Abstract

Licenser rules have originally been introduced in Müller (1999) as a part of a grammar based on discontinuous constituents. We propose licenser rules as a means to avoid underspecified empty elements in grammars with continuous constituents. We applied them to a verb movement analysis of the German main clause with right sentence bracket and to complement extraposition. To reduce the number of unnecessary hypotheses, we extended the licenser rule concept with a licenser binding technique. We compared the licenser rule approach to an approach based on underspecified traces with respect to processing performance. In our experiment, the use of licenser rules reduced the parse time by a factor of 13.5.

1 Introduction

Some linguistic phenomena can be elegantly formalized by assuming phonetically empty elements (traces) which are related to overtly realized antecedents. However, the processing of empty elements is problematic in two ways. First, the parser can hypothesize infinitely many empty elements at any position in the input sentence. Second, empty elements tend to be dramatically underspecified unless information about the antecedent is locally available. This paper addresses the latter problem, but we will touch on the first issue in the context of our actual grammar implementation.

The structure of the paper is as follows. We first illustrate the problem of underspecifed traces by means of German examples. We then discuss related work and state our own contributions. Next, we show how licenser rules can be applied to a verb movement analysis of the German main clause and to complement extraposition. After introducing the licenser binding technique and discussing the problem of spurious ambiguities, we proceed to the experiments.

2 Traces and Underspecification

Sentence (1) is an example of a German main clause:

(1) *gestern liess ihn sein Vater ausschlafen*
yesterday let him his father sleep-late
'yesterday, his father let him sleep late'

In German main clauses, the predicate complex is split into a left and a right sentence bracket. The left sentence bracket contains the finite verb (*liess* in the above example) and the right sentence bracket contains all other verbal elements (*ausschlafen*). Each verbal element can contribute its own complements to the predicate complex, and these complements can be permuted almost freely between the two sentence brackets. To bridge the gap between the left and the right sentence bracket, it is common to assume a trace (an empty verbal head) which acts as the sentence-final counterpart of the sentence-initial finite verb:

(2) *gestern liess$_i$ ihn sein Vater ausschlafen* t_i

Empty verbal heads allow the German predicate complex to be analyzed locally, but they pose a great challenge for bottom-up parsing. In actual implementations such as Carpenter and Penn (2003), empty verbal heads typically are underspecified. In particular, the number and types of their complements are not sufficiently constrained. This leads to a large number of superfluous hypotheses, i.e. VPs which do not meet the requirements of the sentence-initial finite verb.

This problem is not limited to empty heads. In his analysis of partial verb phrase fronting in German, Müller (2005) assumes a trace which represents the fronted partial verb phrase within the right sentence bracket:

(3) *(seiner Tochter erzählen)$_i$ wird$_j$ er das wohl* t_i *müssen* t_j
his daughter tell will he this probably have-to
'he will probably have to tell this to his daughter'

The modal verb *müssen* subcategorizes for a verbal complement whose arguments it attracts. If the verbal complement is an underspecified trace t_i, the subcategorization information of the verbal complex t_i *müssen* is underspecified as well.

3 Contributions and Related Work

Approaches for processing traces more efficiently have been proposed in several publications. Johnson and Kay (1994) are mainly concerned with the fact that an infinite number of traces can hypothesized at any position in the input sentence. They suggest to associate each lexical entry with a bounded number of traces. Each parse can consume only those traces which are provided by the lexical items occuring in the sentence. Thus, the number of traces in any single parse is bounded and the parser is guaranteed to terminate (at least if the grammar does not permit infinite recursion). Besides demonstrating how traces can be assigned to lexical items in several GB analyses, they note that lexical items could be used to partially specify their associated traces.

Geißler (1994) and Batliner et al. (1996) adopt a similar idea for the processing of German main clauses. Whenever a lexical item of a sentence-initial finite verb is accessed, the corresponding empty verbal head is made available to the parser. As this approach establishes the relation between the trace and its antecedent, the empty verbal head is fully specified.

