solutions S.



Two criteria are widely used for evaluation. *Recall* refers to the completeness of a solution, while *precision* means correctness of the provided solutions. A frequently used combination is the *f-measure.*

$$\begin{aligned}\text{recall } r &= |G \cap R|/|R| \\ \text{precision } p &= |G \cap R|/|G| \\ \text{f-measure } f &= 2/(1/r + 1/p)\end{aligned}$$

It is clear that an ideal recall of one can easily be achieved by identifying G with O. Analogously, a precision of one is always possible with $G = \emptyset$ (by definition). However, the goal is to achieve both, a high recall and a high precision.

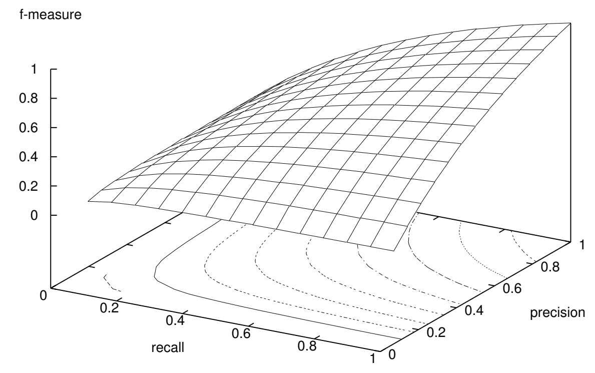


Figure 5.1: Recall and precision.

5.3 Robustness

This section looks at the property of *robustness* in greater detail and describes experiments which investigate the claim that weighted constraint dependency grammars can contribute to a robust system behavior.

5.3.1 Motivation

Although we mentioned robustness several times already (for instance, in Sections 3.5.2 and 3.5.4), no explicit definition has yet been given. A crisp or even formal definition for robustness seems difficult because the term inherently refers to the notion of unseen, unexpected and unpredictable events which are in turn difficult to grasp.

System robustness is sometimes defined as the

> "... extent of the ability of a system [...] to continue to function despite the existence of faults in its component subsystems or parts."
> —— T1A1 [2000, → system robustness]

This definition addresses an internal robustness where the faults occur internally to the system while the environment or type of input stays constant. In the context of natural language processing, however, robustness usually means robustness against changes of the external world:

- A grammar writer cannot possibly anticipate the complete variability of future inputs, e.g., ungrammatical input or constructions that are not yet covered by the grammar. A robust system manages to deal with such cases (maybe with degraded performance). The rest of this section focuses on this aspect of robustness.
- Some input sources for NLP systems are potentially faulty, e.g., recognition uncertainty in noisy environments for speech or blurred writing for optical character recognition. Section 5.6 deals with the additional ambiguity in the input of a parser when dealing with speech.
- Sometimes additional external requirements are demanded dynamically from a system, e.g., temporal pressure or a request for correction or a more detailed analysis. Section 5.5 discusses how a component can react meaningfully to temporal requirements.

We follow Chanod [2001] in the particular respect that the scope of robustness is not limited to parsing of linguistic phenomena that are currently not covered by the grammar. This is in contrast to approaches that exclusively deal with the latter aspect under the term of extra-grammaticality [Weng 1993].

> "Robustness is about exploring all constructions humans actually produce, be they grammatical, conformant to formal models, frequent or not. Linguistic phenomena, regardless of their oddity or frequency, account for the meaning of whatever segment of text they appear in. [...] In this view, robustness is a matter of broadth and depth of linguistic analysis. Altogether."
> —— Chanod [2001, p. 188]

In particular, the following questions are investigated in this section:

- What are the prerequisites for robust parsing?
- Does the inclusion of graded preferences lead to increased robustness?
- Can WCDGs cope with errors in the linguistic input in the sense that small deviations do not hamper the correctness of the output and only multiple faults lead to more and more undesired results?

5.3.2 Means to achieve robustness

At least two properties play an important role in robust system behavior: the ability to accommodate inconsistent evidence and the use of multiple knowledge sources.[1]

1. Inconsistencies are ubiquitous in natural language (cf. Introduction), not only in ungrammatical input, but also within the grammatical system. The soft constraints of WCDG allow the grammar writer to incorporate fall-backs for faulty input and uncertain knowledge as well as different kinds of preferences (cf. Section 3.5). All these conditions can be violated during the analysis of an utterance. Furthermore, the weights facilitate a fine-grained gradation so that constraint violations are treated differently depending on the severity of the error.

2. The second major reason for robust behavior besides weighted constraints results from the inclusion of (partially redundant) information originating from different knowledge sources, which is made possible by the simultaneous analysis on multiple levels (cf. Section 3.4.7).

   Figure 5.2 on the facing page gives an example of the fruitful mutual influence of different levels.[2] On the syntax level, the utterance shows an ambiguity with regard to the attachment of the prepositional phrase. If it can be assumed, however, that the system has strong evidence for the domain information that only Peter has access to the telescope, this knowledge can propagate through the levels and finally determine the syntactic PP-attachment.

   The inclusion of redundant information is necessary to compensate errors or unexpected cases. If only pre-defined, unambiguous and well-formed sentences are to be analyzed, additional levels such as domain information about ownership only lead to a greater depth of analysis but this could also be achieved by a sequential architecture.

5.3.3 Experiments

In order to verify the claim that WCDGs are well suited for robust analysis, an experiment has been conducted [Menzel & Schröder 1998a] where both the error degree of the input and the amount of supportive evidence in the grammar have been artificially controlled. Syntactic errors have systematically been introduced into a single utterance (cf. Example E-68).

E-68	Der	Mann$_{\text{nom}}$	besichtigt	den	Marktplatz$_{\text{acc}}$.
	The	man	visits	the	marketplace.

The following error metric has been used.

- **Case agreement**: Confusing nominative and accusative case counts as a double fault for both the subject and the object while an erroneous genitive case is a single error. Dative case was omitted because it produces the same results as genitive.

[1]Chanod [2001, p. 196] speaks about inclusion of background knowledge and tractable constraint relaxation mechanisms in this context.

[2]Compare this to the use of semantic constraints proposed by Maruyama [1990a] and Harper et al. [1993] as described in Section 2.7.

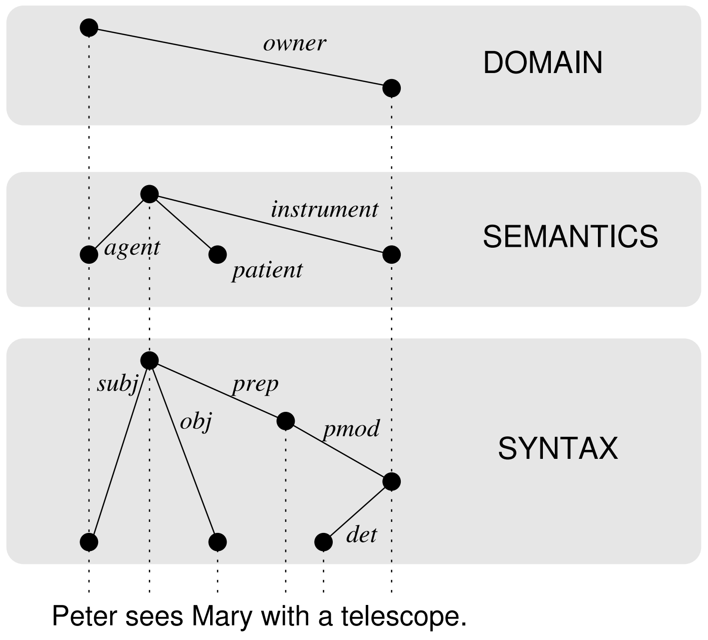


Figure 5.2: Dependency structures for three levels of analysis: Certain information from the domain level propagates through semantics to the syntactic level.

- **Number agreement**: Violating the number agreement between verb and subject again counts as two error points.
- **Word order**: The word order (subject before object) is not mandatory although preferred in German. We therefore increase the error count by only one in the case of a marked word order.

The resulting 72 variants (with error counts from zero to seven) have then been analyzed in three different contexts where domain and semantic information either support, contradict or are neutral with respect to the desired outcome.

Figure 5.3 on the following page summarizes the results. The use of semantic and domain-specific knowledge greatly enhances the syntactic robustness in the supportive case. While 50% of the results for sentences with an error measure of three were wrong when only a syntactic level was represented, even 83% of the utterances with an error count of five are interpreted correctly when enough semantic and domain-specific support is available. Naturally, the robustness is reduced if the additional evidence contradicts the intended interpretation.

Although we emphasized the robustness against syntactic deviations, the robust behavior is symmetric with respect to the different levels. Thus, positive information on the syntactic level also helps to find a semantic interpretation. This can be best seen in the three-dimensional Figure 5.4 on page 137 where the accuracy is shown depending on syntactic and, at the same time, on semantic and domain-specific deviation. Whether semantics helps syntax or vice versa is just a matter of interpretation depending on the point of view. Different levels are mutually beneficial.

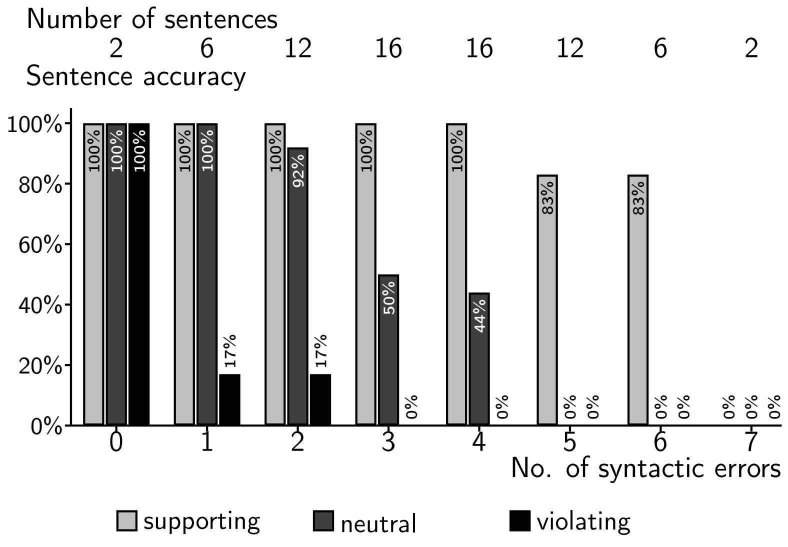


Figure 5.3: Percentage of correct analyses depending on the number of syntactic errors and, additionally, parameterized by the kind of semantic and domain-specific support. The number of sentences for each number of syntactic errors is given above the bar chart.

Foth, Menzel & Schröder [submitted] replicated the experiment for the larger Stellingen grammar and real utterances that have been artificially distorted by changing the inflection of words, deleting a word from the sentence and reversing the word order of two siblings in a constituent. They confirm the increased robustness due to soft constraints.

However, a word of caution is appropriate here. The synergistic effect of different representational levels is, of course, highly dependent on the actual grammar. While the experiment described in this section has proven that truly redundant information from different knowledge sources can have an important beneficial impact on the robustness of the overall system, such behavior cannot be found for all kinds of grammars. The levels of syntax and semantics of the Stellingen grammar, for instance, are highly interwoven so that not all the levels sustain their independence. Additionally, the semantic evidence is very poor in the domain of the corpus used, i.e., appointment scheduling dialogues, since semantically weak verbs are used almost exclusively and domain information, such as information about the calendars of the participating dialogue partners, is completely omitted. When trying to analyze regular, i.e., largely grammatical sentences from that domain using the Stellingen grammar but switching off the semantic levels, one finds that the accuracy is not necessarily lower than with semantic levels. Table 5.1 on the facing page gives experimental results which confirm that the quality of a syntactic analysis even increases slightly for the Stellingen grammar when semantic levels are omitted from the analysis. It has to be assumed that in this case the robustness does not benefit from the semantic levels either. WCDGs with tightly coupled levels behave in this respect similar to integrated approaches to syntax and semantics as found, for instance, in HPSG.

Of course, adding semantic levels always provides a richer or deeper analysis.

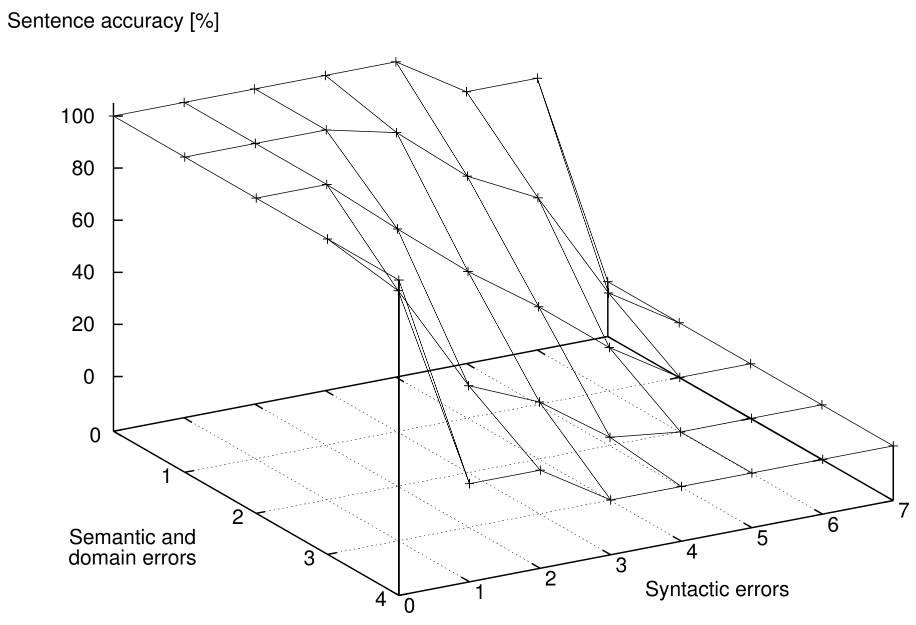


Figure 5.4: Percentage of correct analyses depending on the number of syntactic errors and on a semantic and domain-specific error count. A violation of sortal restrictions or domain information counts two error points while the absence of semantic and domain knowledge is counted as one error respectively. This results in semantic judgments in the range from zero to four error points. The different levels of representation are treated in a completely symmetrical way.

	Tree search		**Heuristic search with tabu list**	
	with sem.	**without sem.**	**with sem.**	**without sem.**
Accuracy (all)	90.85%	93.35%	97.46%	97.65%
Recall (all)	90.89%	93.44%	97.44%	97.65%
F-measure (all)	90.87%	93.40%	97.45%	97.65%
Accuracy (solved)	97.43%	97.29%	97.46%	97.65%
Recall (solved)	97.48%	97.39%	97.44%	97.65%
F-measure (solved)	97.46%	97.34%	97.45%	97.65%

Table 5.1: Accuracy, recall and f-measure (for the syntactic level) for experiments with and without semantic levels using the Stellingen grammar.

5.3.4 Related work

The problem of robustness in natural language processing is nowadays mainly tackled through partial parsing techniques [Abney 1996a] and probabilistic grammars [Charniak 1993; Young et al. 1997] in computational linguistics.

Finite-state automata and finite-state transducer [Abney 1990; Abney 1996a; Karttunen 1993; Karttunen et al. 1997; Karttunen, Gaál & Kempe 1997; Koskenniemi 1990; Koskenniemi, Tapanainen & Voutilainen 1992] recognize structures and transform them into larger 'chunks'. These chunks often cover the input sentences only partially and are defined not to include recursive elements. Finite-state techniques process texts extremely fast because usually no backtracking is allowed.[3]

Shallow parsing [Federici, Montemagni & Pirelli 1996], in contrast, leaves certain decisions underspecified which are difficult to make locally, for instance attachment of prepositional phrases. Early versions of the formalism of *constraint grammar* [Karlsson 1990; Karlsson et al. 1995] which is based on manual finite-state rules and aims at syntactic dependency structures also omitted the precise attachment position in some cases by, for instance, only constraining the governor of a word to be a noun to the left. The successor of the constraint grammar parser for English (ENGCG) is not subject to this drawback; it establishes complete linkages between words [Tapanainen & Järvinen 1997; Järvinen & Tapanainen 1997].

Chart parsers [Earley 1970; Kay 1986] in bottom-up mode also lend themselves to partial parsing because partial structures are already contained in the chart when a complete analysis fails [Ehrlich & Hanrieder 1996].

Probabilistic approaches to NLP [Chelba & Jelinek 1998; Collins 1999; Charniak 1993; Magerman 1994; Ratnaparkhi 1998; Charniak 2000; Brill 1996; Bangalore & Joshi 1999] are blossoming because they overcome some of the intrinsic problems of manual and non-probabilistic grammars. They always yield the 'most plausible' result based on some kind of probability model. Hence they usually achieve complete disambiguation and feature a high degree of error tolerance. One of the most important advantages is that probabilistic grammars are usually automatically induced from (annotated) corpora, thus solving the problem of grammar acquisition and easing portability.

All of the above approaches share one drawback: They cannot argue how good an input utterance is. They hide the notions of deviation and grammatical error either behind too small a depth of analysis or behind the term of joint probability which does not allow a distinction between unlikely and deviant events. WCDGs, in contrast, have an explicit notion of grammatical rules and preferences which easily identify faults (cf. Section 5.4).

This capability is shared by some extensions of standard parsing which either employ a robust version of a grammar [Goeser 1992; Uszkoreit 1991; Erbach 1993] or augment the parsing procedure with additional operations such as generalized rules in a chart parser [Mellish 1989] or word deletions and insertions in a GLR parser for speech applications [Rosé & Lavie 1997; Rosé & Lavie 2001].

Using semantic and possibly domain-dependent information to improve parsing is a well-established technique. Head-driven phrase structure grammar [Pollard & Sag 1987; Pollard & Sag 1994, HPSG], for instance, tightly couples syntactic and semantic aspects in a single feature structure so that on the one hand constraints about the semantic type of complements are checked during

[3]Instead, a repair mechanism is suggested in some cases [Abney 1990]. Note that the use of finite-state machinery does not automatically guarantee efficient algorithms. Two-level morphology [Koskenniemi 1983; Karttunen & Beesley 1992], for instance, is computationally expensive because there are choice points in the processing of the input, i.e., the underlying machinery needs to be non-deterministic [Barton, Berwick & Ristad 1987, Chapter 5].

structure building but on the other hand the slightest incompatibility leads to parsing failures. The consideration of semantic aspects within the CDG framework has already been suggested by Maruyama [1990b, p. 35] and also by Harper et al. [1993]. However, both only add semantic *constraints* which take effect directly on the syntax structure and cannot be violated by a valid solution. Thus semantic information is used to reduce the ambiguity, but no independent semantic structure is built. Autonomous parallel structures for syntax and semantics in the context of CDG have first been proposed by Menzel [1995].

5.3.5 Summary

We identified two prerequisites for robust behavior: (a) the ability to deal with inconsistencies between the input and the grammar and (b) the simultaneous consideration of all available knowledge sources so that the loss of syntactic evidence (possibly due to inconsistencies) does not lead to parsing failures. The potential of the WCDG parsing approach for robust parsing has been investigated in an artificial framework of controlled but highly systematic linguistic deviations. Even without any semantic support, the grammatical system showed a remarkable degree of robustness against syntactic errors. The inclusion of additional semantic and domain-specific support further boosted the immunity against syntactic faults. Not unexpectedly, semantic evidence that contradicts the desired outcome hampers the system with regard to robustness. We have found that the relation between syntactic and semantic evidence is one of mutual support, i.e., not only the syntactic analysis is easier with semantic backup but a semantically odd or new sentence can also be analyzed correctly when the syntactic evidence is clear. Such situations arise when new statements in a discourse violate the expectations or assumptions of a hearer. A replication of the experiment with a larger corpus is desirable. However, corpora of utterances which show systematic faults are not available, especially when the deviations have to be annotated. Foth, Menzel & Schröder [submitted] try to overcome this data acquisition problem by using a procedure which semi-automatically introduces a number of error types into a given corpus of sentences.

5.4 Error diagnosis

5.4.1 Motivation

Computer assisted language learning (CALL) systems should help the learner to practice the use of a foreign language and to overcome individual difficulties.[4] Goals pursued include, for instance, vocabulary drill, practicing of isolated grammatical constructions and – more sophisticated – exercising free rephrasing of statements or interactive pronunciation training [Menzel et al. 2000; Menzel et al. 2001]. Off the shelf systems usually employ simple techniques such as multiple-choice, fill-in the gaps or other methods that only require a small list of valid answer keys. If a CALL system is supposed to give comprehensive and highly accurate feedback, however, a parsing component must be included which meets at least two requirements:

[4]Cf. European Association for Computer Assisted Language Learning (EUROCALL) (`http://www.eurocall.org/`) which, among others, publishes the ReCALL journal (partially available from their web page) and organizes the annual EUROCALL conferences.

Robustness: Since it must be expected that the learner enters highly deviating sentences, the system must be able to cope with such input and it must provide a reasonable interpretation even for non-standard utterances.

Diagnosis: A suitable parsing component must not only be capable of processing deviant input, it must also be able to provide error detection and error explanation facilities at the same time. Robust parsers based on statistical models or on neural networks, for instance, often cannot offer this additional meta-information because they lack an explicit notion of linguistic rules.

Good user feedback is only feasible if, on the one hand, the intended meaning of an utterance can be inferred and if, on the other hand, the location and type of mistakes can be determined.[5]

This section tries to answer the question of whether and how the WCDG parser can be used as a diagnosis component in a CALL system.

5.4.2 Prototype

WCDGs provide good support for both tasks that have been identified as prerequisites for advanced CALL systems. Robustness is an inherent property of WCDGs (cf. Section 5.3) and diagnosis is easily achieved with WCDGs because grammatical faults more or less directly correspond to constraint violations. Figure 5.5 on the next page presents the outline of an architecture for a CALL system based on a WCDG parsing component that yields two types of results, a linguistic structure and a set of constraint violations.

A prototypical CALL system [Menzel & Schröder 1998b; Menzel & Schröder 1998a; Menzel & Schröder 1999] has been implemented based on this idea. A picture of a small town scene (cf. Figure 5.6a on the facing page) has been used as the contextual embedding for the user but a textual representation is equally possible.

The contextual information has also been encoded as constraints so that the language system could use some domain information as background knowledge. Since grammatical and domain knowledge are processed with the same mechanisms of the parser, and constraint violations are used to indicate faults, not only syntactic and semantic errors but also errors regarding the propositional content of the context can be diagnosed. The following list from Menzel & Schröder [1998b] gives examples of student utterances and diagnoses that are reported by the system in the form of constraint violations at specific positions in the sentence:

- **Partial analysis:** Utterances should be complete main clauses (cf. Section 3.5.4).

E-69	Das	alte	Rathaus.
	The	old	town hall.

  $\longrightarrow$ Missing verb

- **Agreement:** A noun agrees with its article and adjectives with respect to gender, number and case. In general, governors may impose certain restrictions on (morphological) features of their complements (cf. Section 3.4.1).

[5] Depending on the type of error detected, a good feedback probably should not list the errors of the user but instead give positive reinforcement or provide additional exercises.

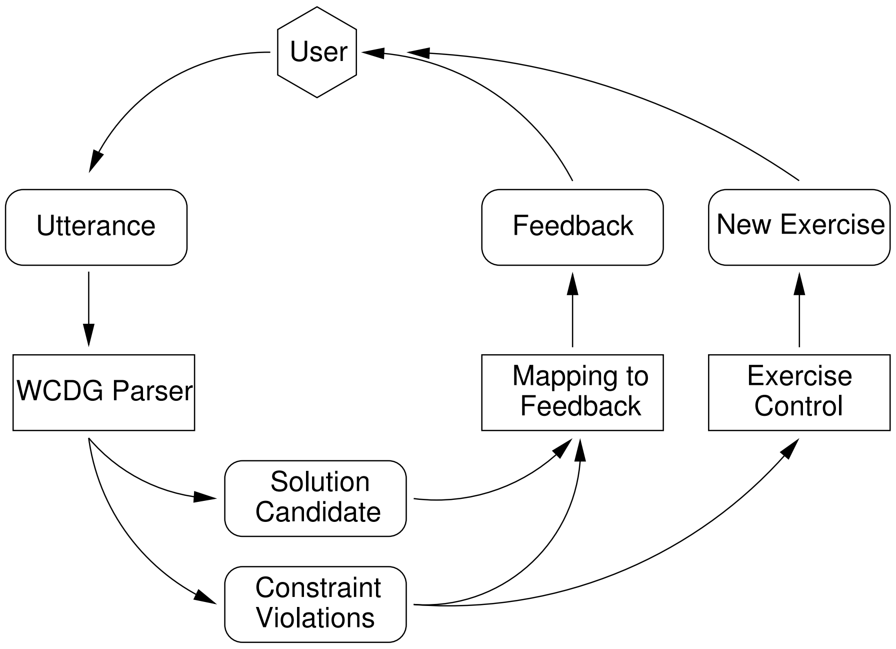
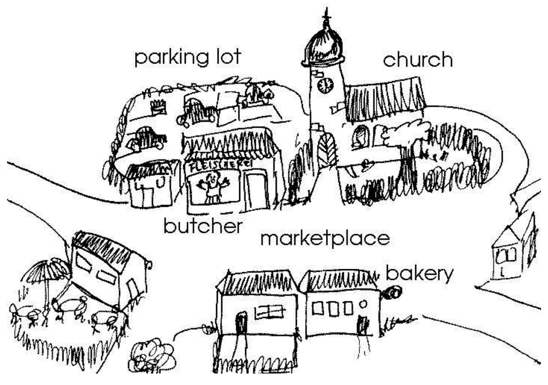


Figure 5.5: Architecture of a CALL system using a WCDG grammar component.



(a) Pictorial embedding

"A man visits a new town. He parks his car in the parking lot. The old church is located at the central marketplace. A butcher's shop is next to the church and a bakery is opposite to the butcher."

(b) Textual embedding

Figure 5.6: The exercise context was made accessible to the student using a picture. Different exercise types might alternatively (or even additionally) require a textual presentation (read to or by the student). In all cases, the relevant contained information must also be encoded in terms of constraints so that the diagnosis component can exploit the 'common background'.

E-70	Die$_{\text{fem}}$	große	Mann$_{\text{masc}}$	schläft.
	The	large	man	sleeps.

$\longrightarrow$ Missing gender agreement for the article 'die'

- **Word order:** Word order rules should be observed; preferences can sometimes be violated and do not always count as errors (cf. Section 3.4.3).

E-71	Der	Mann	die	Stadt	besichtigt.
	The	man	the town	visits.	
	'The man visits the town.'				

$\longrightarrow$ Wrong word order, verb not in second position

- **Auxiliary verb selection:** Verbs determine which auxiliary verb, *haben* (have) or *sein* (be), is used for the perfect form.

E-72	Der	Mann	ist$_{\text{sein}}$	die	Stadt	besichtigt$_{\text{haben}}$.
	The	man	is	the	town	visited.
	'The man has visited the town.'					

$\longrightarrow$ Wrong auxiliary verb form

- **Case frame:** The obligatory case frame of verbs should be filled (cf. Section 3.4.1).

E-73	Der	Mann	besichtigt.
	The	man	visits.

$\longrightarrow$ Missing second argument

- **Sortal restrictions:** The semantic classes of arguments often need to conform with verbal requirements (cf. Section 3.4.6).

E-74	Die	Stadt	schläft.
	The	town	sleeps.

$\longrightarrow$ Violation of sortal restrictions for the first verb argument

- **Contextual restrictions:** The student's utterances should not be in conflict with the given contextual information. Such constraint violations may indicate that the student has comprehension problems reading an introductory text.

E-75	Die	Bäckerei	liegt	neben	der	Fleischerei.
	The	bakery	is	next to	the	butcher.

$\longrightarrow$ Propositional content not supported by the context

The detected faults must, of course, be mapped to an appropriate feedback for the user (cf. Figure 5.5 on the page before). Such mapping may contract multiple errors into one explanation or drop faults altogether because no pedagogically suitable interaction is available.

5.4.3 Related work

Surprisingly, the research communities of CALL and computational linguistics are largely disjoint although it seems natural for an advanced CALL system to include a sophisticated parsing component. This is due to the partially different focus. A CALL engineer has to take additional issues into account, such as a model of the student's current state, pedagogically adequate error explanations, an inspiring user interface and a reasonable and motivating sequence of exercises. On the other hand, computational linguists have failed to come up with a high-precision, general-purpose parser that reliably diagnoses errors. Zock [1996] discusses the relation of CALL and computational linguistics.

The simplest CALL methods still widely in use are based on multiple choice and fill-in-the-blank. Diagnosis only involves a comparison with a stored set of answer keys or pattern matching techniques at most. Two techniques are widely used in advanced parsing components in CALL systems: error rules (generally called 'bug libraries') and constraint retraction. The former are overgenerating grammar rules that model typical cases of students' faults [Weischedel, Voge & James 1978]. Since the robust behavior is not derived from a grammar that models correct language use, all possibly occurring errors have to be anticipated by the grammar writer [Yazdani 1986]. In contrast, constraint retraction [Uszkoreit 1991; Erbach 1993] starts from a correct grammar and softens certain constraints if no complete parse can be found. Often the two techniques are combined, for instance, error rules for word insertions, word deletions and errors of word order and constraint retraction for agreement errors [Schwind 1988; Schwind 1990; Schwind 1995].

A model-based diagnosis [Struss 1992; Davis 1994] that is solely based on descriptions of structure and behavior is difficult to transfer to a diagnosis of errors in natural language utterances since the identification of the internal structure of a sentence is part of the parsing process proper which must be guided by expectations about possible errors. Menzel [1988; 1990] demonstrates the general feasibility of the idea but his approach is limited to a small number of pre-defined structural possibilities. Self [1992] applies the same idea to the general problem of student modeling (cognitive diagnosis).

5.4.4 Summary

The WCDG parsing system has been used as a diagnosis component in a prototypical CALL application. Although no systematic evaluation in the form of a usability study has been conducted, the prototype proves the appropriateness of the approach for language learning exercises, at least in the limited framework that was used. Nevertheless, in order to extend the system to a full-featured CALL system, additional components, such as sophisticated exercise control, student models, tools and utilities for teachers and authors, etc. are required. A particularly helpful module is a correction facility which can generate a corrected version of the student's input. Although generally adequate, the WCDG framework offers only little support for this task. A combination of a WCDG parser and model-based diagnosis [Menzel & Schröder 1998f; Menzel & Schröder 1998d] seems to be a promising approach. The model-based approach not only offers a mechanism of constructing a corrected version of deviant input but additionally extends the range of grammatical faults that can be diagnosed. It needs a structural hypothesis, however, which fortunately can be found by the WCDG parser.[6]

[6]Stockfleth [2000] demonstrates that such a combination is feasible and describes a prototypical implementation.

