Off-line Optimization for Earley-style HPSG Processing

Guido Minnen, Dale Gerdemann, Thilo Götz *
Sonderforschungsbereich 340
University of Tübingen
Wilhelmstr. 113
72074-Tübingen
Germany
E-mail: minnen@sfs.nphil.uni-tuebingen.de

Abstract

A novel approach to HPSG based natural language processing is described that uses an off-line compiler to automatically prime a declarative grammar for generation or parsing, and inputs the primed grammar to an advanced Earley-style processor. This way we provide an elegant solution to the problems with empty heads and efficient bidirectional processing which is illustrated for the special case of HPSG generation. Extensive testing with a large HPSG grammar revealed some important constraints on the form of the grammar.

1 Introduction

Bidirectionality of grammar is a research topic in natural language processing that is enjoying increasing attention (Strzalkowski, 1993a). This is mainly due to the clear theoretical and practical advantages of bidirectional grammar use (see, among others, Appelt, 1987). We address this topic in describing a novel approach to HPSG (Pollard and Sag, 1994) based language processing that uses an off-line compiler to automatically prime a declarative grammar for generation or parsing, and hands the primed grammar to an advanced Earley processor. The developed techniques are direction independent in the sense that they can be used for both generation and parsing with HPSG grammars. In this paper, we focus on the application of the developed techniques in the context of the comparatively neglected area of HPSG generation.

Shieber (1988) gave the first use of Earley's algorithm for generation, but this algorithm does not use the prediction step to restrict feature instantiations on the predicted phrases, and thus lacks goal-directedness. Though Gerdemann (1991) showed how to modify the restriction function to make top-down information available for the bottom-up completion step, Earley generation with top-down prediction still has a problem in that generating the subparts of a construction in the wrong order might lead to massive nondeterminacy or even nontermination. Gerdemann (1991) partly overcame this problem by incorporating a head-driven strategy into Earley's algorithm. However, evaluating the head of a construction prior to its dependent subparts still suffers from efficiency problems when the head of a construction is either missing, displaced or underspecified. Furthermore, Martinović and Strzalkowski (1992) and others have observed that a simple head-first reordering of the grammar rules may still make insufficient restricting information available for generation unless the form of the grammar is restricted to unary or binary rules.

Strzalkowski's Essential Arguments Approach (EAA; 1993b) is a top-down approach to generation and parsing with logic grammars that uses off-line compilation to automatically invert parser-oriented logic grammars. The inversion process consists of both the automatic static reordering of nodes in the grammar, and the interchanging of arguments in rules with recursively defined heads. It is based on the notion of *essential arguments*, arguments which must be instantiated to ensure the efficient and terminating execution of a node. Minnen et al. (1995) observe that the EAA is computationally infeasible, because it demands the investigation of almost all possible permutations of a grammar. Moreover, the interchanging of arguments in recursive procedures as proposed by Strzalkowski fails to guarantee that input and output grammars are semantically equivalent. The Direct Inversion Approach (DIA) of Minnen et al. (1995) overcomes these problems by making the reordering process more goal-directed and developing a reformulation technique that al-

*The presented research was sponsored by Teilprojekt B4 "Constraints on Grammar for Efficient Generation" of the Sonderforschungsbereich 340 "Sprachtheoretische Grundlagen für die Computerlinguistik" of the Deutsche Forschungsgemeinschaft. The authors wish to thank Paul King, Detmar Meurers and Shuly Wintner for valuable comments and discussion. Of course, the authors are responsible for all remaining errors.

lows the successful treatment of rules which exhibit head-recursion. Both the EAA and the DIA were presented as approaches to the inversion of parser-oriented grammars into grammars suitable for generation. However, both approaches can just as well take a declarative grammar specification as input to produce generator and/or parser-oriented grammars as in Dymetman et al. (1990). In this paper we adopt the latter theoretically more interesting perspective.

We developed a compiler for off-line optimization of phrase structure rule-based typed feature structure grammars which generalizes the techniques developed in the context of the DIA, and we advanced a typed extension of the Earley-style generator of Gerdemann (1991). Off-line compilation (section 3) is used to produce grammars for the Earley-style generator (section 2). We show that our use of off-line grammar optimization overcomes problems with empty or displaced heads. The developed techniques are extensively tested with a large HPSG grammar for partial VP topicalization in German (Hinrichs et al., 1994). This uncovered some important constraints on the form of the phrase structure rules (phrase structure rules) in a grammar imposed by the compiler (section 4).

