Interactive grammar development with WCDG

Kilian A. Foth **Michael Daum** **Wolfgang Menzel**
Natural Language Systems Group
Hamburg University
D-22527 Hamburg
Germany
{foth,micha,menzel}@nats.informatik.uni-hamburg.de

Abstract

The manual design of grammars for accurate natural language analysis is an iterative process; while modelling decisions usually determine parser behaviour, evidence from analysing more or different input can suggest unforeseen regularities, which leads to a reformulation of rules, or even to a different model of previously analysed phenomena. We describe an implementation of *Weighted Constraint Dependency Grammar* that supports the grammar writer by providing display, automatic analysis, and diagnosis of dependency analyses and allows the direct exploration of alternative analyses and their status under the current grammar.

1 Introduction

For parsing real-life natural language reliably, a grammar is required that covers most syntactic structures, but can also process input even if it contains phenomena that the grammar writer has not foreseen. Two fundamentally different ways of reaching this goal have been employed various times. One is to induce a probability model of the target language from a corpus of existing analyses and then compute the most probable structure for new input, i.e. the one that under some judiciously chosen measure is most similar to the previously seen structures. The other way is to gather linguistically motivated general rules and write a parsing system that can only create structures adhering to these rules.

Where an automatically induced grammar requires large amounts of training material and the development focuses on global changes to the probability model, a handwritten grammar could in principle be developed without any corpus at all, but considerable effort is needed to find and formulate the individual rules. If the formalism allows the ranking of grammar rules, their relative importance must also be determined. This work is usually much more cyclical in character; after grammar rules have been changed, intended and unforeseen consequences of the change must be checked, and further changes or entirely new rules are suggested by the results.

We present a tool that allows a grammar writer to develop and refine rules for natural language, parse new input, or annotate corpora, all in the same environment. Particular support is available for interactive grammar development; the effect of individual grammar rules is directly displayed, and the system explicitly explains its parsing decisions in terms of the rules written by the developer.

2 The WCDG parsing system

The WCDG formalism (Schröder, 2002) describes natural language exclusively as *dependency structure*, i.e. ordered, labelled pairs of words in the input text. It performs natural language analysis under the paradigm of *constraint optimization*, where the analysis that best conforms to all rules of the grammar is returned. The rules are explicit descriptions of well-formed tree structures, allowing a modular and fine-grained description of grammatical knowledge. For instance, rules in a grammar of English would state that subjects normally precede the finite verb and objects follow it, while temporal NP can either precede or follow it.

In general, these *constraints* are defeasible, since many rules about language are not absolute, but can be preempted by more important rules. The strength of constraining information is controlled by the grammar writer: fundamental rules that must always hold, principles of different import that have to be weighed against each other, and general preferences that only take effect when no other disambiguating knowledge is available can all be formulated in a uniform way. In some cases preferences can also be used for disambiguation by approximating information that is currently not available to the system (e.g. knowledge on attachment preferences).

Even the very weak preferences have an influence on the parsing process; apart from serving as tiebreakers for structures where little context is available (e.g. with fragmentary input), they provide an

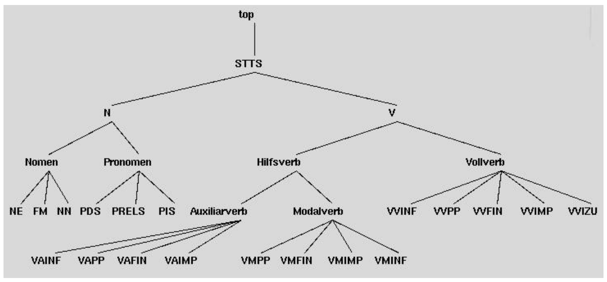


Figure 1: Display of a simplified feature hierarchy

initial direction for the constraint optimization process even if they are eventually overruled. As a consequence, even the best structure found usually incurs some minor constraint violations; as long as the combined evidence of these default expectation failures is small, the structure can be regarded as perfectly grammatical.

The mechanism of constraint optimization simultaneously achieves robustness against *extra-grammatical* and *ungrammatical* input. Therefore WCDG allows for broad-coverage parsing with high accuracy; it is possible to write a grammar that is guaranteed to allow at least one structure for any kind of input, while still preferring compliant over deviant input wherever possible. This *graceful degradation* under reduced input quality makes the formalism suitable for applications where deviant input is to be expected, e.g. second language learning. In this case the potential for *error diagnosis* is also very valuable: if the best analysis that can be found still violates an important constraint, this directly indicates not only where an error occurred, but also what might be wrong about the input.

