A Transformation-based Parsing Technique With Anytime Properties

Kilian Foth, Ingo Schröder, Wolfgang Menzel
(foth | ingo | wolfgang@nats.informatik.uni-hamburg.de)
Fachbereich Informatik, Universität Hamburg
Vogt-Kölln-Straße 30, 22527 Hamburg, Germany

Abstract

A transformation-based approach to robust parsing is presented, which achieves a strictly monotonic improvement of its current best hypothesis by repeatedly applying local repair steps to a complex multi-level representation. The transformation process is guided by scores derived from weighted constraints. Besides being interruptible, the procedure exhibits a performance profile typical for anytime procedures and holds great promise for the implementation of time-adaptive behaviour.

1 Introduction

Parsing procedures always have to be designed around a number of pre-specified requirements which arise from specific conditions of the individual application area in mind. Text retrieval tasks, for instance, can be accomplished already with a rather shallow analysis whereas speed and fail-soft behaviour are of utmost importance. Grammar checking and foreign language learning applications, on the other hand, must provide for highly precise error detection capabilities but differ considerably in their coverage requirements. One of the most demanding combination of target specifications results from the development of spoken language dialogue systems. Since this task is an attempt to model central capabilities of the human language faculty, rather strong criteria have to be met in order to achieve natural communicative behavior on a competitive level:

- **Robustness:**

  A spoken language dialogue system is typically confronted with a rich variety of linguistic constructs and will almost inevitably have to deal with extragrammatical input sooner or later. Also, repairs, hesitations, and other grammatical deviations will frequently produce ungrammatical utterances, while the recognition uncertainty inherent in spoken language input further increases the ambiguity.

  The parsing component must be able to cope with these problems in a robust way. Besides being able to return (possibly partial) analyses even for unexpected and arbitrarily distorted input it is also necessary to provide some kind of measure of how sure the parser is about its results.

- **Complete disambiguation:** Natural language utterances typically exhibit ambiguity when treated in isolation. Nevertheless, a simple enumeration of different (structural) readings almost never can be considered a sensible contribution to a practical language processing task. Although interactive applications can engage the speaker in a kind of clarification dialogue, usually this possibility brings many additional complications and should only be considered a measure of last resort. Instead, a well-designed system should make use of all the available information to obtain a single interpretation of the utterance, which is only abandoned if the user explicitly signals a communication failure.

- **Multiple-source disambiguation:** A vast variety of knowledge sources can contribute to disambiguation: Syntactic constraints, semantic preferences, prosodic cues, domain knowledge, the dialogue history, etc. All the available knowledge should be put to use as soon as possible so that local ambiguity will not create a large space of useless hypotheses during processing. For reasons of perspicuity and accessibility the integration of these knowledge sources should be organized in a way which maintains their modularity. Only then can the respective contributions of individual components be evaluated and properly balanced against each other.

- **Time-aware behavior:** Three closely related aspects must be considered with respect to the temporal behavior of a language processing component: Efficiency, incremental processing and temporal adaptivity. Whereas efficiency always has been an issue of major concern, explorations into incremental and time-adaptive parsing attracted more attention only recently [GKWS96, Amt99, Men94]. Since speech unfolds in time, speaking time is a valuable resource and an immediate response capability can be achieved only if incoming information is processed in an on-line fashion. Temporal adaptivity, on the other hand, is the capability of a component to dynamically control its processing regime depending on how much time is available to complete the task. In principle, such an *anytime behaviour* can be achieved by trading time against quality. Therefore a baseline performance will be required which allows the quality of available results to grow monotonically as more effort is made, and which is robust enough so that results of slightly reduced quality can still be considered being acceptable in a certain sense.

  Obviously, the most natural measure of external temporal pressure is given by the speaking rate of the dialogue partner. Thus temporal adaptivity makes sense first of all under an incremental processing scheme. Its basic mechanisms, however, can also be studied in the far simpler non-incremental case.

This paper investigates a non-standard parsing approach, which attempts to reconcile two different kinds of robustness, namely robustness against unexpected and ill-formed input and robustness against external temporal pressure. It is based on the application of constraint satisfaction techniques to the problem of structural disambiguation and allows the parser to include a wide variety of possibly contradicting informational contributions. Different solution procedures are presented and compared taking into account solution quality and the observed temporal behavior.