However, the antecedent of a verbal trace need not always be lexical. Counterexamples are fronted partial verb phrases (see previous section) and coordinated sentence-initial finite verbs in German:

(4) *sie (suchte und fand)$_i$ die Lösung* t_i.
she looked-for and found the solution
'she looked for the solution and found it.'

Müller (1999) introduced the concept of licenser rules to avoid underspecified verbal traces in his analysis of fronted partial verb phrases. In essence, licenser rules make information about a lexical or phrasal antecedent available locally. Licenser rules will be discussed in greater detail in the next section.

Our contributions are the following: we applied licenser rules to a grammar with continuous constituents. This is novel as licenser rules allow for non-adjacent daughters and were originally proposed for a grammar based on discontinuous constituents. In particular, we used licenser rules in an analysis of the German main clause and complement extraposition. Finally, we extended the licenser rule concept with a *licenser binding* mechanism. This technique allows to further reduce the number of superfluous hypotheses arising from the use of traces. The effect of licenser binding was assessed experimentally.

As one reviewer pointed out, the problem of underspecified traces also occurs for natural language generation. In the solution proposed by Shieber et al. (1990), the overtly realized antecedent can be thought of as being generated at the position of the trace. Then, the antecedent is replaced by an empty element. The empty element in turn is specified according to the antecedent. This solution is related to the licenser rule approach in that it generates a trace after its (phrasal or lexical) antecedent has been derived, incorporating all necessary information from the latter.

4 Licenser Rules

A licenser rule is a (typically discontinuous) binary production rule whose right-hand side contains an argument marked as the licenser argument. In HPSG terminology, a licenser argument has the property that it does not contribute to the phonological information of the mother sign. Or, from the parser's point of view, the application of a licenser rule results in a chart edge covering exactly the same words as the edge which instantiates the non-licenser argument. Further, it can be specified whether the licenser is supposed to be positioned before or after the non-licenser. Thus, a licenser schema can be interpreted as a unary rule which uses a licenser for one (or both) of the following purposes:

- Information contained in the licenser can be used to prevent the resulting edge from being underspecified.
- The presence of the licenser triggers the application of the unary rule. This can avoid unnecessary hypotheses if the resulting edge can only be part of a complete parse if there is a matching licenser.

An example for the former case is the trace-based analysis of the German main clause, whereas the latter case applies to complement extraposition. A more detailed account of how licenser rules are applied to those phenomena will be given in the following sections.

4.1 German Main Clauses with Right Sentence Bracket

As has been argued in Section 2, a trace-based analysis of the German main clause poses a particular challenge for bottom-up parsing. In the following we will adopt the HPSG analysis presented in Müller (2005). The right sentence bracket is assumed to contain an empty verbal head representing a sentence-final finite verb, such that the predicate complex can be analyzed locally. The predicate complex can then combine with its complements and adjuncts, eventually constituting a VP. The LOCAL value of the empty verbal head is duplicated in its head feature DSL, which is percolated to the verbal head's maximal projection (the VP). The sentence-inital finite verb finally subcategorizes for a VP with a matching DSL value, thus closing the gap between the left and the right sentence bracket.

To prevent the empty verbal head from being underspecified, we use the licenser schema shown in Figure 1. It basically combines the empty verbal head with its verbal complement. The NON-LICENSER-DTR represents the verbal complement and LICENSER-DTR is a sentence-initial finite verb. The empty verbal head is only implicit in this schema. The licenser daughter provides all information necessary to fully specify the empty verbal head: the DSL value of its complement is identical to the LOCAL value of the empty verbal head. Like this it is ensured that all maximal projections of the empty verbal head will meet the requirements of the licensing sentence-initial verb. This schema is implemented by means of a discontinuous licenser rule stating that the licenser daughter may appear anywhere to the left of the non-licenser daughter. In our grammar, we used licenser rules

