Gefördert vom
bmb+f
Bundesministerium für
Bildung, Wissenschaft,
Forschung und Technologie

Verbmobil
Verbundvorhaben

Modularizing Codescriptive Grammars for Efficient Parsing

Walter Kasper
Hans-Ulrich Krieger

DFKI

Vm
Report 140
July 1996

July 1996

Walter Kasper
Hans-Ulrich Krieger

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken

Tel.: (0681) 302 - 5282
Fax: (0681) 302 - 5341
e-mail: {kasper,krieger}@dfki.uni-sb.de

Gehört zum Antragsabschnitt: 15.8 Nichtsyntaktische Information für die semantische Auswertung

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 01 IV 101 K/1 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

Modularizing Codescriptive Grammars for Efficient Parsing*

Walter Kasper & Hans-Ulrich Krieger

Abstract

Unification-based theories of grammar allow to integrate different levels of linguistic descriptions in the common framework of typed feature structures. Dependencies among the levels are expressed by coreferences. Though highly attractive theoretically, using such codescriptions for analysis creates problems of efficiency. We present an approach to a modular use of codescriptions on the syntactic and semantic level. Grammatical analysis is performed by tightly coupled parsers running in tandem, each using only designated parts of the grammatical description. In the paper we describe the partitioning of grammatical information for the parsers and present results about the performance.

1 Introduction

Unification-based theories of grammar allow for an integration of different levels of linguistic descriptions in a common framework of typed feature structures. In HPSG this assumption is embodied in the fundamental concept of a *sign* Pollard and Sag1987, Pollard and Sag1994. A sign is a structure incorporating information from all levels of linguistic analysis, such as phonology, syntax, and semantics. This structure specifies interactions between these levels by means of coreferences, indicating the sharing of information. It also describes how the levels constrain each other mutually. Such a concept of linguistic description is attractive for several reasons:

1. it supports the use of common formalisms and data structures on all linguistic levels,

*This report is also published in: *Proc. of COLING-96*, August 5-9, 1996, Copenhagen.

2. it provides declarative and reversible interface specifications between these levels,
3. all information is simultaneously available, and
4. no procedural interaction between linguistic modules needs to be set up.

Similar approaches, especially for the syntax-semantics interface, have been suggested for all major kinds of unification-based theories, such as LFG or CUG. Halvorsen and Kaplan1988 call such approaches *codescriptive* in contrast to the approach of *description by analysis* which is closely related to sequential architectures where linguistic levels correspond to components, operating on the basis of the (complete) analysis results of lower levels. In a codescriptive grammar semantic descriptions are expressed by additional constraints.

Though theoretically very attractive, codescription has its price: (i) the grammar is difficult to modularize due to the fact that the levels constrain each other mutually and (ii) there is a computational overhead when parsers use the complete descriptions. Problems of these kinds which were already noted by Shieber1985 motivated the research described here. The goal was to develop more flexible ways of using codescriptive grammars than having them applied by a parser with full informational power. The underlying observation is that constraints in such grammars can play different roles:

- **Genuine constraints** which relate directly to the grammaticality (wellformedness) of the input. Typically, these are the syntactic constraints.
- **Spurious constraints** which basically build representational structures. These are less concerned with wellformedness of the input but rather with output for other components in the overall system. Much of semantic descriptions are of this kind.

If a parser treats all constraints on a par, it cannot distinguish between the structure-building and the filtering constraints. Since unification-based formalisms are monotonic, large structures are built up and have to undergo all the steps of unification, copying, and undoing in the processor. The costs of these operations (in time and space) increase exponentially with the size of the structures.

In the VERBMOBIL project, the parser is used within a speech translation system Wahlster1993, Kay, Gawron, and Norvig1994. The parser input consists of word lattices of hypotheses from speech recognition. The parser has to identify those paths in the lattice which represent a grammatically acceptable utterance. Parser and recognizer are incremental and interactively running in parallel. Even for short utterances, the lattices can contain several hundreds of word hypotheses, most of which do not combine to grammatical utterances. Parsing these lattices is much more complex than parsing written text.

The basic idea presented here is to distribute the labour of evaluating the constraints in the grammar on several processors (i.e., parsers). Important considerations in the design of the system were

1. increasing the performance,
2. achieving incremental and interactive behaviour,
3. minimizing the overhead in communication between the processors.

