Subtree Parsing to Speed up Deep Analysis

Kilian Foth, Wolfgang Menzel

(foth | wolfgang@nats.informatik.uni-hamburg.de)

Fachbereich Informatik, Universität Hamburg

Vogt-Kölln-Straße 30, 22527 Hamburg, Germany

Abstract

Within a grammar formalism that treats syntax analysis as a global optimization problem, methods are investigated to improve parsing performance by recombining the solutions of smaller and easier subproblems. The robust nature of the formalism allows the application of this technique with little change to the original grammar.

1 Introduction

Treating natural language parsing as a problem of constraint optimization (Freuder and Wallace, 1992) comes with a number of attractive computational properties. It requires formulating the parsing problem as one of assigning values to variables, where values can be constructed in a way as to describe relationships between the variables themselves. This possibility allows us to interpret particular values as dependencies between word forms in a natural language utterance. Constraints can then be defined which check local structural configurations for being admissible and assign scores to them.

A parser based on these principles attempts to determine a structural description which is optimal with respect to an assessment function accumulating local scores into a global measure of well-formedness. Among the benefits of such an approach some are particularly noteworthy:

Fail-soft robustness: Symbolic grammars are combined with robust processing capabilities typical for stochastic approaches. Due to their graded nature constraints are defeasible and constraint violations can easily be handled as part of the normal decision procedure (Foth, Menzel, and Schröder, submitted). In addition, constraint violations provide rich diagnostic information about the structure under consideration. They can be used to guide the further analysis or to explain expectation violations of different kinds.

Information fusion: Constraints offer an ideal interface to integrate possibly contradictory information from different external sources in a graceful manner. This allows the parser to include hypotheses from shallow processing components like a tagger or a chunker, even if this information is uncertain (Foth and Hagenström, 2002). Confidence scores provided by such components can not only be used to arbitrate in cases of conflict, but also to effectively guide the optimization process towards the desired optimum (Schröder, 2002).

Multilevel disambiguation: Different description levels based on word-to-word relations (e.g. for aspects of syntax, semantics, reference or information structure) can be treated in a quasi-parallel manner. This contributes a kind of structural redundancy which allows the analysis procedure to overcome partial conflicts or information deficits on one level using information from another one (Schröder et al., 2000).

Anytime behaviour: A number of different solution methods for solving the optimization problem is available, including both complete and heuristic ones. They all have in common the ability to trade parsing quality against processing time. Even interruptible algorithms have been devised which are able to provide a solution at an arbitrary point in time. While trying to repair the shortcomings of an initial assumption they are able to supply results of steadily increasing quality by carrying out local transformations, successively improving consistency and coherence of a solution candidate (Foth, Menzel, and Schröder, 2002).

Weighted Constraint Dependency Grammar (WCDG) is a computational framework for investigating the potential of constraint optimization techniques for natural language parsing. So far, however, parsing a sentence has always been treated as a global optimization problem. Since constraint optimization is known to be $\mathcal{NP}$ hard, this incurs a serious efficiency problem. Although its effects can be mitigated somehow by applying heuristic decisions together with the fundamentally time-adaptive nature of the available solution procedures, most longer sentences suffer from quite excessive runtime requirements.

To improve the general accuracy/time ratio in such cases this paper investigates different possibilities to break down large parsing problems into smaller subtasks whose results can later be recombined into a solution for the original problem. The rationale behind such an approach is that many attachment problems which have to be solved when parsing a sentence can also be treated in a fairly restricted context. Solving these local problems as local optimization subtasks might therefore save precious processing time which is needed badly to tackle the notoriously difficult attachment problems on the global sentence level. It can be shown that by freezing the optimum solution for local areas of a sentence, indeed a considerable speedup can be achieved, which in turn leads to an improvement in parsing accuracy if only limited processing time is made available.

Section 2 gives a short introduction to WCDG and the repair-based transformation method used throughout the paper. Experiments to determine useful criteria for sentence segmentation are reported in Section 3, while Section 4 investigates different options for handling dependency relations crossing segment boundaries.

