Decision Procedures for Dependency Parsing Using Graded Constraints

Wolfgang Menzel and **Ingo Schröder**
(menzel | ingo.schroeder@informatik.uni-hamburg.de)
Fachbereich Informatik, Universität Hamburg
Vogt-Kölln-Straße 30, 22527 Hamburg, Germany

Abstract

We present an approach to the parsing of dependency structures which brings together the notion of parsing as candidate elimination, the use of graded constraints, and the parallel disambiguation of related structural representations. The approach aims at an increased level of robustness by accepting constraint violations in a controlled way, combining redundant and possibly conflicting information on different representational levels, and facilitating partial parsing as a natural mode of behavior.

1 Introduction

Language understanding is based on a variety of contributions from different representational levels. From this perspective, one of the most attractive features of dependency based grammar models seems to be their relational nature which allows to accommodate various kinds of relationships in a very similar fashion. Since the basic representational framework is a rather general one it can be (re-)interpreted in many different ways. Thus, dependency relations lend themselves to model the surface syntactic structure of an utterance (with labels like subject-of, direct-object-of, determiner-of, etc.), its thematic structure (with labels like agent-of, theme-of, etc.) and even the referential structure (with labels like referential-identity, part-of, possessor-of, etc.). This representational similarity obviates the necessity to integrate too many disparate informational contributions into a single tree-like representation. Instead, representational levels can be separated from each other in a clean manner with appropriate mappings being defined to relate the different components to each other.

Another less obvious advantage of dependency formalisms is their suitability for the application of eliminative parsing techniques. In contrast to the traditional view on parsing as a constructive process, which builds new tree structures from elementary building blocks and intermediate results, eliminative approaches organize structural analysis as a candidate elimination procedure, removing unsuitable interpretations from a maximum set of possible ones. Hence, parsing is constructed as a strictly monotonic process of ambiguity reduction.

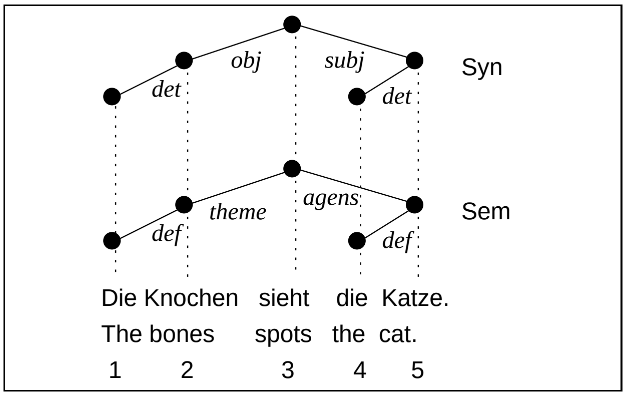

Figure 1: Collection of dependency trees: Each tree represents a description level.

In this paper we describe different algorithmic solutions to eliminative parsing. The novel contribution consists in the use of graded constraints, which allow to model traditional grammar regularities as well as preferences and defaults. Details of the linguistic modeling are presented by Heinecke and Schröder (1998).

2 Eliminative Parsing

The idea of eliminative parsing is not a novel one and virtually every tagger can be considered a candidate elimination procedure which removes items from the maximum set of tags according to different decision criteria. Interestingly, dependency-based parsing can be viewed as a generalized tagging procedure. One of the first parsing systems which built on this property is the Constraint Grammar approach (Karlsson et al., 1995). Underspecified dependency structures are represented as syntactic tags[1] and disambiguated by a set of constraints that exclude inappropriate readings. Maruyama (1990) first tried to extend the idea to allow the treatment of complete dependency structures. Therefore, he has to generalize the notion of a "tag" to pairs consisting of a label and the identifier of the dominating node, i. e., the tagset needs to become sensitive to the individual tokens of the utterance under consideration sacrificing the status of the tagset being fixed a-priori. As in the case of atomic tags, constraints are specified which delete inappropriate dependency relations from the initial space of possibilities. The approach is not restricted to linear input strings but can also treat lattices of input tokens, which allows to accommodate lexical ambiguity as well as recognition uncertainty in speech understanding applications (Harper et al., 1994).

Obviously, it is again the relational nature of dependency models which provides for the applicability of candidate elimination procedures. Since the initial state of the analysis is given by an – admittedly large – set of possible dependency relations per token, the problem space remains finite for finite utterances. An analogous approach for constituency-based grammar models would encounter considerable difficulties, because the number and the kind of non-terminal nodes which need to be included in the tagset remains completely unclear prior to the parsing itself.