2 Parsing As A Consistent Labeling Problem

Although most contemporary unification-based grammars can be said to employ constraints, none of them fulfill the traditional definition of a Constraint Satisfaction Problem (CSP) consisting of a fixed number of *variables*, which receive their *values* from *domains*, i. e. sets of alternative value assignments. The global consistency of a value assignment is defined by means of local *constraints*, which can be understood as sets of admissible value tuples, specified intensionally.[1] A *conflict* then can be defined as a tuple outside a constraint, and a *solution* is a complete value assignment without conflicts. Finally, Consistent Labeling Problems (CLP) are characterized by their domains being discrete, finite and known in advance.

Parsing a given utterance can be viewed as an instance of such a problem in that every word form fulfills a specific function (i. e. complement, specifier, etc.) in a particular analysis. As long as a grammar theory is defined over a finite number of such functions, they can immediately be used as the values of a CSP. Since an utterance often contains multiple instances of the same grammatical function (i. e. a determiner modifying a noun), it always has to be specified in what context the word form fulfills its function. Therefore, every word form in an utterance is annotated with a *pair* consisting of a label (i. e. the grammatical function of the form) and a pointer to another word form, which is modified by the first one. This way, an entire dependency tree can be encoded in a CSP of a fixed size.

Conversely, some word forms require other word forms to fulfill certain functions for them as obligatory complements. Such valence requirements can be modeled as constraint requirements as well. Furthermore additional relations between word forms can be included easily (e. g. semantic arguments or aspects of the informational structure), thus creating a fairly rich structural representation.

Figure 1 shows an example of such a multi-level representation including syntactic dependencies (the arcs above the word forms), semantic arguments (e. g. the arcs labeled agens and patiens) and a bunch of valence requirements (all those with the label prefix n_, e. g. n_subj, n_inf etc.). Admissible combinations between dependency relations on two different representational levels are also defined by means of suitable constraints.

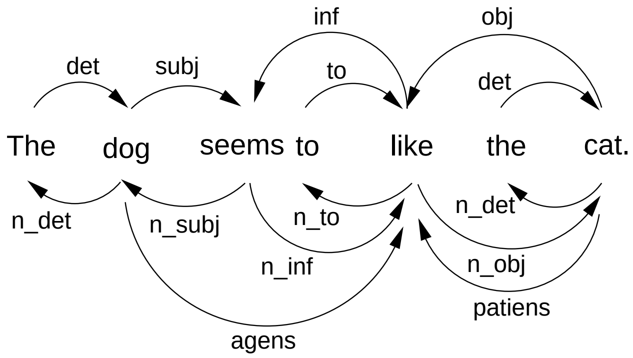

Figure 1: A CDG analysis

The notion of a constraint dependency grammar was first introduced by Maruyama [Mar90]. Harper et al. [HH94] extended the idea in order to cope with ambiguity arising from lexicon lookup and speech recognizer uncertainty. Menzel [MS98a] incorporated soft constraints, i. e., constraints that can be violated by a valid solution if no other solution can be found otherwise. For this purpose, constraints are annotated with a weight or score between zero and one that determines how easily that constraint may be violated. Hard constraints have a weight of zero

[1] From this perspective, the constraint-solving mechanism of a typed unification grammar, like the one underlying e. g. HPSG, might be considered a degenerate CSP with only a single variable, a domain consisting of all possible top-level feature structures, and constraints licensing *parts* of a complex value assignment.

and must not be violated by any solution. Constraints which reflect regularities of the grammar receive a small weight greater than zero while constraints that model mere preferences will be weighted close to one. A solution candidate to the CSP can then be assigned a score by determining the product of the weights of all the constraints which are violated somewhere in the structural description. Under these premises, parsing becomes a multidimensional optimization problem.

For reasons of efficiency at most binary constraints can be allowed, hence the universal expressive power of constraints is not available in practice. This serious shortcoming, however, can be neutralized to a certain degree by approximating higher order constraints using binary ones. Another disadvantage is the lack of a variable binding mechanism like the one which is provided by a unification operator together with the missing notion of a constituent (which, however, is shared with most dependency grammar approaches). Experience with grammar writing has confirmed that nevertheless nontrivial subsets of grammar can be encoded successfully, although some phenomena such as long-distance dependencies can only be modeled approximatively [SMFS].