5.5 Anytime behavior

This section motivates why anytime behavior is important in the context of natural language analysis and investigates whether WCDG parsing obeys the anytime property (cf. Section 4.4.3).

5.5.1 Motivation

An application of natural language processing that interacts with a human user is a kind of real-time system since the pace of the dialogue is largely determined externally, i.e., by the user. Either the system manages to react within a limited amount of time or the user looses his patience and aborts the communication. The more natural such a conversation is supposed to be, the tighter the temporal margin within which the system must act. This is due to the turn-taking behavior of humans who do not pause after the contribution but usually immediately start to formulate an answer. Sometimes it is even appropriate to start the own contribution before the end of the utterance of the partner. However, such a behavior can only be reproduced when parsing occurs incrementally (cf. Section 4.12) and anytime algorithms are employed.

The upcoming subsections present results from WCDG parsing experiments for two kinds of anytime behavior: intrinsic and extrinsic. Anytime behavior as an intrinsic property of the basic solution methods has already been briefly discussed in Chapter 4. An extrinsic anytime property is based on meta-control mechanisms which modify parameters of the basic algorithms so that they show the desired behavior.

5.5.2 Intrinsic anytime behavior

In Section 4.5.1, we realized that the heuristic search methods are generally well-suited for anytime requirements since they immediately start with a first (suboptimal) solution candidate. This has been confirmed with respect to WCDG parsing for the methods of Frobbing and Gls in Section 4.11 in so far as both are able to find a first approximate solution within a fraction of the time that they need to terminate.

Also, following Freuder & Wallace [1992], we conjectured that branch-and-bound should be able to deliver an approximation early and to improve on this intermediate result when the score of a newly encountered candidate falls below the current bound. The fact that a first solution without violated hard constraints can be identified relatively soon (Figure 4.23 on page 111) is a first evidence that this property can actually be found in problems of natural language parsing.

The rest of this section tries to support these claims experimentally.

Tree search

Figure 5.7 on the next page shows how the accuracy of intermediate solutions develops depending on processing time when using the systematic tree search method (cf. Section 4.8).[7]

[7]Wallace [1996] also suggests the distance of an intermediate solution candidate to the optimal solution as a measure for quality in anytime curves.

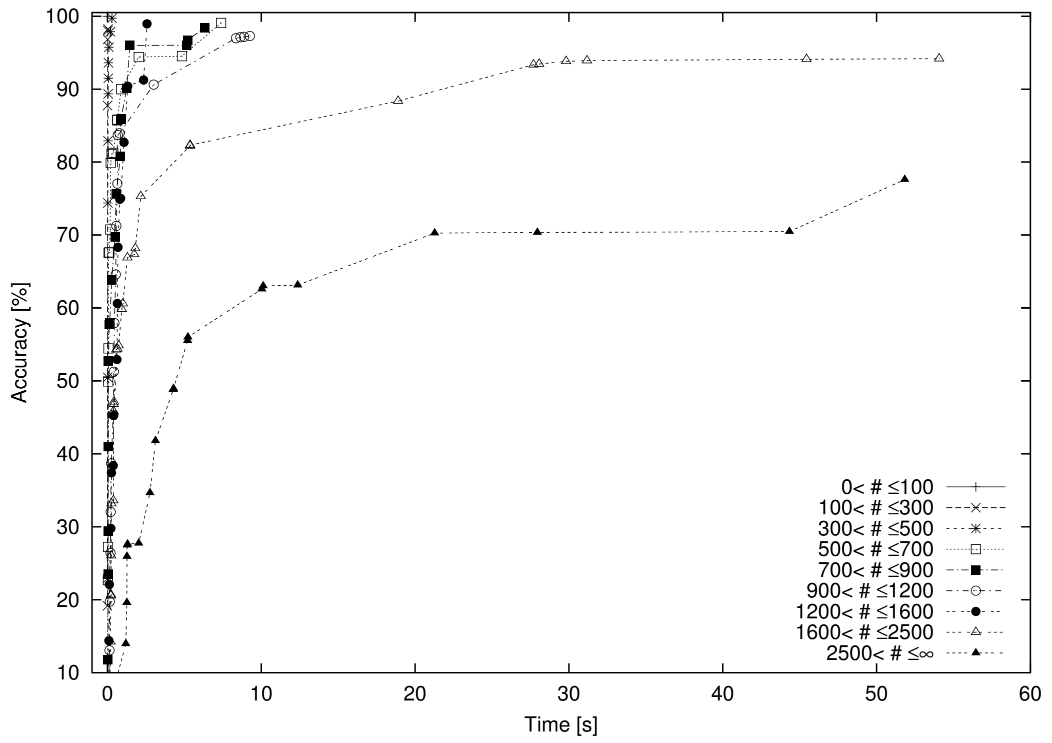


Figure 5.7: Accuracy (as percentage of correct links) of intermediate solutions depending on processing time for the solution method Netsearch (cf. Section 4.8). Problems of varying difficulty (as identified by the number of constraint values, cf. Section 4.11) have been distinguished.

It can in fact be confirmed that approximate solutions start to become available after a short period of time. Note however that an accuracy of some 50% after five seconds for the class of the hardest problems can mean quite different things: Either half of the problem instances have been solved correctly or all problem instances have solutions which are only 50% correct. Only the latter case can be counted as anytime behavior because only then a quality improvement over time can be assumed for individual problem instances.

Time [s]	0.05	0.1	0.5	1	2	5	10	20	30	52
Tree Search [%]	62.5	68.8	81.3	87.9	91.5	94.6	96.4	97.3	98.2	98.7
Heuristic Search [%]	90.6	98.7	100	100	100	100	100	100	100	100

Table 5.2: Fraction of problems with soft solutions after a certain period of time.

Table 5.2 lists the percentage of problems for which a first solution is available after a certain period of time. Although a number of over 94.6% after five seconds may seem high at first glance, it also means that no solution has been found for the remaining 5.4%. It must be assumed that no real anytime behavior can be attributed to the tree search because no solution at all can be found for some problems even after a long time of computing.

To summarize: It has to be concluded that the predicted suitability of branch-and-bound for anytime behavior [Freuder & Wallace 1992] can only partially be confirmed for the problem of WCDG parsing. Two reasons can be identified:

1. The Netsearch procedure immediately terminates a branch in the search space when a hard constraint is violated. Hence, it can only deliver intermediate results that already satisfy all hard constraints. This is the main reason for the observed delay in producing a first solution.

2. Netsearch employs an optimized agenda strategy which is a mixture between depth-first and best-first (cf. Section 4.3.2). Different variants of agenda strategies are imaginable and the more it resembles depth-first, the earlier one can expect a first solution.

Heuristic search

We now turn to heuristic search variants and focus on the procedure Frobbing (cf. Section 4.9.1) which uses the min-conflicts heuristics and a tabu list.

Figure 5.8 on the next page shows how the accuracy of the intermediate solutions develops depending on processing time when using the heuristic search. Note that the accuracy range in Figure 5.8 on the facing page differs from the one in Figure 5.7 on the page before.

It can be clearly seen that almost immediately after the start of the procedure a first solution is available that already features a considerably high accuracy of over 80%. For smaller problems, the maximum accuracy is reached after less than 10 seconds while it seems to still improve after 60 seconds for the largest problems. Considering also the information in Table 5.2 on the preceding page, it can be concluded that the parsing procedure sustains the intrinsic property of anytime behavior typical for heuristic search methods.

It should be noted however, that there is still a lot of variance in the quality of a solution after a specific period of processing time. The performance profiles for WCDG parsing are instance-specific and a probabilistic performance profile can only be given for a corpus of utterances (cf. Section 4.4.3).

Figure 5.9 on page 148 gives examples of the development of the solution quality over time for individual parsing problems.

The problems in Figures 5.9a on page 148 through 5.9c on page 148 have been selected so that they have approximately the same number of constraint values and are, therefore, of comparable difficulty. Accuracy is given depending both on absolute time in a constant range of 10 seconds and on relative time, i.e., relative to the time needed until the particular parsing process terminates. For comparison, the candidate score $\phi^{-}_{\otimes}(\mathrm{SC})$ is also shown. Note that a solution procedure has access only to the score but not the accuracy of intermediate solutions.

The accuracies for sentence n002k015-1 in Figure 5.9a on page 148 and sentence n004k023-1 in Figure 5.9b on page 148 both show a similar run of the curves over relative time. However, the absolute runtime is quite different: The second example needs only one fifth of the time of the first example to find the solution although they seem to be of similar difficulty judging from the number of constraint values.

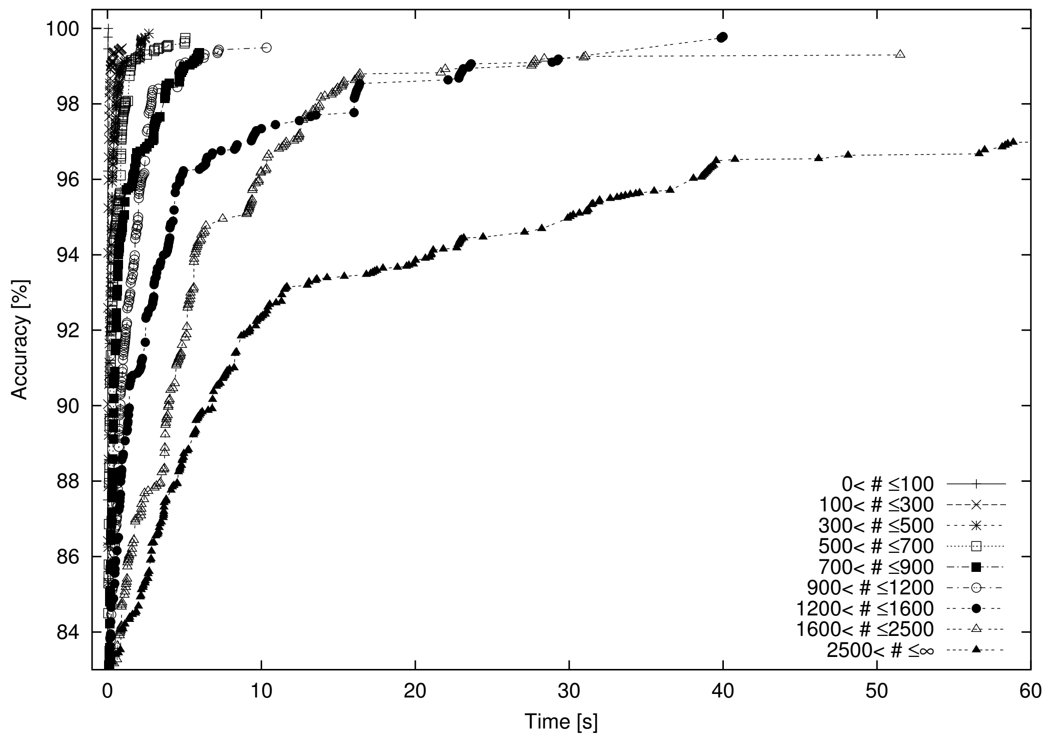
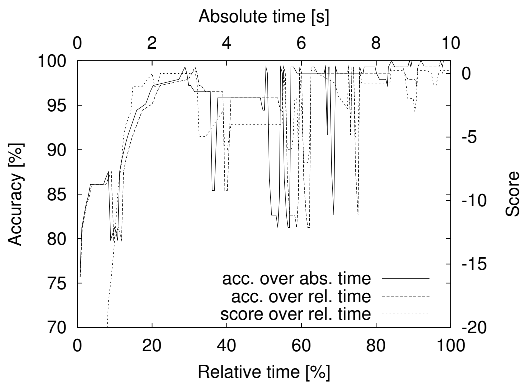


Figure 5.8: Accuracy (as percentage of correct links) of intermediate solutions depending on processing time for the solution method Frobbing (cf. Section 4.9.1). Problems classes of varying difficulty (as identified by the number of constraint values, cf. Section 4.11) have been distinguished.

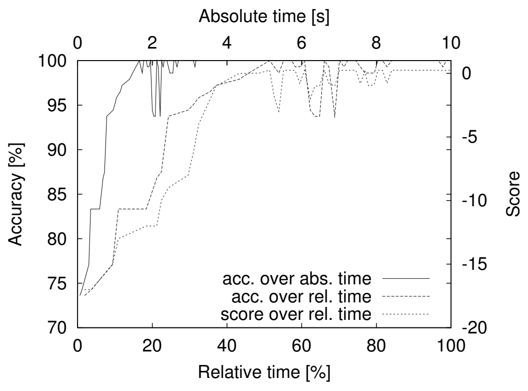
Figure 5.9c on the following page shows that the quality of intermediate solutions can be quite irregular over time. That does of course not mean that the quality of a solution that is returned later must be worse than an earlier one since the search procedure can always remember good candidates and deliver the best intermediate solution so far but it has to use the score as the only criterion since the accuracy is not available to the search method.

Finally, Figure 5.9d on the next page shows an unrelated phenomenon that can easily be observed in some performance profiles: The solution method Frobbing obviously finds the same intermediate candidates multiple times as can be seen from the regular run of the curves to the right of the figure. Although a tabu list (cf. Section 4.9.1) is used to avoid short cycles, this cannot guarantee the procedure not to return to previously visited candidates, a property that is typical for heuristic search variants.

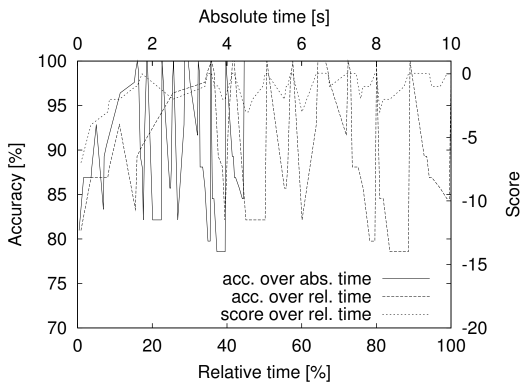
To summarize: The solution method Frobbing shows a runtime characteristic that is typical for weak anytime algorithms. Approximate solutions are available almost immediately. Although probabilistic performance profiles (based on a given corpus) can be derived, the profiles of individual problems vary to a large degree.



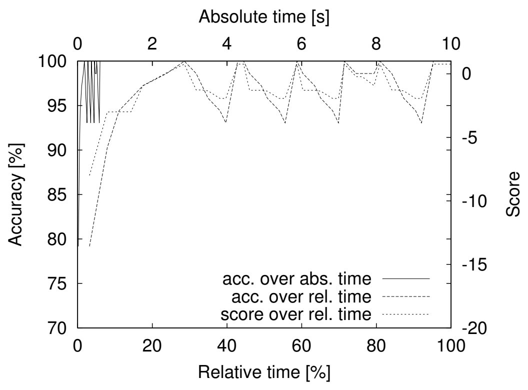
(a) individual performance profile (n002k015-1, 1,012 values)



(b) differing performance profile (n004k023-1, 1,129 values)



(c) irregular performance profile (n005k016-2, 1,021 values)



(d) performance profile showing cycles (n002k026-1, 290 values)

Figure 5.9: Individual performance profiles.

5.5.3 Extrinsic anytime behavior

A system with an extrinsic anytime property is characterized by a two-stage architecture. A meta-control mechanism supervises the underlying algorithm and controls its behavior, e.g., by adjusting parameters when some modification of the parsing process seems appropriate in order to achieve anytime behavior. Parameters can either belong to the basic algorithm itself or to the grammar employed.[8]

Intuitively one would think that the meta-control mechanism requires some leeway in the processing characteristics so that it can apply some kind of pressure to the underlying system to

[8]Given this admittedly vague definition, even the heuristic search variant guided local search (cf. Section 4.9.2), for instance, might be characterized as a two-stage architecture because a mechanism modifies the augmented assessment function and controls the basic local search. Actually, guided local search is often called a meta-heuristics [Voudouris 1998]. However, we will consider here only systems where the meta-control modifies the underlying process in order to achieve a better anytime behavior.

accelerate the computation. For WCDG parsing, we are willing to sacrifice parts of the robustness that the system shows in normal operation mode (cf. Section 5.3) in order to improve the efficiency of the system under temporal pressure. In this case, only the standard sentences will be parsed correctly; deviating utterances will not be analyzed accurately (or not at all) any longer. This behavior is consistent with human language understanding under difficult conditions, such as reading in a hurry or comprehension difficulties when listening to someone speaking in a foreign language.

A series of experiments were run to verify whether the WCDG parsing system is suitable for such a trade-off where a speedup is traded against decreased robustness. An actual meta-control mechanism was not built because a component that dynamically adjusts the parameters of the system only makes sense when temporal pressure can be observed. This however is only feasible (or natural) in natural language parsing when the analysis works incrementally and the distance between the position of the currently analyzed word and the horizon of the language input becomes too large. Incremental analysis with WCDG, however, is still in its infancy (cf. Section 4.12).

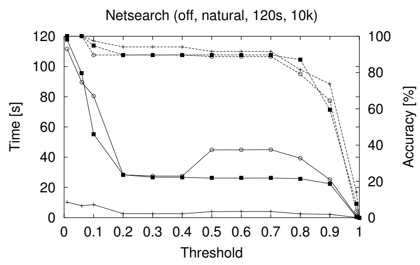
For the systematic tree search procedure Netsearch two means of applying pressure to the system have been tried:

- **Tighter grammar:** All constraints with a weight smaller than the parameter p are made hard, i.e., their weight is re-adjusted to zero. This has two consequences:
  1. All sentences that do not violate any of the constraints c that encode knowledge about severe violations ($\phi(c) < p$), can be analyzed faster than before because deviating local alternatives can be discarded early during the analysis.
  2. The system parses deviating utterances that violate one or more of the tightened constraints into a wrong structure or not at all because the correct analysis has been removed by the now hard constraint.
- **Pre-setting the bound:** A pre-set bound value allows branch-and-bound search methods to prune parts of the search space early. In particular those problems for which it takes a long time to find a first solution and therefore a first bound, can benefit from such a step.

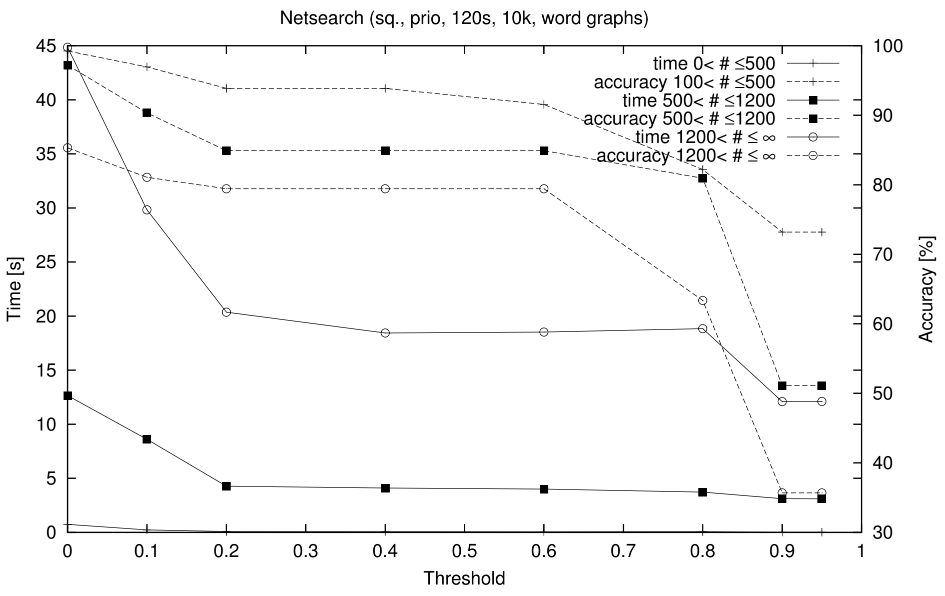
Figure 5.10 on the following page relates a parameter p which represents either the threshold for grammar tightening or the pre-set bound to the achieved accuracy and the computation time needed.

First of all, the expected behavior that the quality decreases and the efficiency grows with increased parameter can be observed for both variants of applying pressure.[9] But only the

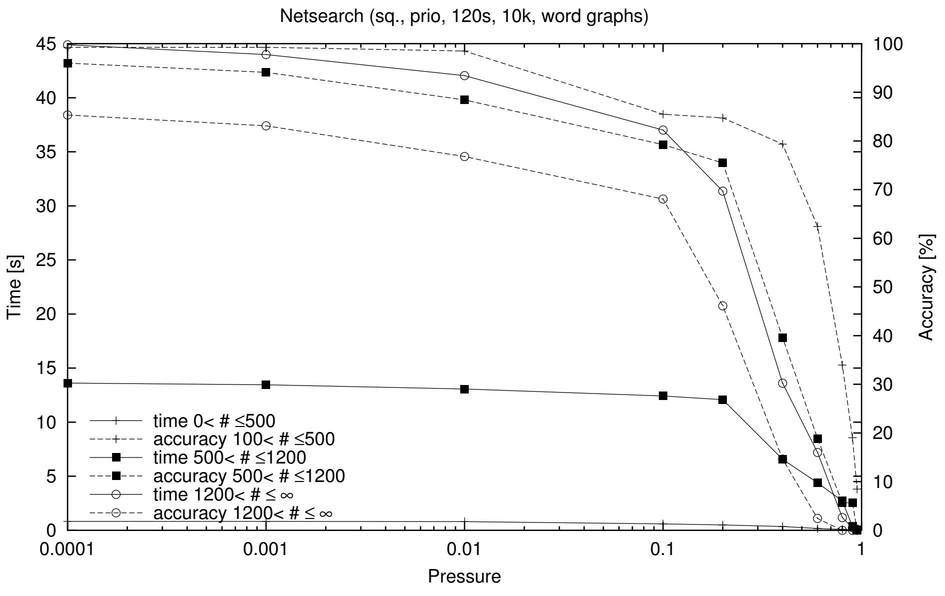
[9]



Note that the results reported in this section largely depend on the grammar employed. The figure to the left, for instance, is similar to Figure 5.10a on the next page but has been produced using an old version of the Stellingen grammar and a subset of the utterances. Although the principal observations are the same, the effects are even more pronounced. The main reason for these differences is the extended corpus of utterances.



(a) External time pressure through tightened grammar.



(b) External time pressure through pre-set bound.

Figure 5.10: Systematic tree search procedure under external time pressure.

grammar threshold (cf. Figure 5.10a) shows the favorable behavior of the accuracy degrading to a lesser degree and later than the efficiency improves. The run of the curves as a hysteresis guarantees that a considerable speedup can be achieved at the cost of negligible losses in accuracy. The pre-set bound in Figure 5.10b does not show such a behavior.

For the method of Frobbing, an internal parameter *p* determines how long detours may get during a local search. Figure 5.11 gives the achieved accuracy as well as the time needed depending on the applied pressure.

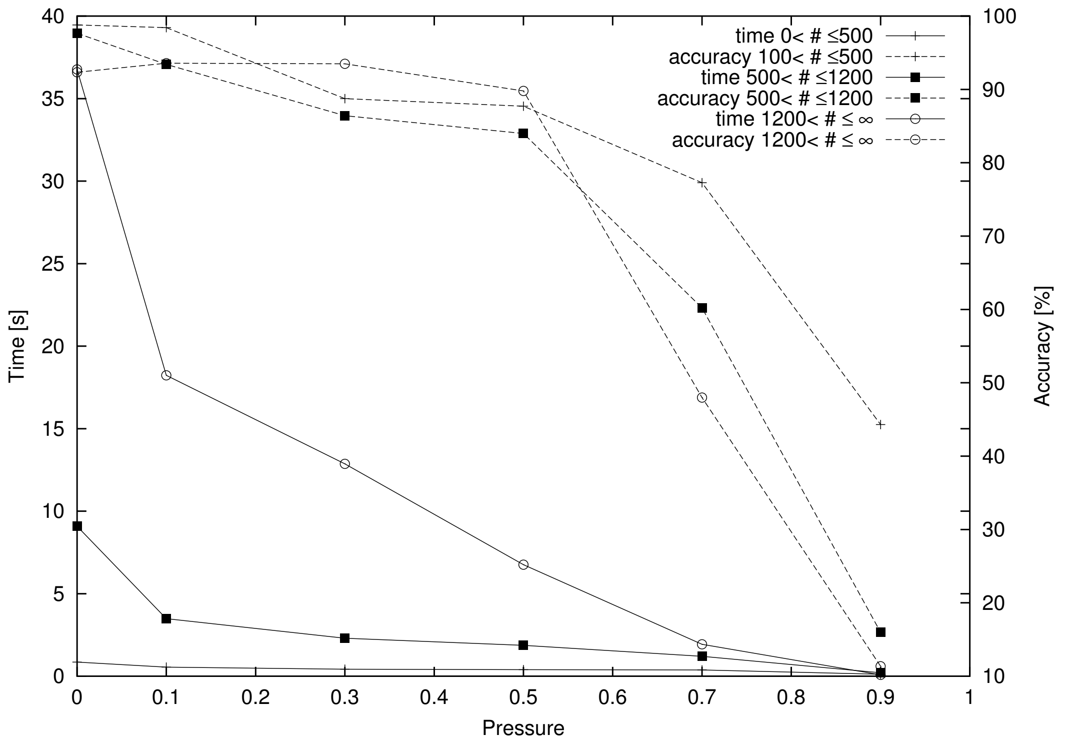


Figure 5.11: Heuristic search method with tabu list under external time pressure.

Obviously, a moderately increased pressure leads to a minor decrease in accuracy and almost no speedup for small problems. For the interesting cases of larger problems, however, a huge speedup and again almost no decrease in accuracy can be observed. This behavior is welcome since smaller problems are solved quickly anyway while large problems benefit most from the applied pressure. The outline of the curves again resembles a hysteresis. Since a suitable parameter is available that a meta-control can use to fine-tune the temporal behavior of the algorithm, the property of extrinsic anytime behavior can be confirmed for the heuristic search method when applied to the problem of WCDG parsing.

5.5.4 Related work

Various classification schemes for anytime algorithms have been proposed in the literature. Wahlster & Tack [1998] distinguish two types of resource-sensitive algorithms. Resource-adapted means that the reasoning about the strategy takes place during the design phase; the required resources are fixed at runtime. In contrast a variable amount of resources is necessary for resource-adaptive and resource-adapting processes. The former embarks on a fixed processing strategy while the latter may change its processing regime during runtime. The classes of resource-adaptive and resource-adapting procedures can be roughly mapped to the interruptible and contract anytime algorithms (as defined by Russell & Zilberstein [1991]) respectively. However, contract algorithms need to know ahead of time how much time is available while resource-adapting methods can change their parameter during runtime. We add the terms of intrinsic and extrinsic anytime algorithms which again roughly correspond to the distinction between interruptible and contract algorithms but emphasize the separation between basic algorithm and meta-reasoning device. We call an algorithm an intrinsic anytime algorithm when the basic method can be interrupted at any time and still yields a reasonable result. In contrast, an extrinsic anytime algorithm offers parameters that enable an external component to control the trade-off between speed and quality. Parameters may have to be adjusted before the start of the algorithm or can be adapted during runtime. Figure 5.12 summarizes these classification schemes.

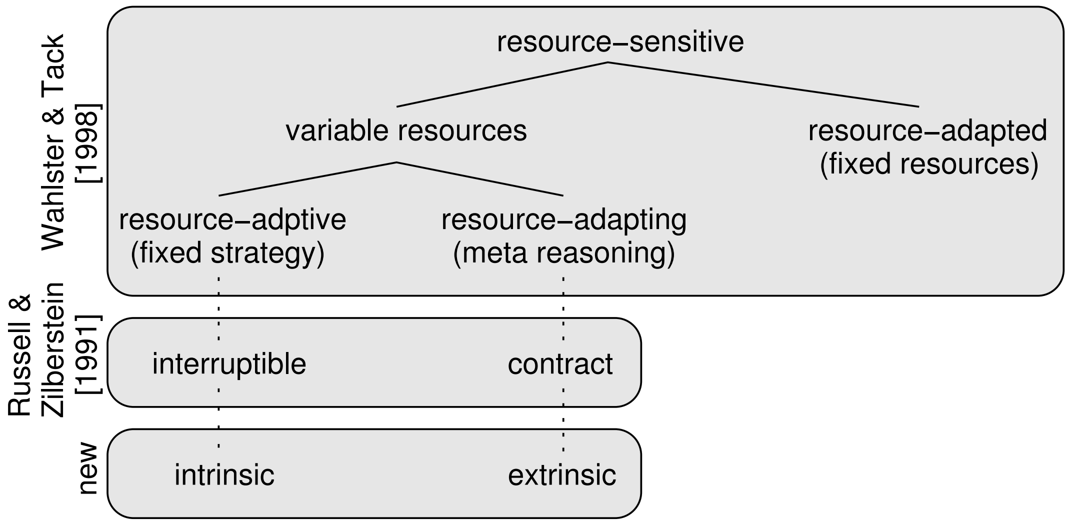

Wahlster & Tack [1998]
resource−sensitive
variable resources
resource−adapted (fixed resources)
resource−adptive (fixed strategy)
resource−adapting (meta reasoning)
Russell & Zilberstein [1991]
interruptible
contract
new
intrinsic
extrinsic

Figure 5.12: Classification of anytime algorithms.

Menzel [1994] introduced the distinction between weak anytime algorithms whose performance profiles are instance-specific and strong anytime algorithms with generally valid performance profiles.

In the context of natural language processing (NLP), Wahlster [1993] proposes anytime algorithms for NLP modules in the speech-to-speech translation system Verbmobil.[10] Relatively few references to the anytime property can be found for natural language parsing. Menzel [1994] suggests anytime algorithms in the context of CDG for the first time. However, he focuses on the view of CDG parsing as successive ambiguity reduction which facilitates an introspection of the remaining work load and therefore a goal-oriented scheduling of delayed constraint applications. Unfortunately, the corresponding techniques have proven not to be directly applicable for WCDG parsing. Görz & Kessler [1994a] investigate the potential of chart parsing techniques as anytime algorithms.[11] However, their view is quite narrow; they only focus on possible points of interruption for a chart parser: after application of the fundamental rule, after a complete unification etc. One particular problem with their approach is that a chart parser obviously needs a

[10]As far as we know the ideas were first mentioned by Wahlster [1992] at his invited lecture at COLING-92.

[11]Görz & Kessler [1994b] also give details about an underlying protocol for interfaces to anytime algorithms.

hardly predictable amount of time before a first solution can be returned. Ehrlich & Hanrieder [1996, p. 29] claim that their island chart parser obeys the anytime property simply because it acknowledges a timeout. Their notion of anytime algorithms is of course only a superficial one since it is in no way guaranteed (or made plausible) that the (expected) quality of the procedure actually degrades gracefully under tightened timeout values.