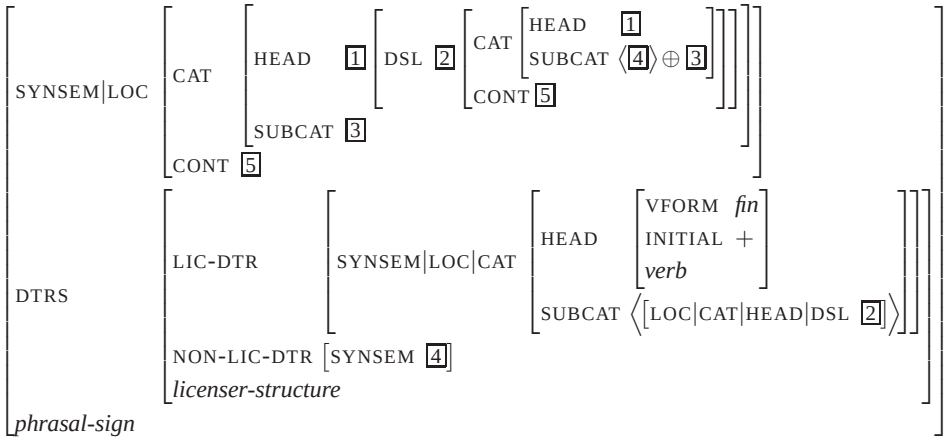

Figure 1: Licenser schema for German main clauses with right sentence bracket.

specifically for analyzing German main clauses with a right sentence bracket. A trace-based analysis of main clauses without right sentence bracket would be very costly, as the empty verbal head would have to be hypothesized at virtually every position in the sentence. If no right sentence bracket is present, we therefore resort to a left-branching structure as proposed in Crysmann (2003a).

4.2 Complement Extraposition

An efficient HPSG solution for the extraposition of adjuncts was proposed by Crysmann (2005). In HPSG with continuous constituents, the extraposition of complements is typically accounted for by means of a non-local dependency mechanism. Keller (1995) uses a lexical rule to move an extraposed complement from the SUBCAT list to an EXTRA set (a unary dominance schema or extraposition traces could be used alternatively). The EXTRA set is percolated by the Nonlocal Feature Principle until its members are eventually bound to matching phrases.

(Müller, 1999, p. 252) notes that this approach unnecessarily inflates the search space: a phrase with an extraposed complement is hypothesized even if no matching phrase is present. In a grammar with discontinuous constituents, this problem does not arise because a phrase and its extraposed complement form a discontinuous constituent. In grammars with continuous constituents, one can use licenser rules to reduce the search space. Our analysis is based on Keller (1995) and on the INERT/ACTIVE percolation approach proposed by Crysmann (2005) to avoid spurious ambiguities. However, as we use a licenser rule instead of a lexical rule, we can ensure that a non-local dependency is introduced only if there is a matching phrase somewhere to the right.

5 Licenser Binding

We have extended the licenser rule concept with a *licenser binding mechanism*. Our basic assumption is the following: in a parse of a complete sentence, each edge serving as a licenser also has to appear as a non-licenser at some point of the derivation. More precisely: if a licenser rule produces an edge e, the licenser edge has to appear as a sibling of some edge e' derived from e.

It is possible to early reject edges which will never satisfy this requirement. Suppose that there are two edges e_1 and e_x such that e_x has been used as a licenser in the derivation of e_1. If e_1 is combined with an edge $e_2 \neq e_x$ and if e_2 and e_x overlap, then no derivation of the resulting edge will be able to combine with e_x. Therefore, two edges e_1 and e_2 may be combined only if the following *licenser binding constraint* holds:

> *For any edge e_x that has instantiated a licenser argument in the derivation of e_1, either e_2 and e_x do not overlap or $e_2 = e_x$.*

Licenser binding can easily be implemented by adding a licenser set to each chart edge. For edges of lexical entries, the licenser set is empty. If two edges e_1 and e_2 with licenser sets L_1 and L_2 are combined by means of a non-licenser rule, the licenser set of the resulting edge is $L_1 \cup L_2$. If a licenser rule is applied and e_2 is the licenser edge, the resulting licenser set is $L_1 \cup \{e_2\}$.

