T.A.L., 2000, vol. 41, n^{o} 1, pp. 113–142

MODELING DEPENDENCY GRAMMAR WITH RESTRICTED CONSTRAINTS

Ingo SCHRÖDER   Wolfgang MENZEL

Kilian FOTH   Michael SCHULZ*

Résumé - Abstract

In this paper, parsing with dependency grammar is modeled as a constraint satisfaction problem. A restricted kind of constraints is proposed, which is simple enough to be implemented efficiently, but which is also rich enough to express a wide variety of grammatical well-formedness conditions. We give a number of examples to demonstrate how different kinds of linguistic knowledge can be encoded in this formalism.

Dans cet article, l'analyse syntaxique avec une grammaire de dépendance est modélisé comme un problème de satisfaction de contraintes. L'article décrit un type particulier de contraintes, assez simples pour être implementées efficacement, et en même temps assez riches pour exprimer une grande quantité de conditions de bonne formation grammaticale. Nous donnons plusieurs exemples illustrant comment différents types de connaissances linguistiques peuvent être encodés par le formalisme proposé.

Mots Clefs - Keywords

Constraint dependency grammar, natural language parsing, constraint parsing, restricted constraints, constraint satisfaction

Grammaire de dépendance à contraintes, analyse syntaxique du langage naturel, analyse par contraintes, satisfaction de contraintes

* AB NATS, Informatik, Universität Hamburg,
E-Mail: {ingo|wolfgang|foth|micha}@nats.informatik.uni-hamburg.de

INTRODUCTION

Natural language analysis can be viewed as a constraint satisfaction problem (CSP), or more specifically as a consistent labeling problem (Tsang E. 1993), since it is possible to describe the parsing process as choosing a unique governor together with a suitable dependency relation for each word from a finite set of possibilities. Taken together, these selected pairs constitute a dependency structure for the sentence under consideration.

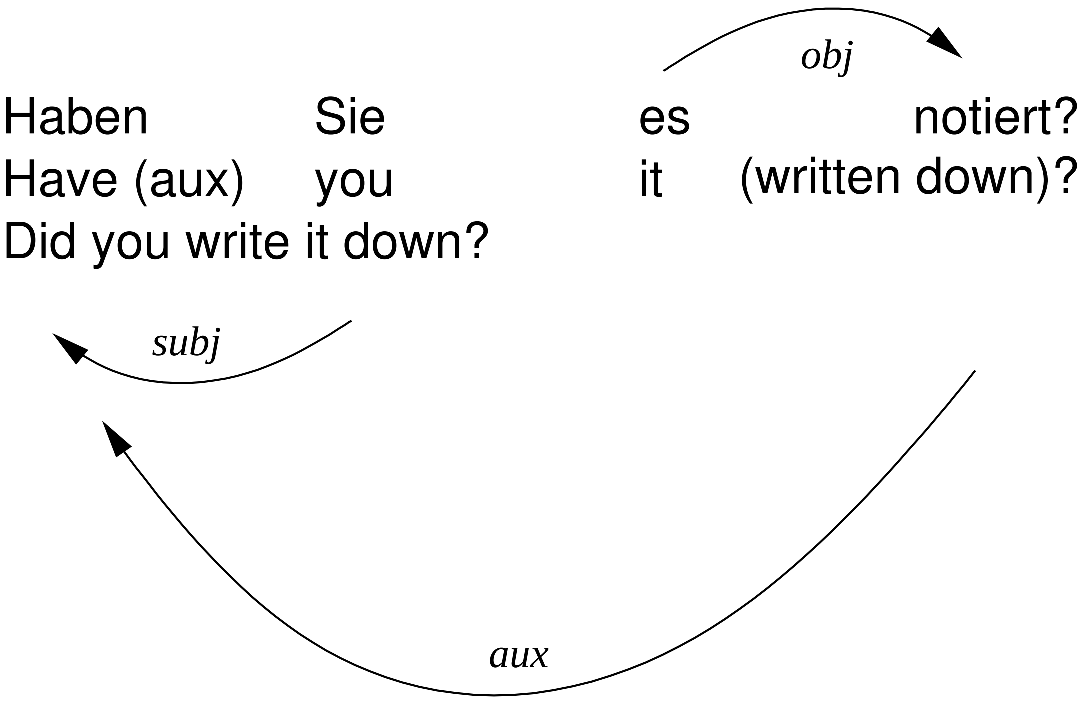


Figure 1. (a) Space of possible dependency arcs (b) Final solution for the example (cf. Figure 2)

Assuming a simplified set of just three different dependency relations `subj(ect)`, `obj(ect)` and `aux(iliary)`, Figure 1(a) shows all nine combinatorically possible dependency edges[1] for each word in a simple four word example sentence.[2] This set is a compact representation of all conceivable structures, in this case $(9+1)^4 = 10,000$ of which $4^3 \cdot 3^3 = 1728$ are connected trees.

Constraints can then be used to identify those edges that constitute a valid solution to the parsing problem, either by deleting inappropriate ones or by selecting the optimal combination of edges based on a plausibility score. Figure 1(b) shows the remaining edges after applying all constraints which describe the unique solution to this parsing problem. Note that there is no need

[1] Additionally, a word can be the root of the tree which is not shown explicitly in Figure 1.

[2] Most of the examples are taken from the Verbmobil (Wahlster W. 1993) corpus of spoken appointment-scheduling dialogues. However, the analyses had to be simplified for presentation purposes.

for an additional component, for instance context-free rules, that generates structures to be checked against compatibility constraints like it is often the case in other constraint-based approaches. Instead constraints are used to build complex structural descriptions by choosing the corresponding edges.

Viewing natural language analysis as such a CSP comes along with a number of advantageous characteristics:

- Constraint satisfaction procedures facilitate robust parsing because they exhibit a fail-soft behavior. Lifting less important constraints makes it possible to find solutions to otherwise over-constrained problems, e. g., in cases of ill-formed sentences.

- The fundamental steps in solving a constraint satisfaction problem involve applying constraints to solution candidates. Since these individual tasks are relatively independent of each other, constraint satisfaction carries a great potential for parallel implementations (Helzerman R. A. & Harper M. P. 1992).

- Constraints are not only useful to distinguish the grammatical from the ungrammatical case, but can also provide a soft decision about the degree of grammaticality of an utterance. This can be achieved by assigning higher penalties to constraints dealing with severe grammar violations and lower penalties to those constraints that can be violated more easily.

- Parsing as a CSP means selecting acceptable dependency edges from a set of possible ones. Since the number of alternatives for a specific dependency edge is finite, it is possible to compare the current choice to its alternatives. This property allows one to gradually improve an intermediate structure as more computation time is spent. Therefore, the selectional nature of CSPs makes it easy to develop so-called anytime algorithms which are able to deliver a result early, but improve the quality of the solution if more time is available (Menzel W. 1994).

- In order to be able to diagnose grammatical faults by language learners, the diagnosis component needs a fine-grained conceptualization of grammar rules. Constraints are well-suited for that purpose.

The formalization of the parsing problem as a CSP allows the system designer to carry over these interesting properties to natural language applications. In order to do so, the set of possible structures for an utterance has to be defined in terms of finite domains for a set of constraint variables. Dependency structures are particularly well-suited for this kind of formalization since they perfectly fit the notion of selecting a value from the domain of a constraint variable: variables correspond to word forms in the utterance and values are pairs of possible governors and dependency relations.

Several applications may benefit from the CSP approach to natural language analysis:

- The inherent robustness of the CSP techniques seems especially beneficial for systems that process spontaneous speech because it this nearly impossible to anticipate the variability of actually occuring constructions (including ungrammatical ones).

- Sophisticated second language learning systems need an elaborate diagnosis component, which can be designed to cover syntactic correctness, semantic plausibility and some aspects of communicative appropriateness of language use (Menzel W. & Schröder I. 1998b).

- Several possibly contradictory knowledge sources can be integrated into multimodal dialogue systems as long as large parts of the information can be supplied as or translated into constraints (Menzel W. & Schröder I. 1998).