2 WCDG parsing as transformation-based repair

Constraints in a WCDG are conditions defined on partial structural descriptions consisting of two dependency edges at most. They have access to lexical and positional information associated to the word forms involved and the label of the dependency relation established between them.

If the word forms of a sentence are used to define a set of variables which receive as values pairs consisting of (1) another word form which they are attached to and (2) the relation with which they modify their governor, parsing becomes a procedure of constraint satisfaction: All variables get assigned a value which is compatible with all the constraints of the grammar. Together with the special subordination `ROOT` which, if assigned to a variable, actually avoids the subordination of the corresponding word form, and a built-in global constraint prohibiting cyclical subordination, this definition of a constraint satisfaction problem guarantees that any admissible value assignment corresponds to one or possibly several dependency trees.

To accommodate lexical ambiguity (and if desired speech recognition uncertainty) the constraint solver works on word graphs instead of simple word form sequences. In addition to the task of determining a structural description of the input sentence it has also to select among the available alternatives in the graph, accomplishing lexical and structural disambiguation at the same time. Another built-in global constraint ensures that only alternatives corresponding to a single path through the graph are considered.

Solutions to a WCDG parsing problem may violate some constraints as long as no better structural interpretation is available. To rank competing hypotheses in such cases, constraints are associated with weights between zero and one, where higher values correspond to less important constraints. Dependency edges under consideration receive a confidence score which is calculated as the product of weights from all the constraints violated by that edge. For complex structures scores are also accumulated by multiplying them out, and the description with the maximum score can be determined.

Constraint weights serve different purposes in WCDG:

- They keep the parsing problem manageable even in spite of pervasive constraint violations. Parsing quality degrades gracefully in cases of extragrammatical input or limited temporal resources.

- They provide a criterion to decide between competing solutions to a parsing problem. If desired, a complete disambiguation can almost always be achieved.

- They guide the parsing problem towards the desired optimum.

- They allow the integration of uncertain knowledge, including default information and preferences, which however is outvoted if conclusive counter-evidence is available.

Simply being logical formulae on the admissibility of certain fragments of a dependency structure, constraints are neutral with respect to the parsing strategy adopted. They can be used to remove dependency relations from an underspecified space of structural hypotheses (eliminative parsing) or to license structures as generated by an appropriate mechanism. Generating structural hypotheses might be achieved by successively extending a partial structure with new dependency edges in a best-first manner or through deriving a new structure from an existing one by applying a local repair transformation to it.

The solution method employed in the following experiments is of the latter kind. It starts out with a complete but possibly incorrect dependency structure, and then successively tries to change those edges that violate important constraints so that the error is repaired. If an error can be repaired and a higher global score achieved at the same time, the new structure is considered more appropriate and transformation continues from there.[1] If the repair has introduced a different error, this error is attacked next. Transformation stops when every important error in the structure has either been repaired or defied attempts of repair.

There are different ways of limiting the time spent by the repair process: a strict time limit can be set after which the parser should stop, a target score can be specified, below which no partial structure should be considered, or a threshold can be given to indicate which errors should be considered for repair. Even with no limiting factors, perfect accuracy is usually only achieved on short sentences, due to both modelling and search errors.

Due to the typical performance profile of such a parsing procedure, which has an early solution (albeit usually a bad one) available very soon and continues to improve it over time, there is a general time-quality trade-off: Given the strict time-out condition, a speedup naturally translates into a higher quality of analysis.

Although this solution method is far more efficient than a complete search of the problem space (empirically, it does not take exponential time even when no time limit is set), it is still too slow for use on long sentences with several dozen words. It is therefore appropriate to search for further possibilities of speeding up the computation without incurring too much loss of accuracy.

3 Parsing of subtrees

As it is, a WCDG considers parsing as a global constraint optimization problem in which any variable may influence the values allowed for any other. For instance, uniqueness constraints are naturally imposed as all-quantified formulas, and in fact *all* constraints are all-quantified in the WCDG formalism (selective application is possible by qualifying the formula with a particular

[1] Although achieving a higher score does not necessarily imply that the analysis is more accurate, this is usually the case.

premise). This means that WCDG parsing is fundamentally not a recursive process, because larger phrases are not built out of smaller ones. In fact, no phrases are explicitly built at all, only relations between the words in the surface structure are found.