2 Advanced Earley Generation

As Shieber (1988) noted, the main shortcoming of Earley generation is a lack of goal-directedness that results in a proliferation of edges. Gerdemann (1991) tackled this shortcoming by modifying the restriction function to make top-down information available for the bottom-up completion step. Gerdemann's generator follows a head-driven strategy in order to avoid inefficient evaluation orders. More specifically, the head of the right-hand side of each grammar rule is distinguished, and distinguished categories are scanned or predicted upon first. The resulting evaluation strategy is similar to that of the head-corner approach (Shieber et al., 1990; Gerdemann and Hinrichs, in press): prediction follows the main flow of semantic information until a lexical pivot is reached, and only then are the head-dependent subparts of the construction built up in a bottom-up fashion. This mixture of top-down and bottom-up information flow is crucial since the top-down semantic information from the goal category must be integrated with the bottom-up subcategorization information from the lexicon. A strict top-down evaluation strategy suffers from what may be called *head-recursion*, i.e. the generation analog of left recursion in parsing. Shieber et al. (1990) show that a top-down evaluation strategy will fail for rules such as VP → VP X, irrespective of the order of evaluation of the right-hand side categories in the rule. By combining the off-line optimization process with a mixed bottom-up/top-down evaluation strategy, we can refrain from a complete reformulation of the grammar as, for example, in Minnen et al. (1995).

2.1 Optimizations

We further improved a typed extension of Gerdemann's Earley generator with a number of techniques that reduce the number of edges created during generation. Three optimizations were especially helpful. The first supplies each edge in the chart with two indices, a *backward index* pointing to the state in the chart that the edge is predicted from, and a *forward index* pointing to the states that are predicted from the edge. By matching forward and backward indices, the edges that must be combined for completion can be located faster. This indexing technique, as illustrated below, improves upon the more complex indices in Gerdemann (1991) and is closely related to OLDT-resolution (Tamaki and Sato, 1986).

$$
\begin{array}{l}
1)\ \mathit{active}(X_0 \rightarrow X_1 \bullet X_2, 1, 2) \\
\quad \vdots \\
2)\ \mathit{active}(X_2 \rightarrow \bullet\, Y_1\, Y_2, 2, 3) \\
\quad \vdots \\
3)\ \mathit{active}(X_2 \rightarrow Y_1 \bullet Y_2, 2, 4) \\
\quad \vdots \\
4)\ \mathit{passive}(X_2 \rightarrow Y_1\, Y_2 \bullet, 2)
\end{array}
$$

Active edge 2 resulted from active edge 1 through prediction. The backward index of edge 2 is therefore identified with the forward index of edge 1. Completion of an active edge results in an edge with identical backward index. In the case of our example, this would be the steps from edge 2 to edge 3 and edge 3 to edge 4. As nothing gets predicted from a passive edge (4), it does not have a forward index. In order to use passive edge 4 for completion of an active edge, we only need to consider those edges which have a forward index identical to the backward index of 4.

The second optimization creates a table of the categories which have been used to make predictions from. As discussed in Gerdemann (1991), such a table can be used to avoid redundant predictions without a full and expensive subsumption test. The third indexes lexical entries which is necessary to obtain constant-time lexical access.

The optimizations of our Earley-generator lead to significant gains in efficiency. However, despite these heuristic improvements, the problem of goal-directedness is not solved.

2.2 Empty Heads

Empty or displaced heads present the principal goal-directedness problem for any head-driven generation approach (Shieber et al., 1990; König, 1994; Gerdemann and Hinrichs, in press), where empty head refers not just to a construction in which the head has an empty phonology, but to any construction in which the head is partially unspecified. Since phonology does not guide generation, the phonological realization of the head of a construction plays no part in the generation of that construction. To better illustrate the problem that underspecified heads pose, consider the sentence:

Hat Karl Marie geküßt?
Has Karl Marie kissed?
"Did Karl kiss Mary?"

for which we adopt the argument composition analysis presented in Hinrichs and Nakazawa (1989): the subcat list of the auxiliary verb is partially instantiated in the lexicon and only becomes fully instantiated upon its combination with its verbal complement, the main verb. The phrase structure rule that describes this construction is [1]