The anytime property has been more widely investigated in the area of dialogue planning and natural language generation. The goal of the READY project [Wahlster et al. 1995] is to model the cognitive resources of the human dialogue partner. The cognitive capacity of humans during a natural language dialogue is modeled by a temporal Bayes network [Schäfer & Weyrath 1996]. Weis [1997] presents a framework for the evaluation of action plans and utterances which verbalize these plans. It is based on hierarchical planning using plan operators and can be used as a basis for resource-adaptive action and dialogue planning. The system is also presented from the point of view of user interface design [Jameson et al. 1999] with examples from a car repair scenario in which a naïve driver is to be instructed over a mobile phone [Weyrath 1997]. Wahlster et al. [1998] describe the system REAL which answers where-questions under time pressure, i.e., it must first locate the object under consideration and then verbalize the location in a suitable way. The system is time sensitive in two different ways: Firstly, it must take the resource limitation of the user and the situation into account, e.g., the proximity of the next crucial situation for a traffic information system in a fast-moving car, and secondly, the algorithms themselves should have the anytime property. An alternative approach to anytime dialogue planning is presented by Carletta [1992] who bases her computational model on extensive corpus analyses. Carletta, Caley & Isard [1993] try to simulate hesitations and self-repairs in human dialogues through anytime techniques. Donaldson & Cohen [1997] develop an anytime model for turn-taking in discourse. Interestingly, they use similar techniques (dynamic constraint satisfaction problems, local search) in their work.

5.5.5 Summary

The potential of WCDG parsing for the anytime property has been experimentally investigated. Two types of anytime behavior are distinguished.

- The *intrinsic anytime behavior* refers to the capability of the basic algorithm to suggest an approximate solution early and improve on that intermediate candidate if more time is available. For both search variants, tree search and heuristic search, this characteristic can be confirmed when a corpus as a whole is considered. Tree search, however, fails to provide approximate solutions soon for some very large problems and can therefore only be partially classified as an anytime algorithm. Heuristic search, in contrast, improves first candidates from the very beginning. Both algorithmic types show great variances over different problem instances so that they have to be classified as weak anytime algorithms.
- The *extrinsic anytime behavior* refers to the availability of parameters in the basic algorithm that allows a meta-level method to control the trade-off between speed of computation and quality of results. Suitable parameters have been identified in the form of a threshold for grammar weights in the case of tree search and in the form of a limitation of the length of detours in the case of heuristic search. In contrast, a pre-set bound for the branch-and-bound method does not show the favorable behavior of large efficiency improvements being achieved by sacrificing only a small amount of accuracy.

5.6 Interfacing a WCDG with a speech recognizer

5.6.1 Motivation

Analyzing spoken language instead of written language is a challenging task because the acoustic ambiguity, i.e., the uncertainty which words have actually been uttered, is added to the already ambitious process of finding the underlying structure of a sequence of words.

Figure 5.13 on the facing page shows the input and output of a typical speech recognizer in which a digitized waveform of the speech segment is analyzed into a weighted directed graph.[12] This so-called word hypotheses graph consists of ordered vertices and edges which are labeled with words and weighted with acoustic scores[13]. Overlapping edges represent alternative interpretations of the same speech segment. The cheapest path through the graph is the acoustically preferred sentence hypothesis. Word graphs are widely used because they allow for a compact representation of recognition ambiguities. Section 2.3.1 describes word graphs from the point of view of WCDGs.

This section suggests an architecture for the integration of acoustic scores into the WCDG parsing system and describes results from a number of experiments in which different instances of such a combination are explored. The following questions are of particular interest:

- Is it at all tractable that a WCDG parser processes realistic word graphs?
- Can acoustic scores help the grammatical system to select the correct sentence structure?
- Can a WCDG parser successfully re-rank the sentence hypotheses of a recognizer?

5.6.2 Combination of acoustic scores and constraint weights

Although the WCDG system uses word hypotheses graphs as standard input and can therefore be classified as a sequentially coupled system, we have so far ignored the question of how to combine acoustic scores in a word graph and constraint weights from a (manual) WCDG.

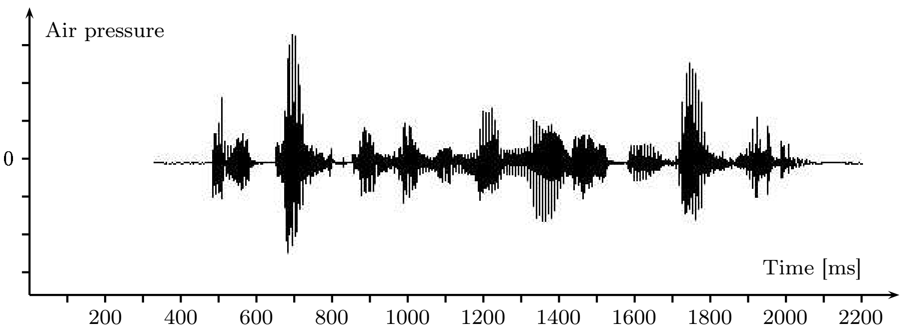
[12]State-of-the-art speech recognizers convert the digitized waveforms first into multi-dimensional feature vectors over a small window of samples and then compute the more probable paths through a large *hidden Markov model* that has words as states and outputs the observed feature vectors. The details of speech recognition systems are beyond the scope of this thesis [Rabiner 1990; O'Shaughnessy 1990; Eppinger & Herter 1993; Schukat-Talamazzini 1995].

[13]Typical speech recognizers nowadays employ two statistical models based on acoustic probabilities and sentence prior probabilities respectively. This is feasible because the probability $P(w|X)$ (where w is a sequence of words and X is a sequence of acoustic observations) that is to be maximized [Schukat-Talamazzini 1995, for instance] can be re-written as follows (using the Bayes rule):

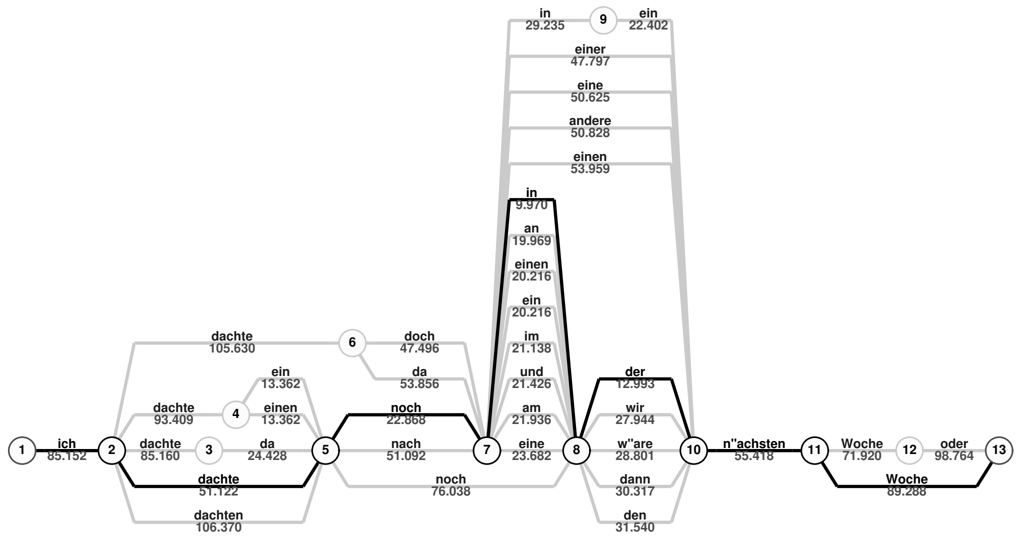
$$w^* = \underset{w}{\arg\max}\, P(w|X) = \underset{w}{\arg\max} \frac{P(w) \cdot P(X|w)}{P(X)} = \underset{w}{\arg\max} \quad \underbrace{P(w)}_{\substack{\text{language}\\ \text{model}}} \quad \cdot \quad \underbrace{P(X|w)}_{\substack{\text{acoustic}\\ \text{model}}}$$

The acoustic model contains information about the pronunciation of the basic unit (often phones) while the language model takes the word sequence information into account. The latter is often realized through an n-gram model, although it is of course generally agreed that an n-gram is too weak a model of natural language.

Hence, it is not quite precise to talk about an acoustic score since partially grammatical evidence from the language model is included. To ease the presentation, we nevertheless use the term 'acoustic score' for the assessment of a hypothesis edge originating from a speech recognizer. Although possible in general, the speech recognizer used for the experiments in this section does not offer to entirely switch off the language model.



(a) Input: Waveform (air pressure vs. time) of a speech signal



(b) Output: Word hypotheses graph with the acoustically preferred path marked by black edges

Figure 5.13: Word graphs as an interface to a speech recognizer.

In principle, one should be able to multiply the acoustic scores directly with the CDG weights since ideally the former are the combined transition and output probabilities of the Markov model. There are two factors against this:

- Speech recognition systems often output scores that are no longer real probabilities. Due to different normalizations and simplifications, the scores should merely be used to rank the competing alternatives.

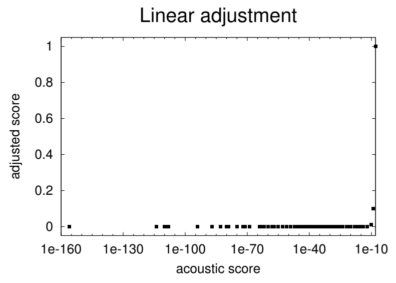
- Acoustic scores are by several magnitudes smaller than usual constraint weights.[14] This means that their scale usually does not fit the scale of a constraint system. While the difference between, say, scores 10^{-18} and 10^{-39} (from the word graph in Figure 5.13b on the page before) might be small enough for a speech recognizer to include both hypotheses for the same speech segment in a relatively small word graph, it is highly unlikely that a grammatical score resulting from constraint violations reaches the same numerical range.

We therefore tried different methods for the integration of acoustic scores.

Let b be the best and w the worst score in a word graph. An original acoustic score p is adjusted by applying a function $a : \mathbb{R} \mapsto [0, 1]$.

- **Linear:** Normalize the emitted scores in the range $[0, 1]$.

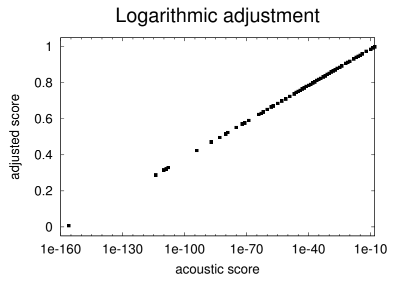
$$a(p) = \frac{p - w}{b - w} \approx \frac{p}{b}$$



The graph to the right shows the transformed scores of the example word graph. The smaller acoustic scores are *much* smaller than the largest one and as a consequence the majority of the values is very close to zero.

- **Logarithmic:** Normalize the logarithms of the scores in the range $[0, 1]$.

$$a(p) = \frac{\log(p/w)}{\log(b/w)}$$



This function distributes the acoustic scores much more evenly over the interval $[0, 1]$. Only the worst scores are close to zero. This kind of adjustment is used in all subsequent experiments.

The normalized word graph score is made available to the constraint system through the function `acoustics()` which has been added to the constraint language (cf. Section 2.5.3). Constraint C-41 is a constraint template (with parameter α) that rescales the acoustic score so that it always falls into the interval $[\alpha, 1.0]$. A small value for α leads to a greater influence of the acoustic score while a larger value α makes the inevitable constraint violations caused by the acoustics constraint less decisive. A value of one renders the constraint ineffective because the score of the constraint becomes one in any case then.

C-41 `{X} : Acoustics : [`α`+(1.0 − `α`)*acoustics(X@id)] :`
`X.level=SYNTAX -> false;`
Integrate the adjusted acoustic score.

By using the truth constant `false` in the constraint body and restricting the application to a single level, it is guaranteed that every (adjusted) acoustic score is integrated exactly once.

[14]Often the acoustic scores 'behave' like negative logarithmic probabilities [Nöth & Plannerer 1993] to prevent arithmetic underflow in a computer program.

5.6.3 Reduction of word graph density

It is intuitively clear that (ceteris paribus) larger word graphs represent harder problem instances because the additional recognition uncertainty has to be compensated by the parser. A common measure for the size of word graphs is the *word graph density* which is defined as the ratio between the number of hypotheses edges in the word graph and the number of words in the reference sentence. In general, a word graph with a higher density contains more alternative word sequences and thus a greater recognition uncertainty than one with a low density. Conversely, the probability that a word graph contains the reference utterance decreases when the density is low.[15]

In order to be able to systematically compare the efficiency and accuracy of the parser under varied word graph sizes, we created word graphs for a small corpus of 25 test sentences that were taken from the Stellingen corpus (cf. Section 5.2.1) and are thus covered by the grammar. The Verbmobil [Wahlster 2000] recognizer for German [Waibel et al. 2000] was configured to output word graphs with an average density of 10, and the manually segmented utterances were read rather than spontaneously spoken. The resulting word graphs have an actual mean density of 8.5 (sd 4.9). It is quite likely (although unknown to the author, cf. Section 5.6.5) that the used sentences (the word sequence, not the waveform) are contained in the training material of the speech recognizer since both are taken from the standard Verbmobil corpus. This, however, does not constitute a serious problem here because we are only interested in the impact of using word graphs (of different size) on our parser, not in a general evaluation of the speech recognizer or the overall system.

For each of the initial word graphs a sequence of smaller and smaller word graphs is created where the less dense word graphs are subgraphs of the larger ones. Starting from the maximum word graphs, the density has been successively reduced by removing word hypotheses with a low acoustic score. The employed algorithm (cf. Figure 5.14 on the next page) uses the following utility functions on word graphs G (cf. Definition 2.3 on page 22): $f : G \mapsto \mathbb{R}$ (forward score), $b : G \mapsto \mathbb{R}$ (backward score) and $p : G \mapsto \mathbb{R}$ (best path score).[16]

$$f(\langle a,z,w,s\rangle) = \begin{cases} 0.0 & \text{iff} \quad z = \max_G \\ \min\limits_{e'=\langle z,z',w',s'\rangle} s' + f(e') & \text{else} \end{cases}$$

$$b(\langle a,z,w,s\rangle) = \begin{cases} 0.0 & \text{iff} \quad a = \min_G \\ \min\limits_{e'=\langle a',a,w',s'\rangle} s' + b(e') & \text{else} \end{cases}$$

$$p(\langle a,z,w,s\rangle) = b(\langle a,z,w,s\rangle) + s + f(\langle a,z,w,s\rangle)$$

The forward score $f(e)$ is the accumulated score of the cheapest path, i.e., minimal with regard to the acoustic weight, starting at the end node of edge e and ending at the end of the word graph. Analogously, the backward score $b(e)$ is the accumulated score of the cheapest path from

[15] Amtrup [1999, p. 38] suggests a different measure for the size of a word graph that is based on the number of paths in the word graph. In general, the two measures correlate to quite some degree.

[16] Sixtus & Ortmanns [1999] use similar definitions.

the start of the word graph to the start node of the edge and the best path score $p(e)$ is the accumulated score of the minimal path that goes through edge e. The idea behind this definition is that the absolute weight of an edge is rather inadequate for assessing its quality but the best path score is better suited since it incorporates the topological context of the edge.

```
1 procedure ReduceWordGraphDensity(G, R, M)
2                        ; G WORD GRAPH, R REFERENCE, M MAXIMUM DENSITY
3 while $|G|/|R| > M$ do                              ; SMALL ENOUGH
4   set $X := \{e \in G \mid p(e) = \min_{e' \in G} p(e')\}$   ; EDGES TO BE REMOVED
5   set $G := G \backslash X$
6   remove dangling edges from $G$
7 done
```

Figure 5.14: Reduction of word graph density in pseudo code.

Note that it is not always possible for the algorithm to return a word graph with the exact requested density because the removal of one hypothesis may trigger the deletion of additional edges that are not members of a complete path anymore ('loose ends' in the graph). This post-processing is not included in Figure 5.14. Figure 5.15 on the facing page shows a selection of reduced word graphs that are derived from the word graph in Figure 5.13b on page 155. The word graph in Figure 5.15a on the facing page, for instance, has a density smaller than the requested value of four due to the optimization described above.

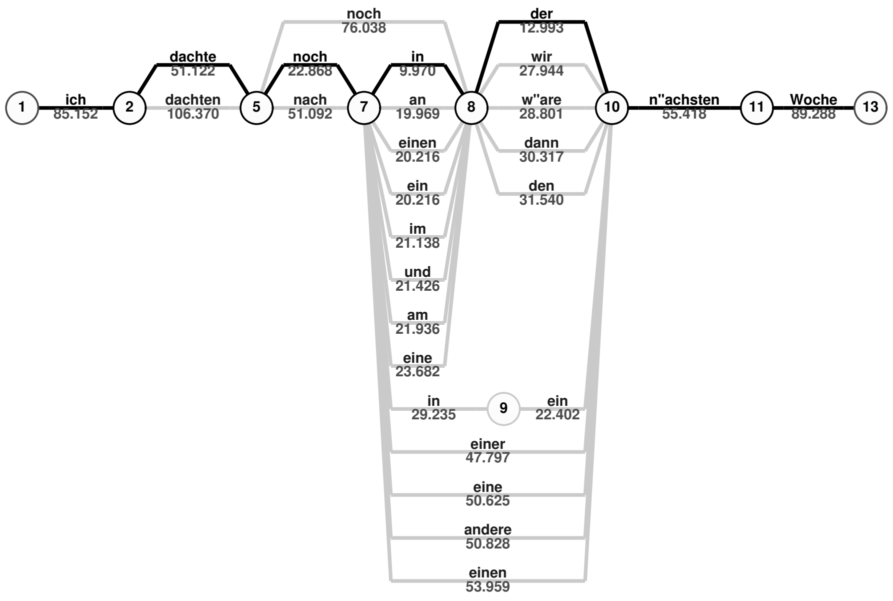
Of the 25 word graphs with minimal density of (up to) one, 64% already contain the reference utterance, i.e., the sentence that was read to the speech recognizer. This figure increases to 80% for densities of up to five where it reaches its maximum; conversely, 20% of the utterances are not contained in any of the corresponding word graphs. The relatively high sentence accuracy means that the recognizer would return the correct sentence in 64% of the cases when asked for the best sentence hypothesis (i.e., a linear word graph). It should be noted that even an ideal parser can only improve the sentence accuracy from 64% to 80% at best unless it adds new arcs to the word graph. This relatively small margin means that we should not expect too big an improvement with respect to sentence accuracy from an integration of a parser, i.e., it is difficult for a parser to correct recognition errors.

5.6.4 Impact of recognition uncertainty on parsing

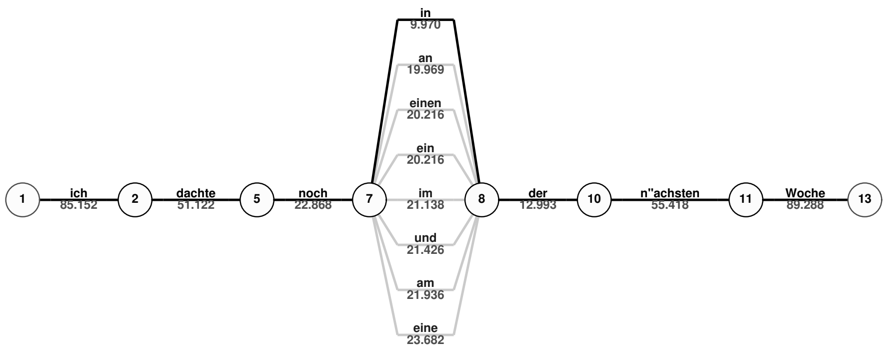
This section describes some experimental findings regarding efficiency and accuracy using the word graphs with increasing density.

The move from smaller word graphs to larger ones has two contrary effects with respect to parsing accuracy.

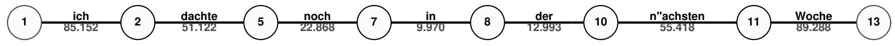
- On the one hand, larger word graphs contain more alternatives and are therefore more difficult to analyze. Multiple word sequences that differ considerably from the reference sentence may be grammatical and some may even be preferred by the grammar. This tendency leads to a decreased accuracy.



(a) Density 3.857143 (27/7, requested ≤ 4)



(b) Density 2.0 (14/7, requested ≤ 2)



(c) Density 1.0 (7/7, requested $= 1$)

Figure 5.15: Word graphs of different densities. Larger word graphs contain more acoustic alternatives.

- On the other hand, the more alternative paths in a word graph, the higher the probability that the correct one is included (which is the motivation for using a word graph with ambiguities in the first place).

Table 5.3 on the next page gives the f-measure values for different acoustics integration schemes, different word graph densities and two solution methods.

Acoustics		$\alpha = 0.1$				$\alpha = 0.9$				$\alpha = 1.0$ (no integration)			
Density		all	1	5	max.	all	1	5	max.	all	1	5	max.
# Problems		182	25	17	25	182	25	17	25	182	25	17	25
# TS Sol.		104	21	8	13	128	24	10	16	128	24	10	16
# HS Sol.		182	25	17	25	182	25	17	25	182	25	17	25
Time TS [s]		70	29	104	50	77	10	91	90	77	9	82	86
Time HS [s]		160	42	199	177	116	19	132	138	114	12	121	146
TS Sol.	**SYN**	84.7	92.7	77.4	77.8	76.6	91.4	59.1	76.8	73.9	91.4	54.9	75.9
	SEM	88.5	95.1	85.6	83.8	81.4	92.8	73.0	77.8	78.2	92.8	68.5	76.6
	TOT	89.0	94.8	86.0	85.6	82.7	92.5	74.4	80.9	79.9	92.5	69.9	79.7
TS All	**SYN**	48.4	77.8	36.4	40.5	53.8	87.8	34.8	49.1	52.0	87.8	32.3	48.6
	SEM	50.6	79.9	40.3	43.6	57.3	89.1	43.0	49.8	55.0	89.1	40.3	49.0
	TOT	50.8	79.6	40.5	44.5	58.2	88.8	43.8	51.8	56.2	88.8	41.1	51.0
HS	**SYN**	66.3	90.8	62.9	56.7	68.6	89.6	61.8	66.9	66.3	89.9	55.5	63.7
	SEM	77.5	91.8	79.2	70.5	77.9	90.9	76.2	74.9	75.4	90.9	72.5	72.9
	TOT	77.8	92.3	78.5	71.7	78.4	91.3	76.1	76.8	75.9	91.4	71.2	74.3

Table 5.3: Parsing f-measure (in percent) and average computation time for experiments with varying degrees of integration of acoustic scores. Values of α refer to Constraint C-41 on page 156. For densities, 'all' means mean of all problem instances, 1 and 5 mean the average f-measure of those word graphs with requested density of 1 and 5 respectively. 'max.' means the problem instances with maximum density per utterance (may be smaller than 5). Note that different numbers of problem instances are available for the different word graph densities (in particular for density 5). 'Time' is average computation time needed per solved instance. 'TS' stands for systematic tree search, 'HS' for heuristic search with tabu list. 'All' means f-measure relative to all instances, 'Sol.' means f-measure relative to the solved instances only. `SYN` is the f-measure for syntax edges only. `SEM` is the mean f-measure of all semantic edges (`AGENT`, `EXP`, `PATIENT`, `QUALITY` and `THEME`). `TOT` is the mean f-measure of all edges on all levels.

The following observations can be made:

- The integration of acoustic scores with the parameter α set to 0.1 leads almost always to worse results compared to the lack of integration. Obviously, the acoustic scores get too large weights in this case leading to neglect of grammatical information.

- A parameterization with $\alpha = 0.9$ is (almost) consistently better than no integration. Acoustics and grammar are better balanced in this case.

- Significantly better f-measure values are achieved for minimal word graphs (with a density of one) compared to those with a density of five or maximum density. This result is somewhat disappointing since we had hoped that the combination of acoustics and grammar would not only reproduce the result from a sequential architecture but improve on it.

  In an experiment of re-ranking paths in word graphs using a CDG with hard constraints only, Harper et al. [1999a] could achieve an increase in sentence accuracy from 90% to 94% when using a random sample of word graphs and from 0% to 37% when using only those word graphs that contain the target sentence but not as the highest ranked one.

- Since some word graphs have a maximum density smaller than five, only a reduced number of utterances are taken into account for the columns with density 5 which is why values for density 5 often do not lie between those for density 1 and maximum density.
- Although the semantic levels `SEM` score higher values than the syntactic level `SYN` and slightly lower values than all levels together, no principal difference in f-measure can be found for the different levels.
- Larger word graphs take over-linearly longer to parse than smaller ones except when using a parameter setting of $\alpha = 0.1$. This can best be seen when comparing the average computation time for densities one and five. The word graphs are five times greater (in terms of word edges) but the procedures only need 3.59 to 4.74 longer for $\alpha = 0.1$. For $\alpha = 0.9$ the tree search method requires 9.1 times longer while the heuristic search needs 6.95 times longer. This observation is consistent with our findings in Section 4.11 that the heuristic search methods scale better for large problems. Heuristic search takes 10.1 times longer for word graphs of density five when no integration of acoustic scores is done (compared to 6.95 with $\alpha = 0.9$), i.e., there is a *relative* speedup for larger problems when integrating the acoustic score. In absolute numbers, the heuristic search still takes slightly longer with acoustic integration than without in most cases.
- Finally, it can be seen that the heuristic search achieves a higher f-measure when all problem instances are considered. Again the reason is that the transformation-based search can *always* return an analysis while the tree search fails to do so sometimes. For those cases where a solution could be found, the tree search scores higher f-measure values (but only because it omits the difficult instances).

It should be noted that Table 5.3 on the facing page only gives average numbers; individual problem instances may vary considerably. Figure 5.16 on the next page shows parsing time and f-measure against word graph density for an individual non-representative sentence (n001k011-1 from the Stellingen corpus). Here the integration not only leads to a much better f-measure but also achieves a considerable speedup. While the time needed for the analysis seems to 'explode' when not considering the acoustic scores, only a moderate increase in parsing time can be observed with acoustic scores integrated.

5.6.5 Evaluation with unknown sentences

It has already been noted that 64% of the minimal word graphs contain ('are') the reference utterance and that this figure increases to 80% for a density of five. Although this leaves some room for improvement for the parser, the baseline is remarkably high. Reasons probably include the following:

- All utterances in our corpus have been manually split into 'sentence-like' segments. Consequently, they are much shorter (namely, 8 words on average per utterance) than the average turn length of 22 words per turn [Niemann et al. 1998, p. 2] in Verbmobil.[17] Shorter utterances, of course, have a much greater probability of being returned as the best sentence hypothesis by a speech recognizer.

[17]They are still longer than the utterances of the ATIS (Air Travel Information System) corpus [Marcus, Santorini & Marcinkiewicz 1993].

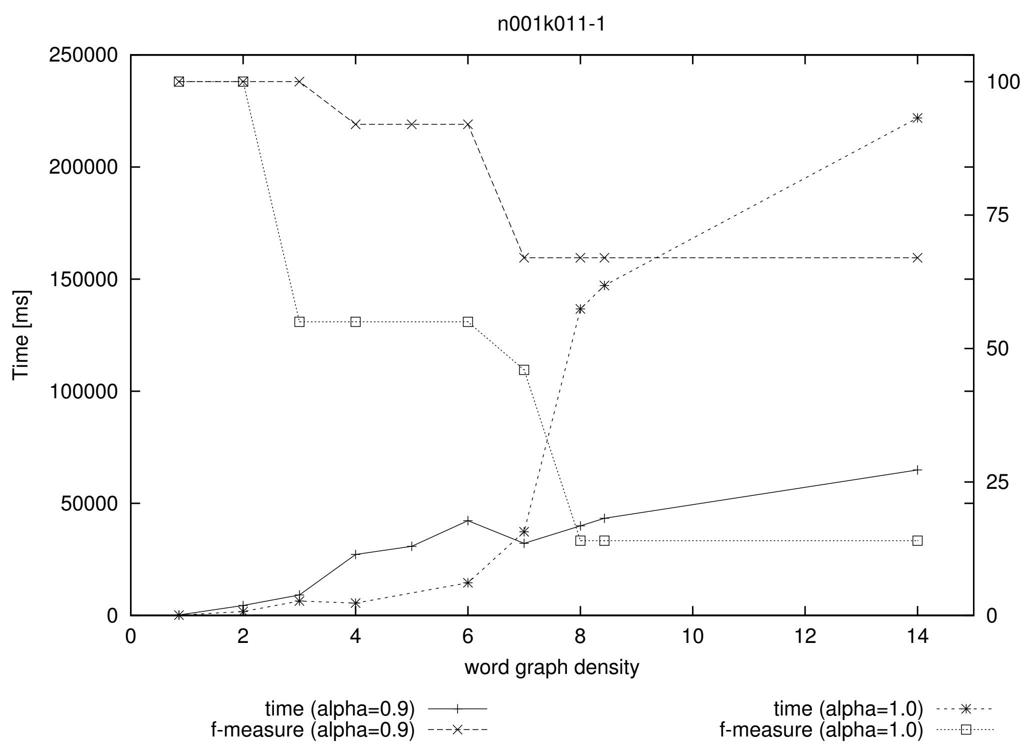


Figure 5.16: Parsing time and f-measure depending on the word graph density for sentence n001k011-1.

- In our experimental setting, the sentences have been read instead of sponteanously spoken. Additionally, no context neighboring the sentence was used and considerable effort has been spent to avoid non-speech noise.

- The sentences used in the experiment are taken from the first parts of the Verbmobil corpus. Therefore, it is highly likely that the employed speech recognizer had already incorporated them into its language model.

At least the last point seems contra-productive for our purpose if we want to verify whether the additional grammatical information used by the parser can help to correct errors made by the speech recognizer. If the recognizer already has access to that knowledge, the effect cannot be as pronounced as in the case where new knowledge is used.

Therefore, a second experiment has been conducted where 50 additional utterances that are not included in the language model[18] have been analysed with the same recognizer which, however, has been configured for an even higher word graph density of 30. The resulting word graphs have an average density of 24.7 (sd 22.6). In 31 of the 50 cases, the reference utterance is contained as the acoustically preferred hypothesis, i.e., the sentence accuracy is almost as high (62% compared to 64%) as in the previous experiment. Furthermore, the reference is identical to

[18]More specifically, the first 50 utterances (manually segmented) from dialogue `g460ac` that is part of the official test set of the speech recognizer (Hagen Soltau, personal communication) have been used in this experiment.

the best hypothesis in three more cases if one ignores minor spelling variations (*unsere*→*unsre*, *ganze*→*Ganze*, ...) resulting in a sentence accuracy of 68%. Seven word graphs do not contain the reference and for the remaining nine instances the reference is contained in the word graphs but not as the best hypothesis.