Maruyama was the first to view parsing as a CSP with finite domains when he proposed the use of Constraint Dependency Grammars (CDG) in an interactive machine translation system (Maruyama H. 1990a). It has since then been extended in several directions, e. g., the integration with a speech recognizer (Harper M. P. *et al.* 1993), the introduction of preference reasoning techniques and the disambiguation in a multilevel representation (Heinecke J. *et al.* 1998).

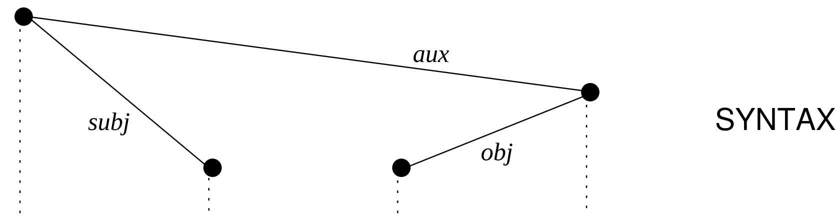
Whereas Maruyama (Maruyama H. 1990b) has shown the weak generative capacity of the general CDG to be strictly greater than that of CFG, this paper concentrates on the question whether the expressiveness of a restricted kind of constraints is sufficient for modeling major parts of a natural language grammar. Therefore, it puts more emphasis on providing practical models for actually observed phenomena. The rest of the paper is organized as follows: Section 1 describes the characteristics of the constraint parsing system. The main part of the paper in Section 2 looks at different natural language phenomena and provides possible constraint modelings for them. Section 3 motivates and describes the use of 'soft' constraints, i. e., constraints that may be violated by a solution. Finally, the conclusions look at the advantages of the approach and give an outlook to further research.

1. PREREQUISITES

This section introduces CDGs, explains how to actually express grammatical constraints and describes the limitations that are enforced in a practical solution.

1.1. Constraint Dependency Grammars

Constraint Dependency Grammars aim at assigning dependency structures to natural language utterances.[3] In such a structure each word is subordinated under another one, thus describing a dependency relation between the two. These dependency edges are usually assumed not to form cycles, and therefore, dependency structures are traditionally depicted as trees. The edges can be enriched with labels providing a further differentiation. Figure 2 shows a possible dependency analysis which is equivalent to the presentation in Figure 1(b).



Haben Sie es notiert?
Have (aux) you it (written down)?
Did you write it down?

Figure 2. A 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. CDG, in contrast, is not limited to one structural level or, more vividly, CDG can be used to build any number of dependency structures in parallel, not just a single syntactic one. Figure 3, for instance, describes some functor-argument relations for the previous example where the dependent is considered to be the argument.

Within the framework of CDG, 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, they are actually built simultaneously so that syntactic evidence not only influences the semantic structure but also vice versa.

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 like semantics (cf. for example Section 2.6), auxiliary ones can be used (cf. for example Section 2.1.3) to further constrain the structures on a main level.

[3] CDGs are a declarative formalism, i. e., they can be used for parsing as well as for generation. However, no generation algorithm has been proposed so far.

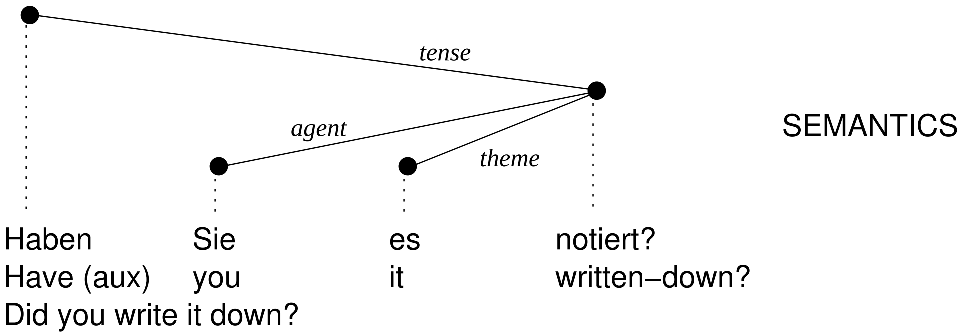


Figure 3. A semantic analysis belonging to the syntactic analysis of Figure 2

Generally, dependency analysis is well-suited for partial parsing (Mitjushin L. 1992). While usually all dependency edges on each level should form a single tree, this is not enforced by CDG. If no single tree can be found, say, for the syntax level, an analysis may consist of disconnected subtrees each representing a partial parse. We will return to this issue in Section 3.1.

Note that although a grammar writer will almost always refer to an established dependency theory (Tesnière L. 1959; Kunze J. 1972), the formalism of CDG does not enforce a specific one. CDG may be used as long as only dependency relations between word forms are considered, only a limited number of 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 I. A. 1988) employ semantic units more complex than word forms which therefore cannot be modeled directly. Semantic modeling in CDG is restricted to some phenomena, e. g., functor-argument structures.

1.2. What are constraints?

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.[4]

(C1) $\texttt{\{X\} : SubjectVerbNoun : Subject :}$

$\texttt{X.level=SYNTAX} \wedge \texttt{X.label=subj} \rightarrow \texttt{X}\downarrow\texttt{cat=NOUN} \wedge \texttt{X}\uparrow\texttt{cat=FINVERB}$

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 C1. If it evaluates to false, the constraint is said to be violated, not

[4] All constraints in this paper are simplified; they are not intended to be actually used in a real grammar. Some constraints are extracted from a larger grammar of German so that they may not be valid for different languages. However, all constraints demonstrate a specific technique or idea; they are not meant as a suggestion how to treat a specific linguistic phenomenon in full.

fulfilled or not satisfied. The first line in Constraint C1 serves more technical purposes by introducing a variable that is a placeholder for a dependency edge (`{X}`), assigning a name to the constraint (`SubjectVerbNoun`) and defining a group the constraints belongs to (`Subject`). Finally, the last line contains an optional comment.

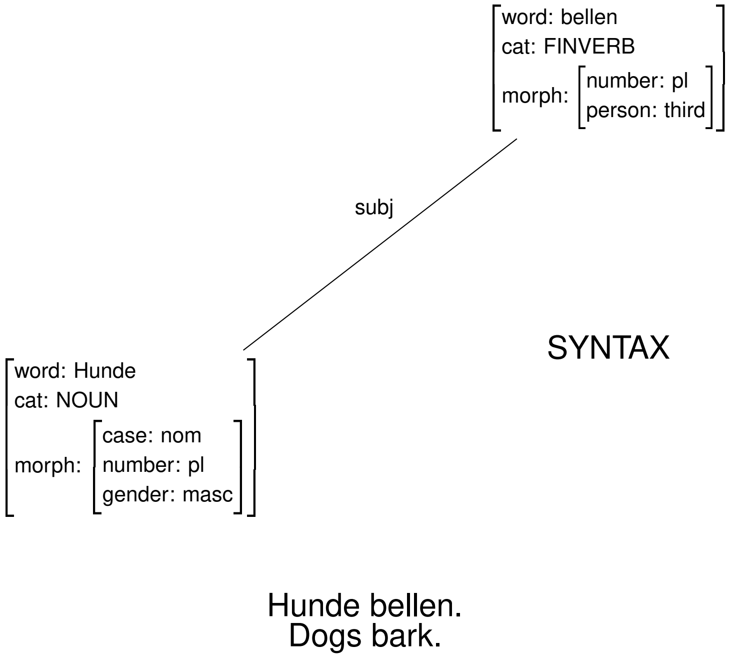


Figure 4. A dependency edge which satisfies Constraint C1

Constraints have access to different bits of information that are associated with the elements of the considered dependency (cf. Figure 4):[5]

- Level of analysis: If multiple structures are built at the same time, constraints need to know which structure the particular edge belongs to. In Constraint C1 the term `X.level` refers to the level of dependency edge `X`.
- Label: Dependencies are further differentiated by labels. Examples in Figure 2 are `subj`, `obj` and `aux`. In Constraint C1 the term `X.label` refers to the label of the dependency edge `X`.
- Lexical information of the dependent: Constraints have access to a lexicon which associates lexical information with word forms. Examples are category, case or number of the word form. In Constraint C1 the term `X↓cat` refers to the lexical feature `cat` of the dependent of dependency

[5]Note that although the lexical entries resemble the well-known feature structures typical for unification-based grammars like HPSG (Pollard C. & Sag I. 1994), unification is *not* a valid operation in CDG (cf. remark at the end of Section 1.2), nor are the lexical entries re-entrant, i. e., there is no structure sharing, internal or external. Essentially, the lexical entries are just structured information containers.

edge `X`. Here, the arrow pointing down pictorially indicates the lower end of the edge.