Eliminative approaches to parsing come along with a number of interesting properties which make them particularly attractive as computational models for language comprehension.

1. As long as constraint checking is restricted to strictly local configurations of dependency relations the decision procedures inherits this locality property and thus exhibits a considerable potential for concurrent implementation (Helzerman and Harper, 1992).
2. Since partial structural descriptions are available concurrently they can be compared in a competitive manner. Note however that such a comparison imposes additional synchronization and communication requirements on parallel realizations.
3. As the eliminative approach considers parsing a procedure of disambiguation, the quality of the results to be expected becomes directly related to the amount of effort one is prepared to spend. This is a clear contrast to constructive methods which, upon request usually will attempt to generate alternative interpretations, thus leading to a corresponding decrease of clarity about the structural properties of the input utterance (in terms of Karlsson et al. (1995)).
4. The progress of disambiguation can easily be assessed by constantly monitoring the size of value sets. Moreover, under certain conditions the amount of remaining effort for obtaining a completely disambiguated solution can be estimated. This appears to be an important characteristic for the development of anytime procedures, which are able to adapt their behavior with respect to external resource limitations (Menzel, 1994; Menzel, 1998).

3 Graded Constraints

Both the comparison of competitive structural hypotheses as well as the adaptation to resource limitations require to generalize the approach by allowing constraints of different strength. While traditional constraints only make binary decisions about the well-formedness of a configuration the strength of a constraint additionally reflects a human judgment of how critical a violation of that particular constraint is considered. Such a grading, expressed as a penalty factor, allows to model a number of observations which are quite common to linguistic structures:

- Many phenomena can more easily be described as preferences rather than strict regularities. Among them are structural conditions about attachment positions or linear ordering as well as selectional restrictions.

[1] In this framework tags denote, for instance, the subject of the sentence, a determiner modifying a noun to the right, a preposition modifying a noun to the left etc. However, only the category of the dominating node is specified, not its exact identity.

- Preferences usually reflect different frequencies of use and in certain cases can be extracted from large collections of sample data.
- Some linguistic cues are inherently uncertain (e. g., prosodic markers), and therefore resist a description by means of crisp rule sets.

By introducing graded constraints the parsing problem becomes an optimization problem aiming at a solution which violates constraints that are as few and as weak as possible. This, on the one hand, leads to a higher degree of structural disambiguation since different solution candidates now may receive a different score due to preference constraints. Usually, a complete disambiguation is achieved provided that enough preferential knowledge is encoded by means of constraints. Remaining ambiguity which cannot be constrained further is one of the major difficulties for systems using crisp constraints (Harper et al., 1995). On the other hand, weighed constraints allow to handle contradictory evidence which is typical for cases of ill-formed input. Additionally, the gradings are expected to provide a basis for the realization of time adaptive behavior.

One of the most important advantages which can be attributed to the use of graded constraints is their ability to provide the mapping between different levels in a multi-level representation, where many instances of preferential relationships can be found. This separation of structural representations facilitates a clear modularization of the constraint grammar although constraints are applied to a single computational space. In particular, the propagation of gradings between representational levels supports a mutual compensation of information deficits (e. g., a syntactic disambiguation can be achieved by means of semantic support) and even cross-level conflicts can be arbitrated (e. g., a syntactic preference might be inconsistent with a selectional restriction).

Combining candidate elimination techniques, graded constraints, and multi-level disambiguation within a single computational paradigm aims first of all at an increased level of robustness of the resulting parsing procedure (Menzel and Schröder, 1998). Robustness is enhanced by three different contributions:

1. The *use of graded constraints* makes constraint violations acceptable. In a certain sense, the resulting behavior can be considered a kind of constraint retraction which is guided by the individual gradings of violated constraints. Therefore, a "blind" weakening of the constraint system is avoided and hints for a controlled application are preserved.
2. The *propagation of evidence among multiple representational levels* exploits the redundancy of the grammar model about different aspects of language use in order to compensate the loss of constraining information due to constraint retraction. Naturally, the use of additional representational levels also means an expansion of the search space, but this undesired effect can be dealt with because once a single point of relative certainty has been found on an arbitrary level, the system can use it as an anchor point from which constraining information is propagated to the other levels. For instance, if selectional restrictions provide enough evidence for a particular solution, an ambiguous case can be resolved. Even contradictory indications can be treated in that manner. In such a case conflict resolution is obtained according to the particular strength of evidence resulting from the observed constraint violations.
3. The *seamless integration of partial parsing* is achieved by allowing arbitrary categories (not just finite verbs) to serve as the top node of a dependency tree. Of course, these configurations need to be penalized appropriately in order to restrict their selection to those cases where no alternative interpretations remain. Note that under this approach partial parsing is not introduced by means of an additional mechanism but falls out as a general result of the underlying parsing procedure.