3 XCDG: A Tool for Parsing and Modelling

An implementation of constraint dependency grammar exists that has the character of *middleware* to allow embedding the parsing functionality into other natural language applications. The program XCDG uses this functionality for a graphical tool for grammar development.

In addition to providing an interface to a range of different parsing algorithms, graphical display of grammar elements and parsing results is possible; for instance, the hierarchical relations between possible attributes of lexicon items can be shown. See Figure 1 for an excerpt of the hierarchy of German syntactical categories used; the terminals correspond to those used the Stuttgart-Tübingen Tagset of German (Schiller et al., 1999).

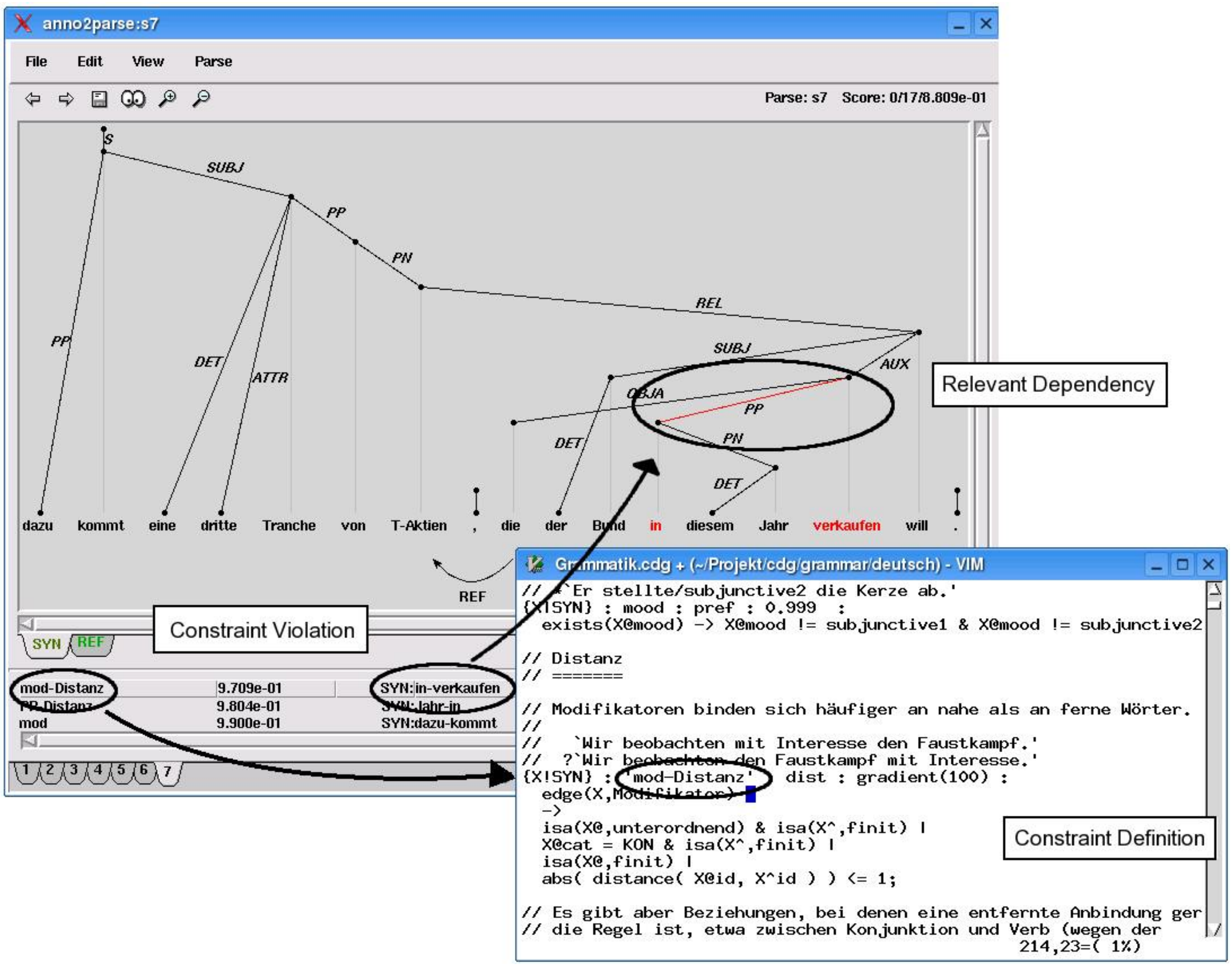
More importantly, mean and end results of parsing runs can be displayed graphically. Dependency structures are represented as trees, while additional relations outside the syntax structure are shown as arcs below the tree (see the referential relationship REF in Figure 2). As well as end results, intermediate structures found during parsing can be displayed. This is often helpful in understanding the behaviour of the heuristic solution methods employed.