- Lexical information of the governor: In Constraint C1 the term $X{\uparrow}\mathtt{cat}$ refers to the lexical feature `cat` of the governor of dependency edge `X`. Again, the arrow pointing up pictorially represents the upper end of the edge.
- Sentence information: The term $X{\downarrow}\mathtt{pos}$ returns the position of the dependent in the sentence and $X{\uparrow}\mathtt{id}$ is used to identify word forms in an utterance unambiguously.
- Predicates: Constraints may use predicates that test for various properties of word forms or dependency edges. The predicate `root()`, for instance, tests whether the word form is the root of the tree.

Constraint C1 imposes the following condition on all dependency edges in the structure: if the level of the edge is `SYNTAX` and the label is `subj` then the lexical feature `cat` of the dependent must be `NOUN` and the lexical feature `cat` of the governor must be `FINVERB`.

We can now extend the notion of constraints to those which do not judge a single dependency edge, but a tuple of edges.

(C2) `{X, Y} : SubjectBeforeObject : WordOrder :`
$\mathtt{X.level{=}SYNTAX} \wedge \mathtt{Y.level{=}SYNTAX} \wedge \mathtt{X{\uparrow}id{=}Y{\uparrow}id} \wedge$
$\mathtt{X.label{=}subj} \wedge \mathtt{Y.label{=}obj} \rightarrow \mathtt{X{\downarrow}pos{<}Y{\downarrow}pos}$
The subject precedes the object.

Constraint C2 looks at two dependency edges: the subject and the object edge of one and the same verb. It is violated in configurations where the subject follows the object.[6]

Note that constraints only carry out a passive check of compatibility. In contrast, operations like unification create new objects as a result and insert them into the space of possible structures. However, CDG depends 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, can indeed be checked while the actual unification is not possible (cf. Section 2.2).

1.3. Limiting the expressiveness of constraints

Actual natural language applications require a task-adequate model of the language, on the other hand they must meet some very practical requirements like e. g., tractability and efficiency. Therefore, CSP algorithms usually

[6]This constraint may be part of a constraint cluster that is used to enforce the SVO order of languages like English.

restrict the expressiveness of constraints in order to allow practical implementations (Tsang E. 1993).

Accordingly, we need to limit the number of simultaneously considered dependency edges to at most two. 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 like non-local dependencies. Figure 5 gives a German example where certain dependencies can only be dealt with by very 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 2.6 and 3.3.

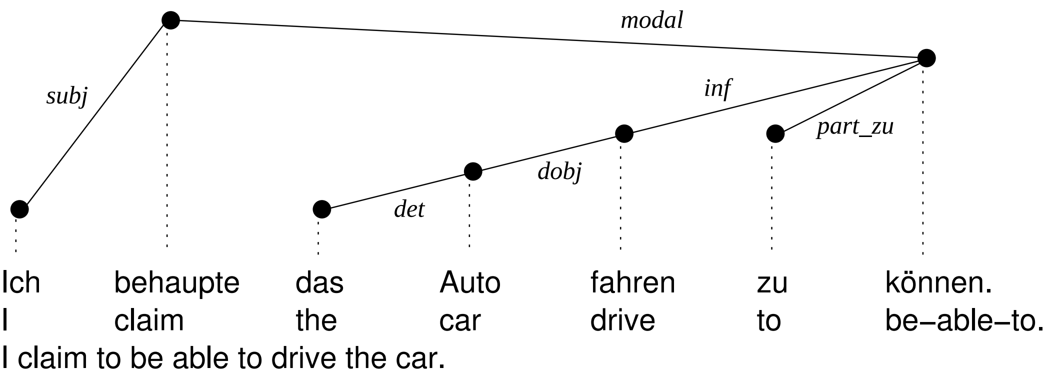


Figure 5. A dependency tree for a German control verb construction

Furthermore, some grammatical restrictions 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 (like the French negation *ne ... pas* or the German circumposition *um ... willen*) must be accompanied by the other part.

But again, we need to restrict the expressive power of constraints for efficiency reasons. Therefore, logical formulas are (implicitely) universally quantified.

To summarize, a constraint can relate at most two dependency edges at the same time and cannot contain any existence quantifiers. However, 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 express existential requirements.

2. DEPENDENCY MODELING

We will now describe how restricted 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, that is, 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.

2.1. Valence

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

- Valence possibility: A lexeme selects its argument, that is, it determines what properties another lexeme must have to be a suitable dependent.
- Valence uniqueness: An argument position (or *slot*) must be uniquely specified, that is, 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, that is, there must be exactly one lexeme in an analysis that fills a specific argument position.

All three conditions can be ensured by restricted constraints.

2.1.1. Valence possibility

In the following Example (1) the verb *verschieben* opens three slots to be filled by appropriate arguments.

(1)	*Wir*	*verschieben*	*das*	*Treffen*	*[auf*	*die*	*nächste*	*Woche].*
	We	*postpone*	*the*	*meeting*	*[until*	*the*	*next*	*week].*

‘We postpone the meeting until next week.’

Figure 6 shows the desired analysis of Example (1) including the correct lexical entries for all word forms.

Modeling the valence possibility of a verb[7] 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. Here, the first two argument positions model the obligatory nominal constituents while the last one refers to an optional prepositional phrase.