We used a mid-size HPSG-kind German grammar written in the $\mathcal{TDL}$ formalism Krieger and Schäfer1994. The grammar cospecifies syntax and semantics in the attributes SYN and SEM. A simplified example is shown in the lexical entry for the verb *come* in Fig. 1.

In the following section, we start with a top-down view of the architecture. After that we will describe the communication protocol between the parsing processes. Then several options for creating subgrammars from the complete grammar will be discussed. The subgrammars represent the distribution of information across the parsers. Finally, some experimental results will be reported.

2 The Architecture

The most important aspect for the distribution of analysis tasks and for defining modes of interaction is that one of the processes must work as a filter on the input word lattices, reducing the search space. The other component then works only with successful analysis results of the previous one. This means, that one parser is in control over the other, whereas the latter one is not directly exposed to the input. For reasons which will become obvious below, we will call the first of these parsers the SYN-*parser*, the second one controlled by the SYN-parser, the SEM-*parser*.

Another consideration to be taken into account is that the analysis should be *incremental* and *time synchronous*. This implies that the SYN-parser should not send its results only when it is completely finished, thus forcing the SEM-parser to wait.[1] *Interactivity* is another aspect we had to consider. The SEM-parser must be able to report back to the SYN-parser at least when its hypotheses failed.

[1]Another problem in incremental processing is that it is not known in advance when an utterance is finished or a new utterance starts. To deal with this, prosodic information is taken into account; see Kasper and Krieger1996 for more details.

$$\begin{bmatrix} \textit{intrans-fin-verb-lex} & \\ \text{INDEX} & \texttt{10281} \\ \text{SC} & \left\langle \begin{bmatrix} \textit{np-nom-type} & \\ \text{SEM} & \begin{bmatrix} \textit{cont2quant} & \\ \text{CONTENT} & \begin{bmatrix}\text{VAR} & \boxed{5}\end{bmatrix}\end{bmatrix} \\ \text{CAT-SYM} & \textit{np-cat} \\ \text{SYN} & \begin{bmatrix} \textit{syntax-type} & \\ \text{LOCAL} & \begin{bmatrix}\text{DET} & \textit{yes}\end{bmatrix} \\ \text{HEAD} & \begin{bmatrix}\text{AGR} & \boxed{7} \\ \text{CASE} & \boxed{1}\,\textit{nom-val}\end{bmatrix}\end{bmatrix}\end{bmatrix} \right\rangle \\ \text{SEM} & \begin{bmatrix} \textit{semantics-type} & \\ \text{QUANT} & \textit{empty-diff-list} \\ \text{CONTENT} & \begin{bmatrix} \textit{rp-type}\wedge\textit{subwff-inst-shr-var} & \\ \text{VAR} & \boxed{4} \\ \text{COND} & \begin{bmatrix} \textit{atomic-wff-type} & \\ \text{INST} & \boxed{4} \\ \text{AGENT} & \boxed{5} \\ \text{PRED} & \texttt{come} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{SYN} & \begin{bmatrix} \textit{syntax-type} & \\ \text{HEAD} & \begin{bmatrix}\text{CASE} & \boxed{1} \\ \text{AGR} & \boxed{7}\end{bmatrix}\end{bmatrix} \\ \text{PHON} & \texttt{"komme"} \end{bmatrix}$$

Figure 1: *The simplified feature structure for the German verb komme (to come).*

This would not be possible when the SEM-parser has to wait till the SYN-parser is finished. This requirement also constrains the exchange of messages.

Incrementality and interactivity imply a steady exchange of messages between the parsers. An important consideration then is that the overhead for this communication should not outweigh the gains of distributed processing. This rules out that the parsers should communicate by exchanging their analysis results in terms of resulting feature structures, since it would imply that on each communication event the parsers would have to analyze the structures to detect changes, whether a structure is part of other already known structures, etc. It is hard to see how this kind of communication can be interleaved with normal parsing activity in efficient ways.