Together with the structural analysis, instances of broken rules are displayed below the dependency graph (ordered by decreasing weights), and the dependencies that trigger the violation are highlighted on demand (in our case the PP-modification between the preposition *in* and the infinite form *verkaufen*). This allows the grammar writer to easily check whether or not a rule does in fact make the distinction it is supposed to make. A unique identifier attached to each rule provides a link into the grammar source file containing all constraint definitions. The unary constraint `'mod-Distanz'` in the example of Figure 2 is a fairly weak constraint which penalizes attachments the stronger the more distant a dependent is placed from its head. Attaching the preposition to the preceding noun *Bund* would be preferred by this constraint, since the distance is shorter. However, it would lead to a more serious constraint violation because noun attachments are generally dispreferred.

To facilitate such experimentation, the parse window doubles as a tree editor that allows structural, lexical and label changes to be made to an analysis by drag and drop. One important application of the integrated parsing and editing tool is the creation of large-scale dependency treebanks. With the ability to save and load parsing results from disk, automatically computed analyses can be checked and hand-corrected where necessary and then saved as annotations. With a parser that achieves a high performance on unseen input, a throughput of over 100 annotations per hour has been achieved.

4 Grammar development with XCDG

The development of a parsing grammar based on declarative constraints differs fundamentally from that of a derivational grammar, because its rules *forbid* structures instead of *licensing* them: while a context-free grammar without productions licenses nothing, a constraint grammar without constraints would allow everything. A new constraint must therefore be written whenever two analyses of the same string are possible under the existing constraints, but human judgement clearly prefers one over the other.



```
mod-Distanz      9.709e-01      SYN:in-verkaufen
PP-Distanz       9.804e-01      SYN:Jahr-in
mod              9.900e-01      SYN:dazu-kommt
```

```
// *`Er stellte/subjunctive2 die Kerze ab.'
{X!SYN} : mood : pref : 0.999  :
  exists(X@mood) -> X@mood != subjunctive1 & X@mood != subjunctive2

// Distanz
// =======

// Modifikatoren binden sich häufiger an nahe als an ferne Wörter.
//
//    `Wir beobachten mit Interesse den Faustkampf.'
//   ?`Wir beobachten den Faustkampf mit Interesse.'
{X!SYN} : 'mod-Distanz'  dist : gradient(100) :
  edge(X,Modifikator)
  ->
  isa(X@,unterordnend) & isa(X^,finit) |
  X@cat = KON & isa(X^,finit) |
  isa(X@,finit) |
  abs( distance( X@id, X^id ) ) <= 1;

// Es gibt aber Beziehungen, bei denen eine entfernte Anbindung ger
// die Regel ist, etwa zwischen Konjunktion und Verb (wegen der
                                                  214,23=( 1%)
```

Figure 2: Xcdg Tree Editor

Most often, new constraints are prompted by inspection of parsing results under the existing grammar: if an analysis is computed to be grammatical that clearly contradicts intuition, a rule must be missing from the grammar. Conversely, if an error is signalled where human judgement disagrees, the relevant grammar rule must be wrong (or in need of clarifying exceptions). In this way, continuous improvement of an existing grammar is possible.

XCDG supports this development style through the feature of *hypothetical evaluation*. The tree display window does not only show the result returned by the parser; the structure, labels and lexical selections can be changed manually, forcing the parser to pretend that it returned a different analysis. Recall that syntactic structures do not have to be specifically allowed by grammar rules; therefore, every conceivable combination of subordinations, labels and lexical selections is admissible in principle, and can be processed by XCDG, although its score will be low if it contradicts many constraints.