$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{subcat} & \langle\rangle \\ \text{cont} & \boxed{5} \end{bmatrix} \longrightarrow$$

$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{fin} & + \\ \text{aux} & + \\ \text{subcat} & \left\langle \boxed{3} | \boxed{4} \right\rangle \\ \text{cont} & \boxed{5} \end{bmatrix}, \boxed{1}, \boxed{2}, \boxed{3}\begin{bmatrix} \text{cat} & \text{v} \\ \text{fin} & - \\ \text{aux} & - \\ \text{subcat} & \boxed{4}\left\langle \boxed{1} | \boxed{2} \right\rangle \end{bmatrix}$$

Though a head-driven generator must generate first the head of the rule, nothing prescribes the order of generation of the complements of the head. If the generator generates second the main verb then the subcat list of the main verb instantiates the subcat list of the head, and generation becomes a deterministic procedure in which complements are generated in sequence. However, if the generator generates second some complement other than the main verb, then the subcat list of the head contains no restricting information to guide deterministic generation, and generation becomes a generate-and-test procedure in which complements are generated at random, only to be eliminated by further unifications. Clearly then, the order of evaluation of the complements in a rule can profoundly influence the efficiency of generation, and an efficient head-driven generator must order the evaluation of the complements in a rule accordingly.

2.3 Off-line versus On-line

Dynamic, on-line reordering can solve the ordering problem discussed in the previous subsection, but is rather unattractive: interpreting grammar rules at run time creates much overhead, and locally determining the optimal evaluation order is often impossible. Goal-freezing can also overcome the ordering problem, but is equally unappealing: goal-freezing is computationally expensive, it demands the procedural annotation of an otherwise declarative grammar specification, and it presupposes that a grammar writer possesses substantial computational processing expertise. We chose instead to deal with the ordering problem by using off-line compilation to automatically optimize a grammar such that it can be used for generation, without additional provision for dealing with the evaluation order, by our Earley generator.

3 Off-line Grammar Optimization

Our off-line grammar optimization is based on a generalization of the dataflow analysis employed in the DIA to a dataflow analysis for typed feature structure grammars. This dataflow analysis takes as input a specification of the paths of the start category that are considered fully instantiated. In case of generation, this means that the user annotates the path specifying the logical form, i.e., the path cont (or some of its subpaths), as *bound*. We use the type hierarchy and an extension of the unification and generalization operations such that path annotations are preserved, to determine the flow of (semantic) information *between* the rules and the lexical entries in a grammar. Structure sharing determines the dataflow *within* the rules of the grammar.

The dataflow analysis is used to determine the relative efficiency of a particular evaluation order of the right-hand side categories in a phrase structure rule by computing the *maximal degree of nondeterminacy* introduced by the evaluation of each of these categories. The maximal degree of nondeterminacy introduced by a right-hand side category equals the maximal number of rules and/or lexical entries with which this category unifies given its binding annotations. The optimal evaluation order of the right-hand side categories is found by comparing the maximal degree of nondeterminacy introduced by the evaluation of the individual categories with the degree of nondeterminacy the grammar is allowed to introduce: if the degree of nondeterminacy introduced by the evaluation of one of the right-hand side categories in a rule exceeds the admissible degree of nondeterminacy the ordering at hand is rejected. The degree of nondeterminacy the grammar is allowed to introduce is originally set to one and consecutively incremented until the optimal evaluation order for all rules in the grammar is found.

3.1 Example

The compilation process is illustrated on the basis of the phrase structure rule for argument composi-

[1] For expository reasons, we refrain from a division between the subject and the other complements of a verb as in chapter 9 of Pollard and Sag (1994). The test-grammar does make this division and always guarantees the correct order of the complements on the *comps* list with respect to the obliqueness hierarchy. Furthermore, we use abbreviations of paths, such as *cont* for *synsem|loc|cont*, and assume that the semantics principle is encoded in the phrase structure rule.

tion discussed in 2.2. Space limitations force us to abstract over the recursive optimization of the rules defining the right-hand side categories through considering only the defining lexical entries.