An ideal parser can, therefore, be expected to improve the sentence accuracy from 68% to at most 86%. However, this is of course a theoretical upper bound. It is also likely that grammatical sentences in the larger word graphs that are different from the reference will be selected by the parser.

In order to get a more realistic upper bound of what our parser can achieve on the data, the new lexical items and grammatical constructions from those nine word graphs which contain the reference as a suboptimal path have been added to the grammar and the parser was run on them. Note that this experiment is too optimistic since only those sentences are considered that bear some potential for improvement. It is not at all easy to avoid errors in the other cases.

Table 5.4 presents the word accuracy and recall[19] for these nine cases resulting from different parser runs (tree search vs. heuristic search, different acoustics integration, $\alpha = 0.9$). The values for the best (acoustic) hypothesis is also given for comparison.

Word graph	**Density**	**Length**	**Parser**		**Best hypothesis**	
			Accuracy	**Recall**	**Accuracy**	**Recall**
24	21	8	46.2 - 66.7%	75.0%	66.7%	75.0%
25	12	6	50.0%	50.0%	83.3%	83.3%
28	7	11	90.0 - 90.9%	81.8 - 90.9%	90.9%	90.9%
29	23	8	37.5 - 55.6%	37.5 - 62.5%	71.4%	62.5%
36	3	6	66.6 - 83.3%	66.6 - 83.3%	83.3%	83.3%
40	2	6	83.3 - 100.0%	83.3 - 100.0%	83.3%	83.3%
41	6	8	71.4 - 85.7%	62.5 - 75.0%	85.7%	75.0%
45	2	12	90.9%	83.3%	80.0%	66.7%
49	4	5	75.0%	60.0%	100.0%	60.0%

Table 5.4: Word accuracy and recall for unknown utterances.

The outcome is not too promising. Although there are some cases where the parser improves the suboptimal best hypothesis of the speech recognizer, the general conclusion is that it is still better to use the best hypothesis than to operate on larger word graphs. Reasons for this behavior include the following.

- The quality of the produced word graphs is still very high so that the improvement that a parser can even *potentially* achieve is limited. Amtrup, Heine & Jost [1997] report *word accuracies* for speech recognizers on the Verbmobil corpus of 70% to 84% for word graph densities of one which go up to 96% for densities of ten (and larger) while the word graphs used in the experiments of this section showed a *sentence accuracy* of up to 68%.

  The question of whether the WCDG parser can yield more favorable results when using more realistic word graphs is subject to further research. But it must be taken into account

[19]Word accuracy and recall are based on a standard Levenstein distance with equal weights for substitutions, deletions and insertions.

that favorable results can be achieved with word graphs that are similar in quality to ours: The word graphs used in Harper et al. [1999a, p. 3] contain the correct sentence in 98% of the cases (compared to 80% in our case) and in 83% (compared to 64%) the acoustically preferred path is the correct sentence.

- Parts of the parser failures are based on search errors rather than modeling errors. That means that if we were not limiting the computation time, the correct solution would finally drop out of the process.
- The interface between acoustics and grammar is still naïve. More work must go into fine-tuning the interface.
- Most importantly, the parser and the grammar have not been designed with speech parsing in mind but were written for analysis of linear sentences (cf. Section 5.2.1). In particular, phenomena of spontaneous speech (such as self corrections, hesitations, repetitions and sentence abortions) and word recognition errors (for instance, *wir*→*sie*, *sie*→*die*) have not been taken into account when modeling the grammar.[20] Therefore no constraints exist that discriminate between different words that are temporal alternatives.

5.6.6 Related work

Harper et al. [1994] discuss different degrees of tightness in coupling a speech recognizer with a parsing component. Tight integration means that the individual processes and knowledge bases cannot be separated while the latter are relatively independent in a loosely integrated system. In a tightly coupled architecture, recognition and parsing are interleaved so that the predictions of the language component can complement the acoustic scores in the search for the best sequence of words. A language model based on n-grams is still often used in tightly and moderately coupled architectures, but lately alternative approaches received more attention, for instance, filtering of bi-gram follow-sets using unification-based CFG [Hauenstein & Weber 1994] as well as direct use of stochastic CFG [Jurafsky et al. 1995] or headword dependencies [Chelba & Jelinek 1998]. Tight (and partially moderate) integration, however, turns out to be difficult for at least three reasons:

- The complexity of the parser is incorporated into the simpler (with regard to complexity) stage of recognition. This can lead to efficiency problems.
- It is not always clear which probabilities to use for the language part and how to train them.
- Only a subset of grammar models and parsing strategies are appropriate for the task since they must operate in a time-synchronous left-or-right manner. In particular, they must be able to assign probabilities to prefix analyses.

A loose coupling is achieved when the recognizer passes a space of hypotheses (in the form of a word graph) to the parser and the latter chooses the best one or acceptable ones from this set.

[20] The parsers in van Noord [2001] and Rosé & Lavie [2001], for instance, explicitly deal with speech phenomena. Heeman [2001] argues for speech recognizers that model spontaneous speech events and pass output to a parser that is enriched with information about such events.

Chappelier et al. [1999] describe a general framework for such a sequential coupling. Note that this more closely resembles the standard view of sequential processing stages where syntactic parsing strictly follows speech analysis without feedback [Amtrup & Weber 1998; Amtrup 1999; Amtrup 2000; Rosé & Waibel 1996; Rosé & Lavie 2001; Bod 1998]. It sometimes seems to be more a question of personal research preferences of whether one considers a parser a more sophisticated form of language model for a speech recognizer or a speech recognizer a dedicated device that delivers (ambiguous and possibly erroneous) input for a parser.

Harper et al. [1999a] report that a loose coupling of a speech recognizer [Harper et al. 1994] and a manual CDG (with only hard constraints) results in an improvement in sentence accuracy. Using an automatically induced CDG [Harper, Hockema & White 1999; White 2000] further boosts the selection power [Harper et al. 2000]. Also see Johnson [2000, Sections 5.5 and 5.6] for experiments with a hand-written as well as induced CDG grammars.

In addition to the loose-tight distinction, coupled recognizer-parser combinations can be distinguished depending on whether they offset the acoustic with the grammatical scores. Systems where the acoustic weight is actually combined with a grammar score of some kind are difficult to find. Ehrlich & Hanrieder [1996, p. 31] combine the acoustic assessment with a score that describes how well a hypothesis fits pragmatic predictions based on dialogue knowledge [Görz & Hanrieder 1995]. Multidimensional weights consisting of, for instance, acoustic score, number of skip edges, number of maximum projections and n-gram statistics are suggested by van Noord [2001] along with appropriately defined update functions and orderings. However, no uniform framework but a cascaded decision procedure is used.

5.6.7 Summary

For the first time an interface between a speech recognizer and the WCDG parser has been designed and experimentally investigated. Although the time needed for analysis increased more than linearly in the size of the word graph, the parser nevertheless succeeded in parsing of word graphs with densities ranging from 3 to 18.25. The number of word arcs of up to 184 arcs per word graph increased considerably when compared to the case of linear sentences. The results suggest that the parser benefits from the integration of acoustic scores from the speech recognizer, independently of the employed solution method. However, as can be expected from the results in Section 4.11, the heuristic search component is superior to the tree search method for large word graphs. At least in the limited setting that has been assumed in these experiments, it was not possible for the parser operating on word graphs to reach the sentence accuracy of the acoustically preferred path. For the time being, it is still better to let the parser run on the best hypothesis of the speech recognizer and ignore the ambiguity contained in the word graph. This last result should be seen in the context of the favorable results that Harper et al. [2000] achieved using a word graph filtering method based on a hard CDG. A direct comparison of the two approaches is not feasible for several reasons:

- The corpora differ. While Harper et al. [2000] use the (extended) resource management corpus, a subset of the Verbmobil corpus of appointment-scheduling dialogues has been used in this section.
- The speech recognizers differ. In particular, they yield different accuracies and varying word graph densities.

- The evaluation metrics differ. While we measured parsing accuracy on a completely disambiguated sentence, Harper et al. [2000] report sentence accuracies on the reduced word graphs.

Although these differences render the experiments nearly comparable, a closer investigation is subject to further research.

5.7 Integrating a part-of-speech tagger

5.7.1 Motivation

Part-of-speech (POS) taggers assign one or more category tags to words in a sentence. Very different architectures for POS taggers have been derived from a manifold set of frameworks: manual finite state rules [Voutilainen 1995], neural networks [Schmid 1994a], decison trees [Magerman 1995; Schmid 1994b], Markov models [Rabiner 1990; Cutting et al. 1992; Charniak 1993; DeRose 1988], maximum entropy models [Ratnaparkhi 1997b; Ratnaparkhi 1996], error-driven learning of transformation rules [Brill 1993; Brill 1995; Brill 1997; Ngai & Florian 2001] and example-based reasoning [Daelemans et al. 1996; Zavrel & Daelemans 1999]. Implementations for these architectures exist that achieve an accuracy of at least 95% on previously unseen data.

A subsequent parser can benefit from a tagger because either the former can then operate directly on a sequence of category symbols instead of ambiguous word forms or it can at least prune its search space by deleting hypotheses that are incompatible with the categories predicted by the tagger.

This section describes results from experiments with the integration of a part-of-spech tagger in the WCDG parsing system. In particular, we are interested in an interface which takes into account that the tagger prediction may be uncertain or even wrong in some cases and which therefore treats the tagging classification as ‘soft’ information. The following questions are of particular interest:

- Can a tagger reduce the ambiguity so that the parsing process itself becomes faster?
- Is the information provided by a tagger accurate enough to improve the parsing accuracy?
- What are upper bounds for efficiency and accuracy improvements?
- Does it help the parser when the tagger emits more than one symbol per word if the decision is unclear?

Tagging word graphs is more complicated than tagging word sequences [Samuelsson 1997a; Samuelsson 1997b]. Unfortunately, no taggers for word graphs are available so that we restricted the experiments in this section to linear sequences of possibly ambiguous words.

5.7.2 Experimental setup

The TnT tagger [Brants 2000], a state-of-the-art Markov tagger, has been deployed for all experiments.

The tagger was trained on the NEGRA corpus version 2 [Skut et al. 1997b; Brants, Skut & Uszkoreit 1999] which uses the Stuttgart-Tübingen tagset (STTS) [Schiller et al. 1995]. The category inventory of the Stellingen grammar and lexicon is based on the STTS, but departs from it in some details so that a mapping of tags becomes necessary. Some STTS tags, e.g., `FM` (foreign language), `NE` (name) or punctuation marks, have been omitted from the Stellingen tag set because they do not occur in the corpus. If such tags are output by the tagger they are automatically mapped to appropriate tags by default rules, e.g., `NE` → `NN` (proper noun). Most STTS tags correspond to Stellingen tags in a one-to-one way. We make the additional distinction between definite and indefinite articles that the STTS does not show. But a correct mapping from the underspecified STTS tag for determiners to Stellingen tags is feasible because articles belong to the group of closed-class categories and the cases can thus be enumerated. In exceptional cases it was necessary to include specialized mapping rules for particular words. For instance, the word group of *beid-* (*both*) is treated as an adjective in the Stellingen grammar but as an attributive indefinite pronoun with determiner in the STTS. Ambiguous mapping rules are used in cases where the STTS uses a unique tag for words for which the Stellingen grammar provides more than one. For instance, the STTS lists the word group of *haben* (*to have*) as an auxiliary verb only while the Stellingen grammar additionally allows main verb forms. These ambiguous mapping rules lead to incomplete disambiguation even if the tagger emits exactly one tag per word.

Table 5.5 on the next page compares the Stellingen corpus to the NEGRA corpus with regard to POS ambiguity.

Two classification figures are given to characterize the corpora.

POS tag entropy is defined as the entropy for the tag unigram $P(t)$ and determines the minimum number of bits per word necessary to encode the tag sequence in the corpus given the probability distribution of tag unigrams:

$$H = -\sum_{t \in T} P(t) \cdot \log P(t)$$

Average tag ambiguity is simply the mean tagging ambiguity in a corpus $w_1 ... w_n$ under the (unrealistic) assumption that a complete lexicon is available.

$$A = \frac{1}{n} \cdot \sum_{i=1}^{n} |\{t \in T | t \text{ is admissible tag for } w_i\}|$$

It can be seen that the tag set size for the Stellingen corpus is reduced from 55 to 33 tags. The Stellingen corpus is more ‘uniform’ than the NEGRA corpus since both the mean ambiguity per word (1.23<1.61) and the entropy (3.76<4.27) are smaller for the Stellingen corpus. Of course, the corpus used for training is much larger than the one used for testing.

- 224 sentences, spontaneous dialogues
- 442 word types, 1,941 word tokens
- 33 tags
- Average ambiguity A=1.230602
- POS tag entropy H=3.761071

#T	Type no.	Type perc.	Token no.	Token perc.
1	357	92.727	1,467	79.598
2	25	6.494	327	17.743
3	3	0.779	49	2.659
	385	100.000	1,843	100.000

(a) Stellingen corpus

- 20,602 sentences, newspaper text
- 51,272 word types, 355,096 word tokens
- 55 tags
- Average ambiguity A=1.611544
- POS tag entropy H=4.273873

#T	Type no.	Type perc.	Token no.	Token perc.
1	49,189	95.937	238,545	67.178
2	1,884	3.675	45,586	12.838
3	164	0.320	46,789	13.176
4	32	0.062	20,090	5.658
5	1	0.002	2,715	0.765
6	1	0.002	1,363	0.384
7	1	0.002	8	0.002
	51,272	100.000	355,096	100.000

(b) NEGRA corpus

Table 5.5: POS statistics for the Stellingen and the NEGRA corpus.

The TnT tagger achieves 96.7% accuracy [Brants 2000] on the NEGRA corpus. Its quality has been independently confirmed by Megyesi [2001]. On the Stellingen corpus it achieves an accuracy of 93.489% when emitting exactly one tag per word. The tag ambiguity in the output nevertheless is 1.07 tags/word because of the ambiguous mapping rules (see above). The accuracy on the Stellingen corpus is reduced in comparison to that on the NEGRA corpus because firstly the corpora for training and testing are different and secondly because the differing tag sets require a mapping that inevitably loses some information. When allowing multiple tags per word (beam 1,000), the accuracy increases to 98.318% with a tag ambiguity of 1.51 tags/word.

5.7.3 Interface

In Section 5.6.2 it was discussed how an acoustic score can be integrated into the WCDG parsing system. Here we adopt a similar approach but can omit the adjustment of scores because the tagger scores are in the same range as the WCDG scores. Again, we use a constraint template (with parameter α) which allows flexible adjustment of the weight the tagger information should receive by projecting the tagger score to the interval $[\alpha, 1.0]$ (cf. Constraint C-42). By setting $\alpha = 0.0$ possible structures that are not licensed by the tagger can be eliminated; for values $\alpha > 0.0$ unsupported selections are only penalized but not completely discarded.

```
C-42 {X} : Tagging : [α+(1.0 − α)*pts(X@id)] :
     X.level=SYNTAX -> false;
     Integrate the tagger score.
```

The function `pts()` (POS tag score) makes the score that a tagger assigns to a category available to the grammar writer. The function must be added to the constraint language (cf. Section 2.5.3).

By restricting the constraint to the syntactic level, it is guaranteed that the score is integrated exactly once for each word. Table 5.6 exemplarily shows the relation between tagger output, tagging score function `pts()` and constraint weight for the example sentence n001k000-2.

word	tagger output: tags & prob.	X@cat	pts(X@id)	weight
wann	ADV 0.862 PWAV 0.148	ADV	0.862	0.7758
		PWAV	0.148	0.1332
		else	0.0	0.1
wäre	FIN 0.500 VAUXFIN 0.500	X@cat=FIN	0.5	0.45
		X@cat=VAUXFIN	0.5	0.45
		else	0.0	0.1
es	PPER 1.00	X@cat=PPER	1.0	0.9
		else	0.0	0.1
Ihnen	PPER 1.00	X@cat=PPER	1.0	0.9
		else	0.0	0.1
denn	ADV 0.873 KON 0.124 KOKOM 0.003	X@cat=ADV	0.873	0.7857
		X@cat=KON	0.124	0.1116
		X@cat=KOKOM	0.003	0.0027
		else	0.0	0.1
recht	ADJP 0.617 ADV 0.375 NN 0.008	X@cat=ADJP	0.617	0.5553
		X@cat=ADV	0.375	0.3375
		X@cat=NN	0.008	0.0072
		else	0.0	0.1

Table 5.6: Tagger output, POS tag score function and constraint weight for $\alpha = 0.1$.

Note that the value of the POS tag score function `pts()` is zero for all categories that are not predicted by the tagger. In particular, if the tagger emits a single category the function value is one for this category and zero for all others.

5.7.4 Experimental results

This section describes the experiments that were run to examine the impact of tagger information on the WCDG parsing system.

The heuristic search method with tabu list (cf. Section 4.9.1) is used in this evaluation and the runs have been limited to 60 seconds processing time. Accuracy of unlabeled dependency edges is reported.

We will look at efficiency and accuracy for four types of tagging modes:

1. *No tagger* is used in a baseline experiment.
2. The *ideal tagger* ‘knows’ the correct tags; it is used to approximate an upper bound of what we can expect from integrating a tagger device.
3. A *single tag tagger* emits a single tag for each word.
4. The *multi tag tagger* may emit more than one tag (along with a probability estimation) for a single word when the probabilities of more tags are within a certain beam (here 1,000).

For the last three modes, experiments with the parameter α of Constraint C-42 on page 168 set to 0.0, 0.1 and 0.9 respectively have been conducted.

224 utterances from the Stellingen corpus have been analyzed for all settings. We excluded five sentences from the evaluation because they were not solved either in the baseline experiment or in one of the experiments with $\alpha > 0.0$. We did so in order to simplify the evaluation and enhance the comparability of the results.[21] The parameter setting of $\alpha = 0.0$ requires special mentioning because the information provided by a tagger is integrated as a hard constraint in this case and the search space is pruned rigorously. This results in 31.5% unanalyzed sentences for the single tag tagger and 8.7% failures for the multi tag tagger.

Figure 5.17 on the next page looks at accuracy and efficiency for the idealized assumption that we have a perfect tagger available which always predicts the correct tag. The experiment can tell us what we can expect from a tagger integration at best.

The system with the ideal tagger integrated is both faster and more accurate. In Figure 5.17a on the facing page, it is clearly visible that information about the correct part-of-speech can help to make the parsing problems easier to solve. The largest effect can be observed for a setting with $\alpha = 0.0$; the computation time is approximately halved. This comes as no surprise because all alternatives with wrong part-of-speech tags are strictly discarded from the space of admissible hypotheses in this case. For a soft integration with parameter α set to 0.1, the effect is still very pronounced while no final conclusion can be drawn for the weakest kind of integration (with $\alpha = 0.9$). Figure 5.17b on the next page shows a similarly favorable outcome for the accuracy. The achieved results are generally better when the tagging information is incorporated; the best accuracy is achieved for values 0.0 and 0.1 for the parameter α.

Figure 5.18 on page 172 gives analogous information for the single tag tagger.

In Figure 5.18a on page 172, it stands out that the hard integration with $\alpha = 0.0$ again speeds up the parsing process. For the soft variants with $\alpha = 0.1$ or $\alpha = 0.9$, the parsing time does not show significant differences to the baseline with no tagger. Figure 5.18b on page 172 clearly shows that although a hard integration of a single tag tagger can in fact accelerate the analysis, the tagger also makes too many errors so that the accuracy drops dramatically. However, if the predicted tags are considered soft information, the accuracy tends to increase when compared to the baseline.

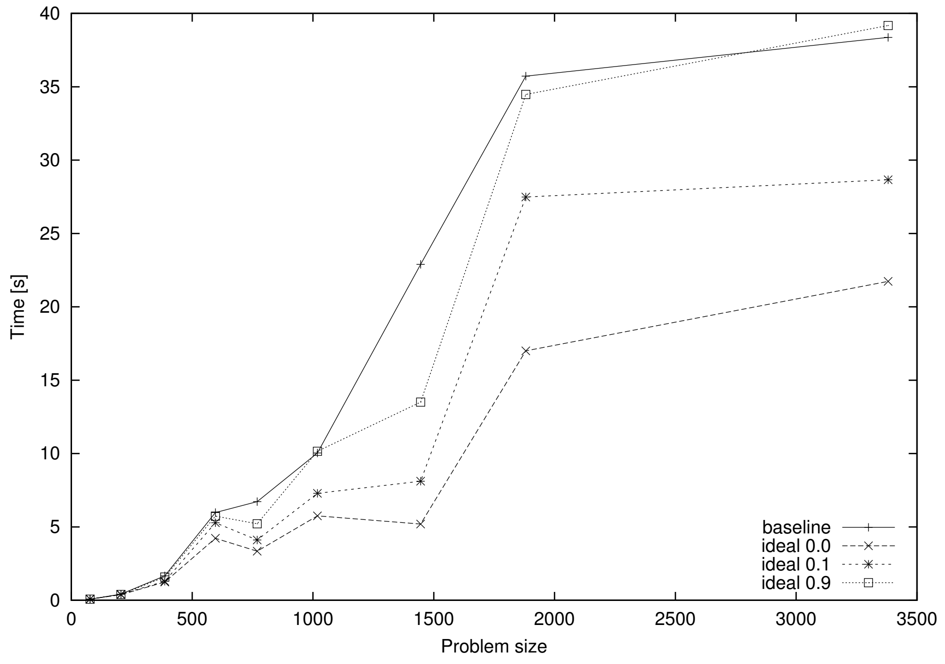
Figure 5.19 on page 173 gives the experimental data for the multi tag tagger.

This time not even the hard integration leads to a speedup (Figure 5.19a on page 173). All attempted variants are similarly efficient with only slight advantages for the $\alpha = 0.0$ setting. The crisp integration again results in a greatly reduced accuracy while the soft integration variants in turn tend to improve the accuracy.

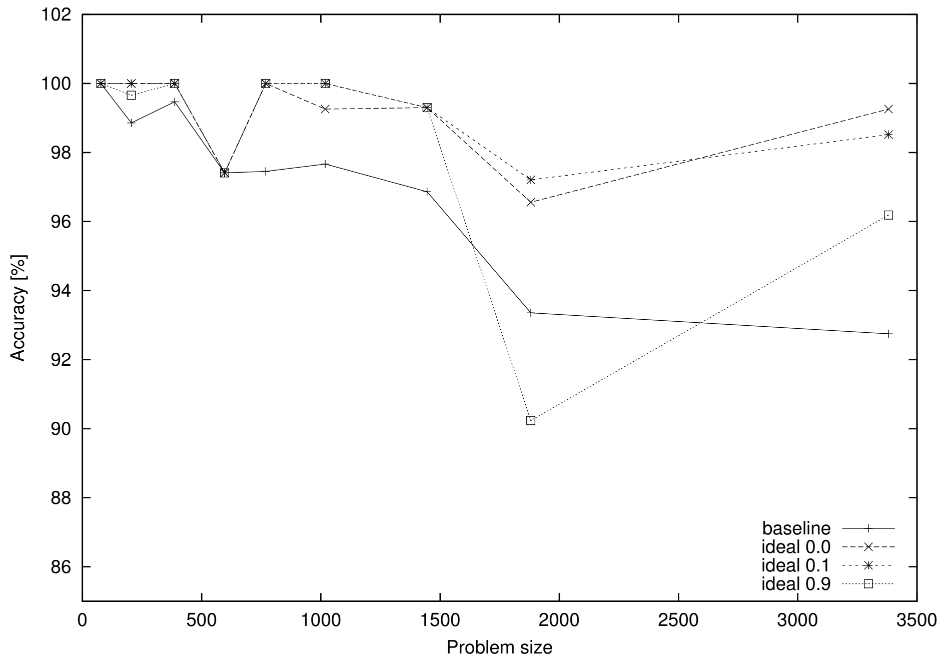
Finally Figure 5.20 on page 174 contrasts the four tagging modes directly by repeating the results for parameter α fixed at 0.1.

Of course, the ideal tagger is unrivaled, both for efficiency and accuracy. The multi tag tagger is slightly faster and more accurate than the single tag tagger. Only the ideal tagger is significantly faster than the rest. Single tag tagger, multi tag tagger and baseline are approximately of the same speed. The baseline experiment results in the worst accuracy.

[21]The named procedures all solved more than 98.6% of the utterances (i.e., less than three unsolved sentences). The only procedure that managed to solve all problems surprisingly is the single tag tagger with $\alpha = 0.1$, not the ideal tagger.

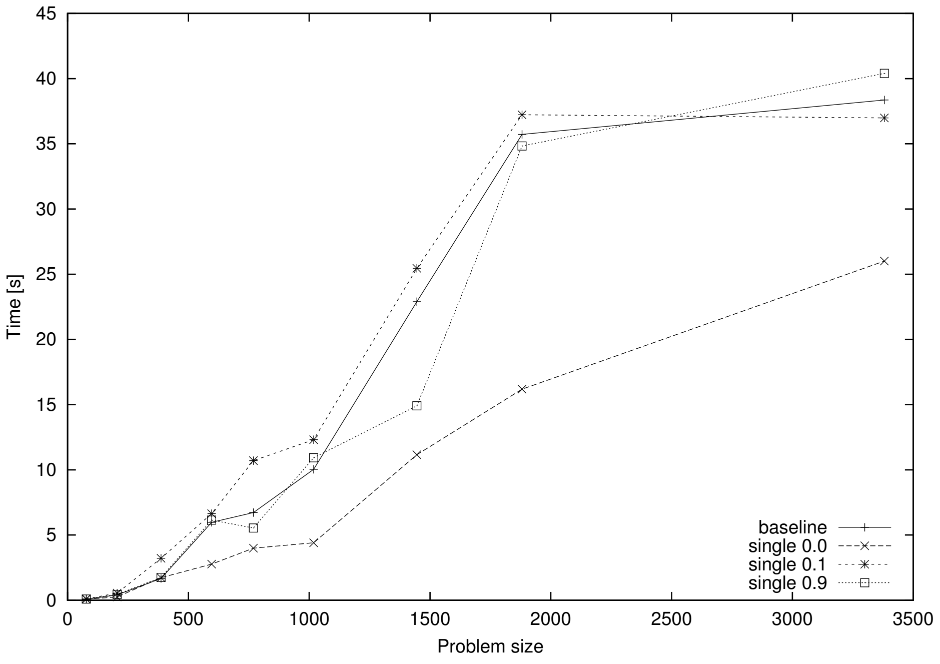


(a) Efficiency

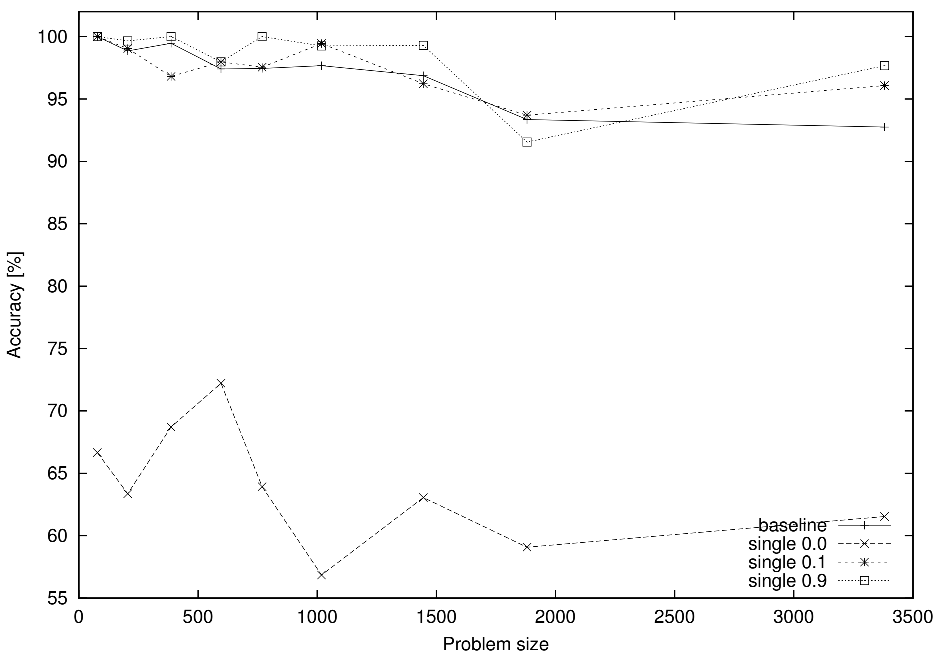


(b) Accuracy

Figure 5.17: Efficiency and accuracy of the WCDG parser using the ideal tagger.