In contrast to this, our approach allows to exploit the fact that the grammars employed by the parsers are derived from the same grammar and thereby "similar" in structure. This makes it possible to restrict the communication between the parsers to information about what rules were successfully or unsuccessfully applied. Each parser then can reconstruct on its side the state the other parser is in—how its chart or analysis tree looks like. Both parsers try to maintain or arrive at isomorphic charts. The approach allows that the parsers never need to exchange analysis results in terms of structures as the parsers should always be able to reconstruct these if necessary. On the other hand, this reconstructibility poses constraints on how the codescriptive grammar can be split up in subgrammars.

The requirements of incrementality, interactivity and efficient communication show that our approach does not emulate the "description by analysis" methodology in syntax-semantics interfaces on the basis of codescriptive grammars.

3 The Parsers and the Protocol

The SYN-parser and the SEM-parser are agenda-driven chart parsers. For speech parsing, the nodes represent points of times and edges represent word hypotheses/paths in the word lattice. The parsers communicate by exchanging *hypotheses*, bottom-up hypotheses from syntax to semantics and top-down hypotheses from semantics to syntax; see Kasper, Krieger, Spilker, and Weber1996 for an in-depth description of the current setup.

- **Bottom-up hypotheses** are emitted by the SYN-parser and sent to the SEM-parser. They undergo verification at the semantic level. A bottom-up hypothesis describes a passive edge (complete subtree) constructed by the syntax parser and consists of the identifier of the rule instantiation that represents the edge and the *completion history* of the constructed passive edge. Having passive status is a necessary but not sufficient condition for an edge to be sent as hypothesis. Whether a hypothesis is sent also depends on other criteria, such as its score.

- **Top-Down hypotheses** result from activities of the SEM-parser, trying to verify bottom-up-hypotheses. To keep the communication efforts low, only failures are reported back to the SYN-parser by sending simply the hypothesis' identifier. This narrows the space of successful hypotheses on the SYN-parser's side (see remarks in Section 4.3.1).

The central data structure by which synchronization and communication between the parsers is achieved is that of a completion history, containing a record on how a subtree was completed. Basically it tells us for each edge in the chart which other edges are spanned. The nodes in the chart correspond to points in time and edges to time intervals spanned. Completion histories are described by the following EBNF:

```
{R<rule-id><edge-id><start><end>{E<edge-id>}* |
 L<lex-id><edge-id><start><end>}+
```

<rule-id>, <lex-id>, <edge-id>, <start>, and <end> are integers. R<rule-id> and L<lex-id> denote rules and lexicon entries, resp. <edge-id> uniquely identifies a chart edge. Finally, <start> and <end> specify the start/end point of a spanning edge.

This protocol allows the parsers to efficiently exchange information about the structure of their chart without having to deal with explicit analysis results as feature structures. Since the SEM-parser does not directly work on linguistic input, there are two possible parsing modes:

- **Non-autonomous parsing.** The parsing process mainly consists of constructing the tree described by the completion history, using the semantic counterparts of the rules which led to a syntactic hypothesis. If this fails, it is reported back to the SYN-parser.

- **Quasi-autonomous parsing.** The parser extends the chart on its own through prediction and completion steps. Obviously, this is only possible after some initial information by the SYN-parser, since the SEM-parser is not directly connected to the input word lattice.

4 Compilation of Subgrammars

In the following, we discuss possible options and problems for the distribution of information in a cospecifying grammar. Our approach raises the question which of the parsers uses what information. This set of information is what we call a *subgrammar*. These subgrammars are generated from a common source grammar.

4.1 Reducing the Representational Overhead by Separating Syntax and Semantics

An obvious choice for splitting up the grammar was to separate the linguistic levels (strata), such as syntax and semantics. This choice was also motivated by the observation that typically the most important constraints on grammaticality of the input are in the syntactic part, while most of the semantics is purely representational.[2] A straightforward way to achieve this is by manipulating grammar rules and lexicon entries—for the SYN-parser, we recursively delete the information under the SEM attributes and similarly clear the SYN attributes to obtain the subgrammar for the SEM-parser. We abbreviate these subgrammars by G_{syn} and G_{sem} and the original grammar by G. This methodology reduces the size of the structures for the SYN-parser to about 30% of the complete structure. One disadvantage of this simple approach is that coreferences between syntax and semantics disappear (we call the collection of these common reentrancies the *coref skeleton*). This might lead to several problems which we address in Section 4.2. Section 4.3 then discusses possible solutions.