Constraint dependency grammar is a purely declarative formalism. This property makes it amenable to a variety of problem solving strategies that can be compared, e. g. with respect to their temporal behaviour. The possibility to add further representational levels supports the integration of knowledge contributions from very different sources into a single solution space without sacrificing the strict modularity of the grammar and of the structural representation. Of course, this possibility is limited to only those knowledge sources that can meaningfully attach information to single word forms. A single interpretation of the incoming utterance can be obtained by using all available evidence, including minor preference indicators like ordering, distance or default cases. A truly ambiguous sentence will usually allow several analyses with only small differences between their scores, which can be ignored if desired.

The approach exhibits a remarkable robustness against unexpected and ill-formed input [MS98b], which obviously can be attributed to three important characteristics:

- the use of weighted constraints, which provides for the accommodation of conflicting evidence and therefore makes the analysis of deviating structures possible,
- the redundancy between loosely coupled representational levels, which allows conflicting information on one level to be overridden by sufficient evidence from a complementary one, and
- the possibility to license arbitrary categories as an acceptable, but in most cases highly disfavored, top node of a dependency tree, thus introducing a partial parsing scheme as a natural extension of the normal mode of operation.

Note that the resulting robust behavior follows immediately from the fundamental principles of the approach and no error rules or special operations become necessary.

Two other characteristics of the approach contribute to the rich potential for obtaining the desired anytime behaviour. In contrast to other parsing approaches the space of all possible analyses for constraint dependency parsing is always finite and very regularly structured. Parsing therefore becomes a process of selection between different analyses with virtually identical formal properties, which considerably facilitates their mutual comparison.

3 Solution Procedures

3.1 Consistency-based Methods

The canonical method for solving a constraint satisfaction problem is to establish a certain degree of consistency in it by deleting incompatible values from its domains, and then select an assignment to all constraint variables from the remaining values. This approach contrasts strongly with common parsing methods that are *constructive* in nature. Usually, grammatical structures are built up recursively from simpler structures, and ultimately from the information associated with the lexical items present in an utterance. Thus, the number of structures available increases over time. In contrast, the achievement of consistency is an *eliminative* process: The more progress is made, the fewer values remain in the problem. An attractive property of this kind of parsing is that it can be exactly determined, at any time, how much progress has already been made and how much work remains to be done until disambiguation is completed. This information will be of great use to a time-aware solution procedure.

Various well-defined degrees of consistency can be achieved in a CSP, and general algorithms exist to establish any desired degree of consistency. Consistency-based methods are used extensively in a constraint dependency parser by Harper et al. [HHZ$^+$95]. However, this approach does not lend itself to robust processing of deviating input. Since consistency algorithms only remove those values that cannot appear in any solution, only hard constraints are effective. If a constraint grammar predominantly employs soft constraints, a consistency algorithm may remove very few values or none at all.

A suitable algorithm for consistency in a partial CSP should remove all those values that do not appear in the *optimal* solution—a property that is much more difficult to determine. The usual consistency algorithm will find a value that cannot appear in a solution by noting that it cannot appear in a valid n-ary assignment. A similar approach for partial CSP would be to select those values for deletion that only occur in n-ary assignments with low scores. The obvious method is to define a fixed limit and consider all scores below it unacceptable; this has much the same effect as employing the unmodified algorithms on a grammar in which more constraints are hard.

Another method known as *pruning* [MS98b] goes one step further. While a consistency algorithm cannot guarantee how much progress it will achieve, a pruning method will invoke a selection function at regular intervals to select exactly one value for deletion. If this function uses a fixed amount of time, an exact appraisal can be given not only of the amount of work to be done, but also of the actual runtime left until termination.

To guarantee that a value is selected for deletion within the allotted time, the selection function will usually have to be heuristic in nature. A simple selection function mimics the behaviour of a 2-consistency algorithm: Tables of mutual support are constructed for all pairs of domains in the problem. The support of a value v from another domain d can be defined as the maximal or the average compatibility of v with any value from d, or in a more elaborate way. The value whose maximal support by any other domain is smallest is selected for deletion.

Since the globally optimal solution may consist of values that are locally suboptimal, in general this method of assessing values exclusively by *local* information may remove the wrong values from a problem. While the CDG formalism ensures that the remaining values form a complete assignment, in general it cannot be guaranteed that this assignment will be the optimal solution, or even a grammatically valid one. Thus, a heuristic consistency algorithm may fail without result even though there is a valid solution, which defies the purpose of robust processing.