This simple variant of licenser binding has the disadvantage that it interfers with ambiguity packing as proposed by Oepen and Carroll (2000): it may happen that two otherwise identical chart edges cannot be packed because they have

different licenser sets. However, the above idea can be straightforwardly generalized to a variant which does not impair ambiguity packing. The basic idea is that a chart edge should bear a disjunction of licenser sets rather than a single licenser set. If two edges e_1 and e_2 with licenser set disjunctions $L_{11} \vee ... \vee L_{1n}$ and $L_{21} \vee ... \vee L_{2m}$ are combined by rule application, the disjunction of the resulting edge is $(L_{11} \cup L_{21}) \vee (L_{11} \cup L_{22}) \vee ... \vee (L_{1n} \cup L_{2m})$. If e_2 is packed onto e_1, the disjunction of the latter is extended to $L_{11} \vee ... \vee L_{2m}$. It now holds that a chart edge can be safely rejected as soon as for each of its licenser sets the licenser binding constraint has been violated at some point of the derivation.

In order to simplify the bookkeeping which is necessary for the above generalization, we actually use a more restricted variant of licenser binding. In general, we do not allow the packing of two edges with different licenser sets. The single exception are edges which were produced by the same licenser rule with the same non-licenser edge. The licenser sets of such edges will only differ with respect to a single element, namely the licenser edge of the preceding licenser rule application. This case is particularly interesting, as the packed edges can actually be ignored in the unpacking phase.

6 Spurious Ambiguities

A general problem arising from licenser rules are spurious ambiguities. The licenser is expected to take on a very specific role with respect to the non-licenser at some later point in the derivation. To a certain degree, this is enforced by the specification of the trace and by the licenser binding constraint. However, it can still happen that the licenser does not take on the appropriate role:

(5) *sie* *habe*$_i$ *gesagt* t_i *er* *habe*$_j$ *es* *gewusst* t_j
she has said he has it known
'she said that he knew it'

The two instances of the auxiliary verb *habe* are syntactically and semantically identical. Therefore, the verbal complex *gewusst* t_j can be licensed by habe$_i$, even though habe$_j$ finally serves as the antecedent. As the "intended" licensing is also possible, we get one spurious ambiguity. Note that the licenser constraint is not violated: each licenser appears as a sibling of some phrase derived from a non-licenser. As mentioned in the previous section, spurious ambiguities of this kind can be reduced as a side-effect of ambiguity packing.

Still, spurious ambiguities are not banned completely. Consider the following scenario. There are two chart edges e_1 and e_2 whose LOCAL values are unifyable, but neither value subsumes the other. Each edge is used as the licenser of the same licenser rule with the same non-licenser edge. As a result, we get two edges $e_1{}'$ and $e_2{}'$ whose feature structures incorporate information (i.e. the LOCAL value) of their respective licenser. Because of this licenser information, neither edge subsumes the other. This in turn implies that ambiguity packing does not apply to $e_1{}'$ and $e_2{}'$.

As the licenser information of $e_1{}'$ and $e_2{}'$ is consistent with both e_1 and e_2, e_1 and e_2 can serve as the antecedent in derivations of both $e_1{}'$ and $e_2{}'$. Consequently, we get two spurious ambiguities in addition to the two proper readings.

The most general way to completely eliminate spurious ambiguities arising from licensing is to filter them out after parsing. This is achieved by "replaying" the unifications for each derivation tree without instantiating the licenser daughters. This operation yields a list of HPSG signs with fully instantiated DAUGHTER features. The spurious ambiguities are filtered out by removing the duplicates from this list.[1] Carrol and Oepen (2005) use such a "replay pass" to reintroduce the semantic features which were removed prior to ambiguity packing. If such a device is already applied for other reasons, the above filtering procedure is relatively cheap.