Nevertheless, these relations contribute to structures that can be viewed as recursively embedded trees. Consider the typical example sentence:

> AT&T hatte sich gegenüber der FCC verpflichtet, | eine von drei Optionen zu verfolgen: | die Anteile an Time Warner zu verkaufen, | die Fernsehproduktionsfirma Liberty Media zu veräußern | oder Kabelfernsehnetze mit 10 Millionen Kunden abzustoßen.
>
> *(AT&T had an obligation to the FCC to follow one of three courses: sell its shares of Time Warner, dispose of the production company Liberty Media, or sell cable TV networks with 10 million customers.)*

At the marked positions, the appropriate syntactic structure can easily be decomposed into five subtrees, each of which might conceivably be established independently of the others. In fact, the grammar of German we use can analyse all five subtrees successfully. Although some complements cannot be found within the subtrees and other words must form ROOT subordinations that are discouraged, the corresponding subtree can still be established.

A natural experiment would be to apply the parser first to the subclauses in isolation and then allow the trees to be recombined in a further step. Since each of the problems to be solved is much smaller than the whole sentence, solving them all should take considerably less time than solving the global problem, which would leave enough resources to recombine them into the target structure afterwards. Of course, the best way, and in fact the only reliable way, of finding this division is to determine the syntactic structure of the entire utterance first, which would defeat the goal of faster analysis. Therefore, an imperfect method of subdividing sentences must be employed.

In written language, an approximate subdivision is given by punctuation, which in German marks all subclauses as well as some other phenomena. Note that in the example sentence, the three punctuation marks correspond to three of the four boundaries between major subtrees (the fourth boundary is marked by the conjunction 'oder', which may or may not introduce a large subtree). For our first experiment, we therefore used subclauses, as indicated by the sentence-internal punctuation, as an imprecise indicator of subtrees. In the first stage of processing, only dependencies between words within each subproblem were allowed. The obtained partial structures were then recombined in the second stage by allowing the ROOT edges to select subordinations outside their subtree.

We also had to decide how much time to spend on analyzing the subproblems and how much on recombining the results. We arbitrarily selected an equal division of processing time: half of