3.2 Enumeration

Most parsers use some kind of search algorithm to enumerate all alternatives for local or global ambiguities arising in the analysis of their input. A great number of search variants has been invented for different parsing applications (top-down vs. bottom-up parsing, depth-first vs. breadth-first search, linear vs. island parsing), and choosing the right method can have dramatic impact on the efficiency of a parser. In general, considering every possible alternative ensures that an algorithm is complete as well as correct, but may require so many resources that it becomes impractical to apply.

Since in a problem in CDG, consistency-based methods cannot guarantee either complete or correct behaviour, a complete method of solution is desirable even if it has other disadvantages. For instance, a complete but inefficient algorithm will still be of great use to the developer of a constraint grammar to check the validity of their model, or to verify the results of an incomplete method.

For the CSP, a complete search of all possible assignments can be conducted that is guaranteed to find the optimal solution.

In the partial CSP, a normal *best first* search can be employed which finds the optimal solution without ever having to expand a partial solution with a lower score. The current implementation of the CDG parser provides a straightforward best-first search in which the variables of a problem are instantiated in a fixed order. This will usually be the order of the words corresponding to the constraint variables. Compared with other parsers, this would be classified as a heuristically driven left-to-right search. It resembles bottom-up parsing in that each word can immediately be integrated into a tree that forms part of the complete dependency structure. A top-down parsing method could be simulated by arranging the search so that every additional dependency edge must modify a word that has already been analyzed, starting with those words that can modify the root of the dependency tree.

Since the CSP is $\mathcal{NP}$-complete, probably any complete solution method will have an exponential worst-case complexity. Although the actual runtime of a complete search algorithm is usually far below the worst possible case, and heuristic re-ordering of both domains and values can greatly improve the efficiency, it is difficult to predict even approximately how long a particular instance of the problem will take to search. Therefore, a complete search is inadequate as a solution method when time-aware behaviour is required. However, in contrast to other methods a complete search can easily enumerate globally near-optimal structures such as those defined by syntactically ambiguous sentences.

3.3 Transformation

An obvious way to overcome the unacceptable temporal behaviour of complete algorithms is to employ suboptimal methods. A strategy that works well in practice is that of heuristic repair. Rather than attempting to build the correct structure by selecting correct values step by step, this method first constructs an arbitrary dependency structure with errors in it and then tries to correct the errors.

Suppose that the dependency structure shown in Figure 2 has been constructed, in which both nouns have been analyzed as subjects of the same verb.

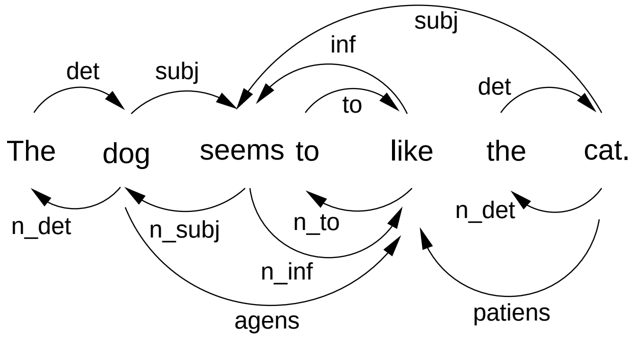

Figure 2: A suboptimal dependency analysis

Figure 3 a) shows part of the corresponding matrix of constraint variables. If a constraint exists that excludes multiple subjects, one of the two annotations with the label 'subj' must be replaced. By analyzing the word 'cat' as an object instead, this assignment can be transformed into the assignment shown as Figure 3 b). Note that when providing an object for the verb 'like' we also changed the corresponding value of the ancillary level OBJ to reflect the fact that the object required by the verb is actually present now. The new assignments represent the optimal analysis shown above as Figure 1.

a)	Syntax	SUBJ	OBJ	...
the_1	det/2	–	–	...
dog_2	subj/3	–	–	...
seems_3	–	n_subj/2	–	...
to_4	to/5	–	–	...
like_5	inf/3	–	–	...
the_6	det/7	–	–	...
cat_7	**subj/3**	–	–	...

$\Longrightarrow$

b)	Syntax	SUBJ	OBJ	...
the_1	det/2	–	–	...
dog_2	subj/3	–	–	...
seems_3	–	n_subj/2	–	...
to_4	to/5	–	–	...
like_5	inf/3	–	**n_obj/7**	...
the_6	det/7	–	–	...
cat_7	**obj/5**	–	–	...