7 Experiments

7.1 The Parser

The following experiments were performed with our Java HPSG parser. A particularity of this parser is that it can process continuous as well as discontinuous rules. In discontinuous rules, the relative order of the rule arguments may or may not be specified. Regardless of the rule type, one or more (but not all) rule arguments can be specified to be licenser arguments. Different indexing structures are maintained to allow for the efficient processing of both types of rules. Further, the parser allows the specification of *relational constraints*. As in the TRALE system, see Haji-Abdolhosseini and Penn (2003), the evaluation of a relational constraint can be blocked and it can introduce non-determinism.

We use *equivalence-based ambiguity packing* rather than the more general subsumption-based packing proposed in Oepen and Carroll (2000). This enables us to efficiently retrieve a candidate set of potentially identical chart edges by means of hashing. The parser employs a special search strategy in order to facilitate the packing of edges that were produced by licenser rules (see Section 5). This is achieved by means of an agenda which uses two alternating phases. In the first phase, the parser tries to derive as many hypotheses as possible without applying licenser rules. The actual licensing takes place in the second phase, after (hopefully) all potential antecedents have been derived.

The parser applies many of the optimizations that have been proposed in the literature. It implements the *quasi-destructive unification algorithm* by Tomabechi (1992) and the *subgraph sharing* technique proposed in Malouf et al. (2000). It employs a *key-driven rule instantiation strategy* which was found to be beneficial in Oepen and Callmeier (2000). Further, the parser makes use of the *rule filter* and a technique for reducing the number of initial chart edges, both as proposed

[1] One might be tempted to simply remove duplicate derivation trees, again ignoring the licenser subtrees. However, this is not feasible in general. For example, non-deterministic procedural attachments such as the `member` relation may lead to hypotheses with identical derivation trees but non-identical feature structures.

in Kiefer et al. (2000). The parser also removes seperable verb prefixes for which there is no matching prefix verb in the chart. This is relevant for our experiments as many verb prefixes are homographs of frequent prepositions and verb prefixes are particularly expensive for an analysis based on underspecified traces.

7.2 The Base Grammar

The German grammar used in the following experiments is largely based on Müller (1999), Müller (2007), Crysmann (2003a) and Crysmann (2005). For a concise list of the covered phenomena we refer to the grammar test results which can be inspected on http://www.tik.ee.ethz.ch/~kaufmann/grammar/test07.html.

7.3 Licenser Rules vs. Underspecified Traces

To compare the performance of our licensing approach with that of an approach based on underspecified traces, we ran experiments with two slightly different grammars. Both grammars were derived from the base grammar by removing the rule for partial verb phrase fronting and disabling licensing for the complement extraposition rules. The grammars differ only in how the analysis of verb movement is implemented. The first grammar applies a licensing rule as discussed in Section 4.1. The second grammar uses underspecified traces. In both grammars, we assume a left-branching structure if there is no right sentence bracket.

The grammar with underspecified traces employs the optimizations proposed by Crysmann (2003b). In particular, we exploit the fact that the non-finite partial verbal complex in the right sentence bracket has a fully specified subcategorization list. This information can be used when the verbal complex is combined with the empty verbal head. The empty verbal head basically inherits the subcategorization list of the non-finite verbal complex. As is common in German HPSG, we assume that the subject is not part of this list. If the verbal complex is headed by a past participle, a subject may be added or not (omitting the subject is necessary to account for passive constructions). If it is headed by an infinitive, one or two underspecified complements are added, thereby allowing for raising and control. If the verbal complex consists of a verb prefix only, we assume a fully underspecified subcategorization list which is restricted to contain at most 5 elements. Underspecified list elements are restricted such that they do not match implausible complements such as determiners.