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

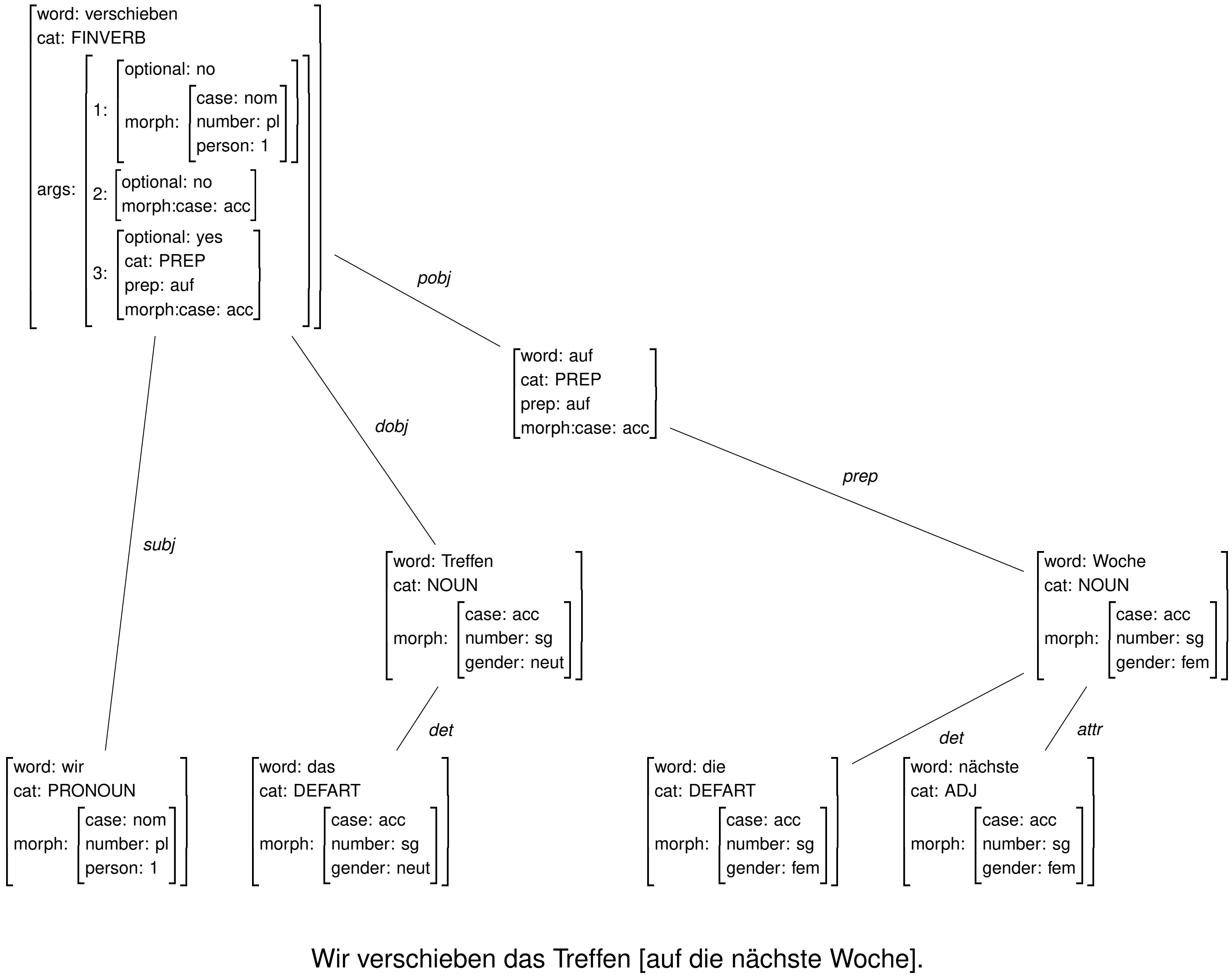


Figure 6. Dependency analysis with correctly chosen lexical entries

Since CDG has 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.

(C3) `{X} : ValenceFirstArg : Valence :`
$$\begin{array}{l} \texttt{X.label=subj} \rightarrow \\ \quad \texttt{X}\uparrow\texttt{args:1:cat=X}\downarrow\texttt{cat} \wedge \\ \quad \texttt{X}\uparrow\texttt{args:1:morph:case=X}\downarrow\texttt{morph:case} \wedge \\ \quad \texttt{X}\uparrow\texttt{args:1:morph:number=X}\downarrow\texttt{morph:number} \wedge \\ \quad \texttt{X}\uparrow\texttt{args:1:morph:person=X}\downarrow\texttt{morph:person} \end{array}$$
The first argument has to fulfill the valence requirements.

Constraint C3, 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 specify explicitly 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.

2.1.2. Uniqueness of valence fillers

In the last section, Constraint C3 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 (2).

(2)	*Wir*	*treffen*	*die*	*Kollegen*
	Pro,nom	Vfin	Det,{nom,acc}	Noun,{nom,gen,acc,dat}
	We	*meet*	*the*	*collegues.*

Only constraints C4 ensures that any two dependency edges do not assign the same function `subj` to different lexemes.

(C4) `{X, Y} : SubjUnique : Valence :`
$$\texttt{X.label=subj} \wedge \texttt{X}\uparrow\texttt{id=Y}\uparrow\texttt{id} \rightarrow \texttt{Y.label}\neq\texttt{subj}$$
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.

2.1.3. 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 at most two edges and must be universally quantified. Fortunately, one can overcome this problem by introducing an auxiliary level. Its purpose is to establish the inverse edge between the two lexemes, that is, whenever a governor dominates its argument on the syntax level, the argument will dominate the governor on the auxiliary level (Maruyama H. 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 7 shows how the syntactic level and an auxiliary level are used in combination.

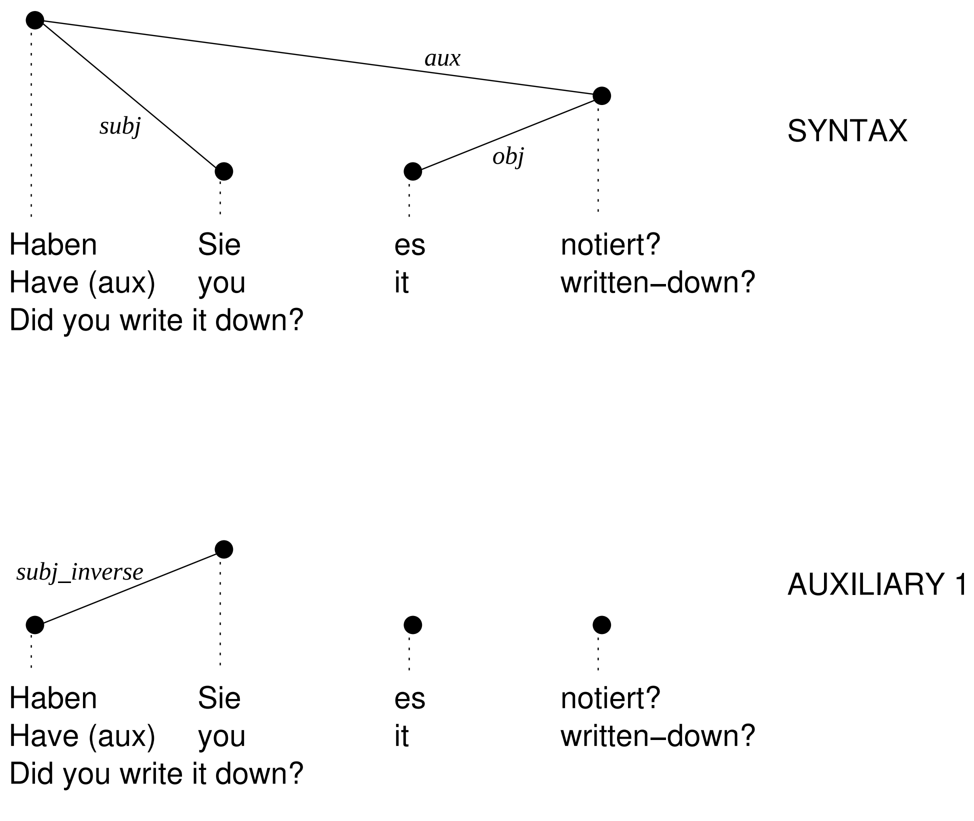


Figure 7. Dependency analysis for the main level `SYNTAX` and an auxiliary level for the first argument

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 8, therefore, contains the same information as Figure 7.

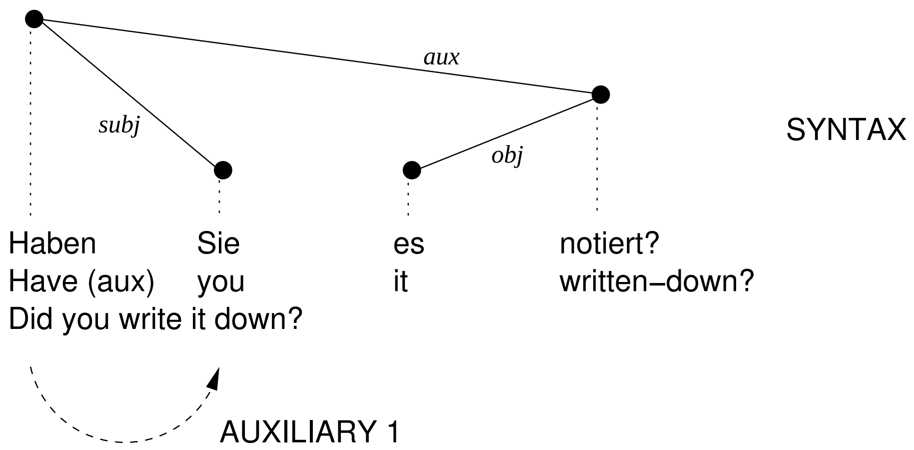


Figure 8. Dependency analysis with an alternative presentation of the first auxiliary level

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 C5: 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`. Constraint C6 ensures that the subject slot of a finite verb is filled.