Unifying the user annotated start category with the left-hand side of this phrase structure rule leads to the annotation of the path specifying the logical form of the construction as bound (see below). As a result of the structure-sharing between the left-hand side of the rule and the auxiliary verb category, the cont-value of the auxiliary verb can be treated as bound, as well. In addition, the paths with a value of a maximal specific type for which there are no appropriate features specified, for example, the path cat, can be considered bound:

$$\begin{bmatrix} \text{cat}_{bound} & \text{v} \\ \text{subcat}_{bound} & \langle\rangle \\ \text{cont}_{bound} & \boxed{5} \end{bmatrix} \longrightarrow$$
$$\begin{bmatrix} \text{cat}_{bound} & \text{v} \\ \text{fin}_{bound} & + \\ \text{aux}_{bound} & + \\ \text{subcat} & \left\langle \boxed{3} \mid \boxed{4} \right\rangle \\ \text{cont}_{bound} & \boxed{5} \end{bmatrix}, \boxed{1}, \boxed{2}, \boxed{3}\begin{bmatrix} \text{cat}_{bound} & \text{v} \\ \text{fin}_{bound} & - \\ \text{aux}_{bound} & - \\ \text{subcat} & \boxed{4}\left\langle \boxed{1} \mid \boxed{2} \right\rangle \end{bmatrix}$$

On the basis of this annotated rule, we investigate the lexical entries defining its right-hand side categories. The auxiliary verb category is unified with its defining lexical entries (under preservation of the binding annotations). The following is an example of such a lexical entry. (Note that subpaths of a path marked as bound are considered bound too.)

$$\begin{bmatrix} \text{phon} & \text{hat} \\ \text{cat}_{bound} & \text{v} \\ \text{fin}_{bound} & + \\ \text{aux}_{bound} & + \\ \text{subcat} & \left\langle \begin{bmatrix} \text{cont}_{bound} & \boxed{5} \end{bmatrix} \right\rangle \\ \text{cont}_{bound} \mid \text{nucleus}_{bound} \mid \text{arg}_{bound} & \boxed{5} \end{bmatrix}$$

The binding annotations of the lexical entries defining the auxiliary verb are used to determine with how many lexical entries the right-hand side category of the rule maximally unifies, i.e., its maximal degree of nondeterminacy. In this case, the maximal degree of nondeterminacy that the evaluation of the auxiliary verb introduces is very low as the logical form of the auxiliary verb is considered fully instantiated. Now we mark the paths of the defining lexical entries whose instantiation can be deduced from the type hierarchy. To mimic the evaluation of the auxiliary verb, we determine the information common to all defining lexical entries by taking their generalization, i.e., the most specific feature structure subsuming all, and unify the result with the original right-hand side category in the phrase structure rule. Because both the generalization and the unification operations preserve binding annotations, this leads (via structure-sharing) to the annotation that the logical form of the verbal complement can be considered instantiated. Note that the nonverbal complements do not become further instantiated. By subsequent investigation of the maximal degree of nondeterminacy introduced by the evaluation of the complements in various permutations, we find that the logical form of a sentence only restricts the evaluation of the nonverbal complements after the evaluation of the verbal complement. This can be verified on the basis of a sample lexical entry for a main verb.

$$\begin{bmatrix} \text{phon} & \text{lieben} \\ \text{cat} & \text{v} \\ \text{fin} & - \\ \text{aux} & - \\ \text{subcat} & \left\langle \begin{bmatrix} \text{cont} & \boxed{6} \end{bmatrix}, \begin{bmatrix} \text{cont} & \boxed{7} \end{bmatrix} \right\rangle \\ \text{cont} \mid \text{nucleus} & \begin{bmatrix} \text{lover} & \boxed{6} \\ \text{loved} & \boxed{7} \end{bmatrix} \end{bmatrix}$$

The relative efficiency of this evaluation leads our compiler to choose

$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{subcat} & \langle\rangle \\ \text{cont} & \boxed{5} \end{bmatrix} \longrightarrow$$
$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{fin} & + \\ \text{aux} & + \\ \text{subcat} & \left\langle \boxed{3} \mid \boxed{4} \right\rangle \\ \text{cont} & \boxed{5} \end{bmatrix}, \boxed{3}\begin{bmatrix} \text{cat} & \text{v} \\ \text{fin} & - \\ \text{aux} & - \\ \text{subcat} & \boxed{4}\left\langle \boxed{1} \mid \boxed{2} \right\rangle \end{bmatrix}, \boxed{1}, \boxed{2}$$

as the optimal evaluation order of our phrase structure rule for argument composition.