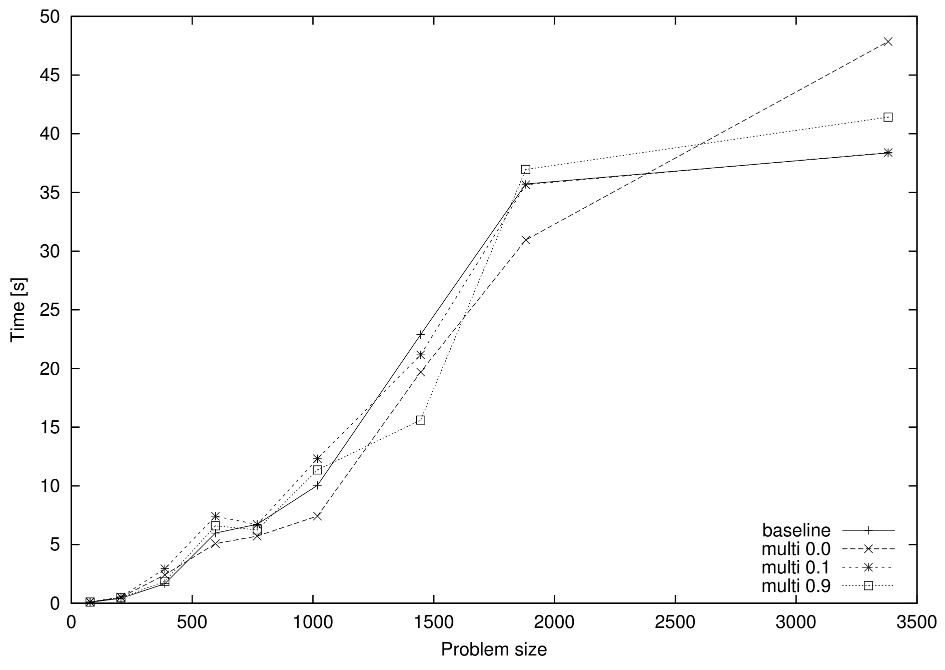


(a) Efficiency

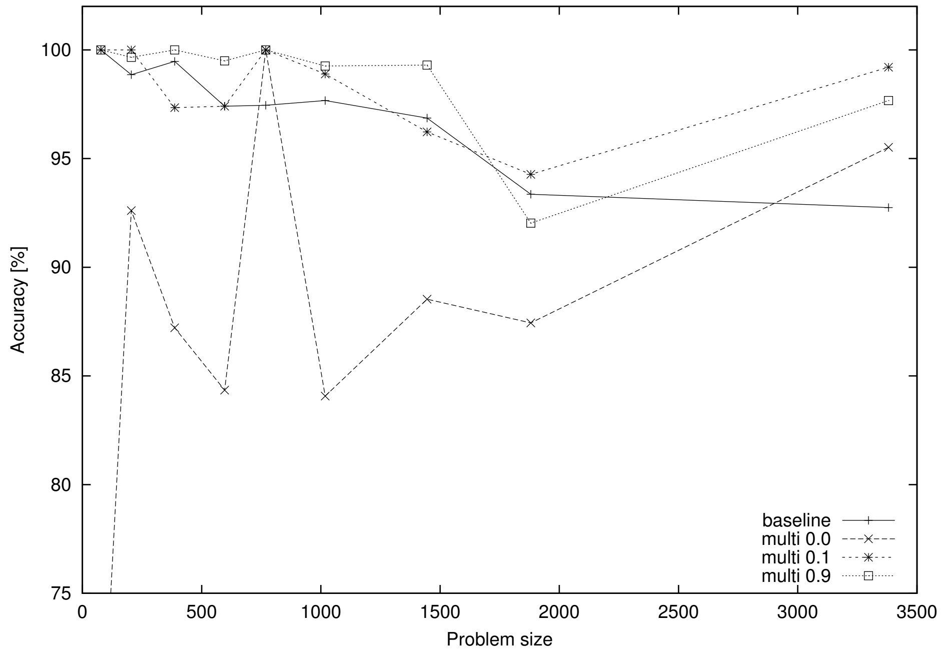


(b) Accuracy

Figure 5.18: Efficiency and accuracy of the WCDG parser using the single tag tagger.

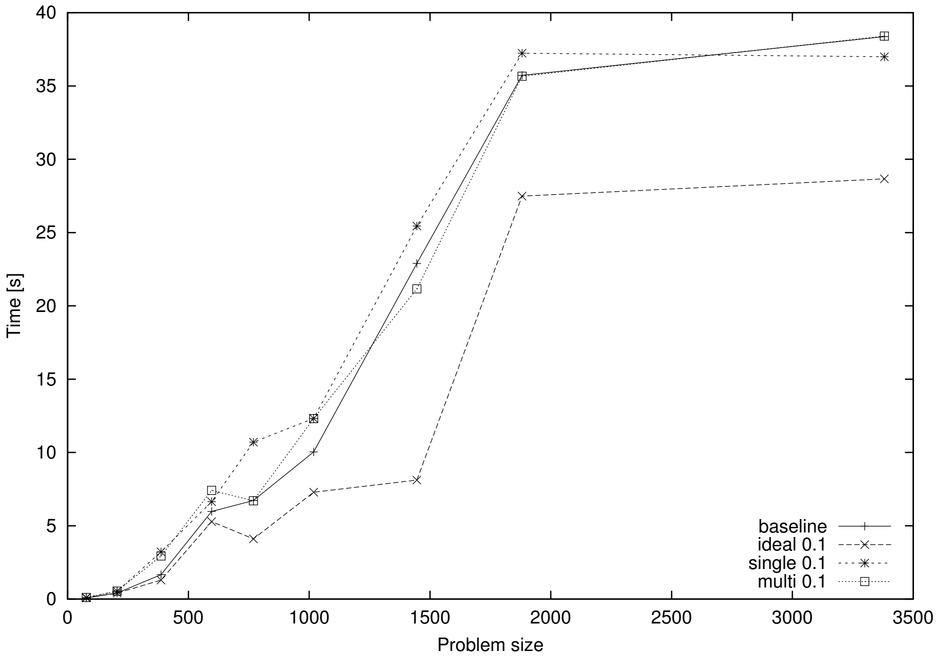


(a) Efficiency

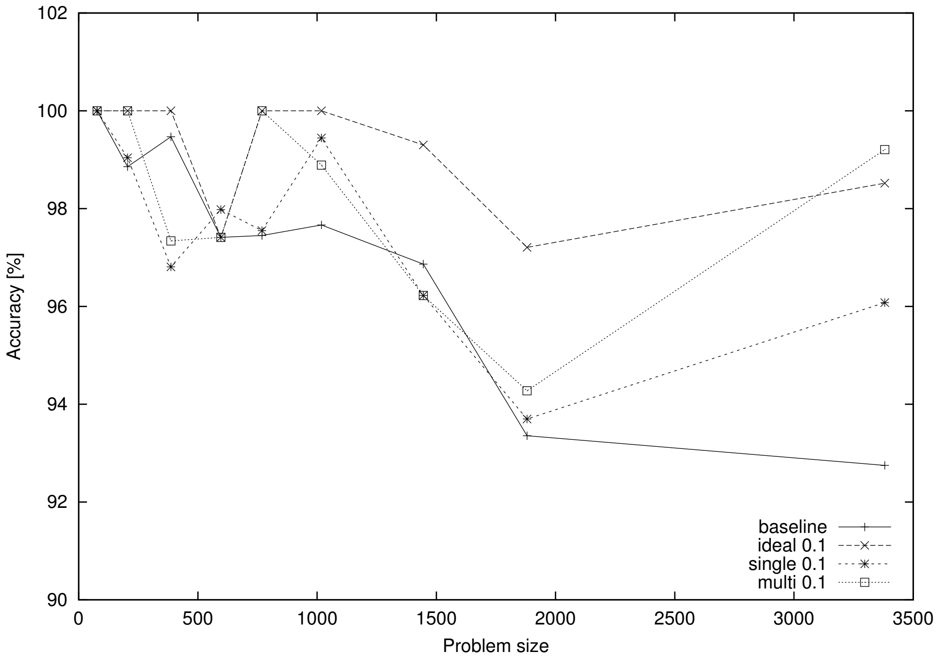


(b) Accuracy

Figure 5.19: Efficiency and accuracy of the WCDG parser using the multi tag tagger.



(a) Efficiency



(b) Accuracy

Figure 5.20: Direct comparison of four tagging modes: no tagger, ideal tagger, single tag tagger and multi tag tagger.

5.7.5 Related work

Part-of-speech tagging is a popular research area because it is both simple and well-understood as well as challenging. Although accuracies of over 90% can easily be achieved with very simple techniques, building a perfect tagger nevertheless requires a complete understanding of natural language, thus rendering the problem 'AI-hard'. Nearly all current tagging approaches are based on local contexts, i.e., no global or structural information is used for the decisions. Taggers are useful for the semi-automatic al construction of tree banks [Marcus, Santorini & Marcinkiewicz 1993; Skut et al. 1997b], information extraction and of course as a preprocessing step before parsing. Charniak et al. [1996] investigate the impact of POS taggers on accuracy and efficiency of a probabilistic context-free grammar chart parser. They conclude that the gain in accuracy achieved by a multi tag tagger is not worth the increased work load when compared to the performance of a single tag tagger and therefore recommend the latter. This is partially incompatible with our findings although one has to consider that the parsing frameworks are completely different.

5.7.6 Summary

Hard and soft integration of an external part-of-speech tagger into the WCDG parsing system has been examined. The potential of such a combination is clearly demonstrated by an ideal tagger which always predicts the correct tag. In this case the accuracy increases by 5 percentage points while reducing the computation time by 50%. The experiments with a realistic tagger show that a hard integration where only those lexical entries are considered by the parser that are compatible with the prediction of the tagger lead to a large decrease in accuracy (and partially an increase in efficiency). This disadvantage may be mitigated in the future when taggers with state-of-the-art precisions of 98% and above are applied to WCDG parsing. A soft integration on the other hand causes an increased accuracy. In short, a soft integration of tagging information allows for more accurate parsing runs but no significant differences in efficiency. The multi tag tagger showed even better results than the single tag tagger.

A seamless integration of exchangeable POS tagging modules into the WCDG system is forthcoming. Hagenström [forthcoming] describes the integration in detail, both conceptually and technically.

5.8 Adjusting constraint weights using corpus data

5.8.1 Motivation

This section describes an experiment where the weights of conflicting constraints are adjusted so that the combined impact of the constraints mimics the observations from a corpus of annotated utterances as closely as possible. The purpose here is two-fold:

- Firstly, we try to answer the question of whether weighted constraints can be used to model linguistic phenomena when inherently conflicting tendencies as well as performance aspects of language production and comprehension are involved.

- Secondly, the parameters of our model, i.e., the weights of the constraints, should be approximated using a corpus of annotated sentences.

It should be noted that although we investigate a linguistic phenomenon largely influenced by performance aspects we do not lay claim to psycholinguistic adequacy. That does not mean that it is impossible to base a psycholinguistic theory on WCDG but such an undertaking would require additional theoretical work and experiments which are beyond the scope of this thesis.

This section only looks at a single phenomenon and only simulates the effects of the small constraint cluster responsible for the assessment of extraposition of relative clauses. We model the observations found for a linguistic phenomenon, but these findings will not directly improve the parsing performance of the grammatical system because the competing alternatives can usually be clearly identified on the surface of the utterance so that they cannot be confused. However, the discussed constraints can prefer one or the other reading when more and different analyses are possible.

Section 5.9 adopts a different approach and estimates the weights of a given grammar using a corpus of completely annotated WCDG structures. It optimizes the parsing performance instead of the prediction quality for the extraction of relative clauses. Hence, the evaluation criteria are more coarse-grained since only accuracy and recall measures are used and no individual phenomena are examined.

5.8.2 Extraposition of relative clauses

"Leser schwitzen nicht, aber sie sind auch nicht trainiert, zwölf Wörter zu speichern; eher die Hälfte. 32 dieser 38 kostbaren Vokabeln also baumeln im Nirgendwo – außer bei jenen sieben Getreuen, die sich verzückt an eine nochmalige Lektüre machen."
—— Schneider [1999]

The potential extraposition of relative clauses from German discontinuous verb groups has been chosen as the grammatical phenomenon to be scrutinized.

E-76	Kaum	jemand	trauert$_i$	dem	alten	Parteichef$_j$
	nach$_i$,	gegen	dessen$_j$	Umgebung	eine	Reihe
	von	Korruptionsprozessen	läuft.			
	Rarely	anybody	mourns	the	former	party leader
	mourn$_{\text{prefix}}$	against	whose	environment	a	series
	of	corruption trials	is pending.			

'Rarely anybody mourns the former party leader against whose environment a series of corruption trials is pending.'

Example E-76 (cf. Figure 5.21 on the next page[22]) has been taken from the training corpus (cf. next section) whereas its counterpart without extraposed relative clause is given as Example E-77 on the facing page.

[22]This figure has been produced with our own `negra2png` tool (available from `http://nats-www.informatik.uni-hamburg.de/~ingo/prg/`) which has been extensively used for visualization during the experiments.

E-77	Kaum	jemand	trauert$_i$	dem	alten	Parteichef$_j$,
		gegen	dessen$_j$	Umgebung	eine	Reihe
	von	Korruptionsprozessen	läuft,	**nach**$_i$.		

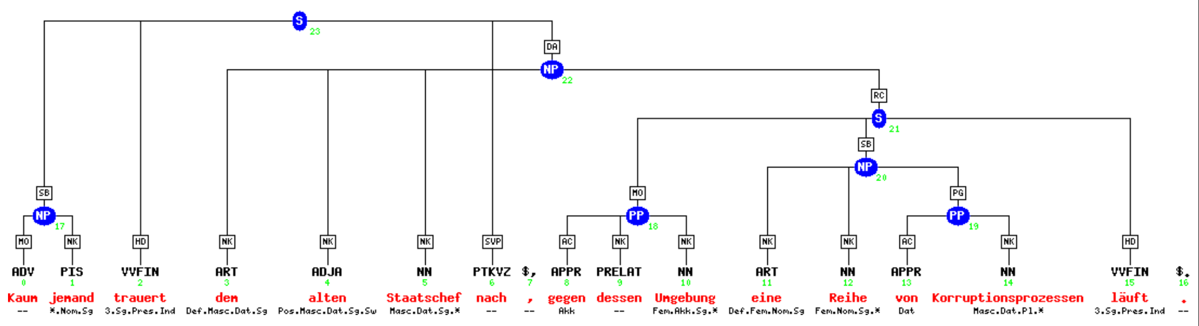

S 23
DA
NP 22
RC
S 21
SB
NP 20
PG
PP 19
MO
PP 18
SB
NP 17
MO NK HD NK NK NK SVP AC NK NK NK NK AC NK HD
ADV PIS VVFIN ART ADJA NN PTKVZ $, APPR PRELAT NN ART NN APPR NN VVFIN $.
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Kaum jemand trauert dem alten Staatschef nach , gegen dessen Umgebung eine Reihe von Korruptionsprozessen läuft .
-- *.Nom.Sg 3.Sg.Pres.Ind Def.Masc.Dat.Sg Pos.Masc.Dat.Sg.Sw Masc.Dat.Sg.* -- -- Akk -- Fem.Akk.Sg.* Def.Fem.Nom.Sg Fem.Nom.Sg.* Dat Masc.Dat.Pl.* 3.Sg.Pres.Ind --

Figure 5.21: Discontinuous phrase structure tree.

Both variants are grammatical. One and the same principle is violated (and also satisfied) in both cases: Parts of a sentence that belong together should occur close to each other. Either the relative clause is separated from its antecedent (as in the first case) or the separable verb prefix appears only long after the verb itself. It is beyond dispute that other factors influence the potential for extraposition as well, such as the nature of linguistic material between noun phrase and relative clause (verbal words only vs. nouns, pronouns etc.), information structure, thematic roles, definiteness, animacy and prosody in the case of speech data [Uszkoreit et al. 1998a; Uszkoreit et al. 1998b]. The linear distance, however, is a particularly simple criterion and is considered exclusively here. We follow Uszkoreit et al. [1998b] who empirically confirm Hawkins [1994]'s principle which predicts a linear sequence of phrases in a sentence such that the processing of the phrases can start as soon as possible. For German relative clauses, Hawkins [1994] therefore considers the distance between relative pronoun and antecedent as well as the length of the relative clause the two main criteria.

5.8.3 Corpus data

As in Section 5.7, the NEGRA corpus version 2 [Skut et al. 1997b; Brants, Skut & Uszkoreit 1999] has been used for these experiments. It consists of German newspaper sentences annotated with part-of-speech tags [Schiller et al. 1995], morphological information and a discontinuous phrase structure (see Figure 5.21 for an example). We neglect the differences between spontaneous language and newspaper text which has been modified by the editorial departments of the publisher.

The corpus contains 20,602 sentences of which 2,332 contain relative clauses. One (0.04%) sentence contains four relative clauses, nine (0.39%) three, 118 (5.06%) two and 2,207 (94.51%) one.

As Uszkoreit et al. [1998b], we only consider relative clauses that depend on a noun phrase (node label `NP`) or a prepositional phrase (node label `PP`) and that are themselves embedded in a sentence.

This excludes 39 relative clauses because they depend on the whole sentence (or other phrases) and 106 relative clauses because they are not (or not directly enough) embedded in a sentence. We consider a relative clause extraposed if the governing phrase contains a gap immediately before the relative clause; otherwise the relative clause is not extraposed. Relative clauses (962 occurrences) that are located at the end of the sentence and adjacent to the noun phrase are further excluded from consideration because the two cases cannot be distinguished properly. This leaves us with 622 cases of extraposed relative clauses and 742 cases of counter examples.

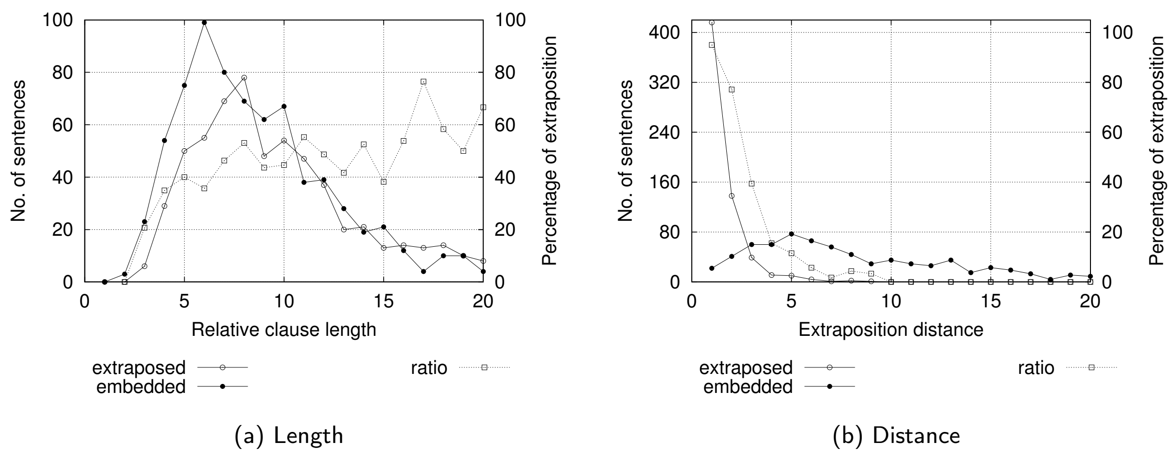


Figure 5.22: Frequency of extraposed and embedded relative clauses depending on (a) the length of the relative clause and (b) the (potential) distance between embedded and extraposed location (limited to lengths and distances not larger than 20).

Figure 5.22a shows the distribution of relative clause lengths for both cases. The relative clauses (up to length 20) have a mean length of 8.565 (limited to length up to: ∞ 9.155, 30 9.009, 25 8.873, 15 8.022) for the embedded case and of 9.471 (limited to length up to: ∞ 10.537, 30 10.132, 25 9.852, 15 8.545) for the extraposed sentences.

The extraposition distance is the number of words (excluding punctuation) between the end of the governing noun phrase and the beginning of the extraposed relative clause or between the end of the relative clause and the end of the embedding sentence. Figure 5.22b gives an overview of how the two distinguished cases are distributed for different extraposition distances. The mean distance is 1.542 for extraposed clauses (with 66.9% having distance 1, 22.2% distance 2, 6.3% distance 3 and only 4.7% having a distance larger than 3). For embedded clauses, the distance is more evenly distributed with only 16.6% having a distance smaller than 4 and a mean of 9.806.

These corpus statistics suggest that both length and distance influence the acceptability of extraposed relative clauses. However, the latter seems to be a much stronger indicator.

5.8.4 Constraints

The two surface criteria described in Uszkoreit et al. [1998b] and confirmed for our corpus in the last section, should now be formulated using (slightly simplified) constraints.

```
C-43 {X, Y} : RCLength : 0.5 :
       X.level=RCEND & Y.level=SYNTAX &
       X@id=Y@id &
       X@cat=RELPRONOUN &
       distance(X^id, X@id)>1 ->
         distance(Y@id, Y^id)>threshold
```

Extraposed relative clauses tend to be long.

Assuming an additional level RCEND (comparable to level REF in Section 3.4.7 but pointing to the end of the noun phrase), we can use Constraint C-43 to express our finding that the extraposed relative clauses are longer on average. However, such a simple approximation is relatively coarse because only two cases are distinguished: lengths that exceed the threshold and those that do not. The same objection holds for Constraint C-44 which penalizes extrapositions over long distances.

```
C-44 {X} : RCDistance : 0.5 :
       X.level=RCEND & X@cat=RELPRONOUN ->
         distance(X^id, X@id)<threshold
```

Actual extraposition distances are relatively small.

Since the acceptability of an extraposition changes gradually depending on the length of the relative clause and the extraposition distance, dynamic constraints (cf. Section 2.5.1) whose weight depends on the actual construction should be employed.

Going back to Figure 5.22b on the facing page, we can conjecture that the 'extraposition rate' follows a function $g(d) = a^{-bd}$ where d is the extraposition distance. For the other criterion (cf. Figure 5.22a on the preceding page), the picture is not as clear but we will assume a linear relation for simplicity reasons here: $f(l) = cl$.

The free parameters a, b and c can be approximated using the nonlinear least-squares Marquardt-Levenberg algorithm [Williams & Kelley 1998, for instance] and the example cases of the corpus. The expected function value has been set to one for extraposed examples and to zero for embedded clauses.[23] This results in concrete formulae $f(l) = 0.0380854l$ and $g(d) = 1.26034^{-1.07837d}$.

So far, the criteria have been considered in isolation. For comparison, we define two more functions that combine the two in an additive and in a multiplicative way. The final functions (with optimized parameters) are $a(l,d) = 1.20533^{-1.61315d} + 0.00697314l$ and $m(l,d) = 1.33276^{-1.27018d} \cdot 0.103924l$.

These functions can now be used as dynamic weights as in Constraint C-45 and Constraint C-46 on the following page. Note that we neglect normalization here.

```
C-45 {X, Y} : DynRCLength : [ 0.038 * distance(Y@id, Y^id) ]  :
       X.level=RCEND & Y.level=SYNTAX &
       X@id=Y@id &
       X@cat=RELPRONOUN &
       distance(X^id, X@id)>1 ->
         false;
```

The shorter an extraposed relative clause the better.

[23]Using the corpus examples directly instead of the percentage of extraposed clauses (as in Figure 5.22 on the facing page) should reflect the fact that we are not going to model the *rate* but find a classifier for our problem. However, varying machine learning approaches, such as decision trees, are feasible here.

```
C-46 {X} : DynRCDistance : [ 1.26**(-1.08*distance(X@id, X^id)) ] :
       X.level=RCEND & X@cat=RELPRONOUN &
       distance(X@id, X^id)>1 ->
         false;
```

The shorter the extraposition distance the better.

Figure 5.23 shows the results of an experiment where the impact of such constraints has been simulated using the examples from the corpus, i.e., the above constraints have been applied to each example and the threshold on the x-axis splits the range into those examples for which an extraposition is preferred and those where it is not. The resulting accuracy on the corpus data is the depending value. This procedure corresponds to a WCDG which decides on the extraposition using only the two mentioned constraints.

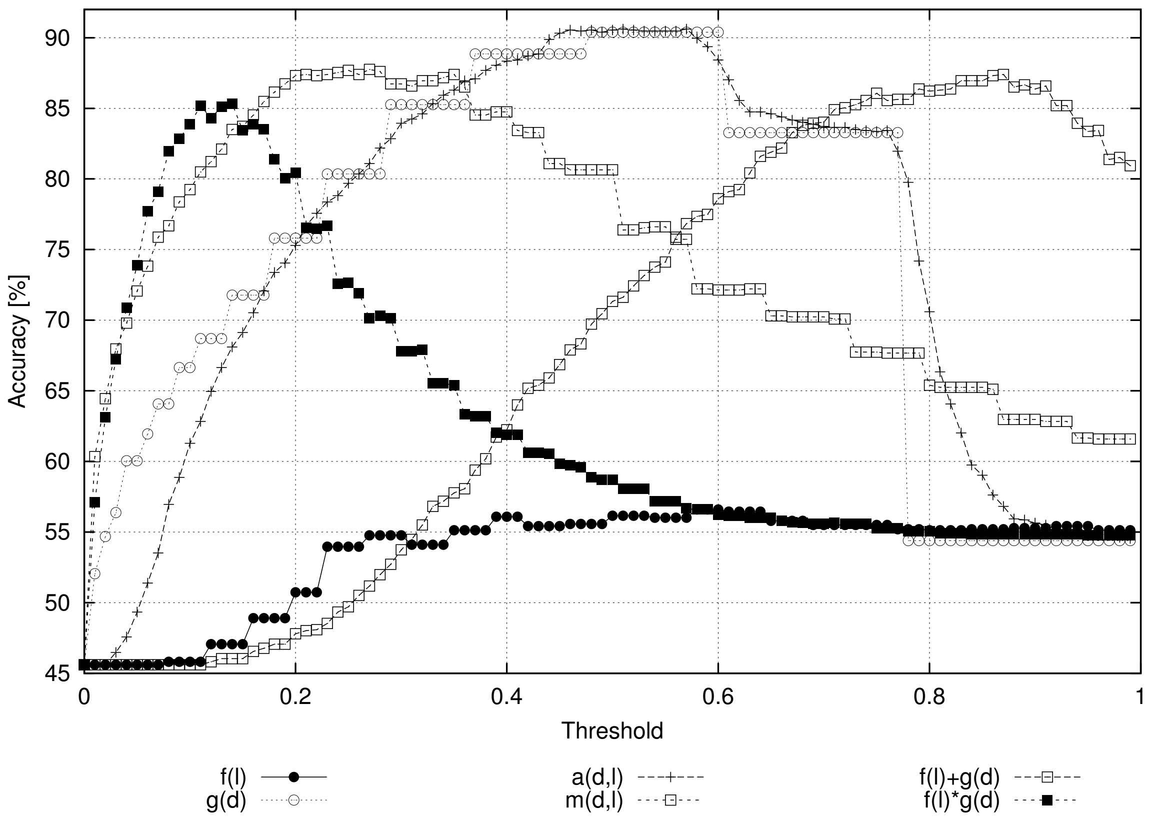


Figure 5.23: Accuracy of six classifiers for relative clause extraposition. The isolated classifiers $f(l)$ and $g(d)$ correspond to the criteria of relative clause length and extraposition distance. $a(l,d)$ and $m(l,d)$ refer to the (additive and multiplicative) combinations. Finally, $f(l)+g(d)$ and $f(l)\cdot g(d)$ are combinations where the components have been optimized individually, but not in combination.

The following observations can be made. As expected from the corpus statistics, the extraposition distance is a much stronger indicator than the relative clause length (and thus should have a much more influential constraint weight). The latter achieves an accuracy of only 56.5% which is hardly above the baseline of 54.4%. The first one, however, predicts the correct outcome in 90.4% of the cases. It is only surpassed by the additive combination $a(l,d)$ which reaches 90.6%.

All classifiers that include the distance criterion achieve at least 85% accuracy for one or the other threshold.

In order to verify the general validity of the approach, the set of examples has been split into disjoint training (1,106 cases), held-out (122 cases) and test (136 cases) sets. The function parameters have been re-adjusted using the same method but using the training set only. Table 5.7 summarizes the results which confirm that even this simple approach manages to generalize from training examples to unseen instances. However, the only reliable estimator for which the threshold parameter stays constant for the three sets is the distance criterion alone. Here almost 90% accuracy can be achieved on the test set using the threshold from the held-out data. All other estimators seem to be 'polluted' by the unreliable clause length criterion.

	$f(l)$	$g(d)$	$a(d,l)$	$m(d,l)$	$f(l)+g(d)$	$f(l)*g(d)$
Threshold for training set	0.54	0.62	0.46	0.27	0.87	0.13
Threshold for held-out data	0.54	0.36	0.52	0.22	0.84	0.1
Threshold for test set	0.54	0.59	0.49	0.35	0.95	0.17
Accuracy at training threshold	89.7%	61.8%	86.8%	83.0%	87.5%	79.4%
Accuracy at held-out data threshold	89.7%	52.9%	89.0%	81.6%	85.3%	72.8%
Accuracy at test set threshold	89.7%	63.2%	90.4%	89.7%	88.2%	86.0%

Table 5.7: Accuracy on the test set.

5.8.5 Related work

The work described in this section is inspired by the first part of Uszkoreit et al. [1998b], a case study about extraposition of German relative clauses that combines methods of linguistic theory and corpus-linguistics in order to better understand performance aspects of language generation and perception. Müller [1999] includes an HPSG analysis for extraposed relative clauses in German which employs performance constraints to restrict the ambiguity. The idea of estimating grammar parameters from annotated corpora is, of course, fundamental in most probabilistic grammars (cf. Sections 1.4 and 5.3.4). Alshawi & Carter [1994] describe an automatic method to balance the contributions of a set of preference functions that rank competing analyses from a natural language system. They report that although scale factors found by using a least-square optimization on a corpus improve on hand-tuned parameters, the quality can be even further improved when the optimization is followed by a hill-climbing procedure.

5.8.6 Summary

This section demonstrated that conflicting criteria for particular linguistic phenomena taken from the linguistic literature can be transfered to a WCDG. In this case dynamic constraints are utilized to encode preferences about the acceptability of the extraposition of relative clauses. The distance between relative pronoun and relative clause as well as the length of the relative clause have been used as the main influential parameters. Constraint weights have been estimated from an annotated corpus which would result in a prediction accuracy of almost 90%. It could be observed that the extraposition distance is a much stronger indicator which may be traced back to the choice of the simple linear approximation function used for the condition of clause length. The focus in this section was on formulating known linguistic conditions as WCDG constraints and on the general possibility of determining constraint weights automatically.

5.9 Learning constraint weights

This section is based on a paper published in the proceedings of the conference 'Recent Advances in Natural Language Processing 2001' [Schröder et al. 2001a].

5.9.1 Motivation

This section describes an experiment in automatic learning of the weights of a WCDG using an evolutionary optimization technique.

Because there are usually very many constraints and because the consequences of their interaction are hardly foreseeable, the manual assignment of appropriate weights to constraints is a labor-intensive and error-prone task. Furthermore, since constraints forbid (or at least disprefer) certain configurations instead of allowing additional constructions (as is the case for, e.g., context-free rules) the extension of a given grammar requires revising old constraints and re-adjusting their weights. An automatic procedure that optimizes the weights for a given grammar can therefore help a grammar writer to focus on the constraints themselves without having to worry that the weights are unbalanced. However, the learning of constraint scores is difficult mainly because constraints can generally encode a large range of conditions, the represented restrictions can overlap or even contradict each other and in general the effects of interactions of weights are complex. Hence, probabilistic models which usually rely on a decomposition of the overall probability into smaller stochastically independent parameters can hardly be applied to this problem. This section aims at answering the following questions:

- What kind of properties must an optimization technique have in order to be applicable to the problem at hand?
- Is it tractable in practice to run such an optimization process?
- Can the procedure improve the manually assigned weights of a given grammar?
- Can the procedure find good weights starting from (almost) random weights?