(C5) `{X, Y} : Aux1SyntaxMapping : ValenceExistence :`
$\texttt{X.level=SYNTAX} \wedge \texttt{Y.level=AUX1} \rightarrow$
$(\ \texttt{X}{\uparrow}\texttt{id=Y}{\downarrow}\texttt{id} \wedge \texttt{X.label=subj} \leftrightarrow \texttt{X}{\downarrow}\texttt{id=Y}{\uparrow}\texttt{id}\ )$

(C6) `{X} : Aux1Existence : ValenceExistence :`
$\texttt{X.level=AUX1} \wedge \texttt{X}{\downarrow}\texttt{cat=FINVERB} \rightarrow \neg\ \texttt{root(X}{\uparrow}\texttt{id)}$

Although these two constraints jointly express an existence condition[8], they fulfill the conditions posed in Section 1.3: at most two edges are considered, and no existential quantifier is used.

2.2. Feature percolation

Feature transport refers to the process of carrying some kind of information along (possibly arbitrarily long) dependency chains. 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.

[8] As a side effect, Constraints C5 and C6 also guarantee the uniqueness of the argument, which has already been enforced by the binary Constraint C4.

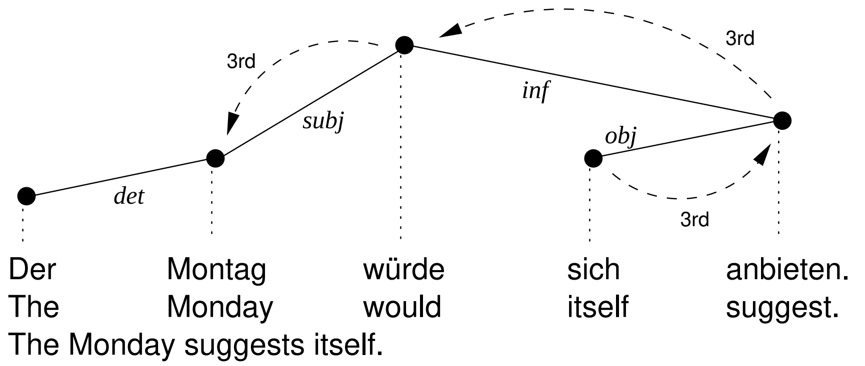


Figure 9. Feature percolation in a dependency tree

In Figure 9, it is necessary to have access to the `person` feature of the reflexive pronoun 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 CDG 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 deprecated and Sections 2.7 and 3.3 introduce mechanisms that are better suited to solve the problem of feature transportation.

2.3. Word order

Word order is a complex phenomenon and some parts of it are (partly implicitly) dealt with in other sections in this paper, e. g., *vorfeld* realization in Section 2.4, linear ordering of complements in Section 1.2 and also in Section 3.2 and projectivity in Section 2.5. 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 usually is the other way round in French (cf. Examples (3) and (4)).

(3) *Ein großer Hund ...*
A big dog ...

(4) *Le traitement automatique ...*
The processing automatic ...
'The automatic processing...'

Since the restriction of word order is directly connected to the dependency between adjective and noun, it is easy to design Constraint C7 that excludes constructions with the wrong word order from the set of well-formed German sentences.

(C7) `{X} : AdjNounWordOrder : WordOrder :`
$\texttt{X.level=SYNTAX} \wedge \texttt{X}\!\downarrow\!\texttt{cat=ADJ} \wedge \texttt{X}\!\uparrow\!\texttt{cat=NOUN} \rightarrow \texttt{X}\!\downarrow\!\texttt{pos} < \texttt{X}\!\uparrow\!\texttt{pos}$
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.

2.4. Vorfeld realization

The word order at sentence level in German is characterized by so-called *satzfelder* (Grewendorf G. *et al.* 1987) which correspond roughly to positions in a sentence which again may be filled by no, one, or more constituents. The *vorfeld* rule of German states that in an indicative main clause, the finite verb is preceded by exactly one constituent. This condition can be formalized by the Constraints C8 to C10. 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 2.1.3. First the Constraint C8 forces each finite verb to select a word to its left as its *vorfeld*. Then constraints C9 and C10 make sure that the main level `SYNTAX` is properly coupled with the auxiliary level VORFELD.

(C8) `{X} : VorfeldInit : Vorfeld :`
$\texttt{X.level=VORFELD} \wedge \texttt{X}\!\downarrow\!\texttt{cat=FINVERB}$
$\rightarrow \neg\ \texttt{root(X}\!\uparrow\!\texttt{id)} \wedge \texttt{X}\!\uparrow\!\texttt{pos} < \texttt{X}\!\downarrow\!\texttt{pos}$
A finite verb must point to the vorfeld to the left.

(C9) `{X,Y} : UnderVerbVorfeld : Vorfeld :`
$\mathtt{X.level{=}SYNTAX} \wedge \mathtt{Y.level{=}VORFELD} \wedge$
$\mathtt{X{\uparrow}id{=}Y{\downarrow}id} \wedge \mathtt{X{\downarrow}pos} < \mathtt{X{\uparrow}pos} \wedge \mathtt{Y{\downarrow}cat{=}FINVERB}$
$\rightarrow \mathtt{Y{\uparrow}id{=}X{\downarrow}id}$
A constituent depending on the verb from the left must occupy the *vorfeld*.

(C10) `{X,Y} : VorfeldUnderVerb : Vorfeld :`
$\mathtt{X.level{=}SYNTAX} \wedge \mathtt{Y.level{=}VORFELD} \wedge \mathtt{Y{\uparrow}id{=}X{\downarrow}id}$
$\rightarrow \mathtt{X{\uparrow}id{=}Y{\downarrow}id}$
The *vorfeld* constituent depends on the finite verb.

2.5. Projectivity

A general rule about constituents is that they are nested structures of adjacent words, that is, 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 paper, we present projection as dotted lines and dependency edges as solid lines in the figures. For example, Figure 10 shows a non-projective dependency tree for a non-projective analysis, with the instances of projectivity violations marked by circles.

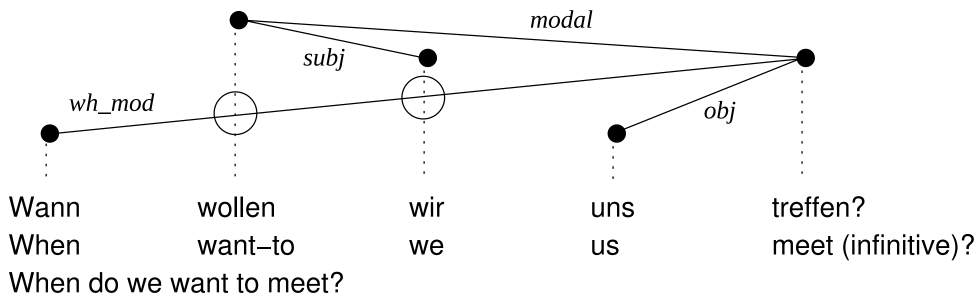


Figure 10. A non-projective dependency tree

For efficiency reasons, one wants to restrict the syntactic analyses to projective ones (Kahane S. *et al.* 1998). Nevertheless, there are some natural language phenomena for which it is difficult to establish a projective analysis, such as the surface structures resulting from instances of 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 10).

For connected dependency trees, projectivity can be enforced by the following constraints C12 and C13.[9] For reasons of simplicity, we have eliminated

[9] Maruyama (Maruyama H. 1990a) used a similar Constraint C11 for the same purpose:

those parts of the constraints that deal with exceptional cases for which we explicitly allow non-projectivity.

(C12) $\{X, Y\}$: ProjectivityDirect : Projectivity :
$X.level{=}SYNTAX \wedge X{\uparrow}id = Y{\downarrow}id \rightarrow$
$Y{\uparrow}pos \leq \min(X{\downarrow}pos, Y{\downarrow}pos) \mid Y{\uparrow}pos \geq \max(X{\downarrow}pos, Y{\downarrow}pos)$
A dependency cannot cover an ancestor.

