

An HPSG Parser Supporting Discontinuous Licenser Rules

February 2007

1 Introduction

We present an HPSG parser which can process discontinuous licenser rules in grammars with continuous constituents. We demonstrate how licenser rules can be used to reduce the parser's search space by applying them to the analysis of German main clauses with a right sentence bracket and to complement extraposition.

Before discussing the parser and the licenser rule technique, we would like to relate it to its intended application. The parser has been developed as a part of a project whose aim is to improve speech recognition by means of a linguistically motivated grammar. To put it simply, the N-best speech recognizer hypotheses are searched for phrases which are grammatically correct. From the presence and the syntactic structure of such phrases we accumulate evidence in favour of the correctness of a given hypothesis.

This particular application makes specific demands on the linguistic subsystem. On the one hand, the grammar should be as precise as possible in order to provide reliable evidence for the correctness of a recognizer hypothesis. On the other, computational efficiency is crucial for carrying out real-world experiments. Consequently, we are disposed to compromise on formal elegance or linguistic adequacy (e.g. on the semantic level) if a "proper" analysis would be computationally too expensive. This tendency is reflected in the analyses discussed in subsequent sections.

A large-coverage German HPSG has been developed in parallel to the parser presented in this article. We will use parts of this grammar to exemplify the licensing technique discussed in section 3. Those who are interested in the coverage of the grammar are referred to the web site <http://www.tik.ee.ethz.ch/~kaufmann/grammar/test.html>, which contains the latest test results on a set of more than 1000 test sentences.

2 Parser characteristics

Our HPSG parser is entirely written in Java and therefore platform-independent. The heart of the system is a *bottom-up chart parser*. It employs a *key-driven* rule instantiation strategy, which was found to lead to a better performance than a head-driven strategy (see [1]). The feature structure unification operation is based on Tomabechi's quasi-destructive algorithm [2] with subgraph sharing [3]. The parser supports *equivalence-based ambiguity packing*. We do not use the more general subsumption-based packing proposed in [4] as that approach entails a considerable computing overhead. With equivalence-based ambiguity packing, hashing can be used to efficiently retrieve a small candidate set of potentially equivalent chart edges.

Feature structures can be further restricted by means of *relational constraints*, which is atypical for a system implemented in a procedural programming language. As in TRALE (see [5]), the evaluation of a relational constraint can be blocked or it can introduce non-determinism. The semantics of the relational constraints have to be specified by means of procedural Java code.

Another feature of the parser is its *morphological analysis* component: each word in a sentence to be parsed is morphologically analysed, and its feature structures are derived from the feature structures of its morphemes. This allows for an elegant treatment of German compound nouns (as the

number of possible compound nouns is virtually unlimited, we would need an infinite full-form lexicon in order to store them). For highly inflected languages such as German, the morphological analysis approach generally results in a more compact lexicon: for a given inflected word, one has to store only a small number of stems instead of an entry for each inflected form.

A particularity of our parser is that it can process two types of rules. *Continuous rules* require their daughters to be adjacent and impose a total linear ordering on them. *Discontinuous rules* allow non-adjacent daughters and may or may not fix their relative order. The grammar writer has to explicitly specify to which type a grammar rule belongs.

3 Licenser Rules

A licenser rule is a (typically discontinuous) binary production rule whose right-hand side contains an argument marked as a *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. Thus, a licenser schema can be interpreted as a unary rule which uses a licenser for one (or both) of the following purposes:

- The presence of the licenser triggers the application of the unary rule. This can avoid unnecessary hypotheses if the edge produced by the unary rule can only be part of a complete parse if there is a matching licenser (for an example see section 3.2).
- Information contained in the licenser can be used to prevent the result of the rule application from being underspecified (see section 3.1).

We have extended this 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. More precisely: if a licenser rule produces an edge e , the edge instantiating the rule's licenser argument has to appear at some point as a sibling of an 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 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 $e_2 = e_x$.

Without this constraint the use of licenser schemata can result in a large number of spurious ambiguities. Note that the constraint does not rule out spurious ambiguities of this kind completely; they still may occur in very rare cases.

The concept of licenser schemata has first been used in the Babel system to avoid underspecified verbal traces in the analysis of fronted partial verb phrases (see [6, p. 357]). However, it has not been reported that something similar to the proposed licenser binding mechanism has been used in this work. Further, the Babel grammar was based on discontinuous constituents, whereas we apply the licenser concept to a grammar with continuous constituents.

3.1 German main clauses with right sentence bracket

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 in the predicate complex can contribute its own complements, and these complements can be permuted almost freely between the two sentence brackets. Consequently, a grammar somehow has to bridge the gap between the left and the right sentence bracket in order to be able to correctly constrain the number and types of complements.

In HPSG with continuous constituents, this is typically achieved by means of empty verbal heads (traces). In the following we will adopt the analysis presented in [7]. 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 feature 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-initial finite verb finally subcategorizes for a VP with a matching DSL value, thus closing the gap between the left and the right sentence bracket.

In actual implementations, the empty verbal head is typically underspecified, in particular with respect to the number and types of its complements. This leads to a large amount of superfluous hypotheses, i.e. VPs which don’t meet the requirements of the sentence-initial finite verb. This can be avoided by using a licenser rule.