After each such change to a parse tree, all constraints are automatically re-evaluated and the updated grammar judgement is displayed. In this way it can quickly be checked which of two alternative structures is preferred by the grammar. This is useful in several ways. First, when analysing parsing errors it allows the grammar author to distinguish *search errors* from *modelling errors*: if the intended structure is assigned a better score than the one actually returned by the parser, a search error occurred (usually due to limited processing time); but if the computed structure does carry the higher score, this indicates an error of judgement on the part of the grammar writer, and the grammar needs to be changed in some way if the phenomenon is to be modelled adequately.

If a modelling error does occur, it must be because a constraint that rules against the intended analysis has overruled those that should have selected it. Since the display of broken constraints is ordered by severity, it is immediately obvious which of the grammar rules this is. The developer can then decide whether to weaken that rule or extend it so that it makes an exception for the current phenomenon. It is also possible that the intended analysis really does conflict with a particular linguistic principle, but in doing so follows a more important one; in this case, this other rule must be found and strengthened so that it will overrule the first one. The other rule can likewise be found by re-creating the original automatic analysis and see which of its constraint violations needs to be given more weight, or, alternatively, which entirely new rule must be added to the grammar.

In the decision whether to add a new rule to a constraint grammar, it must be discovered under what conditions a particular phenomenon occurs, so that a generally relevant rule can be written. The possession of a large amount of analysed text is often useful here to verify decisions based on mere introspection. Working together with an external program to search for specific structures in large treebanks, XCDG can display multiple sentences in stacked widgets and highlight all instances of the same phenomenon to help the grammar writer decide what the relevant conditions are.

Using this tool, a comprehensive grammar of modern German has been constructed (Foth, 2004) that employs 750 handwritten well-formedness rules, and has been used to annotate around 25,000 sentences with dependency structure. It achieves a structural recall of 87.7% on sentences from the NEGRA corpus (Foth et al., submitted), but can be applied to texts of many other types, where structural recall varies between 80–90%. To our knowledge, no other system has been published that achieves a comparable correctness for open-domain German text. Parsing time is rather high due to the computational effort of multidimensional optimization; processing time is usually measured in seconds rather than milliseconds for each sentence.

5 Conclusions

We demonstrate a tool that lets the user parse, display and manipulate dependency structures according to a variant of dependency grammar in a graphical environment. We have found such an integrated environment invaluable for the development of precise and large grammars of natural language. Compared to other approaches, c.f. (Kaplan and Maxwell, 1996), the built-in WCDG parser provides a much better feedback by pinpointing possible *reasons* for the current grammar being unable to produce the desired parsing result. This additional information can then be immediately used in subsequent development cycles.

A similar tool, called Annotate, has been described in (Brants and Plaehn, 2000). This tool facilitates syntactic corpus annotation in a semi-automatic way by using a part-of-speech tagger and a parser running in the background. In comparison, Annotate is primarily used for corpus annotation, whereas XCDG supports the development of the parser itself also.

Due to its ability to always compute the single best analysis of a sentence and to highlight possible shortcomings of the grammar, the XCDG system provides a useful framework in which human design decisions on rules and weights can be effectively combined with a corpus-driven evaluation of their consequences. An alternative for a symbiotic cooperation in grammar development has been devised by (Hockenmaier and Steedman, 2002), where a skeleton of fairly general rule schemata is instantiated and weighed by means of a treebank annotation. Although the resulting grammar produced highly competitive results, it nevertheless requires a treebank being given in advance, while our approach also supports a simultaneous treebank compilation.

References

Thorsten Brants and Oliver Plaehn. 2000. Interactive corpus annotation. In *Proc. 2nd Int. Conf. on Language Resources and Engineering, LREC 2000*, pages 453–459, Athens.

Kilian Foth, Michael Daum, and Wolfgang Menzel. submitted. A broad-coverage parser for German based on defeasible constraints. In *Proc. 7. Konferenz zur Verarbeitung natürlicher Sprache, KONVENS-2004*, Wien, Austria.

Kilian A. Foth. 2004. Writing weighted constraints for large dependency grammars. In *Proc. Recent Advances in Dependency Grammars, COLING 2004*, Geneva, Switzerland.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proc. 40th Annual Meeting of the ACL, ACL-2002*, Philadelphia, PA.

Ronald M. Kaplan and John T. Maxwell. 1996. LFG grammar writer's workbench. Technical report, Xerox PARC.

Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. Guidelines für das Tagging deutscher Textcorpora. Technical report, Universität Stuttgart / Universität Tübingen.

Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. Ph.D. thesis, Department of Informatics, Hamburg University, Hamburg, Germany.