(C13) $\{X, Y\}$: ProjectivityNoCrossing : Projectivity :
$X.level{=}SYNTAX \wedge \min(X{\downarrow}pos, X{\uparrow}pos) < \min(Y{\downarrow}pos, Y{\uparrow}pos) \rightarrow$
$\max(X{\downarrow}pos, X{\uparrow}pos) \geq \max(Y{\downarrow}pos, Y{\uparrow}pos)$
$|$
$\max(X{\downarrow}pos, X{\uparrow}pos) \leq \min(Y{\downarrow}pos, Y{\uparrow}pos)$
Dependency edges do not cross.

Figure 11 proves that these constraints forbid all cases of non-projectivity.

2.6. 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 restricted constraints will often investigate not 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 subcategorisation phenomena. Furthermore, special problems of semantics, e. g., reference resolution, can often be solved by postulating additional levels of analysis (cf. Section 2.7).

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.

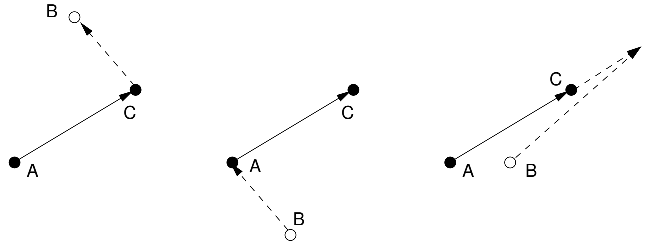
Since in many cases, especially with complex verb groups, the syntactic and semantic structures differ considerably (cf. Figures 2 and 3), an obvious solution would be to create an autonomous semantic level in analogy to the syntactic one. Semantic relations between word forms are represented as dependency edges on that level, and the kind of relation (e. g., the thematic role) can be notated as the label of the edge.[10] 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

(C11) $\{X, Y\}$: Maruyama : Projectivity :
$X{\uparrow}pos < Y{\downarrow}pos \wedge Y{\downarrow}pos < X{\downarrow}pos \rightarrow$
$X{\uparrow}pos \leq Y{\uparrow}pos \wedge Y{\uparrow}pos \leq X{\downarrow}pos$

[10]Word Grammar (Hudson R. A. 1990) uses a similar approach but allows extra edges directly in the syntactic structure, rather than introducing additional levels.

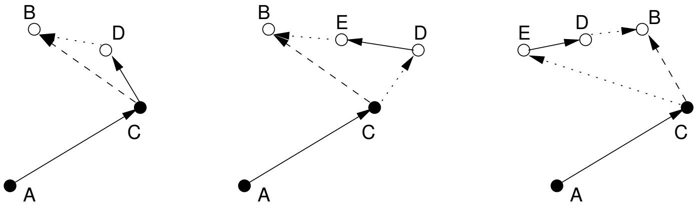
Let V be the set of word nodes in the tree. Let $\rightarrow \subset V \times V$ be the direct subordination relation and $\rightarrow^*$ the transitive closure thereof, the partial order of subsumption. Let further $< \subset V \times V$ be the total order of linear order, i. e., $A < B$ means A appears 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.



(I) (II) (III) (IV)

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



(Ia) (Ib) (Ic)

(Ia) No edge on the path end 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 C12.

(Ib) At least one edge on the path starts between node A and node C and ends to the right of node C. Edges $A \rightarrow C$ and $D \rightarrow E$ are forbidden by Constraint C13.

(Ic) symmetric with configuration (Ib)

Figure 11. Proof

is not necessarily justified for semantics: in contrast to syntactic subordination, it is quite common for a word form to fill more than just one semantic slot.

In Example (5) 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 2.2).

(5) *Der Geschichten erzählende Mann lacht.*
The stories telling man laughs.
'The man telling stories laughs.'

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 dependent, i. e., the relation is reversed if compared to the analysis in Figure 3. 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 12 shows three semantic edges originating from two semantic levels below the syntax tree.

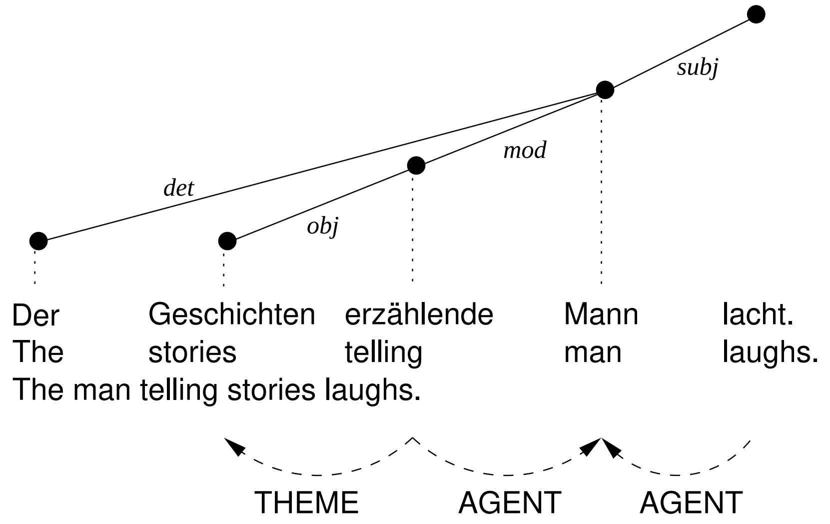


Figure 12. A syntactic dependency tree with two additional levels for thematic roles

The use of arrows in Figure 12 also emphasizes that word forms with open semantic slots select or point to the word forms that fill the slots.

2.7. Additional levels

As already mentioned in Section 2.6 many different problems of natural language analysis can be addressed by the introduction of additional levels in CDG. One such example is reference resolution, i. e., identification of those word forms that anaphoric pronouns refer to.

On a special level, say REF, refering pronouns select their antecedents (cf. Section 2.6 for an in-depth explanation of the general technique of introducing an additional level on which specific selectional processes are carried out). Constraints dealing with properties of those relations between referent and antecedent can then easily be formulated for dependency edges on that level.

In Example (6) the relative pronoun *der* has two possible antecedents: *Ehemann* and *Frau*.

(6)	*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 C14 have recourse 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. Figure 13 presents a possible analysis for Example (6).

(C14) `{X} : RelativeGenderNumber : RelativeSentence :`
`X.level=REF` $\wedge$ `X↓cat=RELPRONOUN` $\rightarrow$
`X↓gender=X↑gender` $\wedge$ `X↓number=X↑number`
Relative pronoun and antecedent agree with respect to gender and number.

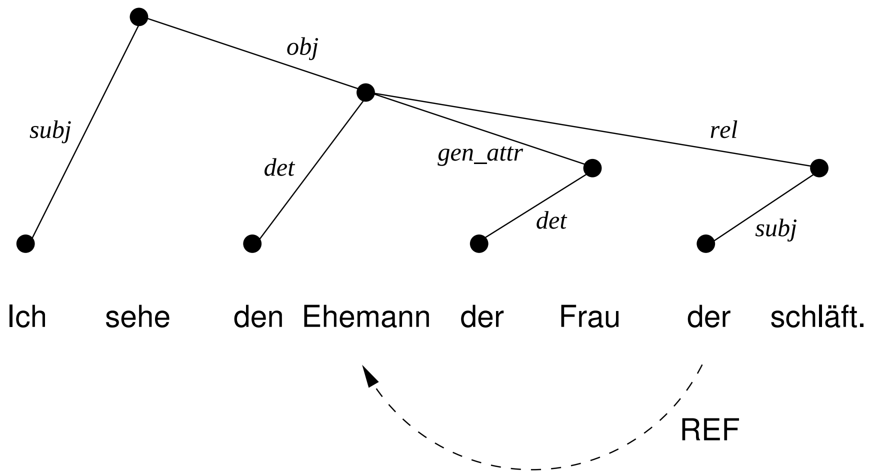


Figure 13. An analysis for a sentence with a relative clause including reference resolution

3. WEIGHTED CONSTRAINTS

Constraints so far have been understood as strict conditions, i. e., a valid dependency structure is not allowed to violate any constraint. It turns out, however, that several aspects of natural language are much better described in terms of weak or soft constraints that encode preferences rather than absolute conditions. While hard constraints might be used for the modeling of the ideal speaker-hearer, soft constraints are indispensable if performance aspects of a speaker or hearer should be modeled. A CDG with soft constraints returns 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 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 solutions.