3.2 Processing Head

The optimal evaluation order for a phrase structure rule need not necessarily be head-first. Our dataflow analysis treats heads and complements alike, and includes the head in the calculation of the optimal evaluation order of a rule. If the evaluation of the head of a rule introduces much nondeterminacy or provides insufficient restricting information for the evaluation of its complements, our dataflow analysis might not select the head as the first category to be evaluated, and choose instead

$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{subcat} & \langle\rangle \\ \text{cont} & \boxed{5} \end{bmatrix} \longrightarrow$$
$$\boxed{3}\begin{bmatrix} \text{cat} & \text{v} \\ \text{fin} & - \\ \text{aux} & - \\ \text{subcat} & \boxed{4}\left\langle \boxed{1} \mid \boxed{2} \right\rangle \end{bmatrix}, \begin{bmatrix} \text{cat} & \text{v} \\ \text{fin} & + \\ \text{aux} & + \\ \text{subcat} & \left\langle \boxed{3} \mid \boxed{4} \right\rangle \\ \text{cont} & \boxed{5} \end{bmatrix}, \boxed{1}, \boxed{2}$$

as the optimal evaluation order. This clearly demonstrates an extremely important consequence of using our dataflow analysis to compile a declarative grammar into a grammar optimized for generation. Empty or displaced heads pose us no problem, since the optimal evaluation order of the right-hand side of a rule is determined regardless of the head. Our dataflow analysis ignores the grammatical head, but identifies instead the 'processing head', and (no less importantly) the 'first processing complement', the 'second processing complement', and so on.

4 Constraints on Grammar

Our Earley generator and the described compiler for off-line grammar optimization have been extensively tested with a large HPSG grammar. This test-grammar is based on the implementation of an analysis of partial VP topicalization in German (Hinrichs et al., 1994) in the Troll system (Gerdemann and King, 1994). Testing the developed techniques uncovered important constraints on the form of the phrase structure rules in a grammar imposed by the compiler.

4.1 Complement Displacement

The compiler is not able to find an evaluation order such that the Earley generator has sufficient restricting information to generate all subparts of the construction efficiently in particular cases of complement displacement. More specifically, this problem arises when a complement receives essential restricting information from the head of the construction from which it has been extracted, while, at the same time, it provides essential restricting information for the complements that stayed behind. Such a case is represented schematically in figure 1 (see next page).

The first processing complement (C1) of the head (H) has been displaced. This is problematic in case C1 provides essential bindings for the successful evaluation of the complement C2. C1 can not be evaluated prior to the head and once H is evaluated it is no longer possible to evaluate C1 prior to C2. An example of problematic complement displacement taken from our test-grammar is given in figure 2 (see next page). The topicalized partial VP "Anna lieben" receives its restricting semantic information from the auxiliary verb and upon its evaluation provides essential bindings not only for the direct object, but also for the subject that stayed behind in the Mittelfeld together with the auxiliary verb. These mutual dependencies between the subconstituents of two different local trees lead either to the unrestricted generation of the partial VP, or to the unrestricted generation of the subject in the Mittelfeld. We handled this problem by partial execution (Pereira and Shieber, 1987) of the filler-head rule. This allows the evaluation of the filler right after the evaluation of the auxiliary verb, but prior to the subject. A head-driven generator has to rely on a similar solution, as it will not be able to find a successful ordering for the local trees either, simply because it does not exist.

4.2 Generalization

A potential problem for our approach constitutes the requirement that the phrase structure rules in the grammar need to have a particular degree of specificity for the generalization operation to be used successfully to mimic its evaluation. This is best illustrated on the basis of the following, more 'schematic', phrase structure rule:

$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{subcat} & \langle\rangle \\ \text{cont} & \boxed{4} \end{bmatrix} \longrightarrow \begin{bmatrix} \text{cat} & \text{v} \\ \text{fin} & + \\ \text{subcat} & \left\langle \boxed{3},\boxed{1},\boxed{2} \right\rangle \\ \text{cont} & \boxed{4} \end{bmatrix}, \boxed{1}, \boxed{2}, \boxed{3}$$