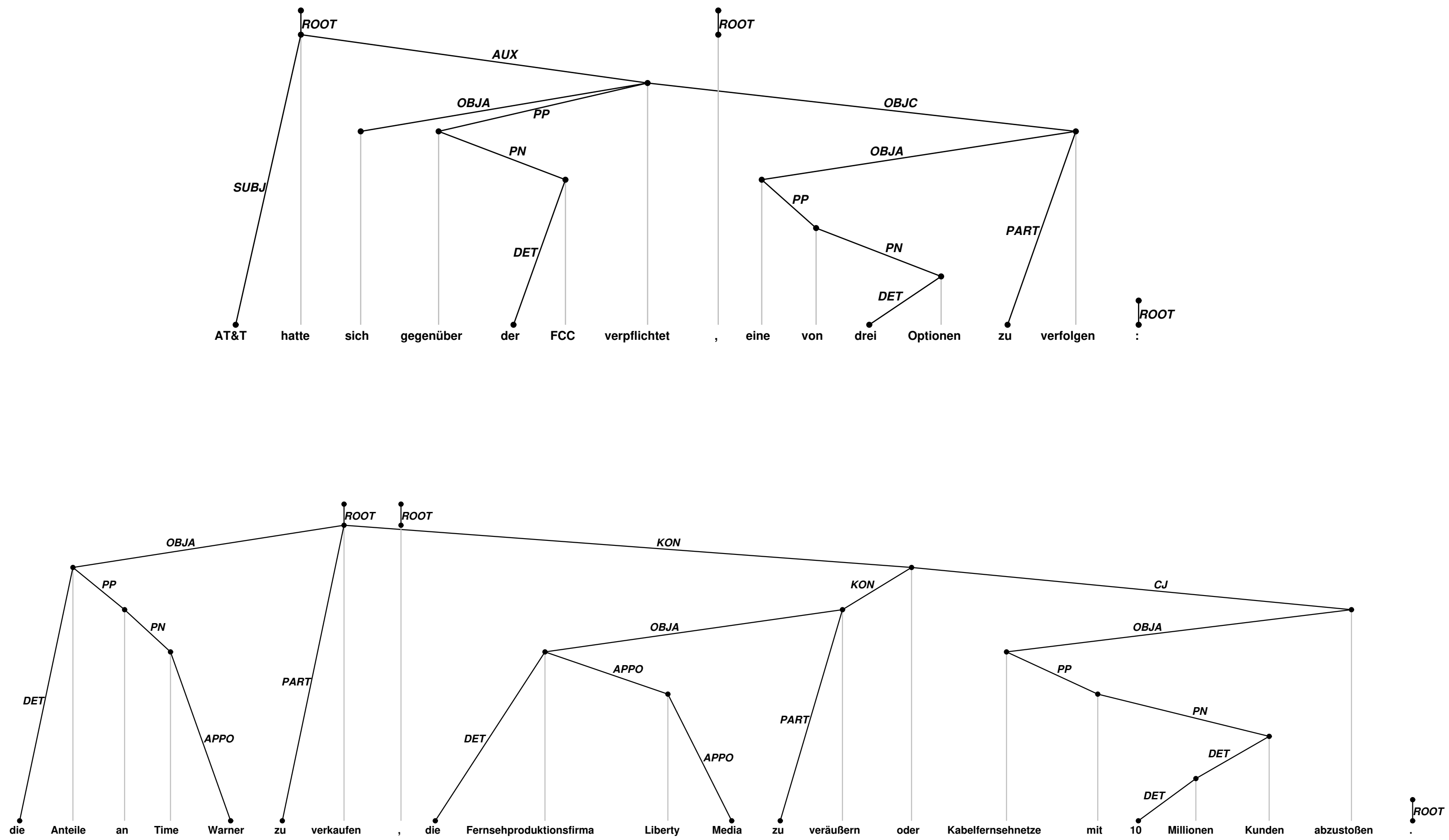

Figure 1: Structural analysis of a German online newscast sentence.

the time was divided up among the subproblems in proportion to their size, and the remaining time was spent on recombining the subtrees.

We employed a corpus of written German extracted from online newscasts of the technical news service `www.heise.de`. This corpus comprises 1894 sentences with an average length of 24 words; when dividing them along internal punctuation, the subclauses have an average length of 10 words.

length	... 10 ... 20 ... 30 ... 40 ... 50 ... 60 ... 70 ...
# sentences	132 609 703 104 25 7 3 1

Figure 2: Distribution of sentence length in the corpus

As a baseline experiment we ran the heuristic solution method on all sentences without any subdivision, with a time limit of 300 seconds per sentence. This was chosen so that most analyses terminate on their own before the limit is reached, so that allowing more time would not improve the results much. On the average, 60 seconds were spent on each sentence and a syntactical accuracy of 79.3% was achieved. All further figures indicate parsing performance relative to this baseline experiment.

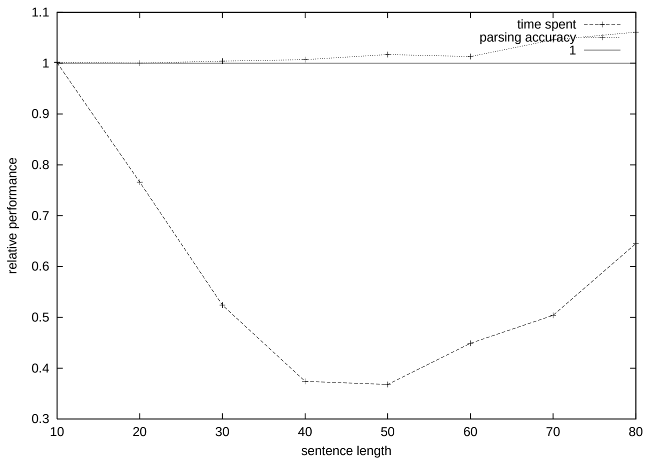

Figure 3: Performance of parsing on subclauses as compared to global parsing

Figure 3 shows the performance of subtree parsing compared against that of normal (global) analysis, displayed by sentence length. It can be seen that the average parsing time over the entire corpus decreases. The efficiency gain grows with the size of the problem, although the trend is reversed for the upper problem classes, where the recombination problem becomes difficult in itself. Also, these very long sentences usually invoked the strict time limit in the baseline case, so that the baseline time figure could not rise above the maximally possible 300 seconds.