Another, more sophisticated way to keep the structures small is due to the type expansion mechanism in $\mathcal{TDL}$ Krieger and Schäfer1995. Instead of destructively modifying the feature structures beforehand, we can employ type expansion to let SYN or SEM unexpanded. This has the desired effect that we do not lose the coreference constraints and furthermore are free to expand parts of the feature structure afterwards. We will discuss this feature in Section 4.4.

4.2 Problems

Obviously, the major advantage of our method is that unification and copying become faster during processing, due to smaller structures. We can even estimate the speedup in the best case, viz., quasi-linear w.r.t. input structure if only conjunctive structures are used. Clearly, if many disjunctions are involved, the speedup might even be exponential.

However, the most important disadvantage of the compilation method is that it no longer guarantees *soundness*, that is, the subgrammar(s) might accept utterances which are ruled out by the full grammar. This is due to the simple fact that certain constraints have been eliminated in the subgrammars. If at least one such constraint is a filtering constraint, we automatically enlarge the language accepted by this subgrammar w.r.t. the original grammar. Clearly, *completeness* is not affected, since we do not add further constraints to the subgrammars.

At this point, let us focus on the estimation above, since it is only a best-case forecast. Clearly, the structures become smaller; however, due to the possible decrease of filter constraints, we must expect an increase of hypotheses in the parser. In fact, the experimental results in Section 5 show that our approach has a differ-

[2] This must be taken *cum grano salis* as it depends on how a specific grammar draws the line between syntax and semantics: selectional constraints, e.g., for verb arguments, are typically part of semantics and are "true" constraints. Also, semantic constraints would have a much larger impact if, for instance, agreement constraints are considered as semantic, too, as Pollard and Sag1994 suggest.

ent impact on the SYN-parser and the SEM-parser (see Figure 2). Our hope here, however, is that the increase of non-determinism inside the parser is compensated by the processing of smaller structures; see Maxwell III and Kaplan1991 for more arguments on this theme.

In general, even the intersection of the languages accepted by G_{syn} and G_{sem} does not yield the language accepted by G—only the weaker relation $\mathcal{L}(G) \subset \mathcal{L}(G_{syn}) \cap \mathcal{L}(G_{sem})$ holds. This behaviour is an outcome of our compilation schema, namely, cutting reentrancy points. Thus, even if an utterance is accepted by G with analysis *fs* encoded as a feature structure, it might be the case that the unification of the corresponding results for G_{syn} and G_{sem} will truly subsume *fs*: $fs \prec fs_{syn} \wedge fs_{sem}$

Let us mention further problems. Firstly, *termination* might change in case of the subgrammars. Consider a subgrammar which contains empty productions or unary (coercion) rules. Assume that such rules were previously "controlled" by constraints which are no longer present. Obviously, if a parser is not restricted through additional (meta-)constraints, the iterated application of these rules could lead to an infinite computation, i.e., a loop. This was sometimes the case during our experiments. Secondly, recursive rules could introduce infinitely many solutions for a given utterance. Theoretically, this might not pose a problem, since the intersection of two infinite sets of parse trees might be finite. However in practice, this problem might occur.

4.3 Solutions

In this section, we will discuss three solution to the problems mentioned before.

4.3.1 Feedback Loop

Although semantics construction is driven by the speech parser, the use of different subgrammars suggest that the speech parser should also be guided by the SEM-parser. This is achieved by sending back *falsified* hypotheses. Because hypotheses are uniquely identified in our framework, we must only send the integer that identifies the falsified chart edge. In the SYN-parser, this information might either lead to a true chart revision process or be employed as a filter to narrow the set of emitted bottom-up hypotheses.

4.3.2 Coref Skeleton

In order to guarantee correctness of the analysis, we might unify the results of both parsers with the corresponding coref skeletons at the end of an analysis. We did not pursue this strategy since it introduces an additional processing step during parsing. Instead, as explained above, it is preferable to employ type expansion here, letting SYN or SEM unexpanded, so that coreferences are preserved. This treatment will be investigated in Section 4.4.