Certainly, all the desired advantages mentioned above become noticeable only if a constraint modeling of grammatical relations can be provided which obeys the rather restrictive locality conditions and efficient implementations of the disambiguation procedure become available.

4 Parsing As Constraint Satisfaction

Parsing of natural language sentences can be considered a constraint satisfaction problem if one manages to specify exactly what the constraint variables should be, how constraints can be used to find appropriate value assignments, and how these value assignments represent specific structural solutions to the parsing problem. These are the questions we address in this section.

The original definition of constraint dependency grammars by Maruyama (1990) is extended to *graded constraint dependency grammars* which are represented by a tuple $\langle \Sigma, L, C, \phi \rangle$. The lexicon Σ is a set of word forms each of which has some lexical information associated with it. The set of representational levels $L = \{\langle l_1, L_1 \rangle, \ldots, \langle l_n, L_n \rangle\}$ consists of pairs $\langle l_i, L_i \rangle$ where l_i is a name of the ith representational level and $l_i^j \in L_i$ is the jth appropriate label for level l_i. Think of $\langle \mathsf{Syn}, \{subj, obj, det\} \rangle$ as a simple example of a representational level.

The constraints from the set C can be divided into disjunct subsets C^i with $C = \bigcup_i C^i$ depending on the constraints' arity i which denotes the number of constraint variables related by the constraint. Mainly due to computational reasons, but also in order to keep the scope of constraints strictly local, at most binary constraints, i. e., constraints with arity not larger then two, are considered: $C = C^1 \cup C^2$.[2]

The assessment function $\phi : C \mapsto [0, 1]$ maps a constraint $c \in C$ to a weight $\phi(c)$ which indicates how serious one considers a violation of that constraint. Crisp constraints which may not be violated at all, i. e., they correspond to traditional constraints, have a penalty factor of zero ($\phi(c) = 0$) while others have higher grades (i. e., $0 < \phi(c) \leq 1$) and thus may be violated by a solution.[3]

[2]The restriction to at most binary constraints does not decrease the theoretical expressiveness of the formalism but has some practical consequences for the grammar writer as he/she occasionally has to adopt rather artificial constructs for the description of some linguistic phenomena (Menzel and Schröder, 1998).

[3]Constraints c with $\phi(c) = 1.0$ are totally ineffective as will become clear in the next paragraphs.

Given a natural language sentence $W = \langle w_1, \ldots, w_m \rangle$ and a graded constraint dependency grammar the parsing problem can be stated as follows: For each representational level l_i and each word of the sentence w_j a constraint variable v_i^j is established. Let the set of all constraint variables be V. The domain $\text{dom}(v_i^j) = L_i \times \{0, 1, \ldots, j-1, j+1, \ldots n\}$ of variable v_i^j, i. e., the set of possible values for that variable, consists of all pairs $\langle l, k \rangle$ where l is an appropriate label for level l_i (i. e., $l \in L_i$) and k is the index of the dominating word w_k (i. e., word w_j is subordinated to word w_k) or zero if the word w_j is the root of the dependency structure on level l_i.

A problem candidate ρ of the parsing problem is a unique value assignment to each of the constraint variables. In other words, for each variable v_i^j a single value $\rho(v_i^j) = d_i^j \in \text{dom}(v_i^j)$ has to be chosen.

The solution is the problem candidate that violates less and/or less important constraints than any other problem candidate. In order to make this intuitive notion more formal the function ϕ is extended to assess not only constraints but also problem candidates ρ.

$$\phi(\rho) = \prod_a \prod_{c \in C^a} \prod_{\vec{v} \in V^a} \phi(c, \rho(\vec{v}))$$
$$\text{where } a, 1 \leq a \leq 2, \text{ is the arity and } \vec{v} \text{ is a tuple of variables}$$

A single constraint c can be violated once, more than once or not at all by a problem candidate since constraints judge local configurations, not complete problem candidates.

$$\phi(c, \vec{d}) = \begin{cases} \phi(c) & : \text{if } \vec{d} \text{ violates } c \\ 1.0 & : \text{else} \end{cases}$$
$$\text{where } \vec{d} \text{ is a (unary or binary) tuple of values}$$

Note that satisfying a constraint does not change the grade of the problem candidate because of the multiplicative nature of the assessing function.