The immediate dominance schema in figure 1 basically combines the empty verbal head with its verbal complement (a partial verb phrase). While the empty verbal head is only implicit in this schema, NON-LICENSER-DTR represents the verbal complement and LICENSER-DTR is a sentence-initial finite verb. The latter provides all information necessary to fully specify the empty verbal head: the DSL value of its complement is equivalent 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.

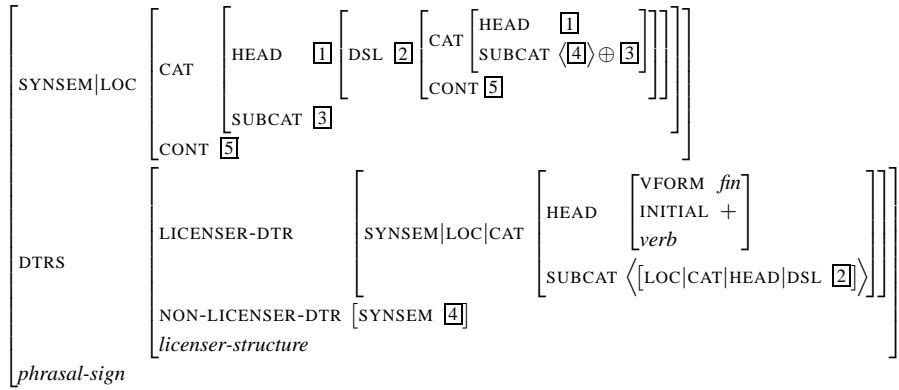


Figure 1: Schema for German main clauses with right sentence bracket.

In our grammar, we used licenser rules 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. In this case, we resort to a left-branching structure as proposed in [8]. The left-branching approach has been challenged mainly on semantic grounds (see e.g. [9, p. 212]), which are, however, not relevant for our application.

3.2 Complement Extraposition

For the extraposition of adjuncts there exist efficient HPSG solutions (see [10]). In HPSG with continuous constituents, the extraposition of complements is typically accounted for by means of a non-local dependency mechanism¹. [11] 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.

[6, 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 as 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 [11] and on the INERT/ACTIVE percolation approach proposed by [10] 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*.

4 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.

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.

There are alternative approaches to avoid underspecified traces. [12] use *fully specified traces* which are attached to the lexicon entries of verbs. However, they do not report any additional constraints to reduce the search space, such as the relative order of the verb and its trace or anything similar in effect to the licenser binding mechanism.

References

- [1] Stephan Oepen and Ulrich Callmeier, “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, Trento, Italy, 2000, pp. 183–194.
- [2] Hideto Tomabechi, “Quasi-destructive graph unification with structure-sharing,” in *COLING*, 1992, pp. 440–446.
- [3] Robert Malouf, John Carroll, and Ann Copestake, “Efficient feature structure operations without compilation,” *Natural Language Engineering*, vol. 6, pp. 29–46, 2000.
- [4] Stephan Oepen and John Carroll, “Ambiguity packing in constraint-based parsing: practical results,” in *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, San Francisco, CA, USA, 2000, pp. 162–169, Morgan Kaufmann Publishers Inc.
- [5] Mohammad Haji-Abdolhosseini and Gerald Penn, *TRALE Reference Manual - Draft*, 2003, http://www.ale.cs.toronto.edu/docs/ref/ale_ref.pdf.

¹Extraposition is considered a local phenomenon. However, the NONLOC feature percolation mechanism lends itself as a convenient means to analyze extraposition.

- [6] Stefan Müller, *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*, Number 394 in *Linguistische Arbeiten*. Max Niemeyer Verlag, Tübingen, 1999.
- [7] Stefan Müller, “Zur Analyse der deutschen Satzstruktur,” *Linguistische Berichte*, vol. 201, pp. 3–39, 2005.
- [8] Berthold Crysmann, “On the efficient implementation of german verb placement in hpsg,” in *Proceedings of RANLP 2003*, 2003.
- [9] Stefan Müller, “Continuous or discontinuous constituents? a comparison between syntactic analyses for constituent order and their processing systems,” *Research on Language and Computation, Special Issue on Linguistic Theory and Grammar Implementation*, vol. 2, no. 2, pp. 209–257, 2004.
- [10] Berthold Crysmann, “Relative clause extraposition in german: An efficient and portable implementation,” *Research on Language and Computation*, vol. 3, no. 1, pp. 61–82, 2005.
- [11] Frank Keller, “Towards an account of extraposition in hpsg,” in *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, San Francisco, CA, USA, 1995, pp. 301–306, Morgan Kaufmann Publishers Inc.
- [12] Anton Batliner, Anke Feldhaus, Stefan Geißler, Andreas Kießling, Tibor Kiss, Ralf Kompe, and Elmar Nöth, “Integrating syntactic and prosodic information for the efficient detection of empty categories,” in *Proceedings of the 16th conference on Computational linguistics*, Morristown, NJ, USA, 1996, pp. 71–76, Association for Computational Linguistics.