To measure the accuracy of the found analyses, the number of structurally correct dependency edges is compared to the baseline case of global heuristic search. The accuracy of the results rises slightly in each problem class. This shows that the time saved by solving smaller problems is actually useful during recombination.

Since punctuation is only available for written language and even then is only an approximate indicator of independent subtrees, we investigated other methods of determining boundaries for subtree parsing. The easiest method is just to assume boundaries every n words. A somewhat more informed experiment uses the brackets inserted by an external chunk parser as boundaries; here the resulting parts are only 2.5 words long on the average. Figure 4 compares the results.

Experiment	Division method	relative parsing time	relative accuracy
1	punctuation	0.431	1.004
2	every 2 words	0.773	0.956
3	every 3 words	0.687	0.964
4	every 5 words	0.532	0.971
5	every 10 words	0.422	0.983
6	every 15 words	0.523	0.988
7	chunk brackets	0.644	0.981

Figure 4: Comparison of different subdivision strategies

When sentences are divided into parts of equal length, parsing accuracy drops slightly; this confirms that the linguistically unmotivated parts are, in fact, somewhat more difficult to parse on their own than true subclauses, even when they are of comparable length (compare lines 1 and 5, lines 2, 3, and 7). The most suitable subdivision for retaining parsing accuracy is that according to sentence-internal punctuation. This obviously reflects the fact that nearly all punctuation does indicate linguistic boundaries of some sort. On the average, only a minor loss in parsing accuracy occurs in any experiment; however, because of the erratic nature of heuristic search many individual sentences are analysed with up to 10% higher or lower accuracy.

Employing punctuation is also nearly the best choice for speeding up the parser, apparently because it strikes a good balance between decreasing the size of the problem and retaining suitable parts. Choosing arbitrary subdivisions has two different effects on parsing time: as the subproblems themselves become smaller, they are solved faster, but at the same time the recombination problem becomes larger and more difficult in itself.

4 Recombining subtrees

So far we have treated every subtree as if it were isolated, i.e. no edge may cross the boundary of the current subproblem, although according to the desired final result, one edge of a subtree actually has to cross a boundary. These edges are forced to find another temporary subordination during the first stage. They will usually adopt the ROOT subordination, which the grammar penalizes because it prefers connected syntax trees. The score of the partial structure is thereby lowered, which makes the problem more difficult to solve. Alternatively, they might find a different (and definitely wrong) subordination within the subtree, which cannot be repaired during recombination. Either way, subdivision makes the entire problem somewhat harder to parse than necessary. We therefore tested several different recombination strategies (reverting to punctuation as a boundary indicator).

What happens if we allow the correct subordination in the first stage already? The optimization subproblem still has fewer variables than the entire problem, but more values for each variable. The disadvantage is that many other non-local attachments are also allowed, all which are wrong, since we cannot know which non-local attachment is the correct one. Line 8 in Figure 5 shows the result: due to the increased complexity of the subproblems, parsing time goes up when compared to experiment 1, and parsing accuracy actually decreases slightly. This shows that the non-local attachments chosen in stage 1 are wrong more often than not.

Experiment	Recombination variant	relative parsing time	relative accuracy
8	nonlocal edges in stage 1	0.531	0.992
9	structure revision in stage 2	0.743	1.005
10	assign weight 1.0	0.433	0.994
11	assign weight 0.99	0.441	0.995
12	assign weight 0.95	0.436	0.995
13	assign weight 0.9	0.391	1.000
14	assign weight 0.8	0.393	1.001
15	assign weight 0.7	0.396	1.001

Figure 5: Comparison of different recombination strategies

Another experiment (line 9) changes the behaviour of the second parsing stage instead. So far, we have tried to recombine the subtrees unchanged, by allowing only ROOT relations to change during stage 2. This succeeds easily if the subproblems have been solved correctly, but it cannot succeed if there was an error in parsing the subtree; this would require revising the partial trees during the recombination step. The disadvantage of this is the same as before: many other alternatives are also allowed, most of which are wrong.