The final solution ρ_s is found by maximum selection.

$$\rho_s = \arg\max_\rho \phi(\rho)$$

Thus the system uniquely determines the dominating node for each of the input word forms. Additional conditions for well-formed structural representations like projectivity or the absence of cyclic dependency relations must be taken extra care of.

In our current implementation the acyclicity property is ensured by a special built-in control structure, while projectivity has to be established by means of specifically designed constraints. This enables the grammar writer to carefully model the conditions under which non-projective dependency structures may occur. Note, however, that there are cases of non-projective structures that cannot be eliminated by using only local (i. e. at most binary) constraints.

Another problem arises from the fact that constraints are universally quantified and existence conditions (like "there must be a subject for each finite verb") cannot be expressed directly. This difficulty, however, is easily overcome by the introduction of "reverse" dependencies on additional auxiliary levels, which are used to model the valency requirements of a dominating node. Since each valency to be saturated requires an auxiliary level, the overall number of levels in a multi-level representation may easily grow to more than some ten.

Moreover, the formal description given so far is only valid for linear strings but not for word graphs. An extension to the treatment of word graphs requires the modification of the notion of a problem candidate. While in the case of linear input an assignment of exactly one value to *each* variable represents a possible structure, this is not valid for word graphs. Instead, only those variables that correspond to a word hypothesis *on one particular path* through the word graph must receive a unique value while all other variables must be assigned no value at all. This additional path condition usually is also not encoded as normal grammar constraints but must be guaranteed by the control mechanism.

5 An Example

To illustrate the formalization we now go through an example. To avoid unnecessary details we exclude the treatment of auxiliary levels from our discussion, thus restricting ourselves to the modeling of valency possibilities and abstracting from valency necessities. The problem is simplified further by selecting an extremely limited set of dependency labels. Consider again the example from Figure 1:

(1) Die Knochen$_{\mathsf{pl}}$ sieht$_{\mathsf{sg}}$ die Katze$_{\mathsf{sg}}$.
The bones sees the cat.
"The cat sees the bones."

Two representational levels, one for syntactic functions and one for semantic case-fillers, are introduced:

$$L = \{ \quad \langle \mathsf{Syn}, \{subj, obj, det\}\rangle, \\ \langle \mathsf{Sem}, \{agent, theme, def\}\rangle \quad \}$$

Figure 2 contains some of the constraints necessary to parse the example sentence. Basically, a constraint consists of a logical formula which is parameterized by variables (in our example X and Y) which can be bound to an edge in the dependency tree. It is associated with a name (e. g., SubjNumber) and a class (e. g., Subj) for identification and modularization purposes respectively. The constraint score is given just before the actual formula. Selector functions are provided which facilitate access to the label of an edge (e. g., X.label) and to lexical properties of the dominating node (e. g., $\mathsf{X}\uparrow\mathsf{num}$) and the dominated one (e. g., $\mathsf{X}\downarrow\mathsf{num}$). Being universally quantified, a typical constraint takes the form of an implication with the premise describing the conditions for its application. Accordingly, the constraint SubjNumber of Figure 2 reads as follows: For each subject (X.label=subj) it holds that the dominated and the dominating nodes agree with each other in regard to number ($\mathsf{X}\downarrow\mathsf{num}=\mathsf{X}\uparrow\mathsf{num}$).

Figure 1 from the introduction graphically presents the desired solution structure which is repeated as a constraint variable assignment in Figure 3.[4]

All (shown) constraints are satisfied by the variable assignment except SubjOrder which is violated once, viz. by the assignment $v^5_{\mathsf{Syn}} = \langle subj, 3\rangle$. Therefore, the structure has a score

[4]The presentation of solutions as dependency trees becomes less intuitive as soon as more levels, especially auxiliary levels, are introduced.

{X} : SubjNumber : Subj : 0.1 :
X.label=subj $\rightarrow$ X↓num=X↑num
'Subjects agree with finite verbs regarding number.'

{X} : SubjOrder : Subj : 0.9 :
X.label=subj $\rightarrow$ X↓pos<X↑pos
'Subjects are usually placed in front of the verb.

{X} : SemType : SelRestr : 0.8 :
X.label $\in$ { agent, theme } $\rightarrow$
type_match(X↑id, X.label, X↓id)
'Verbs restrict the semantic types of their arguments.'

{X, Y} : SubjAgentObjTheme : Mapping : 0.2 :
X↑id=Y↑id $\wedge$ X↓id=Y↓id $\rightarrow$
(X.label=subj $\leftrightarrow$ Y.label=agent) $\wedge$
(X.label=obj $\leftrightarrow$ Y.label=theme)
'The subject is the agent and the object the theme.'