5.9.2 Optimization techniques for constraint weights

Standard estimation techniques for probabilities do not qualify for learning the weights of a given WCDG (see above). Optimizing constraint scores can also be viewed as a constraint satisfaction optimization problem (cf. Section 2.2.4) itself. The constraints are the variables which get weights assigned to them; the scoring function is defined by an objective function which takes into account the accuracy and the efficiency of a candidate on a fixed corpus. Obviously this constraint satisfaction optimization problem is much harder to solve than the WCDG parsing problems that we investigated so far. Firstly, the values are real-valued numbers instead of discrete symbols which define a dependency edge.[24] Secondly, the objective function is extremely expensive to calculate since one evaluation involves solving of all WCDG parsing problems from the chosen corpus. Apparently, it is prohibitive to ask for a complete systematic search procedure (cf. Section 4.3.2) and even the heuristic search (cf. Section 4.3.4) variants that we considered so far are not applicable because either no real conflicts are available (for min-conflicts, cf. Section 4.9.1) or the definition of an appropriate neighborhood is difficult (for guided local search, cf. Section 4.9.2). Given the gigantic search space and an expensive objective function, a search method is needed which on the one hand sparsely draws samples from the search space and on the other hand allows fast moves within the set of candidates.

Genetic algorithms [Holland 1975; Michalewicz 1996] are an optimization technique inspired by biological systems.[25] They maintain an evolving population of candidates, so-called individuals, throughout an iterative process, i.e., from generation to generation. The initial population is either an educated guess or consists of randomized individuals. Each new generation is created by first adding individuals resulting from recombination and mutation to the population and then selecting the 'fittest' ones for the next cycle.

Evolutionary optimization techniques seem appropriate for the problem at hand for two reasons. On the one hand, genetic algorithms explore only a small portion of the neighborhood. It is this selective nature that makes it feasible to find an approximate solution in a reasonable amount of time. On the other hand, the randomness in genetic algorithms allows jumps in the search space, making it possible to reach areas of the search space when other methods get stuck in small isolated areas.

5.9.3 Genetic operators

Each individual consists of the whole set of weights for one particular grammar instance, i.e., a list of real-valued numbers, one for each constraint.[26] Two different genetic operators are applied to the population (cf. Figure 5.24): crossover and mutation.[27]

$$
\left.\begin{array}{cccc} .1 & .2 & .3 & .4 \\ .8 & .6 & .3 & .2 \end{array}\right\} \rightarrow \mathrm{AC} \rightarrow \quad \begin{array}{cccc} \mathbf{.45} & \mathbf{.4} & .3 & \mathbf{.3} \end{array}
$$

$$
\left.\begin{array}{cccc} .1 & .2 & .3 & .4 \\ .8 & .6 & .3 & .2 \end{array}\right\} \rightarrow \mathrm{MC} \rightarrow \left\{\begin{array}{cccc} \mathbf{.1} & \mathbf{.6} & \mathbf{.3} & \mathbf{.4} \\ \mathbf{.8} & \mathbf{.2} & \mathbf{.3} & \mathbf{.2} \end{array}\right.
$$

$$
\begin{array}{cccc} .1 & .2 & .3 & .4 \end{array} \quad \rightarrow \mathrm{M} \rightarrow \quad \begin{array}{cccc} .1 & \mathbf{.25} & .3 & \mathbf{.2} \end{array}
$$

Figure 5.24: Illustration of different genetic operators.

Crossover is the rough analog of reproduction in nature, and for each pair of individuals two different crossover operators are used (with probabilities p_{ac} and p_{mc} respectively):

[24]Strictly speaking, constraint satisfaction optimization problems do not allow real-valued numbers as values because the domains have to be finite. Of course, this could be overcome by assuming a large but limited set of possible constraint weights.

[25]See, for example, Gottlieb [2001] for an overview. Stützle & Hoos [2001] describe the newer *ant colony optimization* approach to combinatorial optimization that is also population-oriented by borrowing techniques from the behavior of ants.

[26]Assuming that the constraints themselves are fixed, an individual completely describes one grammar instance.

[27]Schröder et al. [2001b] provide a brief analysis of the effectiveness of the individual genetic operators.

- **Arithmetic crossover (AC):** The descendant is the pairwise weighted arithmetic mean of the two involved individuals.
- **Multi-point crossover (MC):** The genes of the two involved individuals, i.e., their lists of weights, are 'intertwined'.

Mutation (M) is inspired by random changes in biological genes in nature, e.g., due to cosmic rays. It is applied to a single individual and can be subdivided into three types (with probability p_r, p_a and p_z respectively):

- **Relative mutation:** A weight is randomly increased or decreased.
- **Absolute mutation:** A weight is assigned a completely random number.
- **Mutation-to-zero:** The current weight is set to zero.

During the whole process, the population size is kept constant. Because the individuals produced by the genetic operators are added to the population, a mechanism is needed to select the members of the new generation from the current population.

Selection is based on the objective function or fitness measure which in our experiments is defined as a weighted mean of the f-measure (80%, cf. Figure 5.1 on page 132) and a time index[28] (20%):

- Include the five fittest individuals in the next generation.
- Discard the five worst individuals.
- Replenish the new population with randomly selected individuals where the chance of a survival of an individual is proportional to its fitness.

This procedure guarantees that the best individuals are not lost and the genetic diversity is maintained.

5.9.4 Experiments

The experiments were run using the Stellingen grammar and corpus (cf. Section 5.2.1). The experimental design needed to be extremely careful because the computational load for such experiments is very high. A cluster of some 20 SUN Ultra workstations were used in parallel to compute the objective function values for the individuals of a population. Between five and ten generations were evaluated per day.

[28]The time index is a measure for parsing speed with zero meaning immediate completion and one meaning that the process uses the maximum amount of time allowed by a hard time limit that is necessary for practical reasons.

Baseline

In order to get a feeling for what a trivial algorithm can achieve on our data, we counted the number of 'trivial' edges (with root attachment and null label) in the annotations. The parsing procedure actually has to tackle two tasks, finding the correct structural configuration and disambiguating lexical alternatives. It turns out that if one ignores lexical ambiguity, 83.5% of all edges point to the root and have the null (default) label. This high number is due to our design decisions when modeling the grammar, in particular the use of twelve levels most of which are auxiliary levels and therefore irrelevant for large parts of a sentence (cf. Chapter 3). However, if one chooses a random (i.e., totally uninformed) lexical possibility, the percentage of trivial edges decreases to 47.9% which may be considered the baseline for our experiments.

Improving a manual grammar

The first experiment tries to answer the question of whether it is possible to improve the weights of the carefully crafted manual Stellingen grammar with 485 constraints (free parameters). In an educated guess the following probabilities were selected as experimental parameters: $p_{ac} = 0.25$ (arithmetic crossover), $p_{mc} = 0.25$ (multipoint crossover), $p_r = 0.02$ (relative mutation), $p_a = 0.002$ (absolute mutation) and $p_z = 0.002$ (mutation-to-zero).

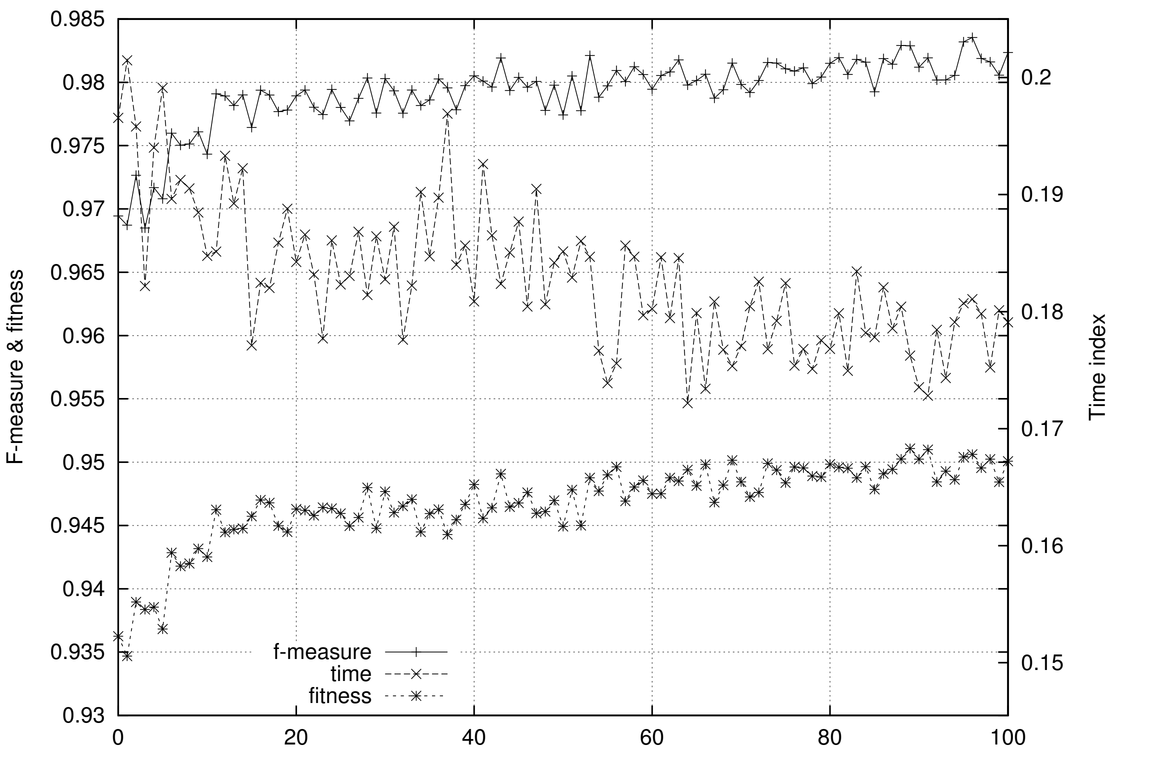


Figure 5.25: F-measure, time and fitness for the experiment of improving manually assigned weights.

Figure 5.25 presents the results from which the following observations can be made. The f-measure increased from 96.9% to 98.4% (reduction of error rate from 3.1% to 1.6% by 48.4%).

Since processing time was measured over the corpus and the time index may be dominated by some few lengthy parsing runs, no further conclusions can be drawn with respect to the efficiency criteria.

The resulting fitness increased from 93.6% to 95.1% (reduction of 'error rate' from 6.4% to 4.9% by 23.4%). The improvements are small but significant and it should be noted that the original grammar was already heavily hand-tuned.

Finding soft weights automatically

The encouraging results from the first experiment suggested another, more ambitious one: Can a similar level of performance be achieved without resorting to the weights of a hand-crafted initial grammar? In this experiment, soft constraints receive random weights in the initial population; however, we assume here that the grammar designer deliberately distinguished between hard and soft constraints. Therefore, absolute mutation and mutation to zero were de-activated which resulted in a reduction of the number of free parameters by 44%. The final parameters were as follows: $p_{ac} = 0.25$, $p_{mc} = 0.25$, $p_r = 0.02$, $p_a = 0.0$ and $p_z = 0.0$. The experimental parameters were changed slightly multiple times after the process seemed to enter a saturation phase. In order to escape this local minimum, additional randomness was introduced into the procedure. The number of generations was increased to 300 for the this experiment.

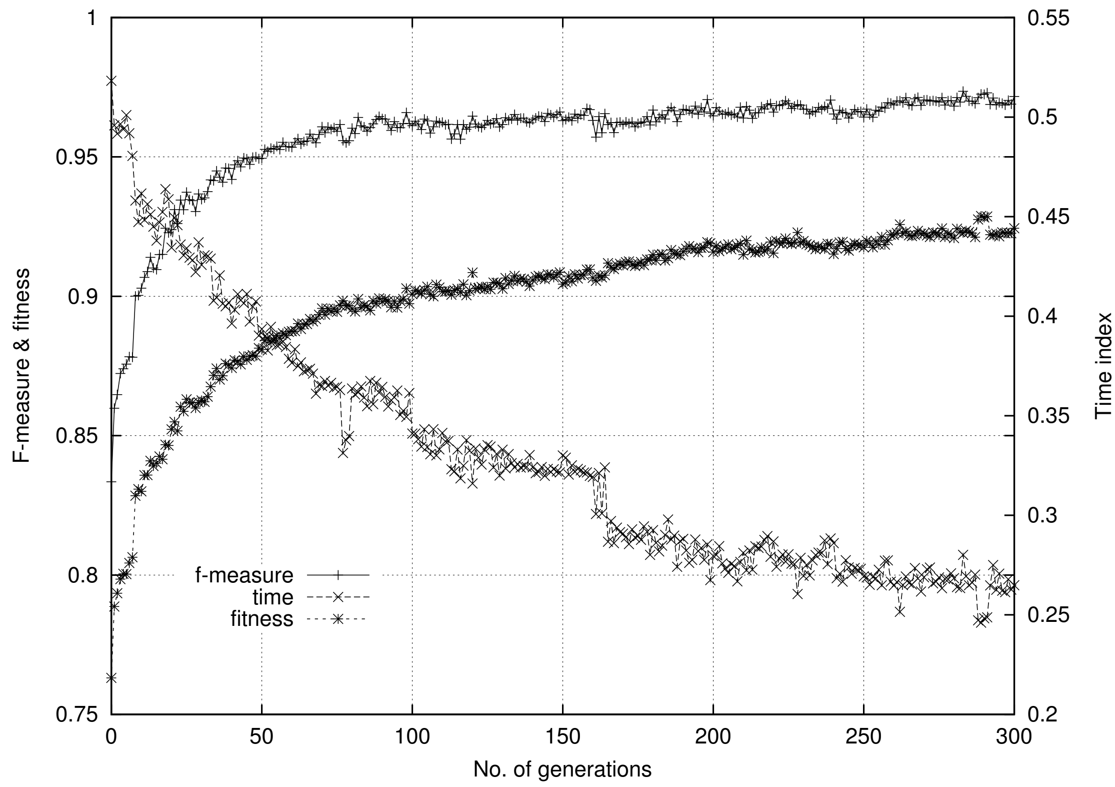


Figure 5.26: F-measure, time and fitness for the experiment of learning soft grammar weights from scratch.

As can be seen from Figure 5.26 on the facing page, the f-measure started at 83.3%, hit 96.6% after 100 generations and finally reached 97.4% (reduction of error rate from 16.7% via 3.4% to 2.6% by 79.6% and 84.4% respectively); the total processing time decreased by 52.5% (via 36.1% at generation 100). The fitness rose from 76.3% via 90.3% after 100 generations to 92.8% (reduction of 'error rate' from 23.7% via 9.7% to 7.2% by 59.0% and 69.6% respectively). Two observations can be made. The learning process was able to reach a similar level of performance as the original manual grammar. Even though a considerable improvement was achieved, it was not possible to attain the quality of the optimized manual grammar weights that resulted from the first experiment.

Results on test set

The results reported so far are intra-experimental, i.e., they are tested on the training set. In order to judge the evolved grammars independently, we tested the individuals from the experiments on a held-out test set of 90 more utterances. For this problem set, the manual grammar has a fitness of 91.5%. This value increases for the final individual of the first experiment to 92.3%. The best random assignment of weights that was used to start the second experiment scored an initial fitness of 77.2% and improved to a fitness of 91.6% (cf. Figure 5.27). In either case, both a speedup of analysis and an increased f-measure were observed.

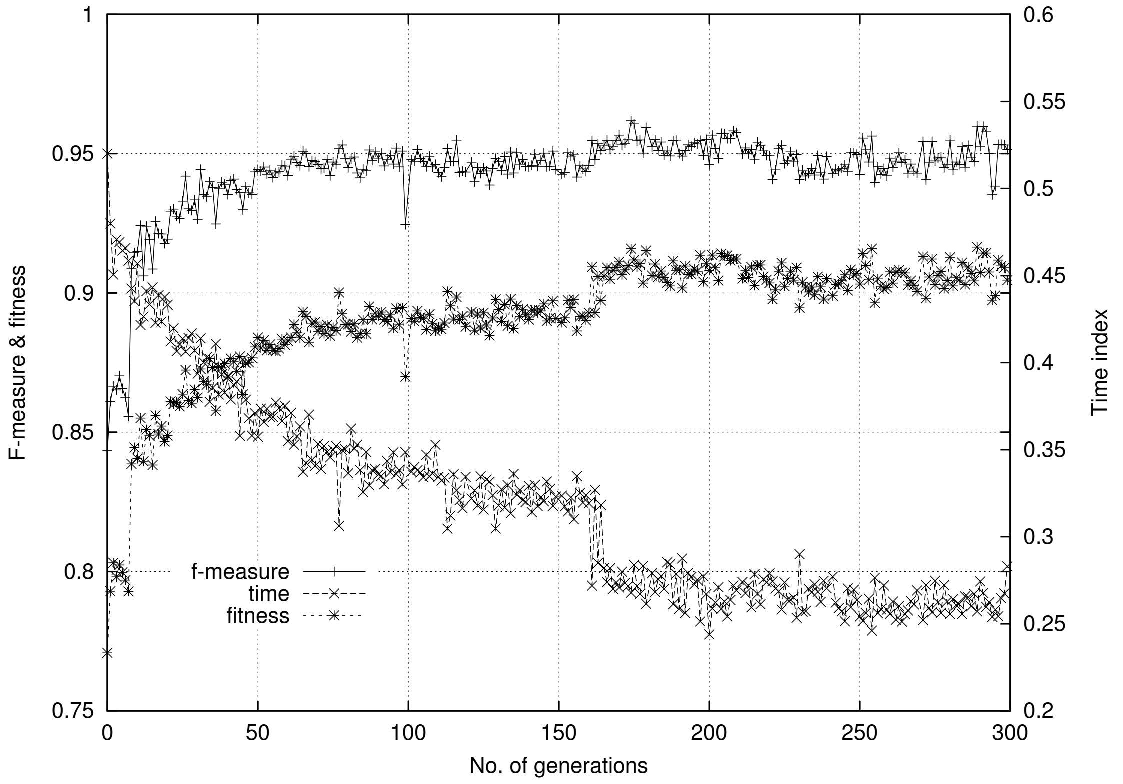


Figure 5.27: F-measure, time and fitness of the best individuals taken from each generation of the second experiment evaluated on the test set.

In order to confirm the robustness of these results with respect to test set variations, the test set was randomly split into two subsets. Both subsets showed exactly the same tendencies and varied only within a three percent margin. A verification of these results on a larger test set seems desirable.

5.9.5 Related Work

Automatic learning of grammars and grammar weights has been investigated in a wide range of research activities. However, few reports can be found for dependency grammars, and indeed very little work deals with genetic algorithms as the primary method.

Optimality theory [Prince & Smolensky 1993, cf. Section 1.4] assumes a strict ordering of constraints with respect to importance, and this hierarchy determines the grammatical utterances. It is assumed that the constraints themselves are universal while the ordering is language-dependent (cf. Section 3.5). Although it can be argued that it is a simpler problem to decide on a hierarchy of constraints than on individual numerical weights, it nevertheless remains a difficult task. As far as learning the hierarchical ordering is concerned, only selected problems within phonology have been studied, mainly from the (cognitively interesting) perspective of unsupervised learning [Tesar 1995; Tesar & Smolensky 1998]. For an alternative learning algorithm for optimality theory see Boersma & Hayes [2000] and its critical assessment in Keller & Asudeh [2002].

This work focuses on the task of learning parameters for a given grammar. The more general problem of finding a complete grammar (including the rules themselves) has been tackled by various approaches in the field of statistical NLP (cf. Sections 1.4 and 5.3.4). For probabilistic context-free grammars, for instance, the rules are extracted from an annotated corpus and the weights are computed from their relative frequencies in the corpus. If no annotated corpus is available, however, the parameters for the context-free rules can be approximated by an efficient training algorithm [Charniak 1993]. Collins [1997] describes a probabilistic model for dependency parsing based on lexical as well as tag affinities and linear distance. An evaluation on the Wall Street corpus [Marcus, Santorini & Marcinkiewicz 1993] is given. Eisner [1996] additionally investigates models based on selectional preferences as well as decisions made during language production, but restricts himself to unlabeled trees. Carroll & Charniak [1992] go one step further by trying to learn complete rules for dependency grammars from unannotated corpora. However, they train from artificially created data. Lankhorst [1993] describes an approach for induction of context-free grammars using genetic algorithms, but this work deals with small artificial languages only. Keller & Lutz [1997] extend the idea to stochastic context-free grammars. Formal languages such as those recognized by push-down automata are induced using genetic algorithms also by Huijsen [1993].

The main difference between statistical methods and the work reported in this section is that the former calculate the parameters more directly, i.e., in a well-defined deterministic way, and can therefore guarantee a certain rate of convergence. Probabilistic methods, however, need a well-defined probabilistic model with stochastically independent parameters (if possible) which can then be estimated from a corpus. Genetic algorithms, in contrast, rely on the randomized generation of new candidates and a powerful selection strategy. Neither is a stochastic model required which is difficult for WCDG [Schröder 1996] nor is the search space systematically explored. It is interesting to note that some parallels can be identified between how the parsing problem for WCDG is understood and how grammar weights are learned, especially in comparison to probabilistic approaches. Parsing with WCDGs is a process of selection from a pre-defined set of possibilities in a similar way that the constraint scores are selected by the genetic algorithm. In contrast, parsing with probabilistic context-free grammars, for instance, is a constructive process driven by re-write rule, and probabilities can also be constructed or deterministically derived from an annotated corpus.

Constraint induction for [W]CDG has been tried in two related ways so far. Schröder [1996] uses an annotated corpus to find a set of constraint instances using a small set of constraint templates and based on smoothed relative frequencies. However, the approach suffers from too small a corpus and limited constraint templates [Schröder 1997b]. For hard constraints, White [2000] introduces so-called abstract role values which are derived from example utterances. An abstract role value covers a whole set of actual configurations and can therefore be used to generalize the characteristics of the training set.

5.9.6 Summary

The possibility of automatic learning of constraint weights for a grammar from annotated corpora has been investigated. The general expressiveness and flexibility of constraints as well as the enormous size of the search space have been identified as obstacles for the application of standard probabilistic techniques and standard search methods respectively. Although requiring still a huge amount of computational effort genetic algorithms prove themselves suitable for the task of optimizing constraint weights. A first experiment confirmed that the quality of a manual grammar (in terms of efficiency and accuracy) can be significantly improved (f-measure error rate reduced by 48.4%). In a second experiment, which started with almost random weights (except hard scores) assigned to the constraints, the quality reached the level of the original manual grammar but not the level of the final optimized grammar resulting from the first run. These results have been validated on a test set. The techniques developed in this section can fully replace the labor-intensive and error-prone task of manual weight assignment to soft constraints. This finding is even more remarkable as only a comparatively small training corpus has been used.

Chapter 6

Conclusion

Gradation is ubiquitous in natural language.

This is the working hypothesis that started this thesis. In the previous five chapters it has been shown that the proposed formalism of weighted constraint dependency grammars (WCDG) can serve as a unified framework for the treatment of gradation in natural language. WCDGs not only offer a well-defined formalism to concisely express what the underlying constraints of natural language understanding are but additionally allow one to ascertain the status of the constraint, be it a strict rule which describes what structures are possibly imaginable or be it a slight preference which only selects the preferred one among a set of ambiguous analyses for a sentence.

6.1 Summary

WCDGs are an extension of constraint dependency grammars (CDG) as defined by Maruyama [1990b], a grammar definition that assigns feasible dependency structures to natural language utterances based on constraints which describe properties of grammatical utterances. In WCDGs, a constraint receives a score between zero and one depending on how important it is for an analysis to fulfill the constraint or on the dynamic degree to which the constraint is satisfied. This generalization of constraints allows the inclusion of grammatical (and other) conditions in the arbitration process that hold often but not always or that should preferably hold but are not absolutely required. Constraint violations are cumulative as supported by psycholinguistic experiments; in particular it is possible for multiple weaker constraints to overrule a stronger constraint whose penalty score is lower than any of the weights of the former constraints.

A constraint language for WCDG has been defined, both syntactically by means of rules in extended Backus-Naur form and semantically by a concise mapping of the WCDG parsing problem to the well-known class of constraint satisfaction optimization problems. Based on this formalization of WCDG, the worst-case complexity of the WCDG recognition problem has been identified as belonging to the class of $\mathcal{NP}$-complete problems and the set of languages generated by WCDGs has been classified as a proper superset of the mildly context-sensitive languages and as a proper subset of the context-sensitive languages.

WCDGs have been shown to be appropriate for modeling a large variety of linguistic phenomena, such as immediate dominance, valence, a subset of German word order, projectivity including exceptional cases and semantic aspects of thematic roles and reference resolution. However, the (computational) restriction to at most binary constraints has also been found to severely limit the expressiveness of the formalism in practical cases, and several techniques to circumvent this restriction have been discussed. The spectrum of application scenarios for weighted constraints includes well-formedness conditions (traditional grammar constraints), preferences, technical means to avoid spurious ambiguities and to 'tag' certain configurations, interfaces for the fusion of information from different sources, partial parsing, approximation of higher constraints and dynamic constraints which carry varying weights depending on the evaluation context.

The reduction of the WCDG parsing problem to constraint satisfaction optimization problems facilitates the deployment of well-studied algorithmic schemes. Three major classes have been discussed and experimentally evaluated:

- Consistency-based (or filtering) methods discard undesired readings by achieving a state of mutual local support that depends on the notion of inconsistency between values. For weighted constraints, inconsistency cannot easily be defined in a way that the ambiguity is effectively reduced and only suboptimal values are discarded. Therefore, it must be concluded that consistency-based methods are of limited use to the WCDG parsing problem although they have proven successful for CDGs with hard constraints only [Maruyama 1990b; Harper et al. 1999a].
- Tree search successively extends partial solutions and systematically explores the entire search space. The methods are complete and sound but require exponential runtime in the worst case so that some kind of pruning has to be applied which of course renders the algorithm incomplete. In practical cases, tree search has been found to be superior in small and medium-sized problems while unsatisfactory in some cases of very large problems because no solution at all is found in the time available.
- Heuristic search immediately starts with a suboptimal solution candidate and iteratively moves on to promising candidates which can be derived from the current candidate by means of simple transformations. The search space is explored unsystematically, i.e., no records about visited regions are kept, and hence an external termination criterion must be used. The accuracy of the returned solutions is high and stays high even for very large problems.

Various claims about the suitability of WCDGs for handling gradation in natural language have been experimentally verified.

- **Claim:** The use of weighted constraints leads to a robust mode of parsing.

  Result: To corroborate the claim, an experiment has been conducted where both the grammar support and the degree of deviation in the input have been systematically controlled. WCDGs have proven to be able to deal with contradictions between the grammar and the input in a fail-soft way, i.e., although the accuracy decreases with an increasing number of inconsistencies it does so in a graceful manner. Inclusion of redundant information from different knowledge sources has been identified as a second means (besides weighted and hence violable constraints) for robust behavior. The interaction between levels is mutually beneficial in this case.

- **Claim:** WCDG can be successfully deployed in applications for natural language learning.

  Result: Applications for natural language learning require a diagnosis component that robustly analyzes utterances and determines the location and the type of faults. WCDG parsing fulfills both requirements: It not only yields analyses even in cases of highly distorted input but also identifies conflicts, i.e., constraint violations at specific positions, which have been used to generate user feedback in a prototypical application.

- **Claim:** WCDGs allow for anytime parsing algorithms.

  Result: The robust behavior of WCDG parsing has been identified as a prerequisite for anytime behavior because it provides the procedure with the necessary room to react to temporal pressure. Two kinds of anytime behavior have been distinguished: intrinsic anytime behavior which refers to inherent characteristics of an algorithm and extrinsic anytime behavior which refers to the availability of parameters in the basic algorithm that allow an external component to control the behavior of the underlying procedure. While the systematic tree search method fails to meet some requirements for the former form and is only adequate for the latter, heuristic search has been found to be suitable for both kinds. These results have not only been found to be valid theoretically, but have also been confirmed in experiments.

- **Claim:** Constraints in WCDGs are flexible enough to integrate external judgments into the parsing process.

  Result: Graded information from a speech recognizer on the one hand and from a part-of-speech tagger on the other hand has been integrated into the WCDG parsing component by means by parameterized constraints. WCDG parsing benefits from the integration of acoustic scores when confronted with large word graphs but nevertheless fails to improve on results with the acoustically preferred (and thus possibly erroneous) path of the word graph. A soft integration of tagging information allows for more accurate parsing runs but no significant differences in efficiency have been found. The multi tag tagger yields even better results than the single tag tagger.

- **Claim:** Constraint weights for WCDGs can be estimated from annotated corpora.

  Result: The scores of a small constraint cluster that handles a single linguistic phenomenon, namely extraposition of German relative clauses, have been approximated from corpus data using the least-squares method. Nearly 90% of the test cases have been correctly classified using a decision procedure based on these constraints. Although this does not directly influence parsing accuracy (since extraposed clauses obviously cannot be confused with embedded ones for a given surface form), it can be assumed that such preferences can indirectly influence the quality of the parser. In a second experiment using genetic algorithms the weights of a complete WCDG have been (a) optimized starting from a manual assignment and (b) learned from scratch. In the first case a considerable improvement has been found and in the second case the final grammar has been as good as a highly-tuned manual grammar.

6.2 Further research

Not every detail of WCDG has been covered in this thesis and new open questions arise as known problems are tackled. This section is a collection of further lines of research that either deal with aspects of WCDG that have been identified as being problematic in this thesis or open up views on new applications of the WCDG parsing approach.

- The anytime property (cf. Sections 4.11 and 4.4.3) in natural language applications is particularly interesting when investigated for an incremental parsing model (cf. Section 4.12) since an external temporal pressure can best be justified when the speed of the parsing component falls behind the rate of incoming language data. Thus, coupling the two aspects such that a meta-control mechanism switches to a faster parsing mode when it detects such time critical situations, but stays with the most thorough strategy otherwise, is especially interesting. Such a coupled feedback device is compatible with the behavior of humans who easily switch between skimming and thoroughly studying a text. Would it be possible to really react to time pressure (at the cost of decreased accuracy)? Can such a parsing component be switched back and forth between fast and slow mode by means of a simple parameter? Is it possible to achieve considerable time savings with only a graceful degradation of the quality?

- WCDGs aim at dependency trees for a number of representational levels as results of the parsing process. These non-standard structures are the main reason why the development, the evaluation and most experiments have been conducted on a relatively small corpus that has been constructed especially for this purpose. Nevertheless, an application of the WCDG techniques to and an evaluation on standard corpora such as the Wall Street Journal [Marcus, Santorini & Marcinkiewicz 1993] or the NEGRA corpus [Skut et al. 1997a] seems highly desirable. Is it possible to not only derive syntactic dependency structures from the phrase structure annotations in these corpora but also dependency structures for other aspects such as thematic roles, at least for subsets of the corpora? How can a large coverage WCDG be built for these corpora? What results can be achieved when lexical information is ignored and a (possibly ambiguous) sequence of category symbols is used for parsing?