Again, relaxing the strict separation between subtrees increases parsing time when compared to experiment 1. The effect is even more pronounced, since it is now the global optimization problem which allows all alternatives. Although the transformational method only changes edges when there is an obvious problem, each repair step that is done now has to compare many more alternatives. At the same time, parsing accuracy increases only slightly.

Rather than indiscriminately allowing more alternatives in the hope of finding the desired relation, it would be preferable to avoid the artificial constraint failures in the first stage altogether. Empirically, most of these are caused by a small number of constraints which pose conditions that are not easily satisfied within a fragment.

For instance, the condition that a finite verb needs a subject is often not fulfilled within a subproblem because the subject is far distant from the verb. Similarly, penalizing sentence fragments is appropriate when parsing entire utterances, but for the subproblems, the roots of the subtrees are actually intended to select ROOT subordinations and should not be punished.

One solution is simply not to use these non-local constraints during stage 1. Since all constraints operate independently of each other, they can easily be switched off. In the grammar of German employed, 16 out of 391 constraints fall into this category and were marked for special treatment during the first step. Line 10 shows the result of this experiment:[2] while parsing time does not change notably compared to experiment 1, a small loss in accuracy occurs.

Obviously, disregarding non-local constraints during stage 1 once again allows unwanted structures as well as correct ones. For instance, when parsing an entire subclause, the subject can usually be found with the substructure, and therefore the subject constraint should not be lifted. Rather than switching the constraints off totally, their importance can be decreased by assigning a different weight to them during the first stage. This still urges the parser to establish, e.g., the subject relation if this is possible, but does not cause too much problems when it is not.

The rest of Figure 5 shows that retaining the problematic constraints at a lower weight is even more effective than simply switching them off: up to a point, they speed up computation further without losing more parsing accuracy. The exact weight which should be assigned to them depends heavily on the weight of other competing constraints in the grammar. As it is, choosing a fixed alternative weight is enough to obtain a slightly better parsing performance. It is probable that choosing an alternative weight for each constraint individually could improve the parsing behaviour further; however, this would require much more work on the part of the grammar writer and would go even further in the direction of writing a special grammar for sentence fragments, while our goal was to increase parsing performance on arbitrary grammars.

5 Conclusions

Breaking down a given constraint optimization problem into a number of smaller ones and recombining the partial results in a second integration stage indeed provided a major step to achieve an accuracy/speed relation which makes parsing of really long sentences feasible. Most notably this acceleration has been achieved without any loss in accuracy.

Similar approaches to build complex dependency trees out of partial analyses have always used chunk boundaries as the primary source of segmentation cues, e. g. (Basili, Pazienza, and

[2] Due to the multiplicative score aggregation in WCDG, a constraint weight of 1.0 renders the constraint ineffective, while lower values assign it progressively more importance. Constraints with weight 0.0 are the most important and correspond to the constraints of a classical (crisp) constraint satisfaction problem.

Zanzotto, 1998) , (Bücher, Knorr, and Ludwig, 2002) or (Aït-Mokhtar, Chanod, and Roux, 2002). In contrast to these results we found chunk boundaries not being optimal segmentation points. Although performing slightly better than arbitrary segments of the same mean length, they obviously introduce too short substrings of the sentence to fully reap the possible benefits of decomposing the optimization problem. Moreover, the quality of chunk boundaries might suffer from the *ad hoc* treatment unchunked words. Best results have been achieved using punctuation marks as linguistically well-motivated segmentation boundaries, which also show a clear quality advantage over arbitrary segments of comparable length. As expected, the payoff of segmentation is largest for medium to longer sized sentences.