Figure 3: Transformation of a dependency structure.

This method has several advantages as opposed to the previous ones:

- Because a complete dependency analysis is maintained at all times, the algorithm may be interrupted at any time and still return a meaningful answer, though not always the optimal one. Thus, it automatically fulfills a strong anytime criterion.
- The constraints that cause conflicts in a suboptimal assignment can suggest which value is inappropriate and what other value should be substituted.
- Because all analyses of a given utterance comprise the same number of values, it is guaranteed that the optimal solution (if any exists) can be constructed from any other assignment by successively replacing one value at a time.

A transformation step is usually defined as the exchange of one value of a constraint variable for another. By this definition, the correct solution of a CSP of degree n can always be reached in not more than n transformation steps, if the correct replacement value is chosen at any point. To accomplish this, however, *every* value in the problem has to be considered as an alternative in each step, whereas a backtracking search only has to try out all values from one particular domain. Obviously the transformation algorithm will encounter a much greater branching factor. However, the search space is now graph-shaped, and so not every alternative must be pursued further because the order in which several values are inserted into an analysis does not matter. Instead, the number of alternatives that are tried out at all can be used as a parameter to speed up the computation.

Obviously, the efficiency of such a repair algorithm depends on its ability to select the correct values for repair. Even a totally uninformed repair method can ultimately find the correct solution, since it is simply a random walk through the problem space. For better results, heuristic decision methods must be found to guide the selection. The simple *hill climbing* method will always choose the value that results in the best immediate improvement. The principal difficulty with this method is that it can get stuck in local optima of the problem space where no immediate improvement is possible.

Different methods exist to allow an algorithm to leave such misleading areas of the search space.

- Occasional downhill steps may be allowed so that an algorithm may escape from a local maximum. For example, in the popular method of *simulated annealing* downhill steps are allowed but gradually discouraged as analysis progresses.
- In another approach, hill climbing does not optimize the score of an analysis itself, but an ancillary cost function that is adapted in each step, so that the local optimum can be turned into an ascent.
- The definition of a transformation step can be changed so that several values may be replaced simultaneously. Assignments which differ in several variables will then become adjacent to the current assignment. Again, the number of values that may be changed in one step can be used as parameter to influence the speed of the algorithm.

A subsequent difficulty is that after such a repair algorithm has converged to the optimal solution, it may not terminate. If the optimal solution still causes some minor conflicts, the algorithm will continually try to repair these conflicts without success, since there is no simple way to distinguish a local optimum from the global one. In this case the individual constraints of the grammar can be used as a taboo criterion: If no repair step is allowed to re-introduce the conflict that prompted it in the first place, termination can be guaranteed.

Although repair-based solution methods cannot guarantee to find the optimal solution in all cases, in practice they achieve results comparable to those of exhaustive methods. This demonstrates that the values contained in a complete parse are helpful in selecting correct values even though some of them may themselves be incorrect [Minton92]. In this way, parsing by transformation can make use of global information without suffering the full combinatorial explosion of a complete search.

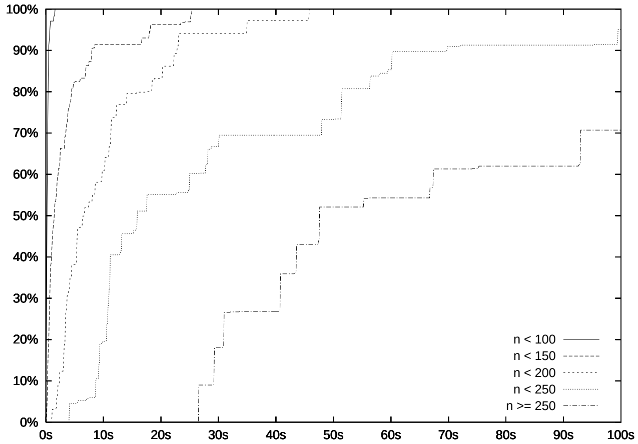

Figure 4: Solution quality as a function of time.