{X, Y} : Unique : Unique : 0.0 :
X.label $\in$ {subj,obj,agent,theme} $\wedge$ X↑id=Y↑id
$\rightarrow$ Y.label$\neq$X.label
'Some labels are unique for a given verb.'

Figure 2: Some of the constraints needed for the disambiguation of the example sentence.

equal to the constraint's score, namely 0.9. Furthermore there is no structure which has a better assessment. The next example is similar to the last, except that the finite verb appears in plural form now.

(2) Die Knochen$_{\text{pl}}$ sehen$_{\text{pl}}$ die Katze$_{\text{sg}}$.
The bones see the cat.
"The bones see the cat."

A solution structure analogous to the one discussed above would have a score of 0.09 because not only the constraint SubjOrder but also the constraint SubjNumber would have been violated. But the alternative structure where the

$$
\begin{array}{lcllcl}
v^1_{\mathsf{Syn}} & = & \langle det, 2\rangle & v^1_{\mathsf{Sem}} & = & \langle def, 2\rangle \\
v^2_{\mathsf{Syn}} & = & \langle obj, 3\rangle & v^2_{\mathsf{Sem}} & = & \langle theme, 3\rangle \\
v^3_{\mathsf{Syn}} & = & \langle root, 0\rangle & v^3_{\mathsf{Sem}} & = & \langle root, 0\rangle \\
v^4_{\mathsf{Syn}} & = & \langle det, 5\rangle & v^4_{\mathsf{Sem}} & = & \langle def, 5\rangle \\
v^5_{\mathsf{Syn}} & = & \langle subj, 3\rangle & v^5_{\mathsf{Sem}} & = & \langle agent, 3\rangle
\end{array}
$$

Figure 3: Constraint variable assignments corresponding to the dependency trees in Figure 1

subj/*agent* and the *obj*/*theme* edges are interchanged (meaning that the bones do the seeing) has a better score of 0.8 this time because it only violates the constraint SemType. This result obviously resembles performance of human beings who first of all note the semantic oddness of the example (2) before trying to repair the syntactic deviations when reading this sentence in isolation.

Thus, the approach successfully arbitrates between conflicting information from different levels, using the constraint scores to determine which of the problem candidates is chosen as the final solution.

6 Constraint Satisfaction Procedures

A lot of research has been carried out in the field of algorithms for constraint satisfaction problems (Meseguer, 1989; Kumar, 1992) and constraint optimization problems (Tsang, 1993).

Although CSPs are NP–complete problems in general and, therefore, one cannot expect a better than exponential complexity in the worst case, a lot of methods have been developed to allow for a reasonable complexity in most practical cases. Some heuristic methods, for instance, try to arrive at a solution more efficiently at the expense of giving up the property of correctness, i. e., they find the globally best solution in most cases while they are not guaranteed to do so in all cases.

This allows to influence the temporal characteristics of the parsing procedure, a possibility which seems especially important in interactive applications: If the system has to deliver a reasonable solution within a specific time interval a dynamic scheduling of computational resources depending on the remaining ambiguity and available time is necessary. While different kinds of search are more suitable with regard to the correctness property, local pruning strategies lend themselves to resource adaptive procedures.

6.1 Consistency-Based Methods

As long as only crisp constraints are considered, procedures based on local consistency, particularly arc consistency can be used (Maruyama, 1990; Harper et al., 1995). These methods try to delete values from the domain of constraint variables by considering only local information and have a polynomial worst case complexity.

Unfortunately, they possibly stop deleting values before a unique solution has been found. In such a case, even if arc consistency has been established one cannot be sure whether the problem has zero, one, or more than one solution because alternative value assignments may be locally consistent, but globally mutually incompatible. Consequently, in order to find actual solutions an additional search has to be carried out for which, however, the search space is considerably reduced already.

6.2 Search

The most straightforward method for constraint parsing is a simple search procedure where the constraint variables are successively bound to values and these value assignments are tested for consistency. In case of an inconsistency alternative values are tried until a solution is found or the set of possible values is exhausted. The basic search algorithm is Branch & Bound which exploits the fact that the score of every subset of variable assignments is already an upper bound of the final score. Additional constraint violations only make the score worse, because the scores of constraints do not exceed a value of one. Therefore, large parts of the search space can be abandoned as soon as the score becomes too low. To further improve the efficiency an agenda is used to sort the search space nodes so that the most promising candidates are tried first. By not allowing the agenda to grow larger than a specified size, one can exclude search states with low scores from further consideration. Note that correctness cannot be guaranteed in that case anymore. Figure 4 presents the algorithm in pseudo code notation.