To compare the coverage of the two grammars, each of them was applied to our set of about 900 grammar development test sentences. It turned out that the grammar based on underspecified traces is in fact more restrictive. This is due to the fact that the partial specifications described above imply very specific assumptions about the grammar. For instance, it is assumed that the finite verb has at least as many (non-verbal) complements as its infinitive verbal complement. However, this is not correct for modal infinitives (6) and for imperative forms of subject control verbs (7):

(6) *das ist nicht zu verachten*
this is not to condemn
'this should not be condemned'

(7) *versucht zu schlafen!*
try to sleep!
'try to sleep!'

It is further assumed that all complements on the infinitive verb's subcat list are "inherited" by the finite verb. This does not comply with the analysis of dative passive presented in Müller (1999). The mentioned problems could be overcome by increasing the amount of underspecification, at the cost of higher processing complexity. However, we decided to stick to the more restrictive grammar.

To compare the parsing performance of the two approaches, both grammars were used to parse the same set of sentences on the same platform (Linux 2.6.16 on a Sun-Fire-X2200-M2-64 with 2 AMD Opteron 2218 processors and 7 GB of memory, Sun Microsystems Java Runtime environment 1.5.0_01). Licenser binding was enabled for the grammar based on licenser rules. The test data consisted of 458 sentences transcribed from three broadcasts of a German news shows (the "Tagesschau"). The sentence lengths ranged from single words up to 37 words, with a mean of 10.8 words.

approach	#edges	#nodes	time (s)
underspecified traces	4739	429341	2.49
licenser rules	908 (-81%)	66542 (-85%)	0.18 (-93%)

Table 1: For each approach, the number of edges, AVM nodes and the parse time are averaged over the 458 sentences.

As the results in Table 1 show, the parsing time could be reduced by a factor of 13.5 by using licenser rules instead of underspecified traces. Note that parsing was aborted if the representations of the AVMs required more than 8 millions of graph nodes. For the grammar based on underspecified traces, early termination occurred in 6 sentences. The grammar with licenser rules never required more than 1.6 millions of AVM nodes.

We further compared the number of readings of full parses and complete phrases for the results produced by the two grammars. The occasional differences could all be attributed to the fact that the grammar based on underspecified traces is more restrictive. This implies that spurious ambiguities as discussed in Section 6 did not occur at all.

7.4 Licenser binding

To quantify the benefit of the licenser binding mechanism, we processed the same set of 458 sentences with and without licenser binding. In contrast to the previous

experiment, we applied the full base grammar which uses licenser rules for partial verb phrase fronting, complement extraposition and German main clauses with right sentence bracket. The experiment was carried out on the same platform as the previous one.

The results are shown in Table 2. It can be seen that the number of edges and the memory consumption (as measured by the number of AVM nodes) are reduced by roughly 25%. The reduction in parse time (-11%) is smaller, but still significant. Note that the base grammar with licenser binding produces even less edges than the more restricted grammar from the previous experiment. This is due to the licensing of complement extraposition, which saves more edges than are produced by the partial verb phrase fronting rule.

approach	#edges	#nodes	time (s)
no licenser binding	1136	93218	0.282
licenser binding	875 (-23%)	67602 (-27%)	0.250 (-11%)

Table 2: For each approach, the number of edges, AVM nodes and the parse time are averaged over all 458 sentences.

8 Conclusions

We propose licenser rules as a technique to very selectively avoid underspecified traces in grammars with continuous constituents, particularly in grammars that are geared towards computational efficiency. We have applied this technique to an analysis of the German main clause with right sentence bracket and have found large performance gains in comparison to an implementation based on underspecified traces. We have further proposed a licenser binding technique to avoid unnecessary hypotheses. Our experiments demonstrate that this technique can yield a significant reduction in the number of chart edges as well as parse time.

Apart from the computational issue, licenser rules may also be advantageous from the grammar developer's point of view. Approaches based on underspecified traces typically need to encode prior knowledge about the formalized language in order to be computationally tractable. Such optimizations introduce redundancy and affect the elegance of the grammar – in fact, they can even reduce its coverage. As licenser rules provide all information about the antecedent, such extra knowledge is not necessary.