In a series of experiments, repair methods have been successfully applied to the the constraint parsing problem. The corpus comprised 200 consecutive utterances from the Verbmobil corpus of spoken dialogue utterances, segmented and slightly regularized by hand. Figure 4 illustrates the average temporal behaviour of a transformation-based algorithm grouped by the number n of constraint variables[2] in the resulting CSP. For each problem class, the average score after the indicated runtime is given relatively to the optimal score attainable.

With the observed increase of the solution quality the parser has a performance profile typical for anytime procedures [BD89]. In 90% of all problems, the solutions found were either identical to those found by a complete search or had even better scores (since the search uses a finite agenda, it may actually become incomplete in large problems). In the final analyses, 99.7% of all dependency links were established correctly.

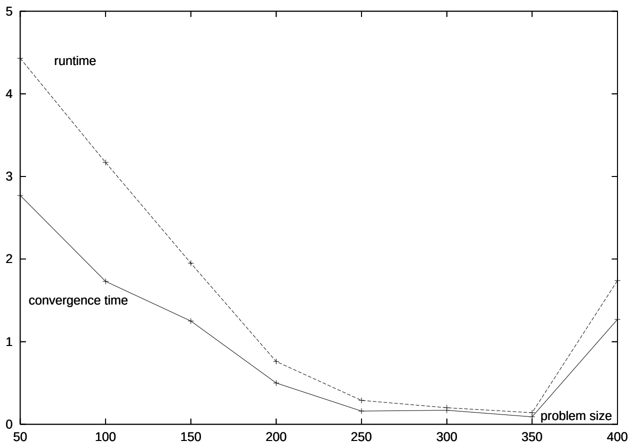

Figure 5: Ratio of runtime for transformation and search methods.

[2]In this particular constraint grammar, a sentence of five words will typically map to a CLP with about 100 variables.

In general, a near-optimal structure will be constructed after a short time. Finding the exact optimal analysis may take considerably more time in medium-sized and large optimization problems. However, particularly in these cases the algorithm will be consistently faster than a complete search. Figure 5 gives the average time that the transformation-based solution method takes to terminate, measured in terms of the runtime of a best-first search of the same problems. Clearly, the repair method is faster in most large problems except for the highest problem class (which only has two members in our corpus, however). Comparison of the time until the last successful repair with the total runtime shows that the algorithm will usually terminate not long after having reached the optimal analysis.

For applications that have to react to varying external time constraints, a parameter is provided that adjusts the number of alternatives to be tried out at each transformation step. In the cases investigated so far, decreasing this parameter can speed up the repair substantially, with an acceleration by a factor of about 3 reducing the accuracy to 86% of correct dependency links.

4 Related work

There is a striking analogy between constraint dependency parsing and customary approaches to the task of tagging natural language data: In both cases each word form in the utterance is annotated with a label from a finite set of alternatives and the approaches differ only in the information content of the labels. Although tagging usually means to classify word forms into (syntactic and semantic) types, the idea can also be extended to the use of functional tags in a straightforward way. Karlsson et al. [KVHA95] use such functional descriptions in Constraint Grammar parsing. Their tagset contains elements like “subject” or “a determiner modifying a noun to the right”. Usually these tags are underspecified, because the exact identity of the modified item is unknown. Constraint Dependency Grammar as discussed in this paper extends this approach to fully specified structural representations. Since the identity of the modified word is now included into the composite tags, the size of the tagset additionally depends on the number of word forms in the given utterance. Moreover, the extension to multi-level disambiguation allows to treat considerably richer representations as compared to what a usual tagger is taking into account.

Another approach using complex tags is Supertagging [BJ99], where complex tree fragments are attached to word forms by means of a stochastic model. These tags represent considerably more structure than values in CDG, which correspond to single dependency edges. However, tags are treated in isolation and the compatibility between adjacent structures is modeled only probabilistically. In order to combine tags to complete parse trees an additional processing step has to be carried out after the tagging itself.

The idea to obtain a structural description of natural language utterances by applying a sequence of transformations which successively modifies an intermediate representation has first been pursued within the framework of parsing as tree-to-tree transduction [BGQA82], although no explicit notion of scoring and quality improvement was involved at that time. The dynamics of the transformation process was fully under the control of the grammar writer, taking into account the precedence ordering implicit in a sequence of rules and some additional means to influence the degree of non-determinism. Transformation-based approaches have later been applied to the problem of syntactic tagging [Bri95]. However, focus was on inducing an appropriate set of transformation rules from the information contained in an annotated corpus.

