An HPSG-Based Generator for German
An Experiment in the Reusability of Linguistic Resources

Johannes Matiasek and Harald Trost
Austrian Research Institute
for Artificial Intelligence*
Schottengasse 3,
A-1010 Vienna, Austria
Email: {john,harald}@ai.univie.ac.at

Abstract

We describe the development of a generator for German built by reusing and adapting existing linguistic data and software. Reusability is crucial for the successful application of NLP techniques to real-life problems since it helps to cut down on both development and adaptation effort. However, combining resources not designed to work together is not trivial. We describe the problems arising when integrating three preexisting resources (FUF, a unification-based generator, an HPSG Grammar for German, and X2MorF, a two-level morphology component) and the adaptations necessary to come up with a wide coverage tactical generator for German.

1 Introduction

A main obstacle for the successful application of NLP is the necessary effort in terms of development and adaptation time. One possible answer to this problem is the use of generic and modular software. An example for a software system developed with this goal in mind is the FUF generator (Elhadad, 1991), a well-documented public domain software written in LISP. Still, it is no straightforward task to employ that kind of software for new applications. Another important step is the declarative definition of linguistic data (grammar and lexicon) which also facilitates reuse in another setting. The reuse of existing resources does not only save efforts but, to a hopefully much minor extent, also creates new tasks to be solved, i.e. the integration of resources not having been designed to work together. How this can be done in an organized way is the topic of this paper.

The work being described here was done in the context of a multilingual text generation system. One of the objectives of the project is to reuse existing resources for those subtasks for which appropriate resources exist. For the German tactical generator[1] an implementation of an HPSG[2] style grammar of German (used for parsing and generation, but on a different software platform) was available inhouse. The FUF generator was chosen as the core component of the system.

However, two problems had to be solved before FUF could be used for the planned purpose. One was the fact that FUF, developed for English, has no suitable morphological component for the rich inflection of German. X2MorF (Trost, 1991), an available morphological component had to be integrated with the FUF generator for this purpose. The other problem was that the existing HPSG-inspired grammar of German could not be directly ported to the FUF formalism.

Before we describe the integration task we will briefly sketch the main characteristics of these resources, emphasizing those aspects which either cause problems for integration or provide the means for performing the integration task.

2 Available Resources

2.1 The FUF Generator

FUF (Elhadad, 1991) is a surface generator for natural language based on the theory of functional unification grammar (Kay, 1979). It employs both phrase structure rules and unification of feature descriptions. Input to FUF is a partially specified feature description which constrains the utterance to be generated. Output is a fully specified feature description subsumed by the input structure, which is then linearized to yield a sentence.

*The work reported here has been carried out within the LRE Project *GIST* (LRE 062-09) and funded by the Austrian *Forschungsförderungsfonds der Gewerblichen Wirtschaft*, Grant 2/329. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian *Bundesministerium für Wissenschaft, Forschung und Kunst*.

[1]The task of a tactical generator is to produce sentential or subsentential phrases corresponding to a semantic input specification and does not include text planning.

[2]Head Driven Phrase Structure Grammar (Pollard and Sag, 1987; Pollard and Sag, 1994)

2.1.1 Grammar Specification in FUF

Grammar and lexicon are specified as one large feature description, containing at least one disjunction (given by the `alt` keyword) ranging over the phrasal and lexical categories of the grammar. The feature `cat` is used to indicate these categories. The feature `lex` associates strings with lexical categories. The trivial grammar of Fig. 1 exemplifies the layout of a FUF grammar.

```
(alt (; --- S (with subject/verb agreement)
      ((cat   s)
       (subj ((cat np)))
       (pred ((cat vp)
              (agr {^ ^ subj agr}))))
      ; --- NP (only proper nouns)
      ((cat np)
       (n  ((cat noun) (proper y))))
      ; --- VP (only intransitive verbs)
      ((cat vp)
       (v   ((cat verb)
             (agr {^ ^ agr}))))
      ; --- Lexicon
      ((cat verb)(lex "laughs")
                 (agr ((pers 3rd)(num sg))))
      ((cat noun)(lex "Mary")
                 (agr ((pers 3rd)(num sg)))
                 (proper y))))
```

Figure 1: A trivial FUF grammar

Pointers are used to enforce structure sharing and provide a means to percolate information within a feature structure.

FUF provides the means to specify a subsumption ordering of *types*, which is useful to express generalizations, and a macro mechanism.

2.1.2 Operational Characteristics

Generation starts from an underspecified input feature structure. FUF unifies the grammar *into* the input structure, i.e. enriches and further constrains it. Alternatives are explored sequentially until one branch succeeds. Thus the input structure never contains disjunctions.