4.3.3 Full-Size Grammar

The most straightforward way to guarantee soundness is simply by employing the full-size grammar in one of the two parsers. This might sound strange, but if one processor basically only verifies hypotheses from the other and does not generate additional hypotheses, the overhead is neglectable. We have used this scheme in that the SEM-parser operates on the full-size grammar, whereas the speech parser directly communicates with the word recognizer. This makes sense since the word lattice parser processes an order of magnitude more hypotheses than the SEM-parser; see Kasper, Krieger, Spilker, and Weber1996 for more details. Because the SEM-parser passes its semantic representation to other components, it makes further sense to guarantee total correctness here.

4.4 Improvements

This section investigates several improvements of our compilation approach, solving the problems mentioned before.

4.4.1 Identifying Functional Strata Manually

Normally, the grammarian "knows" which information needs to be made explicit. Hence, instead of differentiating between the linguistic strata SYN and SEM, we let the linguist identify which constraints filter and which only serve as a means for representation; see also Shieber1985. In contrast to the separation along linguistic levels, this approach adopts a functional view, cutting across linguistic strata. On this view, the syntactic constraints together with, e.g., semantic selection constraints would constitute a subgrammar.

4.4.2 Bookkeeping Unifications

In case that the grammarian is unaware of these constraints, it is at least possible to determine them relatively to a training corpus, simply by counting unifications. Features that occur only once on top of the input feature structures do not specialize the information in the resulting structure (actually the values of these features). Furthermore, unrestricted features (value $\top$) do not constrain the result. For instance,

$$\begin{bmatrix} \text{A} & s \\ \text{B} & \begin{bmatrix} t \\ \text{D} & w \end{bmatrix} \\ \text{C} & u \end{bmatrix} \wedge \begin{bmatrix} \text{A} & v \\ \text{B} & \top \end{bmatrix} = \begin{bmatrix} \text{A} & s \wedge v \\ \text{B} & \begin{bmatrix} t \\ \text{D} & w \end{bmatrix} \\ \text{C} & u \end{bmatrix}$$

indicates that only the path A needs to be made explicit, since its value is more specific than the corresponding input values: $s \wedge v \preceq s$ and $s \wedge v \preceq v$.

4.4.3 Partial Evaluation

Partial evaluation, as known from functional/logic programming, is a method of carrying out parts of computation at compile time that would otherwise be done at run time, hence improving run time performance of programs; see, e.g., Jones, Gomard, and Stestoft1993. Analogous to partial evaluation of definite clauses, we can partially evaluate annotated grammar rules, since they drive the derivation. Partial evaluation means here to substitute type symbols by their expanded definitions.

Because a grammar contains finitely many rules of the above form and because the daughters (the right hand side of the rule) are type symbols (and there are only finitely many of them), a great deal of this partial evaluation process can be performed offline. In contrast to a pure CF grammar with finitely many terminal/nonterminals, the evaluation process must not terminate, due to coreference constraints within feature structures. However, meta-constraints such as *offline parsability* or *lazy type expansion* (see next section) help us to determine those features which actively participate in unification during partial evaluation. In contrast to the previous method, partial evaluation is corpus-independent.

4.4.4 Lazy Type Expansion

We have indicated earlier that type expansion can be fruitfully employed to preserve the coref skeleton. Type expansion can also be chosen to expand parts of a feature structure on the fly at run time.

The general idea is as follows. Guaranteeing that the lexicon entries and the rules are consistent, we let everything unexpanded unless we are enforced to make structure explicit. As was the case for the previous two strategies, this is only necessary if a path is introduced in the resulting structure whose value is more specific than the value(s) in the input structure(s).

The biggest advantage of this approach is obvious—only those constraints must be touched which are involved in restricting the set of possible solutions. Clearly,

number of sentences: 56							
average length: 7.6							
	SynSem	**Syn**		**SemNA**		**SemQA**	
			%		%		%
run time:	30.6	15.2	50	15.4	50	45.8	150
#readings:	1.7	2.1	123	1.7	100	1.8	105
#hypotheses:	53.0	58.1	110	53.9	102	81.3	153
#chart edges:	192.0	215.0	112	58.1	30	301.1	156

Figure 2: *Experimental results of* SYN/SEM *separation.* **SemNA** *represents results for the* SEM*-parser in non-autonomous mode,* **SemQA** *the results for* SEM*-parser as quasi-autonomous semantic parser. The percentage values are relative to* **SynSem**.

such a test should be done every time the chart is extended. The cost of such tests and the on-line type expansions need further investigation.