In order to distinguish constraints of different strength a *penalty factor* or *weight* taken from the interval $[0, 1]$ is assigned to each constraint. A weight near one indicates that the constraint is a very soft one and may very well be violated in a solution. The more the weight approaches zero the more important the constraint becomes. A weight of zero means that the constraint must never be violated.[11] In this sense, traditional hard constraints all have a weight of zero. The weights of violated constraints are combined multiplicatively in order to assess a dependency structure candidate, and the structure with the highest score will be selected as the most plausible analysis of the utterance.

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 weights. 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.

CDG with soft constraints has some properties in common with Optimality Theory (Prince A. & Smolensky P. 1993) which defines a structure as grammatical if it violates only less important conditions than any other structure. Like a weighted CDG it therefore allows to cope with conflicting conditions in the grammar. However, while Optimality Theory draws on universal principles that are strictly ranked, constraints in CDG tend to be more specific. Since their penalty factors are combined multiplicatively, many weak constraints can compensate for a strong one, which has advantages for the disambiguation of ungrammatical utterances. Additionally, Optimality Theory employs some unspecified component GEN that generates a possibly infinite number of structures while CDG uses the complete (but finite) set of subordination possibilities.

[11] Constraints with a weight of one are totally ineffective.

3.1. 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 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 at least partly be analyzed.

(C15) `{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.

(C16) `{X} : AdverbNoRoot : Adverb : 0.1 :`
`X↓cat=ADVERB → ¬ root(X↑id)`
Most often, an adverb is not the root of the tree.

The Constraints C15 and C16, for example, demonstrate this method for adverbs. While Constraint C15 generally allows the subordination of adverbs under verbs, adjectives and adverbs as well as the analysis as the root of the tree, Constraint C16 penalizes the root analysis with a weight of 0.1. Therefore, the subordination under verbs, adjectives and adverbs is highly prefered, but in case no such dependency can be found, an analysis as root is acceptable as well.

(7) *(Sie haben gewonnen!)*
(You have won!)

Toll$_{\text{ADV}}$*!*
Great!

In Example (7), the single adverb *toll* can be analyzed even though no complete sentence, especially no finite verb, can be found. While this example may seem trivial (which it indeed is) the very same mechanism allows an analysis of fragmentary input (cf. Example (8)).

(8) * *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/her 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 14 shows a possible dependency analysis for this utterance.

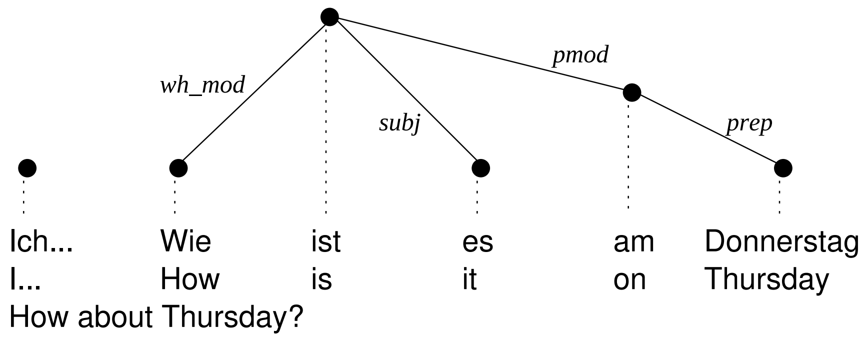


Figure 14. An analysis for an utterance with self correction

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. Note that it is quite easy for a subsequent stage of processing to identify the structure of the main sentence as well as to determine that there was some kind of self repair.

3.2. Preference constraints

Besides contributing to a robust analysis, soft constraints allow the integration of preferential knowledge into the parsing system. By preferences, we mean bits of 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 a sentence.

When one designs a natural language grammar using weighted constraints, one quickly finds that there are very few rules that hold for *every* sentence. Therefore, a whole spectrum of preferences has to be provided:

- 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. As long as the utterance is not too badly distorted, the system should nevertheless find the most plausible analysis. Additionally, often an indication of what errors were made is useful. In order to deal with this kind of errors, constraints have to be written with a weight near zero (because it is a serious violation of grammaticality).

  (C17) `{X} : DetNounCase : Agreement : 0.1 :`
  $\texttt{X.level=SYNTAX} \wedge \texttt{X.label=det} \rightarrow \texttt{X}{\downarrow}\texttt{case=X}{\uparrow}\texttt{case}$
  The determiner agrees with its noun with respect to case.

  For example, Constraint C17 requires that determiner and noun agree in their case features. Nevertheless, the constraint is a soft one, only penalizing a violation with a factor of 0.1, but not forbidding it absolutely.

- The next class of preferential constraints really encode preferences in a literal sense. 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 put the object at the beginning of the sentence:

  (9) *DIESEN Termin mag ich nicht.*
  THIS appointment like I not.
  'THIS appointment, I do not like.'

  Thus, Constraint C2 from Section 1.2 is too strict for German. Softening the constraint with a weight of, say, 0.9, preserves the preferential characteristics of the condition without leading to a system failure in the topicalized case.

  This kind of preferential information blends well with the lexicalized constraints shown in Section 2. For instance, the word form *passt* ('suits$_{3sg}$') 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 prefered types for its second argument (cf. Figure 15). Note that the least prefered sort is `anything` that subsumes everything so that a second argument that is not a human being only leads to a penalization with a weight of 0.5.

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

Figure 15. A lexical entry with preferences for the type of its second argument

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

(10)	*	*Ich*	*gebe*	*andere*	*Termine*	*den*	*Vorrang*
			Adj,{nom,acc}	Noun,dat			
		I	*give*	*different*	*dates*	*the*	*preference.*

‘I prefer different dates.’

Example (10) violates an agreement constraint because the dative noun *Termine* does not agree with its determiner with respect to case. But *andere* 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 *andere* (and, of course, identifying that a case violation occurred).

3.3. Approximation of higher order constraints

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

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 9, a constraint must access at least three dependency edges simultaneously. Constraint C18 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 1.3.