Underspecification of the head of the rule allows it to unify with both finite auxiliaries and finite ditransitive main verbs. In combination with the underspecification of the complements, this allows the rule not only to be used for argument composition constructions, as discussed above, but also for constructions in which a finite main verb becomes saturated. This means that the logical form of the nonverbal complements ($\boxed{1}$ and $\boxed{2}$) becomes available either upon the evaluation of the complement tagged $\boxed{3}$ (in case of argument composition), or upon the evaluation of the finite verb (in case the head of the rule is a ditransitive main verb). As a result, the use of generalization does not suffice to mimic the evaluation of the respective right-hand side categories. Because both verbal categories have defining lexical entries which do not instantiate the logical form of the nonverbal arguments, the dataflow analysis leads to the conclusion that the logical form of the nonverbal complements never becomes instantiated. This causes the rejection of all possible evaluation orders for this rule, as the evaluation of an unrestricted nonverbal complement clearly exceeds the allowed maximal degree of nondeterminacy of the grammar. We are therefore forced to split this schematic phrase structure rule into two more specific rules at least during the optimization process. It is important to note that this is a consequence of a general limitation of dataflow analysis (see also Mellish, 1981).

5 Concluding Remarks

An innovative approach to HPSG processing is described that uses an off-line compiler to automatically prime a declarative grammar for generation or parsing, and inputs the primed grammar to an advanced Earley processor. Our off-line compiler extends the techniques developed in the context of the DIA in that it compiles typed feature structure grammars, rather than simple logic grammars. The approach allows efficient bidirectional processing with similar generation and parsing times. It is shown that combining off-line techniques with an advanced Earley-style generator provides an elegant solution to the general problem that empty or displaced heads pose for conventional head-driven generation.

The developed off-line compilation techniques make crucial use of the fundamental properties of the HPSG formalism. The monostratal, uniform treatment of syntax, semantics and phonology supports

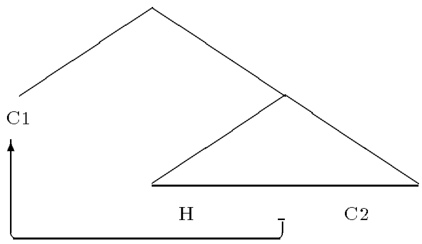
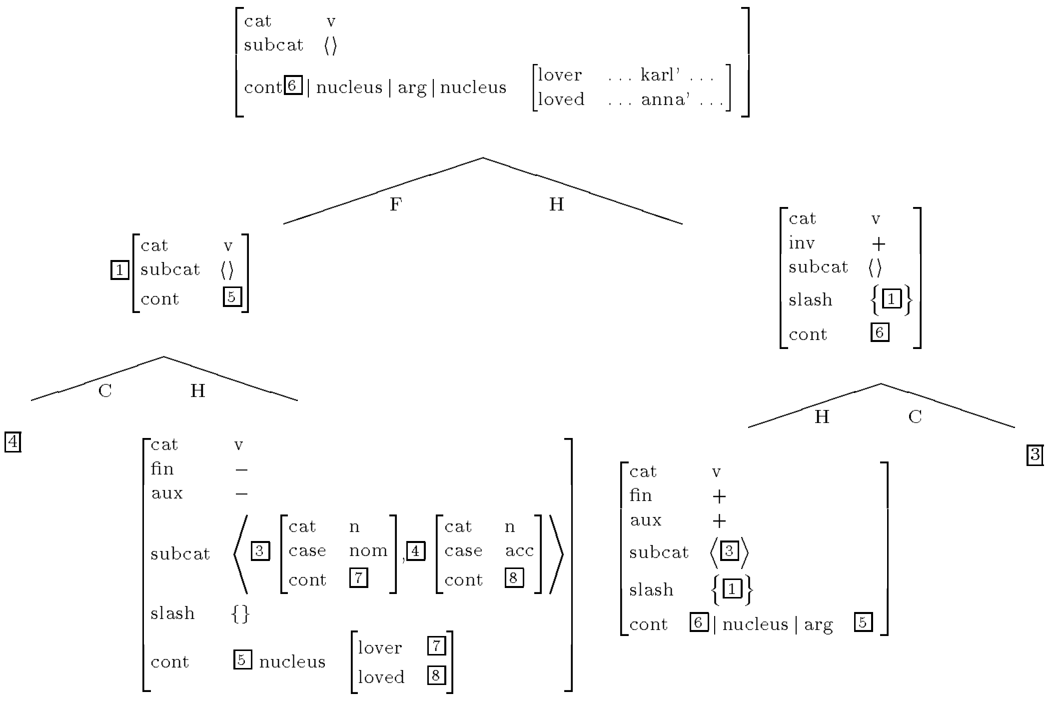


Figure 1: Complement displacement.