Unfortunately, the time requirements of the search algorithms are almost unpredictable since an intermediate state of computation does not give a reliable estimation of the effort that remains to be done.

6.3 Pruning

As explained in Section 6.1 consistency-based procedures use local information to delete values from the domain of variables. While these methods only do so if the local information suffices to guarantee that the value under consideration can safely be deleted, pruning goes one step further. Values are successively selected for deletion based on a heuristic (i. e., possibly incorrect) assessment until a single solution remains (cf. Figure 5). The selection function considers only local information (as do the consistency-based methods) for efficiency reasons. Taking into account global optimality criteria would not help at all since then the selection would be as difficult as the whole problem, i. e., one would have to expect an exponential worst-case complexity.

```
procedure ConstraintSearch
  set b := 0.0                                  ; best score so far
  set r := ∅                                    ; set of solutions
  set a := {⟨∅, V, 1.0⟩}                         ; agenda
  while a ≠ ∅ do                                ; process agenda
    get best item ⟨B, V, s⟩ from agenda a       ; best first
    if V = ∅ then                               ; complete assignment?
      if s = b then                             ; best so far?
        add ⟨B, V, s⟩ to r                      ; equally good
      else
        set r := {⟨B, V, s⟩}                    ; better
        set b := s
      fi
    fi
    select v ∈ V                                ; try next free variable
    set V' := V\{v}
    foreach d ∈ dom(v) do                       ; try all values
      set B' := B ∪ {⟨v, d⟩}
      compute new score s' for B'
      if s' ≥ b then                            ; already worse?
        add ⟨B', V', s'⟩ to agenda
      fi
    done
    truncate agenda a (if desired)
  done
```

Figure 4: Search procedure for constraint parsing: Best-first branch & bound algorithm with limited agenda length (*beam search*)

```
procedure pruning(V)
  while ∃(v_i^j ∈ V) : |dom(v_i^j)| > 1 do
    select ⟨v̄, d̄⟩ to be deleted
    delete d̄ from domain v̄
  done
```

Figure 5: The pruning algorithm repeatedly selects and deletes values from the domain of variables.

Obviously, the selection heuristics plays the major role while pruning.

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.

Formally the following formulas (using the notation of Section 4) determine the value $\bar{d}$ of variable $\bar{v}$ to deleted next:

$$s(v,d) \quad = \quad \min_{v' \in V \setminus \{v\}} \frac{\sum\limits_{d' \in \mathrm{dom}(v')} \mathrm{score}(d,d')^2}{|\mathrm{dom}(v')|}$$

$$\langle \bar{v}, \bar{d} \rangle \quad = \quad \arg\min_{\langle v,d \rangle} s(v,d)$$

where $\mathrm{score}(d,d')$ is the accumulated assessment of the pair of subordinations $\langle d, d' \rangle$:

$$\mathrm{score}(d,d') = \prod_{c \in C^1} \phi(c,d) \cdot \phi(c,d') \cdot \prod_{c \in C^2} \phi(c,d,d')$$

While this heuristics works quite well for linear strings it fails if one switches to word graphs. Figure 6 gives an example of a very simple word graph which cannot be handled correctly by the simple heuristics.

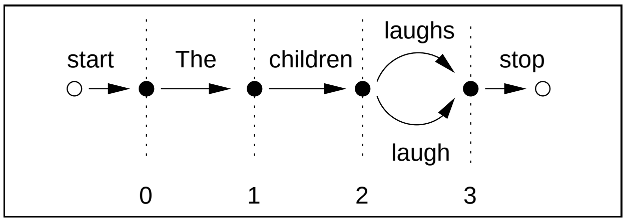

Figure 6: Simple word graph

Alternative word hypotheses whose time spans are not disjunct do not support each other by definition. Therefore, the subordination of children under laugh in Figure 6 is equally disfavored by laughs as is the subordination of children under laughs by laugh. Unfortunately, this lack of support is not based on negative evidence but on the simple fact that laugh and laughs are temporal alternatives and may, thus, not be existent in a solution simultaneously. Since the simple heuristics does not know anything about this distinction it may arbitrarily select the wrong value for deletion.