Licenser rules are a processing technique rather than a formal device. Thus, it seems to be desirable to hide them from the grammar developer. One possible approach might be to introduce traces with parser-specific annotations. These traces are then compiled into the grammar, which amounts to adding licenser rules and removing some of the original rules.

References

Batliner, A., Feldhaus, A., Geißler, S., Kießling, A., Kiss, T., Kompe, R. and Nöth, E. 1996. Integrating syntactic and prosodic information for the efficient detection of empty categories. In *Proceedings of the 16th conference on Computational linguistics*, pages 71–76, Morristown, NJ, USA: Association for Computational Linguistics.

Carpenter, B. and Penn, G. 2003. *The Attribute Logic Engine with TRALE extensions – User's Guide*.

Carrol, J. and Oepen, S. 2005. High efficiency realization for a wide-coverage unification grammar. In Robert Dale, Kam-Fai Wong, Jian Su and Oi Yee Kwong (eds.), *Proceedings of the IJCNLP 2005, Jeju Island, Corea*, pages 165 – 176, Springer-Verlag Berlin Heidelberg.

Crysmann, B. 2003a. On the efficient implementation of German verb placement in HPSG. In *Proceedings of RANLP*.

Crysmann, B. 2003b. On the efficient implementation of German verb placement in HPSG. Conference talk, http://www.coli.uni-saarland.de/~crysmann/papers/bc-RANLP2003-pres.pdf.

Crysmann, B. 2005. Relative Clause Extraposition in German: An Efficient and Portable Implementation. *Research on Language and Computation* 3(1), 61–82.

Geißler, S. 1994. Lexikalische Regeln in der IBM-Basisgrammatik. Verbmobil Report, No. 20, Heidelberg.

Haji-Abdolhosseini, M. and Penn, G. 2003. *TRALE Reference Manual - Draft*. Http://www.ale.cs.toronto.edu/docs/ref/ale_ref.pdf.

Johnson, M. and Kay, M. 1994. Parsing and empty nodes. *Computational Linguistics* 20(2), 289–300.

Keller, F. 1995. Towards an account of extraposition in HPSG. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 301–306, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kiefer, B., Krieger, H.-U. and Nederhof, M.-J. 2000. Efficient and Robust Parsing of Word Hypotheses Graphs. In Wolfgang Wahlster (ed.), *Verbmobil. Foundations of Speech-to-Speech Translation*, pages 280–295, Berlin, Germany: Springer, artificial intelligence edition.

Malouf, R., Carroll, J. and Copestake, A. 2000. Efficient feature structure operations without compilation. *Natural Language Engineering* 6, 29–46.

Müller, S. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Linguistische Arbeiten, No. 394, Tübingen: Max Niemeyer Verlag.

Müller, S. 2005. Zur Analyse der deutschen Satzstruktur. *Linguistische Berichte* 201, 3–39.

Müller, S. 2007. *Head-Driven Phrase Structure Grammar: Eine Einführung*. Stauffenburg Einführungen, Nr. 17, Tübingen: Stauffenburg Verlag.

Oepen, S. and Callmeier, U. 2000. Measure for Measure: Parser Cross-Fertilization. Towards Increased Component Comparability and Exchange. In *Proceedings of the 6th International Workshop on Parsing Technology (IWPT '00), February 23-25*, pages 183–194, Trento, Italy.

Oepen, S. and Carroll, J. 2000. Ambiguity packing in constraint-based parsing: practical results. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 162–169, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Shieber, S., van Noord, G., Pereira, F. and Moore, R. 1990. Semantic-head-driven generation. *Comput. Linguist.* 16(1), 30–42.

Tomabechi, H. 1992. Quasi-Destructive Graph Unification with Structure-Sharing. In *COLING*, pages 440–446.