- An issue related to the application of WCDGs to very large corpora is the treatment of unknown words. No attempt has been made so far to deal with word forms that do not occur in the lexicon, but eventually this task will have to be tackled. A straight-forward approach can simply use an underspecified lexical item each time it encounters a new word. More sophisticated schemes can 'learn' from the discourse and build successively richer information structures for formerly unknown items.

- WCDG parsing degenerates to partial parsing when no connected dependency tree can be found (cf. Section 3.5.4). Applications such as question-answering or information extraction which are interested only in specific parts of a text or utterance can exploit this behavior by using a grammar and a lexicon which quickly parse the irrelevant parts into a shallow structure while the interesting segments can be analyzed completely and in full depth. Research into these kinds of strategies for variable-depth analysis is undertaken in the PAPA[1] project.

[1]For further information see `http://nats-www.informatik.uni-hamburg.de/~papa/`.

- A semantic modeling in WCDG suffers from the limitation to models which allow only relations between word forms (cf. Section 3.4.6). It seems worthwhile, however, to couple the WCDG parsing system with a knowledge-based inference system into which the more complex resolutions and discourse representations can be transferred [Lahres 1997].
- The WCDG paradigm is flexible enough to integrate external assessments by means of special-purpose constraints. Information fusion has been demonstrated for acoustic scores in word graphs (cf. Section 5.6) and for probabilities of part-of-speech tags (cf. Section 5.7). Both types have opened up follow-up questions.
  - Although the parser benefits from taking the acoustic scores into account, the overall success of parsing word graphs is limited because higher accuracies can be achieved by simply parsing the acoustically preferred path. This result should be compared to the findings of Harper et al. [1999a] who managed to achieve a higher sentence accuracy in word graphs that have been filtered using a CDG with hard constraints only.
  - The largest improvements in efficiency have been achieved using a hard integration of tagger information. The accuracy, however, dropped dramatically in these cases, which can be attributed to the relatively low accuracy of the tagger on the corpus used. An interesting question is whether this relation changes when a more accurate tagger (or tag mapping) is utilized.

  A further interesting candidate for information fusion is the inclusion of phrase boundary information from a finite-state 'chunker' [Federici, Montemagni & Pirelli 1996] or from a prosody component [Niemann et al. 1998] which could help the WCDG parser to overcome the inherent efficiency problems with non-projective structures (cf. Section 2.8.2).
- The restriction to binary constraints (cf. Section 2.6) has been motivated by computational needs but on the other hand has been identified as a severe limitation of the expressiveness of the formalism. While this restriction is more important for consistency-based methods (cf. Section 4.6) because they have to build a data structure of size $O(n^{2^a})$ where n is the number of words and a the maximum arity of constraints, it is less critical for tree search and heuristic search which only need to *apply* constraints $O(n^{2^a})$ times. Therefore it seems feasible to allow ternary constraints or even higher order constraints, especially when used sparsely and when combined with constraint indexing techniques (cf. Section 4.13.3).
- The solution method of systematic tree search sometimes shows unpredictable runtime requirements for WCDG parsing (cf. Section 5.5). Research in randomization techniques for systematic search [Gomes, Selman & Kautz 1998] indicates that this disadvantage can be overcome by introducing random elements into the search procedure (without sacrificing the soundness). This hypothesis should be experimentally verified.

6.3 Final remarks

WCDGs have been found to offer solutions for a series of problems in natural language processing such as robustness, integration of preferences, information fusion and explanatory clarity. So far WCDGs have been studied in an academic context but realistic applications begin to appear on the horizon, most probably in the area of computer assisted language learning where both the domain and the coverage of the grammar can be precisely controlled.

Bibliography

Emile H. L. Aarts, Jan H. M. Korst & Peter J. M. van Laarhoven, 1997. “Simulated Annealing”. Chapter 4, pages 91–120. In Aarts & Lenstra [1997b].

Emile H. L. Aarts & Jan Karel Lenstra, 1997a. “Introduction”. Chapter 1, pages 1–17. In Aarts & Lenstra [1997b].

Emile H. L. Aarts & Jan Karel Lenstra, ed.. 1997b. *Local Search in Combinatorial Optimization.* Series in Discrete Mathematics and Optimization. Chichester, New York, Weinheim, Brisbane, Singapore, Toronto: John Wiley & Sons.

Steven Abney. 1990. Rapid incremental parsing with repair. In *Proceedings of the 6th New OED Conference: Electronic Text Research*, pages 1–9, Waterloo, Ontario.

Steven Abney. 1996a. Partial parsing via finite-state cascades. In John Caroll, ed., *Proceedings of the Workshop on Robust Parsing at the Eighth Summer School in Logic, Language and Information (ESSLLI-96)*, pages 8–15.

Steven Abney, 1996b. “Statistical Methods and Linguistics”. Chapter 1. In Klavans & Resnik [1996].

Steven Abney, 1997. “Part-of-Speech Tagging and Partial Parsing”. Chapter 4, pages 118–136. Text, Speech and Language Technology, volume 2, edited by Steve Young and Gerrit Bloothooft. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Hiyan Alshawi & David Carter. 1994. Training and scaling preference functions for disambiguation. *Computational Linguistics*, 20(4):635–648.

Jan W. Amtrup. 1999. *Incremental Speech Translation.* Lecture Notes in Artificial Intelligence, number 1735. Berlin, Heidelberg, New York: Springer Verlag.

Jan. W. Amtrup. 2000. Hypergraph unification-based parsing for incremental speech processing. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT-2000)*, pages 291–292, Trento, Italy.

Jan W. Amtrup, Henrik Heine & Uwe Jost. 1997. What’s in a word graph - evaluation and enhancement of word lattices. In *Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH-97)*, Rhodes, Greece.

Jan. W. Amtrup & Volker Weber. 1998. Time mapping with hypergraphs. In *Proceedings of the Joint Conference COLING/ACL ’98*, pages 55–61, Montréal, Canada.

Srinivas Bangalore & Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

G. Edward Barton, Robert C. Berwick & Eric Sven Ristad. 1987. *Computational Complexity and Natural Language.* Cambridge, MA, USA: MIT Press.

Philippe Blache. 2000a. Constraints, linguistic theories and natural language processing. In D. Christodoulakis, ed., *Natural Language Processing*, Lecture Notes in Artificial Intelligence, number 1835. Springer, Heidelberg.

Philippe Blache. 2000b. Property grammars: a solution for parsing with constraints. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT-2000)*, pages 295–296, Trento, Italy.

Philippe Blache. 2000c. Property grammars and the problem of constraint satisfaction. In *Proceedings of the ESSLLI-2000 workshop on Linguistic Theory and Grammar Implementation.*

Philippe Blache & Frank Morawietz. 2000. A non-generative constraint-based formalism. Technical report, Université de Provence, Laboratoire Parole et Langage.

Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer & Salim Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Workshop on Speech and Natural Language*, New York.

Rens Bod. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* Ph.D. thesis, University of Amsterdam.

Rens Bod. 1998. Spoken dialogue interpretation with the DOP model. In *Proceedings of the 17th Intl. Conf. on Computational Linguistics and 36th Annual Meeting of the ACL (COLING/ACL-98)*, pages 138–144.

Mark Boddy & Thomas L. Dean. 1994. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence Journal*, 67:245–286.

Paul Boersma & Bruce Hayes. 2000. Empirical tests of the gradual learning algorithm. *Linguistic Inquiry*, 32(1):45–86.

Thorsten Brants. 2000. TnT - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA, USA.

Torsten Brants, Wojciech Skut & Hans Uszkoreit. 1999. Syntactic annotation of a German newspaper corpus. In A. Abeillé, ed., *Corpus annotés pour la syntaxe — Treebanks*, ATALA, Paris.

Eric Brill. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the the 31st Annual Meeting of the ACL.*

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Eric Brill. 1996. Transformation-based error-driven parsing. In Harry Bunt & Masara Tomita, ed., *Recent Advances in Parsing Technology.* Kluwer Academic Publishers, Dordrecht.

Eric Brill. 1997. Unsupervised learning of disambiguation rules for part of speech tagging. In *Natural Language Processing Using Very Large Corpora.* Kluwer Academic Publishers.

Norbert Bröker. 1998. Separating surface order and syntactic relations in a dependency grammar. In *Proceedings of the 17th Intl. Conf. on Computational Linguistics and 36th Annual Meeting of the ACL (COLING/ACL-98)*, pages 174–180, Montréal, Canada.

Norbert Bröker, Udo Hahn & Susanne Schacht. 1993. Ein Plädoyer für Performanzgrammatiken. In *Proc. der 4. Fachtagung der Deutschen Gesellschaft für Sprachwissenschaft, Sektion Computerlinguistik, "Deklarative und prozedurale Aspekte der Sprachverarbeitung"*, pages 6–11. DGfS/CL. Hamburg, Germany.

Norbert Bröker, Udo Hahn & Susanne Schacht. 1994. Concurrent lexicalized dependency parsing: the ParseTalk model. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, volume 1, pages 379–385. Kyoto, Japan.

Norbert Bröker, Susanne Schacht, P. Neuhaus & Udo Hahn. 1996. Performanzorientiertes Parsing und Grammatik-Design: das ParseTalk-System. In C. Habel, S. Kanngießer & G. Rickheit, ed., *Perspektiven der Kognitiven Linguistik. Modelle und Methoden.* Opladen: Westdeutscher Verlag, pages 79–126.

Norbert Bröker, Michael Strube, Susanne Schacht & Udo Hahn. 1997. Coarse-grained parallelism in natural language understanding: Parsing as message passing. In D. B. Jones & H. L. Somers, ed., *New Methods in Language Processing.* London, UCL Press, pages 301–318.

Hadumod Bußmann. 1990. *Lexikon der Sprachwissenschaft.* Stuttgart: Alfred Kröner Verlag.

Jean Carletta. 1992. *Risk-taking and Recovery in Task-Oriented Dialogue.* Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh.

Jean Carletta, Richard Caley & Stephen Isard. 1993. A system architecture for simulating time-constrained language production. HCRC/RP 43, HCRC, University of Edinburgh.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures.* Cambridge University Press.

Bob Carpenter & Gerald Penn, 1994. *The Attribute Logic Engine User's Guide Version 2.0.* Laboratory for Computational Linguistics, Carnegie Mellon University, Pittsburgh, PA.

Glenn Carroll & Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the Fall Symposium on Probabilistic Approaches to Natural Language, American Association for Artificial Intelligence (AAAI).*

Lewis Carroll. 1872. *Through the Looking-Glass and What Alice Found There.* London: Macmillan & Co.

Jean-Piere Chanod, 2001. "Robust Parsing and Beyond". Chapter 8, pages 187–204. In Junqua & van Noord [2001].

J.-C. Chappelier, M. Rajman, R. Aragües & A. Rozenknop. 1999. Lattice parsing for speech recognition. In *Proceedings of the 6me conférence sur le Traitement Automatique du Langage Naturel (TALN'99)*, pages 95–104, Cargèse, France.

Eugene Charniak. 1993. *Statistical Language Learning.* Cambridge, MA: The MIT Press.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, Seattle, WA.

Eugene Charniak, Glenn Carroll, John Adcock, Anthony Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael Littman & John McCann. 1996. Taggers for parsers. *Artificial Intelligence*, 85(1-2):45–57.

Ciprian Chelba & Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the Joint Conference COLING/ACL '98*, Montréal, Canada.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions PGIT*, 2:113–124.

Noam Chomsky. 1957. *Syntactic Structures.* The Hague: Mouton.

Noam Chomsky, 1964. "Degrees of grammaticalness". pages 384–389. In Fodor & Katz [1964].

Noam Chomsky. 1965. *Aspects of the Theory of Syntax.* Cambridge, MA, USA: The MIT Press.

Noam Chomsky. 1975. *The Logical Structure of Linguistic Theory.* New York, NY, USA: Plenum Press.

Noam Chomsky. 1981. *Lectures on Government and Binding.* Dordrecht, The Netherlands: Foris.

William F. Clocksin & Christopher S. Mellish. 1987. *Programming in Prolog.* 3rd edition. Heidelberg: Springer-Verlag.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, University of Pennsylvania.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 27–38, Santa Cruz, CA, USA.

Michael John Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, USA.

Jacques Courtin & Daminen Genthial. 1998. Parsing with dependency relations and robust parsing. In Sylvain Kahane & Alain Polguère, ed., *Proceedings of the Joint Conference COLING/ACL Workshop: Processing of Dependency-based Grammars*, Montréal, Canada.

Michael A. Covington. 1990. A dependency parser for variable-word-order languages. Technical Report AI-1990-01, Artificial Intelligence Center, University of Georgia, Athens, GA, USA.

A. P. Cowie, ed.. 1989. *Oxford Advanced Learner's Dictionary of Current English.* Oxford, UK: Oxford University Press.

Doug Cutting, Julian Kupiec, Jan Pedersen & Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Trento, Italy.

Walter Daelemans, Jakub Zavrel, Peter Berck & Steven Gillis. 1996. MBT: A memory-based part of speech tagger-generator. In Eva Ejerhed & Ido Dagan, ed., *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27.

Randall Davis. 1994. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence Journal*, 24(1):347–410.

Simon de Givry & Gérald Verfaillie. 1997. Optimum anytime bounding for constraint optimization problems. In *Proceedings of the AAAI-97 Workshop on Building Resource-Bounded Reasoning Systems*, pages 37–42, Providence, RI, USA.

Thomas Dean & Mark Boddy. 1988. An analysis of time-dependent planning. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88)*, volume 1, pages 49–54, Saint Paul, MN, USA.

Rina Dechter & Avi Dechter. 1988. Belief maintenance in dynamic constraint networks. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88)*, volume 1, pages 37–42, Saint Paul, MN, USA.

Steven J. DeRose. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39.

Wolfgang Domschke, Robert Klein & Armin Scholl. 1996. Taktische Tabus. *c't Magazin für Computertechnik*, 12(1):326.

Toby Donaldson & Robin Cohen. 1997. Computer-generated dialogue as a resource-bounded activity. In *Proceedings of the AAAI-97 Workshop on Building Resource-Bounded Reasoning Systems*, pages 101–103, Providence, RI, USA.

Denys Duchier. 1999a. Axiomatizing dependency parsing using set constraints. In *Proceedings of the 6th Meeting on Mathematics of Language*, pages 115–126, Orlando, FL, USA.

Denys Duchier. 1999b. Set constraints in computational linguistics – solving tree descriptions. In *Proceedings of the Workshop on Declarative Programming with Sets (DPS-99)*.

Denys Duchier. 2001. Lexicalized syntax and topology for non-projective dependency grammar. In *Joint Conference on Formal Grammars and Mathematics of Language (FGMOL-01)*, Helsinki, Finland.

Denys Duchier & Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear order. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics (ACL-2001)*, pages 180–187, Toulouse, France.

Denys Duchier, Claire Gardent & Joachim Niehren, 1999. *Concurrent Constraint Programming in Oz for Natural Language Processing.* Computer Science Department, University of Saarbrücken, Saarbrücken, Germany.

Denys Duchier & Joachim Niehren. 2000. Dominance constraints with set operators. In *Proceedings of the First International Conference on Computational Logic (CL-2000)*, LNCS. Springer.

Denys Duchier & Stefan Thater. 1999. Parsing with tree descriptions: a constraint-based approach. In *Proceedings of the 6th International Workshop on Natural Language Understanding and Logic Programming (NLULP-99)*, pages 17–32, Las Cruces, NM, USA.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13:94–102.

Ute Ehrlich & Gerhard Hanrieder. 1996. Robust speech parsing. In John Caroll, ed., *Proceedings of the Workshop on Robust Parsing at the Eighth Summer School in Logic, Language and Information (ESSLLI-96)*, pages 26–34.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, Denmark.

Bernd Eppinger & Eberhard Herter. 1993. *Sprachverarbeitung.* München, Wien: Carl Hanser Verlag.

Gregor Erbach. 1993. Towards a theory of degrees of grammaticality. Bericht 34, Computerlinguistik, Universität Saarbrücken.

Gisbert Fanselow & Christiane Féry, forthcominga. "Grammatik mit Widersprüchen". In Fanselow & Féry [forthcomingb].

Gisbert Fanselow & Christiane Féry, ed.. forthcomingb. *Sonderheft Kognitionswissenschaft: Konfligierende Regeln: Wege zu formalen Modellen der Kognition.*

Helene Fargier & Jerome Lang. 1993. Uncertainty in constraint satisfaction problems: a probalistic approach. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU-93)*, pages 97–104.

Stefano Federici, Simonetta Montemagni & Vito Pirelli. 1996. Shallow parsing and text chunking: A view on underspecification in syntax. In John Caroll, ed., *Proceedings of the Workshop on Robust Parsing at the Eighth Summer School in Logic, Language and Information (ESSLLI-96)*, pages 35–44.

Dietmar Fünning. 2001. Backmarking – Ein Suchverfahren für Constraint Satisfaction Probleme. Studienarbeit, Fachbereich Informatik, Universität Hamburg, Germany.

Jerry A. Fodor, 1987. "Modules, Frames, Fridgeons, Sleeping Dogs, and the Music of the Spheres". pages 25–36. In Garfield [1987].

Jerry A. Fodor & Jerrold J. Katz, ed.. 1964. *The Structure of Language: Readings in the Philosophy of Language.* Englewood Cliffs, NJ, USA: Prentice-Hall.

Sandiway Fong. 2000. The PAPPI system: Lexical semantics and morpho-syntax. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, China. Demonstration Abstract.

Kenneth I. Forster, 1987. "Binding, Plausibility, and Modularity". pages 63–82. In Garfield [1987].

Kilian Foth. 1998a. CDG Hacker's Guide. Memo HH-14/98, Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Kilian Foth. 1998b. Disjunktive Lexikoninformation im eliminativen Parsing. Studienarbeit, FB Informatik, Universität Hamburg.

Kilian Foth. 1999. Transformationsbasiertes Constraint-Parsing. Diplomarbeit, Fachbereich Informatik, Universität Hamburg.

Kilian Foth, Stefan Hamerich, Ingo Schröder & Michael Schulz. 1999. [X]CDG Benutzerhandbuch – Version 1.1. Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Kilian Foth, Wolfgang Menzel, Horia F. Pop & Ingo Schröder. 2000. An experiment on incremental analysis using robust parsing techniques. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 1026–1030, Saarbrücken, Germany.

Kilian Foth, Wolfgang Menzel & Ingo Schröder. 2000. A transformation-based parsing technique with anytime property. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT-2000)*, pages 89–100, Trento, Italy.

Kilian Foth, Wolfgang Menzel & Ingo Schröder. submitted. Robust parsing with weighted constraints. *Natural Language Engineering: Special Issue on Robust Methods in Analysis of Natural Language Data.*

Kilian Foth, Ingo Schröder & Michael Schulz. 1998. [X]CDG Benutzerhandbuch. Memo HH-13/98, Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Lyn Frazier, 1987. "Theories of Sentence Processing". pages 291–307. In Garfield [1987].

Eugene C. Freuder. 1982. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32.

Eugene C. Freuder. 1985. A sufficient condition for backtrack-bounded search. *Journal of the ACM*, 32(4):755–761.

Eugene C. Freuder, 1988. "Backtrack-free and Backtrack-bounded Search". Chapter 10, pages 343–369. In Kanal & Kumar [1988].

Eugene C. Freuder & Alan K. Mackworth, ed.. 1994. *Constraint-based Reasoning.* Cambridge, MA, USA: MIT Press.

Eugene C. Freuder & Richard J. Wallace. 1992. Partial constraint satisfaction. *Artificial Intelligence Journal*, 58:21–70. Reprinted [Freuder & Mackworth 1994].

H. Gaifman. 1965. Dependency systems and phrase-structure systems. *Information & Control*, (8):304–337.

Jean H. Gallier. 1987. *Logic for computer science.* New York: John Wiley & Sons, Inc.

Jay L. Garfield, ed.. 1987. *Modularity in Knowledge Representation and Natural-Language Understanding.* Cambridge, MA: MIT Press.

Gerald Gazdar & Chris Mellish. 1989. *Natural Language Processing in Prolog.* Wokingham et al.: Addison-Wesley.

Kim Gerdes & Sylvain Kahane. 2001. A formal dependency grammar using a topological hierarchy. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics (ACL-2001)*, pages 220–227, Toulouse, France.

Fred Glover & Manuel Laguna. 1997. *Tabu Search.* Boston: Kluwer Academic Publishers.

Fred Glover & Manuel Laguna. 2000. Tabu search. In P. M. Pardalos & M. G. C. Resende, ed., *Handbook of Applied Optimization.* Oxford Academic Press, Chapter X.

Sebastian Goeser. 1992. Chart parsing of robust grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING '92)*, pages 120–126, Nantes, France.

Carla P. Gomes, Bart Selman & Henry Kautz. 1998. Boosting combinatorial search through randomization. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, USA.

Jens Gottlieb. 2001. Evolutionäre Algorithmen für Constraint-Optimierungsprobleme. *Künstliche Intelligenz*, 1:69–71.

Günther Grewendorf, Fritz Hamm & Wolfgang Sternefeld. 1987. *Sprachliches Wissen. Eine Einführung in die Theorie der grammatischen Beschreibung.* Suhrkamp Taschenbuch Wissenschaft, volume 695. Frankfurt/Main: Suhrkamp.

Günther Görz & Gerhard Hanrieder. 1995. Robust parsing of spoken dialogue using contextual knowledge and recognition probabilities. ESCA Tutorial and Research Workshop on Spoken Dialogue Systems – Theories and Applications.

Günther Görz & Markus Kessler. 1994a. Anytime algorithm for speech parsing? In *Proceedings of the 15th Conference on Computational Linguistics (COLING-94)*, Kyoto, Japan.

Günther Görz & Markus Kessler. 1994b. An anytime protocol for speech parsing. In *Proceedings of the FORWISS/CRIM Workshop: Progress and Prospects of Speech Research and Technology*, volume 1, pages 257–261, St. Augustin, Germany. infix.

Liliane Haegeman. 1991. *Introduction to Government and Binding Theory.* Cambridge, MA, USA: Blackwell, Inc.

Jochen Hagenström. 2001. Kompilation von Constraint-Grammatiken. Studienarbeit, Fachbereich Informatik, Universität Hamburg, Germany.

Jochen Hagenström. forthcoming. Integration und Evaluation eines POS-Taggers im CDG Parsing-System. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, Germany.

R. M. Haralicj & G. L. Elliot. 1980. Increasing tree search efficiency for constraint satisfcation problems. *Artificial Intelligence Journal*, 14:263–313.

Karin Harbusch. 1997. The relation between tree-adjoining grammars and constraint dependency grammars. In *Proceedings of Fifth Meeting on the Mathematics of Language (MOL5)*, pages 38–45, Schloss Dagstuhl, Wadern near Saarbrücken, Germany. DFKI Document No. 97-02.

Mary P. Harper & Randall A. Helzerman. 1994. Managing multiple knowledge sources in constraint-based parsing of spoken language. Technical Report EE 94-16, School of Electrical Engineering, Purdue University, West Lafayette, IN, USA.

Mary P. Harper & Randall A. Helzerman. 1995. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9(3):187–234.

Mary P. Harper, Randall A. Helzermann, C. B. Zoltowski, B. L. Yeo, Y. Chan, T. Steward & B. L. Pellom. 1995. Implementation issues in the development of the PARSEC parser. *Software – Practice and Experience*, 25(8):831–862.

Mary P. Harper, S. A. Hockema & C. M. White. 1999. Enhanced constraint dependency grammar parsers. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*, Honolulu, HI, USA.

Mary P. Harper, L. H. Jamieson, C. D. Mitchell, G. Ying, S. Potisuk, P. N. Srinivasan, R. Chen, C. B. Zoltowski, L. L. McPheters, B. Pellom & R. A. Helzerman. 1994. Integrating language models with speech recognition. In *Proceedings of the AAAI-94 Workshop on the Integration of Natural Language and Speech Processing*, pages 139–146.

Mary P. Harper, Leah H. Jamieson, Carla B. Zoltowski & Randall A. Helzerman. 1993. Semantics and constraint parsing of word graphs. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 63–66, Minneapolis, MN, USA.

Mary P. Harper, M. T. Johnson, L. H. Jamieson, S. A. Hockema & C. M. White. 1999a. Interfacing a CDG parser with an HMM word recognizer using word graphs. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix.

Mary P. Harper, C. M. White, R. A. Helzerman & S. A. Hockema. 1999b. Faster MUSE CSP arc consistency algorithms. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*, Honolulu, Hawai.

Mary P. Harper, C. M. White, Wen Wang, Michael T. Johnson & Randall A. Helzerman. 2000. The effectiveness of corpus-induced dependency grammars for post-processing speech. In *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pages 102–109, Seattle, WA, USA.

Andreas Hauenstein & Hans Weber. 1994. An investigation of tightly coupled time synchronous speech language interfaces using a unification grammar. In *Proceedings of the 12th National Conference on Artificial Intelligence: Workshop on the Integration of Natural Language and Speech Processing*, pages 42–49, Seattle, Washington.

John A. Hawkins. 1994. *A Performance Theory of Order and Constituency.* Cambridge Studies in Linguistics. Cambridge: Cambridge University Press.

Peter A. Heeman, 2001. “Improving Robustness by Modeling Spontaneous Speech Events”. Chapter 5, pages 123–152. In Junqua & van Noord [2001].

Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel & Ingo Schröder. 1998. Eliminative parsing with graded constraints. In *Proceedings of the Joint Conference COLING/ACL-98*, pages 526–530, Montréal, Canada.

Johannes Heinecke & Andreas Nolda. 1998. Dependenzbasiertes Parsing des Deutschen. Abdeckungsbereich und Analyse. Memo B-1/98, Projekt DAWAI, Humboldt-Universität zu Berlin.

Johannes Heinecke & Ingo Schröder. 1998. Multilevel representation of the robust analysis of language. In Bernhard Schröder, Winfried Lenders, Wolfgang Hess & Thomas Portele, ed., *Proceedings of the 4th Conference on Natural Language Processing (KONVENS-98)*, Bonn, Germany.

Peter Hellwig. 1988. Chart parsing according to the slot and filler principle. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, pages 242–244, Budapest, Hungary.

Randall A. Helzerman & Mary P. Harper. 1992. Log time parsing on the MasPar MP-1. In *Proceedings of the 21st International Conference on Parallel Processing*, volume 2, pages 209–217, St. Charles, IL.

Randall A. Helzerman & Mary P. Harper. 1996. MUSE CSP: An extension to the constraint satisfaction problem. *Journal of Artificial Intelligence Research*, 5:239–288.

Alain Hertz, Eric Taillard & Dominique de Werra, 1997. “Tabu search”. Chapter 4, pages 121–136. In Aarts & Lenstra [1997b].

Joachim Hertzberg. 1995. KI-Lexikon: Anytime-Algorithmen. *Künstliche Intelligenz*, 1:40.

Michael Hess. 1996. Computerlinguistik und Logikprogrammierung. *Künstliche Intelligenz*, 3:40–45.

Harald Hesse & Andreas Küstner. 1985. *Syntax der koordinativen Verknüpfung.* Studia Grammatica, number XXIV. Berlin: Akademie Verlag.

Donald Hindle & Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19:103–120.

John H. Holland. 1975. *Adaptation in natural and artificial systems.* The University of Michigan Press.

John E. Hopcroft & Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages, and computation.* Reading, MA: Addison-Wesley.

Eric Horvitz. 1987. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of AAAI-87 Workshop on Uncertainty in Artificial Intelligence.*

Juraj Hromkovič. 1998. *Algorithmics for Hard Problems.* Texts in Theoretical Computer Science. Springer.

Richard Hudson. 2000. Discontinuity. *International Journal Traitement automatique des langues: Les grammaires de dépendance*, 41(1):15–56.

Richard A. Hudson. 1990. *English Word Grammar.* Oxford: Blackwell.

Willem Olaf Huijsen. 1993. Genetic grammatical inference. In *Proceedings of the fourth CLIN meeting*, Groningen, The Netherlands.

Anthony Jameson, Ralph Schäfer, Thomas Weis & André Berthold Thomas Weyrath. 1999. Making systems sensitive to the user’s time and working memory constraints. In M. T. Maybury, ed., *Proceedings of the International Conference on Itelligent User Interfaces (IUI-99)*, New York, NY, USA. ACM.

Fred Jelinek. 1990. Self-organizing language models for speech recognition. In Alex Waibel & Kai-Fu Lee, ed., *Readings in Speech Recognition.* Morgan Kaufmann, San Mateo, CA, pages 450–506.

Michael T. Johnson. 2000. *Incorporating Prosodic Information and Language Structure into Speech Recognition Systems.* Ph.D. thesis, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA.

Aravind Joshi, Leon Levy & M. Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10.

Aravind Joshi & Bangalore Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the International Conference on Computational Linguistics (COLING-94)*, Kyoto, Japan.

Aravind Joshi, K. Vijay-Shanker & D. Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In Peter Sells, S. M. Shieber & T. Wasow, ed., *Foundational Issues in Natural Language Processing.* Bradford, Cambridge, MA, USA.

Aravind K. Joshi & Y. Schabes, 1997. "Tree-Adjoining Grammars". In G. Rozenberg & A. Salomaa, ed., *Handbook of Formal Languages.* Berlin, Germany: Springer.

Timo Järvinen & Pasi Tapanainen. 1997. A dependency parser for English. Technical Report TR-1, Department of General Linguistics, University of Helsinki, Finland.

Jean-Claude Junqua & Gertjan van Noord, ed.. 2001. *Robustness in Language and Speech Technology.* Text, Speech and Language Technology, volume 17. Dordrecht/Boston/London: Kluwer Academic Publishers.

Daniel Jurafsky, Chuck Wooters, Gary Tajchman, Jonathan Segal, Andreas Stolcke, Eric Fosler & Nelson Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of 20th International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 189–192, Detroit, MI, USA.

Sylvain Kahane, forthcoming. "The Meaning-Text Theory". In Vilmos Ágel, Ludwig M. Eichinger, Hans Werner Eroms, Peter Hellwig, Hans Jürgen Heringer & Henning Lobin, ed., *Dependency and Valency: A Handbook of Contemporary Research.* Berlin, Germany: De Gruyter.

Sylvain Kahane, Alexis Nasr & Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *Proceedings of the Joint Conference COLING/ACL '98*, pages 646–652, Montréal, Canada.

Hans Kamp & Uwe Reyle. 1993. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language.* Kluwer Academic Publishers.