A naive extension to the above heuristics would be to base the assessment not on the minimal support from *all* variables but on the combined support from those variables that share at least one path through the word graph with the variable under consideration. But the path criterion is computationally expensive to compute and, therefore, needs to be approximated during pruning. Instead of considering all possible paths through the graph, we compute the maximum support at each time point t and on each level l and select the minimum of these values to be removed from the space of hypotheses:

$$s(v,d,l,t) \quad = \quad \max_{\substack{v' \in V \setminus \{v\} \\ t \in \mathrm{time}(v') \\ \mathrm{level}(v') = l}} \frac{\sum\limits_{d' \in \mathrm{dom}(v')} \mathrm{score}(d,d')^2}{|dom(v')|}$$

$$s(v,d) \quad = \quad \min_{t,l} s(v,d,l,t)$$

$$\langle \bar{v}, \bar{d} \rangle \quad = \quad \arg\min_{\langle v,d \rangle} s(v,d)$$

where $\mathrm{time}(v)$ denotes the time interval of the word hypothesis (cf. Figure 6 or 7) that corresponds to the variable v and $\mathrm{level}(v)$ denotes the representational level of variable v.

For temporally overlapping nodes the procedure selects a single one to act as a representative of all the nodes within that particular time slice. Therefore, information about the exact identity of the node which caused the lack of support is lost. But since the node which gives a maximum support is used as a time-slice representative it seems likely that any other choice might be even worse.

Although preliminary experiments produced promising results (around 3 % errors) it can be expected that the quality of the results depends on the kind of grammar used and utterances analyzed. Since the problem deserves further investigation, it is too early to give final results.

The example in Figure 7 shows a simple case that demonstrates the shortcomings of the refined heuristics. Although these and children are not allowed to occur in a solution simultaneously, exactly these two words erroneously remain undeleted and finally make up the subject in the analysis. First, all values for the article a are deleted because of a missing number agreement with the possible dominating nodes and thereafter the values for the word houses are discarded since the semantic type does not match the selectional restrictions of the verb very well.

The heuristics is not aware of the distinction between the time points and word graph nodes and, therefore, counts the determiner these as supporting the noun children.

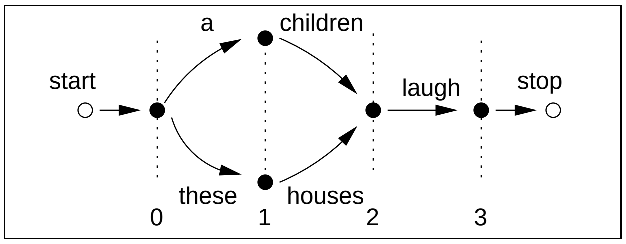

Figure 7: Hypothetic simplified word graph which may be analyzed "incorrectly" by the time slice pruning heuristics.

7 Efficiency Issues

Although pruning strategies bear a great potential for efficient and time-adaptive parsing schemes, the absolute computational expenses for a "blind" application of constraints are still unacceptably high. Additional techniques have to be employed to decrease actual computation times. One of the starting points for such improvements is the extremely large number of constraint evaluations during parsing: A few million constraint checks are quite common for realistic grammars and sentences of even modest size.

Two approaches seem to be suitable for the reduction of the number of constraint evaluations:

- Reduced application of constraints: A detailed analysis of how constraints are applied and under what circumstances they fail shows that most constraint checks are "useless" since the tested constraint is satisfied for some trivial reason. For instance, because most constraints are very specific about what levels are constrained and whether and how the dependency edges are connected, this information can be exploited in order to reduce the number of constraint checks. By applying constraints only to the relevant levels the number of constraint evaluation has been cut down to (at most) 40%. Taking into account the topological structure of the edges under consideration improves the efficiency by another 30% to 50%.

- Reduction of the number of constraint variables: A typical grammar contains a relatively large number of representational levels and for most word forms there are several entries in the lexicon. Since the lexical ambiguity of the word form usually is relevant only to one or very few levels, constraint variables need not be established for all lexical entries and all levels. For instance, the German definite determiner die has eight different morpho-syntactic feature combinations if one only considers variations of gender, case, and number. All these forms behave quite similarly with respect to non-syntactic levels. Consequently, it makes no difference if one merges the constraint variables for the non-syntactic levels except that now less constraint checks must be carried out. By considering the relevance of particular types of lexical ambiguity for constraint variables of different levels one achieves an efficient treatment of disjunctive feature sets in the lexicon (Foth, 1998). This technique reduced the time requirements by 75% to 90% depending on the details of the grammatical modeling. In particular, a clean modularization, both in the constraint set and the dictionary entries, results in considerable gains of efficiency.