Considering the partial results of the initial parsing phase as unmodified parts of the final structure turned out to be superior to more flexible approaches. Allowing the parser to modify the already established partial structures during the second phase gave no additional benefit. This adds some confirming evidence to the approach adopted in (Basili, Pazienza, and Zanzotto, 1998), where chunk internal structures are also frozen and can only be accessed from outside through an explicitly specified handle.

Finally, a further decrease in processing time was achieved by relaxing the non-local constraints avoiding unwanted penalties for local subtrees. This result additionally shows that the potential of WCDG to weaken constraining information instead of simply switching it off is beneficial in many different settings.

The approach to decompose a complex constraint satisfaction problem into a number of simpler ones was mainly inspired by chart parsing techniques, where complex structures are also built out of partial descriptions (Kay, 1986). By avoiding useless duplication of analysis effort through the memoization of rule applications in a well-formed substring table the worst case complexity can even be reduced from exponential to polynomial.

Since dependency grammar lacks the notion of constituency, no equivalent for a spanning edge is directly available. Moreover, it can only be introduced if dependency structures are restricted to projective trees, a rather strong assumption which has been avoided in WCDG to preserve structural parallelisms for all kind of movement phenomena. Under these conditions the idea of associating parts of a sentence with partial analyses cannot easily be transferred to the constraint optimization task at hand. In particular two main differences have to be mentioned: In contrast to the recursive decomposition in a chart parser, here the optimization is divided into a two-step procedure. Moreover, segmentation of the sentence is not achieved as byproduct of a rule application, but established prior to the constraint evaluation itself using evidence external to the grammar.

This certainly points to the most interesting question for further research: How can information from the grammar itself be used to determine the optimum segment length during parsing,

e. g. by noticing that another word seems to be required. If this can be done in a strictly left-to-right manner it might open up an interesting new perspective on the time course of incremental language processing, where the desired close-to-linear behaviour can only be achieved if already derived structures for sentence segments of the past are frozen and treated as complex units in the ongoing optimization process.

References

Aït-Mokhtar, S., J.-P. Chanod, and C. Roux. 2002. Robustness beyond shallowness: incremental deep parsing. *Natural Language Processing*, 8(2/3):121–144.

Basili, R., M. T. Pazienza, and F. M. Zanzotto. 1998. Efficient parsing for information extraction. In *Proc. 13th European Conference on Artificial Intelligence*, pages 135–139, Brighton, UK.

Bücher, Kerstin, Michael Knorr, and Bernd Ludwig. 2002. Anything to clarify? report your parsing ambiguities! In *Proc. 15th European Conference on Artificial Intelligence*, pages 465–469, Lyon, France.

Foth, Kilian A. and Jochen Hagenström. 2002. Tagging for robust parsers. In *2nd Workshop on Robust Methods in Analysis of Natural Language Data, ROMAND2002*, pages 21 – 32, Frascati, Italy.

Foth, Kilian A., Wolfgang Menzel, and Ingo Schröder. 2002. A Transformation-based Parsing Technique with Anytime Properties. In *4th Int. Workshop on Parsing Technologies*, pages 89 – 100, Trento, Italy.

Foth, Kilian A., Wolfgang Menzel, and Ingo Schröder. submitted. Robust parsing with weighted constraints. *Natural Language Engineering*.

Freuder, Eugene C. and Richard J. Wallace. 1992. Partial constraint satisfaction. *Artificial Intelligence*, 58(1–3):21–70.

Kay, Martin. 1986. Algorithm schemata and data structures in syntactic processing. In B. J. Grosz, K. Sparck Jones, and B. Lynn Webber, editors, *Readings in Natural Language Processing*. Morgan Kaufmann, Los Altos, CA, pages 35–70.

Schröder, Ingo. 2002. *Natural Language Parsing with Graded Constraints*. Phd thesis, Dept. of Computer Science, University of Hamburg, Germany.

Schröder, Ingo, Wolfgang Menzel, Kilian Foth, and Michael Schulz. 2000. Modeling dependency grammar with restricted constraints. *International Journal Traitement automatique des langues: Les grammaires de dépendance*, 41(1):113–144.