Possibilities to model grammar by means of contradicting principles are investigated currently in the framework of optimality theory [PS91]. The grammatical principles postulated there are ranked rather than weighted, with higher ranked regularities completely overriding the influence of the lower ones. First applications have been identified in phonology and syntax.

5 Conclusion

A novel approach to parsing as constraint-based structural disambiguation has been presented. By combining techniques for robust parsing (graded constraints, multi-level-disambiguation and partial parsing) with the idea of a transformation-based problem solving mechanism, a parser can be created that shows the typical temporal behavior of an interruptible anytime algorithm.

Further investigations will focus on

1. transferring these procedural characteristics to the case of incremental parsing, thus addressing particularly the problem of processing time for long utterances,
2. a more thorough investigation into time-adaptive behaviour, which should be able to speed up the convergence towards the optimum solution under temporal pressure, at the risk of missing it completely, and
3. the combination of different solution techniques to improve the termination behaviour.

The resulting parsing method mimics human language processing in that it is time-adaptive and robust and therefore lends itself to the implementation of human-machine dialogue systems.

References

[Amt99] Jan W. Amtrup. *Incremental Speech Translation*, volume 1735 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York, 1999.

[BJ99] Srinivas Bangalore and Aravind K. Joshi. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265, 1999.

[BD89] Mark Boddy and Thomas L. Dean. Solving Time-dependent Problems. in *Proceedings 11th Int. Joint Conference on Artificial Intelligence*, Detroit, 1989.

[BGQA82] Ch. Boitet, P. Guillaume, and M. Quezel-Ambrunaz. Implementation and conversational environment of ariane-78. In *Proceedings 9th International Conference on Computational Linguistics, Coling '82*, pages 19–28, Prague, CSSR, 1982.

[Bri95] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[GKWS96] Günther Görz, Marcus Kesseler, Hans Weber, and Jörg Spilker. Research on architectures for integrated speech/language systems in verbmobil. In *Proceedings 16th*

International Conference on Computational Linguistics, Coling '96, pages 484–489, Kopenhagen, Denmark, 1996.

[HH94] Mary P. Harper and Randall A. Helzerman. Managing multiple knowledge sources in constraint-based parsing of spoken language. Technical Report EE-94-16, School of Electrical Engineering, Purdue University, West Lafayette, 1994.

[HHZ+95] Mary P. Harper, Randall A. Helzermann, C. B. Zoltowski, B. L. Yeo, Y. Chan, T. Steward, and B. L. Pellom. Implementation issues in the development of the parsec parser. *Software - Practice and Experience*, 25(8):831–862, 1995.

[KVHA95] Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors. *Constraint Grammar – A Language-Independent System for Parsing Unrestricted Text.* Mouton de Gruyter, Berlin, New York, 1995.

[Mar90] H. Maruyama. Structural disambiguation with constraint propagation. In *Proceedings 28th Annual Meeting of the ACL*, pages 31–38, 1990.

[Men94] Wolfgang Menzel. Parsing of spoken language under time constraints. In T. Cohn, editor, *Proceedings 11th European Conference on Artificial Intelligence*, pages 560–564, Amsterdam, 1994.

[MS98a] Wolfgang Menzel and Ingo Schröder, Constraint-Based Diagnosis for Intelligent Language Tutoring Systems. In *Proceedings IT & KNOWS, XV. IFIP World Computer Congress*, 484–497, Wien and Budapest, 1998.

[MS98b] Wolfgang Menzel and Ingo Schröder. Decision procedures for dependency parsing using graded constraints. In Sylvain Kahane and Alain Polguère, editors, *Proc. of the Joint Conference COLING/ACL Workshop: Processing of Dependency-based Grammars*, Montréal, Canada, 1998.

[Minton92] Minton, Steven, Johnston, Mark D., Philips, Andrew B. and Laird, Philip. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. In *Artificial Intelligence* 58, 161–205, 1992.

[PS91] Alan Prince and Paul Smolensky. Linguistics 247: Notes on connectionism and harmony theory in linguistics. In Technical Report CU-CS-533-91, Department of Computer Science, University of Colorado, Boulder, Colorado, 1991.

[SMFS] Ingo Schröder, Wolfgang Menzel, Kilian Foth and Michael Schulz. Dependency modeling with restricted constraints. Submitted to *Traitement automatique de langage*, Special Issue on Dependency Grammar.