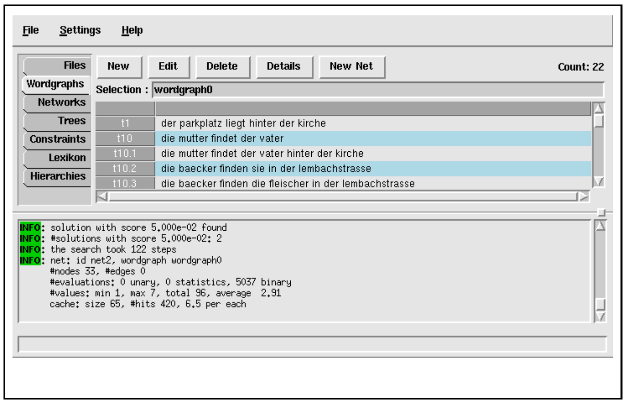

Figure 8: Window of the graphical grammar environment xcdg

In order to support the grammar writer, a graphical grammar environment has been developed (cf. Figure 8). It includes an editor for dependency trees (cf. Figure 9) which allows to detect undesired constraint violations easily.

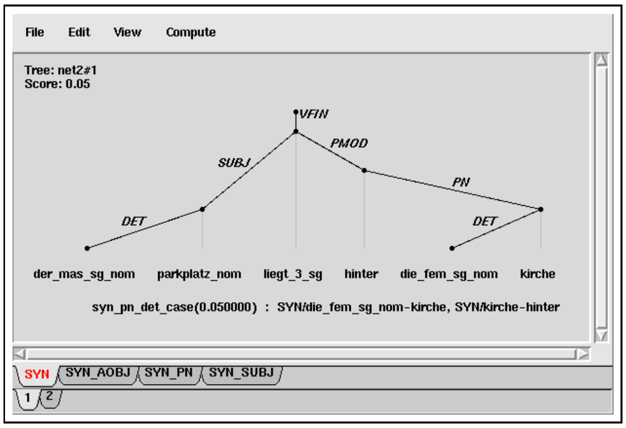

Figure 9: Window of the editor for dependency trees

8 Conclusion

A parsing approach aiming at dependency structures for different representational levels has been presented. The approach improves in robustness by assessing partial structures, integrating multiple representational levels, and employing partial parsing techniques. Knowledge about the grammar but also extralinguistic information about the domain under consideration is encoded by means of graded constraints which allows for the arbitration between conflicting information. Different decision procedures for the defined parsing problem have been introduced and some efficiency issues have been discussed.

The approach has successfully been applied to a number of modestly sized projects (Menzel and Schröder, 1998; Heinecke et al., 1998).

Further investigations will focus on possibilities for incremental processing of speech input and the realization of resource adaptive behavior.

References

Kilian Foth. 1998. Disjunktive Lexikoninformation im eliminativen Parsing. Studienarbeit, FB Informatik, Universität Hamburg.

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, and 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, Randall A. Helzermann, C. B. Zoltowski, B. L. Yeo, Y. Chan, T. Steward, and B. L. Pellom. 1995. Implementation issues in the development of the PARSEC parser. *Software – Practice and Experience*, 25(8):831–862.

Johannes Heinecke and Ingo Schröder. 1998. Robust analysis of (spoken) language. In *Proc. KONVENS '98*, Bonn, Germany.

Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel, and Ingo Schröder. 1998. Eliminative parsing with graded constraints. In *Proc. Joint Conference COLING/ACL '98*.

Randall A. Helzerman and Mary P. Harper. 1992. Log time parsing on the MasPar MP–1. In *Proceedings of the 6th International Conference on Parallel Processing*, pages 209–217.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors. 1995. *Constraint Grammar – A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin, New York.

Vipin Kumar. 1992. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13(1):32–44.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Annual Meeting of the ACL*, pages 31–38, Pittsburgh.

Wolfgang Menzel and Ingo Schröder. 1998. Constraint-based diagnosis for intelligent language tutoring systems. In *Proceedings of the IT&KNOWS Conference at the IFIP '98 Congress*, Wien/Budapest.

Wolfgang Menzel. 1994. Parsing of spoken language under time constraints. In A. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 560–564, Amsterdam.

Wolfgang Menzel. 1998. Constraint Satisfaction for Robust Parsing of Spoken Language. *Journal for Experimental and Theoretical Artificial Intelligence*, 10:77–89.

Pedro Meseguer. 1989. Constraint satisfaction problems: An overview. *AI Communications*, 2(1):3–17.

E. Tsang. 1993. *Foundations of Constraint Satisfaction*. Academic Press, Harcort Brace and Company, London.