Laveen Kanal & Vipin Kumar, ed.. 1988. *Search in Artificial Intelligence.* New York, NY, USA: Springer-Verlag.

Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, pages 168–173, Helsinki, Finland.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä & Arto Anttila, ed.. 1995. *Constraint Grammar – A Language-Independent System for Parsing Unrestricted Text.* Berlin, Germany: Mouton de Gruyter.

L. Karttunen, Tamás Gaál & André Kempe. 1997. Xerox finite-state tool.

Lauri Karttunen. 1993. Finite-state lexicon compiler. Technical Report ISTL-NLTT-1993-04-02, Xerox Palo Alto Research Center, Palo Alto, CA.

Lauri Karttunen & Kenneth R. Beesley. 1992. Two-level rule compiler. Technical Report ISTL-92-02, Xerox Palo Alto Research Center, Palo Alto, CA.

Lauri Karttunen, J.-P. Chanod, G. Grefenstette & A. Schiller. 1997. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):1–24.

Martin Kay. 1984. Functional unification grammar: A formalism for machine translation. In *Proceedings of the 22nd Annual Meeting of the ACL*, pages 75–78, Standford University, CA, USA.

Martin Kay. 1986. Algorithm schemata and data structures in syntactic processing. In Barbara J. Grosz, Karen Sparck Jones & Bonnie L. Webber, ed., *Readings in Natural Language Processing.* Morgan Kaufmann, Los Altos, CA, USA, pages 35–70.

Bill Keller & Rüdi Lutz. 1997. Learning stochastic context-free grammars from corpora using a genetic algorithm. Cognitive Science Research Paper 446, School of Cognitive and Computing Sciences, University of Sussex.

Frank Keller. 2000a. Evaluating competition-based models of word order. In Lila R. Gleitman & Aravid K. Joshi, ed., *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, pages 747–752, Mahawah, NJ, USA. Lawrence Erlbaum.

Frank Keller. 2000b. *Gradience in Grammar.* Ph.D. thesis, University of Edinburgh.

Frank Keller, 2001. "Experimental Evidence for Constraint Competition in Gapping Constructions". In Gereon Müller & Wolfgang Sternefeld, ed., *Competition in Syntax*, pages 211–248. Berlin, Germany: Mouton de Gruyter.

Frank Keller & Ash Asudeh. 2002. Probabilistic learning algorithms and Optimality Theory. *Linguistic Inquiry*, 33(2).

S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220:671–680.

Karl-Heinz Kiyek & Friedrich Schwarz. 1989. *Mathematik für Informatiker 1.* Stuttgart: B. G. Teubner.

Judith Klavans & Philip Resnik, ed.. 1996. *The Balancing Act: Combining Symbolic and Statistical Approaches to Language.* Cambridge, MA, USA: MIT Press.

Kimmo Koskenniemi. 1983. *Two-level Morphology: A General Computational Model for Word-form Recognition and Production.* Ph.D. thesis, Department of General Linguistics, University of Helsinki, Helsinki, Finland.

Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, pages 229–232, Helsinki, Finland.

Kimmo Koskenniemi, Pasi Tapanainen & Atro Voutilainen. 1992. Compiling and using finite-state syntactic rules. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 156–162, Nantes, France.

Brigitte Krenn & Christer Samuelsson. 1996. The linguist's guide to statistics. ftp://coli.uni-sb.de/pub/coli/doc/stat/stat_cl.ps.gz.

Vipin Kumar. 1992. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13(1):32–44.

Jürgen Kunze. 1972. *Die Auslaßbarkeit von Satzteilen bei koordinativen Verbindungen im Deutschen*. Schriften zur Phonetik, Sprachwissenschaft und Kommunikationsforschung, number 14. Berlin, Germany: Akademie Verlag.

Jürgen Kunze. 1975. *Abhängigkeitsgrammatik*. Studia Grammatica, number XII. Berlin, Germany: Akademie Verlag.

John Lafferty, Daniel Sleator & Davy Temperley. 1992. Grammatical trigrams: A probabilistic model of link grammar. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*. Also available as technical report CMU-CS-92-181.

Bernhard Lahres. 1997. A semantic level of description for constraint dependency grammars. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, Germany.

Marc M. Lankhorst. 1993. A genetic algorithm for the induction of context-free grammars. In *Proceedings of the fourth CLIN meeting*, Groningen, The Netherlands.

Vincenzo Lombardo. 1992. Incremental dependency parsing. In *Proceedings of the Annual Meeting of the ACL*, Delaware, Newark, USA.

Alan K. Mackworth. 1977. Consistency in networks of relations. *Artificial Intelligence Journal*, 8:99–118.

Alan K. Mackworth & Eugene C. Freuder. 1985. The complexity of some polynominal network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence Journal*, 25:65–74.

David M. Magerman. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL-95)*.

M. Marcus, B. Santorini & M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2).

Mitchell P. Marcus. 1987. Deterministic parsing and description theory. In Peter Whitelock, Mary McGee Wood, Harold Somers, Rod Johnson & Paul Bennett, ed., *Linguistic Theory and Computer Applications*. Academic Press, London, UK, pages 69–112.

William Marslen-Wilson & Lorraine Komisarjevsky Tyler, 1987. “Against Modularity”. pages 37–62. In Garfield [1987].

Hiroshi Maruyama. 1990a. Constraint dependency grammar. Technical Report RT0044, IBM Research, Tokyo Research Laboratory.

Hiroshi Maruyama. 1990b. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Annual Meeting of the Association of Computational Linguistics (ACL-90)*, pages 31–38, Pittsburgh, PA.

Hiroshi Maruyama, Hideo Watanabe & Shiho Ogino. 1990. An interactive Japanese parser for machine translation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, pages 257–262, Helsinki.

Beáta Megyesi. 2001. Comparing data-driven learning algorithms for PoS tagging of swedish. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2001)*, pages 151–158, Carnegie Mellon University, Pittsburgh, PA, USA.

Chris S. Mellish. 1989. Some chart-based techniques for parsing ill-formed input. In *Proceedings of the 27th Annual Meeting of the ACL*, pages 102–109, Vancouver, Canada.

Igor A. Mel’čuk. 1988. *Dependency Syntax: Theory and Practice.* SUNY Series in Linguistics. New York: State University of New York Press.

Wolfgang Menzel. 1988. Error diagnosing and selection in a training system for second language learning. In *Proceedings 12th International Conference on Computational Linguistics (COLING-88)*, pages 414–419, Budapest.

Wolfgang Menzel. 1990. Anticipation-free diagnosing of structural faults. In *Proceedings 13th International Conference on Computational Linguistics (COLING-90)*, pages 422–424, Helsinki.

Wolfgang Menzel. 1994. Parsing of spoken language under time constraints. In A. Cohn, ed., *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, pages 560–564, Amsterdam.

Wolfgang Menzel. 1995. Robust processing of natural language. In *Proceedings of the 19th German Annual Conference on Artificial Intelligence (KI-95)*, pages 19–34, Berlin.

Wolfgang Menzel. 2000. Theory and applications in computational linguistics – Is there common ground? In *Proceedings of the Symposium “Computerlinguistik: Divergenz oder Synergie”*, Heidelberg, Germany.

Wolfgang Menzel, Daniel Herron, Patrizia Bonaventura & Rachel Morton. 2000. Automatic detection and correction of non-native english pronunciation. In *Proceedings of the Workshop on Intergrating Speech Technology in the (Language) Learning and Assistive Interface*, pages 49–56, Dundee, UK.

Wolfgang Menzel, Daniel Herron, Rachel Morton, Dario Pezzotta, Patrizia Bonaventura & Peter Howarth. 2001. Interactive pronounciation training. *ReCALL*, 13(1):67–78.

Wolfgang Menzel & Ingo Schröder. 1998a. Constraint-based diagnosis for intelligent language tutoring systems. In José Cuena, ed., *Proceedings of the IT&KNOWS Conference at the IFIP ’98 Congress*, Wien/Budapest.

Wolfgang Menzel & Ingo Schröder. 1998b. Constraint-based diagnosis for intelligent language tutoring systems. Bericht FBI-HH-B-208/98, Fachbereich Informatik, Universität Hamburg.

Wolfgang Menzel & Ingo Schröder. 1998c. Decision procedures for dependency parsing using graded constraints. In Sylvain Kahane & Alain Polguère, ed., *Proceedings of the Joint Conference COLING/ACL-98 Workshop: Processing of Dependency-based Grammars*, pages 78–87, Montréal, Canada.

Wolfgang Menzel & Ingo Schröder. 1998d. Error diagnosis for language learning systems. In *Proceedings of the 2nd International Conference on Natural Language Processing and Industrial Applications (NLPIA-98)*, pages 45–51, Moncton, Canada.

Wolfgang Menzel & Ingo Schröder. 1998e. Error diagnosis in a language tutoring system. In *Proc. of "NLP in CALL"*, UMIST Manchester.

Wolfgang Menzel & Ingo Schröder. 1998f. Model-based diagnosis under structural uncertainty. In Henri Prade, ed., *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 284–288, Brighton, UK.

Wolfgang Menzel & Ingo Schröder. 1999. Error diagnosis in a language tutoring system. *Special ReCALL publication: Language Processing in CALL*, pages 20–30.

Pedro Meseguer. 1989. Constraint satisfaction problems: An overview. *AI Communications*, 2(1):3–17.

Zbigniew Michalewicz. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer Verlag. Third, Revised and Extended Version.

Ian Miguel & Qiang Shen. 2001a. Solution techniques for constraint satisfaction problems: Advanced approaches. *Artificial Intelligence Review*, 15:269–293.

Ian Miguel & Qiang Shen. 2001b. Solution techniques for constraint satisfaction problems: Foundations. *Artificial Intelligence Review*, 15:243–267.

Steven Minton, Mark D. Johnson, Andrew B. Philips & Philip Laird. 1992. Minmizing conflicts: A heuristic repair method for constraint-satisfcation and scheduling problems. *Artificial Intelligence Journal*, 58:161–205. Reprinted [Freuder & Mackworth 1994].

Leonid Mitjushin. 1992. High-probability syntactic links. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 930–934, Nantes, France.

Stefan Müller. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Linguistische Arbeiten, number 394. Tübingen, Germany: Max Niemeyer Verlag.

Roger Mohr & Thomas C. Henderson. 1986. Arc and path consistency revisited. *Artificial Intelligence Journal*, 28:225–233.

Bernard A. Nadel, 1988. "Tree Search and Arc Consistency in Constraint-Satisfaction Problems". Chapter 9, pages 287–342. In Kanal & Kumar [1988].

Peter Neuhaus & Norbert Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *Proceedings of the 35th Annual Meeting of the ACL and 8th Conference of the EACL*, pages 337–343, Madrid, Spain.

Grace Ngai & Radu Florian. 2001. Transformation based learning in the fast lane. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001).*

H. Niemann, E. Nöth, A. Batliner, J. Buckow, F. Gallwitz, R. Huber & V. Warnke. 1998. Using prosodic cues in spoken dialog systems. In Y. Kosarev, ed., *Proceedings of the International Workshop on Speech and Computer*, pages 17–28, St. Petersburg, Russia.

Andreas Nolda & Johannes Heinecke. 1999. Hierarchien, Lexikoneinträge, Analyse-Ebenen, Constraints. Memo B-1/99, Projekt DAWAI, Humboldt-Universität zu Berlin.

Elmar Nöth & B. Plannerer. 1993. Schnittstellendefinition für den Worthypothesengraphen. Verbmobil Memo 2, Universität Erlangen-Nürnberg.

Bernard Nudel. 1983. Consistent-labeling problems and their algorithms: Expected complexities and theory-based heuristics. *Artificial Intelligence Journal*, 21:135–178.

Douglas O'Shaughnessy. 1990. *Speech Communications.* Reading, MA: Addison-Wesley.

Fernando Pereira. 2000. Formal grammar and information theory: Together again. *Philosophical Transactions of the Royal Society*, pages 1239–1253.

Fernando C. N. Pereira & David H. D. Warren. 1980. Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence Journal*, 13:231–278.

Oliver Plaehn. 1999. Probabilistic parsing with discontinuous phrase structure grammar. Diplomarbeit, Computerlinguistik, Universität Saarbrücken.

Oliver Plaehn. 2000. Computing the most probable parse for a discontinuous phrase structure grammar. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT-2000)*, Trento, Italy.

Carl Pollard & Ivan Sag. 1987. *Information-Based Syntax and Semantics, Volume 1: Fundamentals.* Stanford University, CA, USA: Centre for the Study of Language and Information.

Carl Pollard & Ivan Sag. 1994. *Head-driven Phrase Structure Grammar.* Center for the Study of Language and Information Science. Chicago, IL, USA: The University of Chicago Press.

Alan Prince & Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Technical Report RuCCS #2, Rutgers University, Center for Cognitive Science, Piscataway, NJ.

Y. Qu, C. P. Rosé & B. Di Eugenio. 1996. Using discourse predictions for ambiguity resolution. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Copenhagen, Denmark.

M. O. Rabin. 1980. Probabilistic algorithm for primality testing. *Journal of Number Theory*, 12:128–138.

Lawrence R. Rabiner. 1990. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel & Kai-Fu Lee, ed., *Readings in Speech Recognition.* Morgan Kaufmann, San Mateo, CA, pages 267–290. See also Errata http://www.media.mit.edu/ rahimi/rabiner/rabiner-errata/.

Owen Rambow & Aravind K. Joshi. 1994. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, ed., *Current Issues in Meaning-Text Theory*. Pinter, London, UK.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, PA.

Adwait Ratnaparkhi. 1997a. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, USA.

Adwait Ratnaparkhi. 1997b. A simple introduction to maximum entropy models for natural language processing. Technical Report 97-08, Institute for Research in Cognitive Science.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

J. A. Robinson. 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41.

Carolyn Penstein Rosé & Alon Lavie. 1997. An efficient distribution of labor in a two stage robust interpretion process. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, USA. cmp-lg/9706021.

Carolyn Penstein Rosé & Alon Lavie, 2001. "Balancing Robustness and Efficiency in Unification-augmented Context-Free Parsers for Large Practical Applications". Chapter 10, pages 239–269. In Junqua & van Noord [2001].

Carolyn Penstein Rosé & Alex Waibel, 1996. "Recovering From Parser Failures: A Hybrid Statistical/Symbolic Approach". Chapter 8. In Klavans & Resnik [1996].

Stuart J. Russell & Eric Wefald. 1991. *Do the right thing: studies in limited rationality*. Cambridge, MA, USA: The MIT Press.

Stuart J. Russell & Shlomo Zilberstein. 1991. Composing real-time systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 212–217, Sydney, Australia.

Christer Samuelsson. 1997a. Extending n-gram tagging to word graphs. In Ruslan Mitkov, Nicolas Nicolov & Nikolai Nikolov, ed., *Proceedings of the Second Conference on Recent Advances in Natural Language Processing (RANLP-97)*, pages 21–26, Tzigov Chark, Bulgaria.

Christer Samuelsson. 1997b. A left-to-right tagger for word graphs. In *Proceedings of the 5th International Workshop on Parsing Technologies (IWPT-97)*, Boston, MA, USA.

Ralph Schäfer & Thomas Weyrath. 1996. Einschätzung von verfügbarer Arbeitsgedächtniskapazität mit temporalen Bayesschen Netzen. In *Proceedings of ABIS-96: Workshop on Adaptivity and User Modeling in Interative Software Systems*, Dortmund.

Thomas Schiex. 1992. Possibilitic constraint satisfaction problems or "how to handle soft constraints?". In *Proceedings of the 8th Conference of Uncertainty in Artificial Intelligence*, pages 269–275, Stanford, CA, USA.

Thomas Schiex, Hélène Fargier & Gérald Verfaillie. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of 12th International Conference on Artificial Intelligence (IJCAI-95)*, Montréal, Canada.

Anne Schiller, Simone Teufel, Christine Stöckert & Christine Thielen. 1995. Vorläufige Guidelines für das Tagging deutscher Textcorpora mit STTS. Draft of 14th November 1995.

Matthias Schlesewsky. forthcoming. Konfliktresolution beim Sprachverstehen. In Fanselow & Féry Fanselow & Féry [forthcomingb].

Helmut Schmid. 1994a. Part-of-speech tagging with neural networks. In *Proceedings of the International Conference on Computational Linguistics (COLING-94)*, pages 172–176, Kyoto, Japan.

Helmut Schmid. 1994b. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Wolf Schneider. 1999. Sprachkritik. *ZEIT*, 44.

Uwe Schöning. 1992a. *Logik für Informatiker.* 3. revised edition. Mannheim: BI Wissenschaftsverlag.

Uwe Schöning. 1992b. *Theoretische Informatik kurz gefaßt.* Mannheim: BI Wissenschaftsverlag.

Ingo Schröder. 1995. Analyse natürlicher Sprache durch Beschränkungserfüllung. Studienarbeit, Fachbereich Informatik, Universität Hamburg.

Ingo Schröder. 1996. Integration statistischer Methoden in eliminative Verfahren zur Analyse von natürlicher Sprache. Diplomarbeit, Fachbereich Informatik, Universität Hamburg.

Ingo Schröder. 1997a. Benutzerhandbuch des CDG-Parsers 0.8. Memo HH-4/97, Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Ingo Schröder. 1997b. Ergebnisse statistischen CDG-Parsings. Memo HH-6/97, Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Ingo Schröder. 1997c. Pruning-Strategien. Memo HH-8/97, Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Ingo Schröder. 1997d. Syntax der Eingaben in den CDG-Parser 0.8. Memo HH-3/97, Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Ingo Schröder. 1997e. Unsicherheit vs. keine negative Evidenz. Memo HH-5/97, Projekt DAWAI, Fachbereich Informatik, Universität Hamburg.

Ingo Schröder, Wolfgang Menzel, Kilian Foth & Michael Schulz. 2000. Modeling dependency grammar with restricted constraints. *International Journal Traitement automatique des langues: Les grammaires de dépendance*, 41(1):113–144.

Ingo Schröder, Horia F. Pop, Wolfgang Menzel & Kilian A. Foth. 2001a. Learning grammar weights using genetic algorithms. In *Proceedings of the Third Conference on Recent Advances in Natural Language Processing (RANLP-2001)*, Tzigov Chark, Bulgaria.

Ingo Schröder, Horia F. Pop, Wolfgang Menzel & Kilian A. Foth. 2001b. Learning weights for a natural language grammar using genetic algorithms. In *Proceedings of the International Conference on Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems (EUROGEN-2001)*, Athens, Greece.

Ernst Günther Schukat-Talamazzini. 1995. *Automatische Spracherkennung.* Braunschweig/Wiesbaden: Verlag Vieweg.

Michael Schulz. 1999. Inkrementelle Analyse natürlicher Sprache mit Constraints. Studienarbeit, Fachbereich Informatik, Universität Hamburg.

Michael Schulz. 2000. Parsen natürlicher Sprache mit gesteuerter lokaler Suche. Diplomarbeit, Fachbereich Informatik, Universität Hamburg.

Michael Schulz & Wolfgang Menzel. submitted. Parsing natural language using guided local search. In *Proceedings of the 15th European Conference on Artificail Intelligence (ECAI-2002)*, Lyon, France.

Camilla Schwind. 1988. Sensitive parsing: Error analysis and explanation in an intelligent language tutoring system. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 608–613, Budapest.

Camilla Schwind. 1990. An intelligent language tutoring system. *International Journal of Man-Machine Studies*, 33:557–579.

Camilla Schwind. 1995. Error analysis and explanation in knowledge based language tutoring. *Computer Assisted Language Learning*, 8(4):295–324.

John Self. 1992. Cognitive diagnosis for tutoring systems. In Bernd Neumann, ed., *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92)*, pages 699–703, Vienna, Austria. Wiley.

Peter Sells. 1987. *Lectures on Contemporary Syntactic Theories.* CSLI Lectures Notes Number 3.

Bart Selman, 1999. “Greedy Local Search”. In Wilson & Keil [1999].

Stuart M. Shieber. 1984. The design of a computer language for linguistic information. In *Proceedings of the 10th International Conference on Computational Linguistics (COLING-84)*, Standford, CA, USA.

Stuart M. Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lectures Notes Number 4.

Stuart M. Shieber. 1992. *Constrained-Based Grammar Formalisms.* Cambridge, MA, USA: Bradford.

Achim Sixtus & Stefan Ortmanns. 1999. High quality word graphs using forward-backward pruning. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-99)*, Phoenix, AZ, USA.

Wojciech Skut, Thorsten Brants, Brigitte Krenn & Hans Uszkoreit. 1997a. Annotating unrestricted German text. In *Proceedings of 6. Fachtagung der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft*, Heidelberg, Germany.

Wojciech Skut, Brigitte Krenn, Thorsten Brants & Hans Uszkoreit. 1997b. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, Washington, D. C., USA.

Daniel D. Sleator & Davy Temperley. 1993. Parsing English with a link grammar. In *Proceedings of the 3rd International Workshop on Parsing Technologies.*

Daniel D. K. Sleator & Davy Temperley. 1991. Parsing English with a link grammar. Technical Report CMU-CS-91-196, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Gert Smolka. 1995. The Oz programming model. In Jan van Leeuwen, ed., *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, pages 324–343.

Wolfgang Sternefeld. 1998. Suboptimale syntaktische Strukturen im Deutschen. Project Proposal, SFB 441.

Niels Stockfleth. 2000. Integration von CDG-Parsing und Fehlersimulation. Diplomarbeit, Fachbereich Informatik, Universität Hamburg.

Peter Struss. 1992. Knowledge-based diagnosis: An important challenge and touchstone for AI. In Bernd Neumann, ed., *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 863–874, Vienna, Austria.

Thomas Stützle & Holger H. Hoos. 2001. Ameisenalgorithmen zur Lösung kombinatorischer Optimierungsprobleme. *Künstliche Intelligenz*, 1:45–51.

ANS T1A1. 2000. American national standard for telecommunications - telecom glossary 2000. T1A1 Technical Subcommittee on Performance and Signal Processing.

Pasi Tapanainen & Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, Washington, DC, USA.

Bruce Tesar. 1995. *Computational Optimality Theory.* Ph.D. thesis, Department of Computer Science, Unversity of Colorado.

Bruce Tesar & Paul Smolensky. 1998. The learnability of optimality theory. *Linguistic Inquiry*, 29(2):229–268.

Lucien Tesnière. 1959. *Eléments de syntaxe structurale.* Paris: Klincksieck.

M. Thiel, 1987. "Weighted Parsing". In Leonard Bolc, ed., *Natural Language Parsing Systems*, pages 137–159. Heidelberg, Germany: Springer.

John C. Trueswell, Micheal K. Tanenhaus & Susan M. Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language*, 33:285–318.

Edward Tsang. 1993. *Foundations of Constraint Satisfaction.* London: Academic Press.

Hans Uszkoreit. 1987. *Word Order and Constituent Structure in German.* CSLI Lectures Notes Number 8.

Hans Uszkoreit. 1991. Strategies for adding control information to declarative grammars. Research Report RR-91-29, DFKI GmbH.

Hans Uszkoreit, Thorsten Brants, Denys Duchier, Brigitte Krenn, Lars Konieczny, Stephan Oepen & Wojciech Skut. 1998a. Studien zur performanzorientierten Linguisitk. Aspekte der Relativsatzextraposition im Deutschen. CLAUS Report 99, Universität des Saarlandes.

Hans Uszkoreit, Thorsten Brants, Denys Duchier, Brigitte Krenn, Lars Konieczny, Stephan Oepen & Wojciech Skut. 1998b. Studien zur performanzorientierten Linguisitk. Aspekte der Relativsatzextraposition im Deutschen. *Kognitionswissenschaft*, 7(3).

Gertjan van Noord, 2001. “Robust Parsing of Word Graphs”. Chapter 9, pages 205–238. In Junqua & van Noord [2001].

Gérard Verfaillie & Thomas Schiex. 1994a. Dynamic backtracking for dynamic constraint satisfaction problems. In *Proceedings of the ECAI-94 Workshop on Constraint Satisfaction Issues Raised by Practical Applications*, Amsterdam, The Netherlands.

Gérard Verfaillie & Thomas Schiex. 1994b. Solution reuse in dynamic constraint satisfaction problems. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 307–312, Seattle, WA, USA.

Stefan Voß, Silvano Martello, Obrahim H. Osman & Catherine Roucairol. 1999. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization.* Boston, MA, USA: Kluwer Academic Publishers.

Chris Voudouris. 1998. Guided local search - an illustrative example in function optimisation. *BT Technology Journal*, 16(3):46–50.

Christos Voudouris & Edward Tsang. 1995. Partial constraint satisfaction problems and guided local search. Technical Report CSM-250, Department of Computer Sciense, University of Essex, Colchester, UK.

A. Voutilainen. 1995. A syntax-based part of speech analyser. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL-95)*, pages 157–164.

Wolfgang Wahlster. 1992. Computational models of face-to-face dialogs: Multimodality, negotiation and translation. Invited lecture at COLING-92; not included in the proceedings.

Wolfgang Wahlster. 1993. Verbmobil: Translation of face-to-face dialogs. In *Proceedings of the 3rd European Conference on Speech Communication and Technology*, pages 29–38, Berlin, Germany.

Wolfgang Wahlster, ed.. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation.* Berlin, Heidelberg: Springer-Verlag.

Wolfgang Wahlster, Anselm Blocher, Jörg Baus, Eva Stopp & Harry Speiser. 1998. Ressourcenadptierende Objektlokalisation: Sprachliche Raumbeschreibung unter Zeitdruck. Bericht 141, Universität des Saarlandes, KI-Labor am Lehrstuhl für Informatik.

Wolfgang Wahlster, Anthony Jameson, Alassane Ndiaye, Ralph Schäfer & Thomas Weis. 1995. Ressourcenadaptive Dialogführung: ein interdisziplinärer Forschungsansatz. *Künstliche Intelligenz*, 6:17–21.

Wolfgang Wahlster & Werner Tack. 1998. SFB 378: Ressourcenadaptive kognitive Prozesse. Bericht 143, Universität des Saarlandes, KI-Labor am Lehrstuhl für Informatik.

Alex Waibel, Hagen Soltau, Tanja Schultz, Thomas Schaaf & Florian Metze, 2000. "Multilingual Speech Recognition". pages 33–45. In Wahlster [2000].

Richard J. Wallace. 1996. Analysis of heuristic methods for partial constraint satisfaction problems. In Eugene C. Freuder, ed., *Principles and Practice of Constraint Programming – CP '96*, Lecture Notes in Computer Science 1118. Springer, Berlin.

Richard J. Wallace & Eugene C. Freuder. 1995a. Anytime algorithms for constraint satisfaction and SAT problems. In *Proceedings of the IJCAI Workshop on Anytime Algorithm and Deliberation Scheduling*, Montréal, Quebec.

Richard J. Wallace & Eugene C. Freuder. 1995b. Heuristic methods for over-constrained constraint satisfaction problems. In *Proceedings of the CP 1995 Workshop on Over-Constrained Systems.*

David Waltz. 1975. Understanding line drawings of scenes with shadows. In P. Winston, ed., *The Psychology of Computer Vision*. McGraw-Hill, New York, pages 19–91.

Heinz Josef Weber. 1992. *Dependenzgrammatik: Ein Arbeitsbuch.* Tübingen, Germany: Gunter Narr Verlag.

Thomas Weis. 1997. Resource-adaptive action planning in a dialogue system for repair support. In *Proceedings of KI-97: Deutsche Jahrestagung für Künstliche Intelligenz.*

Ralph M. Weischedel, Wilfried M. Voge & Mark James. 1978. An artificial intelligence approach to language instruction. *Artificial Intelligence Journal*, 10:225–240.

Fuliang L. Weng. 1993. Handling syntactic extra-grammaticality. In *Proceedings of the 3rd International Workshop on Parsing Technologies (IWPT-93)*, Tilbiurg, The Netherlands.

Thomas Weyrath. 1997. Einschätzung von Zeitdruck und Arbeitsgedächtnisbelastung in ressourcenbeschränkten Dialogen: Eine explorative empirische Untersuchung. In *Proceedings der KogWis-97: 3. Fachtagung der Gesellschaft für Kognitionswissenschaft*, Friedrich-Schiller-Universität Jena.

Christopher M. White. 1995. Converting context-free grammars to constraint dependency grammars. Master Thesis, Purdue University.

Christopher M. White. 2000. *Rapid Grammar Development and Parsing: Constraint Dependency Grammar with Abstract Role Values.* Ph.D. thesis, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA.

Christopher M. White, Mary P. Harper, T. Lane & R. A. Helzerman. 1997. Inductive learning of abstract role values derived from a constraint dependency grammar. In *Proceedings of the Automata Induction, Grammatical Inference, and Language Acquisition Workshop at the Fourteenth International Conference on Machine Learning*.

Jan Wielemaker, 2000. *SWI-Prolog 3.3.4 Reference Manual*. Department of Social Informatics (SWI), Roeterstraat 15, 1018 WB Amsterdam, The Netherlands.

Thomas Williams & Colin Kelley. 1998. gnuplot – an interactive plotting program.

Robert A. Wilson & Frank Keil. 1999. *The MIT Encyclopedia of Cognitive Science*. Cambridge, MA, USA: Bradford.

Mats Wirén. 1992. *Studies in Incremental Natural-Language Analysis*. Ph.D. thesis, Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Niklaus Wirth. 1986. *Compilerbau*. Stuttgart, Germany: Teubner.

Mihalis Yannakalis, 1997. "Computational complexity". Chapter 2, pages 19–55. In Aarts & Lenstra [1997b].

Mazoud Yazdani. 1986. Intelligent tutoring systems: An overview. *Expert Systems*, 3(3):154–162.

Steve Young, Joop Jansen, Julian Odell, Dave Ollason & Phil Woodland, ed.. 1997. *The HTK Book*. Cambridge, UK: Entropic.

Jakub Zavrel & Walter Daelemans. 1999. Recent advances in memory-based part-of-speech tagging. In *Actas del VI Simposio Internacional de Comunicacion Social*, Santiago de Cuba, Cuba.

Michael Zock. 1996. Computational linguistics and its use in real world: The case of computer assisted-language learning. In *Proceedings of the 16th International Conference on Computational Linguistics, Panel on Computer Assisted-Language Learning*, pages 1002–1004, Copenhagen, Denmark.

Carla B. Zoltowski, Mary P. Harper, Leah H. Jamieson & Randall A. Helzerman. 1992. PARSEC: A constraint-based framework for spoken language understanding. In *Proceedings of the International Conference on Spoken Language Processing*, pages 249–252, University of Alberta.