(C18)
```
{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 come in two flavors:

- Only a subset of the configurations that are not well-formed is forbidden by the approximation. The rest may actually occur that 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.

Constraint C19 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.

(C19) $\{\mathtt{X,\ Y}\}$: $\mathtt{ReflexivePronounAntecedent}$: $\mathtt{Approximation}$: 0.5 :
$\mathtt{X.level{=}SYNTAX} \wedge \mathtt{Y.level{=}SYNTAX} \wedge$
$\mathtt{X.label{=}subj} \wedge \mathtt{Y.label{=}obj} \wedge \mathtt{X{\uparrow}pos{<}Y{\uparrow}pos} \wedge$
$\mathtt{Y{\downarrow}cat{=}REFLEXIVE} \wedge \mathtt{X{\uparrow}cat{=}AUX} \wedge \mathtt{Y{\uparrow}cat{=}INFINITIVE}$
$\rightarrow \mathtt{X{\downarrow}morph{:}person{=}Y{\downarrow}morph{:}person}$

CONCLUSION

We have shown in this paper that a wide variety of grammatical phenomena can be modeled either directly using restricted constraints or indirectly by incorporating additional labels or representational levels of analysis. A further class of grammatical constructions can be analyzed by approximating ternary conditions by means of binary constraints, thus mitigating the limitations of the restricted formalism to a certain degree.

Several disambiguation algorithms for the proposed formalism have been implemented (Menzel W. & Schröder I. 1998a; Foth K. 1999) and embedded into a platform for grammar development and parsing experiments. The system includes the parser itself, a graphical user interface and a SQL database of sample analyses. It has been used to model a number of non-trivial grammars, among them one for the Verbmobil domain and another one for a prototypical foreign language learning application. The number of constraints ranges from 200 to 600 and up to fifteen levels were used simultaneously. The experience with these applications has confirmed some quite attractive properties of the approach:

- Restricted constraints allow one to implement a parsing procedure that can solve the theoretically NP-complete constraint satisfaction problem with reasonable time and space requirements in most practical cases.
- It allows robust analysis of deviating utterances without any special provision. Given that an utterance is not totally ill-formed, the constraints can determine which linguistic rules were violated and still provide the best analysis possible. This way, not only a one-dimensional score but also a qualitative assessment of grammaticality can be provided.
- As a result of this, an utterance will never be gratuitously rejected because of minor inconsistencies. A constraint-based analyser is therefore

well suited as a diagnostic tool to support second-language acquisition: since all constraints are formulated explicitly, a learner can be told precisely what mistake has been made, and even what correction might remove the mistake.

- Usually only a single optimal analysis is returned by the parser. Therefore a complete disambiguation is achieved which is hardly possible with strict constraints only (Harper M. P. *et al.* 1995).
- CDG allows the formalization of different levels of representation, such as syntactic and semantic information, in a uniform manner. The interdependence between these levels can be made as tight or loose as desired. Natural language can thus be parsed in a holistic way.
- The inherent robustness against errors as well as the use of all kinds of available information for disambiguation is typical of human language understanding. In this respect, the constraint-based approach resembles human language analysis more closely than many other formalisms.

Further investigations will focus on the following three directions to pursue the considerable potential of the constraint-based approach in these areas:

- Anytime behavior: The simultaneous availability of all hypotheses enables the analysis process to maintain a measure of how ambiguous the solution still is. The possibility to dynamically adapt the behavior of the disambiguation procedure must be investigated more thoroughly.
- Incrementality: In fact, using an adaptive algorithm only makes sense if there are external time requirements to adapt to. If, for instance, an interactive system is expected to behave adequately, the speech rate suggests itself as such a reference. Furthermore, human language processing proceeds incrementally, and therefore incremental analysis is needed in order to engage in cooperative turn-taking in a spoken dialogue system. Although the first steps towards an incremental version of CDG parsing have been taken (Schulz M. 1999), more investigations are necessary.
- Concurrency: Constraint satisfaction problems are well suited for parallel solution procedures. Although a massively parallel implementation has been realized for the basic CDG parsing paradigm (Helzerman R. A. & Harper M. P. 1992), little has been done so far to extend the parallel implementation to use weighted constraints.

ACKNOWLEDGEMENTS

This research has been partly funded by the German Research Foundation “Deutsche Forschungsgemeinschaft” under grant no. Me 1472/1-2. We would like to thank Johannes Heinecke, Jürgen Kunze and Andreas Nolda for a fruitful cooperation and many discussions about principles and details of dependency modeling.

REFERENCES

FOTH, Kilian (1999) : "Transformationsbasiertes Constraint-Parsing", Diplomarbeit, Fachbereich Informatik, Universität Hamburg.

GREWENDORF, Günther ; HAMM, Fritz ; STERNEFELD, Wolfgang (1987) : *Sprachliches Wissen. Eine Einführung in die Theorie der grammatischen Beschreibung*, Suhrkamp, *Suhrkamp Taschenbuch Wissenschaft*, volume 695, Frankfurt/Main.

HARPER, Mary P. ; JAMIESON, Leah H. ; ZOLTOWSKI, Carla B. ; HELZERMAN, Randall A. (1993) : "Semantics and constraint parsing of word graphs", *in Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 63–66, Minneapolis, MN.

HARPER, Mary P. ; HELZERMANN, Randall A. ; ZOLTOWSKI, C. B. ; YEO, B. L. ; CHAN, Y. ; STEWARD, T. ; PELLOM, B. L. (1995) : "Implementation issues in the development of the PARSEC parser", *Software – Practice and Experience*, vol. 25, n° 8, pp. 831–862.

HEINECKE, Johannes ; KUNZE, Jürgen ; MENZEL, Wolfgang ; SCHRÖDER, Ingo (1998) : "Eliminative parsing with graded constraints", *in Proceedings of the Joint Conference COLING/ACL-98*, Montréal.

HELZERMAN, Randall A. ; HARPER, Mary P. (1992) : "Log time parsing on the MasPar MP-1", *in Proceedings of the 21st International Conference on Parallel Processing*, pp. 209–217, St. Charles, IL.

HUDSON, Richard A. (1990) : *English Word Grammar*, Blackwell, Oxford.

KAHANE, Sylvain ; NASR, Alexis ; RAMBOW, Owen (1998) : "Pseudo-projectivity: A polynomially parsable non-projective dependency grammar", *in Proceedings of the Joint Conference COLING/ACL '98*, pp. 646–652, Montréal.

KUNZE, Jürgen (1972) : *Die Auslaßbarkeit von Satzteilen bei koordinativen Verbindungen im Deutschen*, Akademie Verlag, *Studia Grammatica 24*, Berlin.

MARUYAMA, Hiroshi (1990a) : *Constraint Dependency Grammar*, Rapport technique n° RT0044, IBM Research, Tokyo Research Laboratory.

MARUYAMA, Hiroshi (1990b) : "Structural disambiguation with constraint propagation", *in Proceedings of the 28th Annual Meeting of the Association of Computational Linguistics (ACL-90)*, pp. 31–38, Pittsburgh, PA.

MEL'ČUK, Igor A. (1988) : *Dependency Syntax: Theory and Practice*, State University of New York Press, *SUNY Series in Linguistics*, New York.

MENZEL, Wolfgang ; SCHRÖDER, Ingo (1998a) : "Decision procedures for dependency parsing using graded constraints", *in Proceedings of the Joint Conference COLING/ACL-98 Workshop: Processing of Dependency-based Grammars,* S. Kahane ; A. Polguère (eds.), pp. 78–87, Montréal.

MENZEL, Wolfgang ; SCHRÖDER, Ingo (1998b) : "Model-based diagnosis under structural uncertainty", *in Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98),* H. Prade (ed.), pp. 284–288, Brighton.

MENZEL, Wolfgang ; SCHRÖDER, Ingo (1998) : "Constraint-based diagnosis for intelligent language tutoring systems", *in Proceedings of the IT&KNOWS Conference at the IFIP '98 Congress,* J. Cuena (ed.), Wien/Budapest.

MENZEL, Wolfgang (1994) : "Parsing of spoken language under time constraints", *in Proceedings of the 11th European Conference on Artificial Intelligence,* A. Cohn (ed.), pp. 560–564, Amsterdam.

MITJUSHIN, Leonid (1992) : "High-probability syntactic links", *in Proceedings of COLING-92: 14th International Conference on Computational Linguistics*, pp. 930–934, Nantes.

POLLARD, Carl ; SAG, Ivan (1994) : *Head-driven Phrase Structure Grammar*, The University of Chicago Press, Chicago.

PRINCE, Alan ; SMOLENSKY, Paul (1993) : *Optimality Theory: Constraint Interaction in Generative Grammar*, Technical Report n° RuCCS #2, Piscataway, NJ, Rutgers University, Center for Cognitive Science.

SCHULZ, Michael (1999) : "Inkrementelle Analyse natürlicher Sprache mit Constraints", Studienarbeit, Fachbereich Informatik, Universität Hamburg.

TESNIÈRE, Lucien (1959) : *Eléments de syntaxe structurale*, Klincksieck, Paris.

TSANG, Edward (1993) : *Foundations of Constraint Satisfaction*, Academic Press, London.

WAHLSTER, Wolfgang (1993) : "Verbmobil – Translation of face-to-face dialogs", *in Proceedings of the 3rd European Conference on Speech Communication and Technology (EUROSPEECH-93)*, pp. 29–38, Berlin.