5 Experimental Results

This section presents experimental results of our compilation method, indicating that the simple SYN/SEM separation does not match the distinction between true and spurious constraints, mostly due to semantic selectional constraints (see Fig. 2). The measurements have been obtained w.r.t. a corpus of 56 sentences/turns from 4 dialogs in the VERBMOBIL corpus.

The column **Syn** shows that parsing with syntax only takes 50% of the time of parsing with the complete grammar (**SynSem**). The number of readings, hypotheses, and chart edges only slightly increase here. The column **SemNA** shows the results for operating the SEM-parser in non-autonomous mode, that is, simply verifying/falsifying hypotheses from the SYN-parser. The parsing time of the coupled system is slightly higher than that for SYN-parser alone, due to the fact that the SEM-parser can only terminate after the SYN-parser has sent its last hypothesis. Nevertheless, the overall time is still only 50% of the system with the complete grammar (a sequential coupling only improves the overall run time for **SemNA** only by 5–10%). This illustrates that the efficiency of the parallel running system mainly depends on that of the SYN-parser. The column **SemQA** shows the results for the SEM-parser in quasi-autonomous mode. Since no syntactic constraints are involved in filtering, we expect a considerable increase in processing time and number of hypotheses. In fact, our measurements indicate that syntax (in our grammar) provides most of the genuine constraints.

These results show that the modularization of the grammar and the distribution of its information lead to a considerable increase in parsing efficiency, thus improving the computational applicability of codescriptive grammars.

6 Conclusions

Linguistic theories like HPSG provide an integrated view on linguistic objects by providing a uniform framework for all levels of linguistic analysis. Though attractive from a theoretical point of view, their implementation raises questions of computational tractability. We subscribe to that integrated view on the level of linguistic descriptions and specifications. However, from a computational view, we think that for special tasks not all that information is useful or required, at least not all at the same time.

In this paper we described attempts to make a more flexible use of integrated linguistic descriptions by splitting them up into subpackages that are handled by special processors. We also devised an efficient protocol for communication between such processors and addressed a number of problems and solutions, some of which need further empirical investigation. The results obtained so far indicate that our approach is very promising for making efficient use of codescriptive grammars in natural language analysis.

References

Halvorsen, Per-Kristian and Ronald M. Kaplan. 1988. Projections and Semantic Description in Lexical-Functional Grammar. In *Proceedings of 5th Generation Computer Systems*, pages 1116–1122.

Jones, Neil D., Carsten K. Gomard, and Peter Stestoft 1993. *Partial Evaluation and Automatic Program Generation.* New York: Prentice Hall.

Kasper, Walter and Hans-Ulrich Krieger. 1996. Integration of Prosodic and Grammatical Information in the Analysis of Dialogs. Verbmobil Report.

Kasper, Walter, Hans-Ulrich Krieger, Jörg Spilker, and Hans Weber. 1996. From Word Hypotheses to Logical Form: An Efficient Interleaved Approach. Verbmobil Report.

Kay, Martin, Jean Mark Gawron, and Peter Norvig. 1994. *Verbmobil. A Translation System for Face-to-Face Dialog.* CSLI Lecture Notes, volume 33. Chicago University Press.

Krieger, Hans-Ulrich and Ulrich Schäfer. 1994. $\mathcal{TDL}$—A Type Description Language for Constraint-Based Grammars. In *Proceedings of COLING-94*, pages 893–899.

Krieger, Hans-Ulrich and Ulrich Schäfer. 1995. Efficient Parameterizable Type Expansion for Typed Feature Formalisms. In *Proceedings of IJCAI-95*, pages 1428–1434.

Maxwell III, John T. and Ronald M. Kaplan. 1993. The Interface between Phrasal and Functional Constraints. In *Computational Linguistics*, Vol. 19, No. 4, pages 571–590.

Pollard, Carl and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics.* Vol. 1: Fundamentals. CSLI Lecture Notes, Volume 13. Stanford: CSLI.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar.* Chicago: University of Chicago Press.

Shieber, Stuart M. 1985. Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms. In *Proceedings of ACL-85*, pages 145–152.

Wahlster, Wolfgang. 1993. Verbmobil: Übersetzung von Verhandlungsdialogen. Verbmobil Report.