Anna	lieben	wird	Karl.
Anna	love	will	Karl.

"Karl will love Anna"

Figure 2: Example of problematic complement displacement.

dataflow analysis, which is used extensively to provide the information upon which off-line compilation is based. Our compiler uses the type hierarchy to determine paths with a value of a minimal type without appropriate features as bound. However, the equivalent of this kind of minimal types in untyped feature structure grammars are constants which can be used in a similar fashion for off-line optimization.

References

Appelt, Douglas. 1987. Bidirectional Grammars and the Design of Natural Language Generation Systems. In *Proceedings of* TINLAP-*3*, Las Cruces, New Mexico, USA.

Dymetman, Marc; Pierre Isabelle and François Perrault. 1990. A Symmetrical Approach to Parsing and Generation. In *Proceedings of* COLING-*90*, Helsinki, Finland.

Gerdemann, Dale. 1991. *Parsing and Generation of Unification Grammars.* Doctoral dissertation. University of Illinois. Published as Beckman Institute Cognitive Science technical report, number CS-91-06. Urbana-Champaign, Illinois, USA.

Gerdemann, Dale and Erhard Hinrichs. in press. Some Open Problems in Head-driven Generation. In *Linguistics and Computation.* CSLI Lecture Notes. Stanford, California, USA.

Gerdemann, Dale and Paul King. 1994. The Correct and Efficient Implementation of Appropriateness Specifications for Typed Feature Structures. In *Proceedings of* COLING-*94*, Kyoto, Japan.

Hinrichs, Erhard and Tsuneko Nakazawa. 1989. Subcategorization and VP Structure in German. Paper presented to the '3rd Symposium on Germanic Linguistics' at Purdue University. Published as SFB 340 technical report. Tübingen, Germany.

Hinrichs, Erhard; Detmar Meurers and Tsuneko Nakazawa. 1994. *Partial* VP *and Split* NP *Topicalization in German: An* HPSG *Analysis and its Implementation.* SFB 340 technical report nr. 58. Tübingen, Germany.

König, Esther. 1994. Syntactic-Head-Driven Generation. In *Proceedings of* COLING-*94*. Kyoto, Japan

Martinović, Miroslav and Tomek Strzalkowski. 1992. Comparing Two Grammar-based Generation Algorithms: A Case Study. In *Proceedings of* ACL-*92*, Newark, Delaware, USA.

Mellish, Chris. 1981. The Automatic Generation of Mode Declarations for Prolog Programs. In *Proceedings of the Workshop on Logic Programming and Intelligent Systems*, Los Angeles, California, USA.

Minnen, Guido; Dale Gerdemann and Erhard Hinrichs. 1995. Direct Automated Inversion of Logic Grammars. In *New Generation Computing*, volume 13, number 2.

Pereira, Fernando and Stuart Shieber. 1987. *Prolog and Natural Language Analysis.* CSLI Lecture Notes, number 10. Stanford, California, USA.

Pollard, Carl and Ivan Sag. 1994. *Head-driven Phrase Structure Grammar.* University of Chicago Press, Chicago, Illinois, USA.

Shieber, Stuart. 1988. A Uniform Architecture for Parsing and Generation. In *Proceedings of* COLING-*88*. Budapest, Hungary.

Shieber, Stuart; Gertjan van Noord; Fernando Pereira and Robert Moore. 1990. Semantic Head-Driven Generation. In *Computational Linguistics*, volume 16, number 1.

Strzalkowski, Tomek (editor). 1993a. *Reversible Grammar in Natural Language Processing.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

Strzalkowski, Tomek. 1993b. A General Computational Method for Grammar Inversion. In Strzalkowski 1993a.

Tamaki, Hisao and Taisuke Sato 1986. OLD Resolution with Tabulation. In *Proceedings of the Third International Conference on Logic Programming.* Berlin, Germany