When unification at the current level is complete, i.e. nothing further can be added to the input structure, every substructure of the input representing a category is recursively unified with the grammar. This process is repeated breadth first until all constituents are leaves.

To determine which substructures have to be processed recursively FUF employs two methods. The default strategy collects all substructures of the current level having a `cat` feature. Explicit specification of subconstituents is also possible via the special feature `cset` (constituent set). If `cset` is present, FUF performs recursion on these explicitly given substructures only. E.g., the default strategy operates on category `s` in Fig. 1 as if `(cset (subj pred))` had been specified. When specifying `(cset (pred))` only, no recursion would be performed on `subj`.

2.1.3 Linearization

The recursive unification process handles only the dominance relations of the grammar. In order to account for linear ordering of the resulting tree shaped feature structure, FUF performs a linearization process after unification has finished. Linear precedence of constituents is specified in the grammar using the special feature `pattern`. Only constituents mentioned in a pattern are realized during linearization. Thus, the simple grammar in Fig. 1 has to be enriched: `(pattern (subj pred))` has to be added at `(cat s)`, `(pattern (n))` has to be added at `(cat np)` and `(pattern (v))` is needed at `(cat vp)`. Lexical categories don't need a pattern feature.

Patterns need not specify an absolute ordering. E.g., `(...a ...b ...)` specifies that constituent `a` has to precede `b`. More such partial patterns may be specified, pattern unification leads to all legal constituent combinations.

Linearization traverses the tree, extracts the strings found in the `lex` feature of the leaves, and flattens this structure according to the `pattern` directives found.

2.2 The HPSG Grammar for German

In HPSG (Pollard and Sag, 1987; Pollard and Sag, 1994), the fundamental objects of linguistic analysis are signs modeled by typed feature structures and constrained by global principles. HPSG does not employ phrase structure rules. Instead, very general dominance schemata are given. Which arguments a lexical head takes is lexically specified in its SUBCAT list. Also adjunction is specified lexically; the adjunct is seen as the semantic head which selects the kind of signs it modifies, the modified sign remains the syntactic head of the resulting phrase. Long distance dependencies are handled in HPSG not in terms of movement but via structure sharing of the values of a SLASH feature percolating the "moving" constituent.

The grammar for German follows the version of HPSG given in (Pollard and Sag, 1994) rather strictly, deviating only in the following aspects:

- The Subcategorization Principle is given in a binary branching fashion.
- The argument structure of lexical heads is enriched. Thus generalizations concerning case assignment and argument reduction phenomena can be captured in a principled fashion (see Heinz and Matiasek (1994)).
- Verb second position is handled by a mechanism resembling the notion of head movement of GB-theory.

2.3 X2MorF

X2MorF (Trost, 1991) is a morphological component based on two level morphology (Koskenniemi, 1983). In two-level morphology morphophonology is treated by means of rules that

$$\begin{bmatrix} \text{PHON} & \textit{"geht"} \\ \text{SYNSEM|LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & {}_{verb}\begin{bmatrix}\text{VFORM} & \textit{finit}\\ \text{TENSE} & \textit{present}\\ \text{PERS} & \textit{3rd}\\ \text{NUM} & \textit{sg}\end{bmatrix} \\ \text{SUBCAT} & \left\langle \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & {}_{noun}\begin{bmatrix}\text{CASE} & \textit{nom}\\ \text{PERS} & \textit{3rd}\\ \text{NUM} & \textit{sg}\end{bmatrix} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONT} & {}_{nom\_obj}\begin{bmatrix}\text{INDEX} & \boxed{1}\end{bmatrix} \end{bmatrix}\end{bmatrix} \right\rangle \end{bmatrix} \\ \text{CONT} & {}_{psoa}\begin{bmatrix}\text{REL} & \textit{walk}\\ \text{ACTOR} & \boxed{1}\end{bmatrix} \end{bmatrix}\end{bmatrix}$$

```
((cat     verb)
 (lex     "geht")
 (head    ((vform finit)
           (tense present)
           (pers  3rd)
           (num   sg)))
 (concept walk)
 (args
   ((actor
      ((cat  np)
       (head ((case nom)
              (pers 3rd)
              (num  sg))))))))
```

Figure 2: Lexical Entry for *"geht"* in HPSG and in FUF

map between the lexical representation of a word and its surface form. Morphology proper on the other hand is viewed as a simple concatenation process governed by a regular grammar.

X2MorF augments standard two-level morphology in two ways. First, it replaces the continuation class mechanism with a feature-based word grammar and lexicon. This is an important prerequisite for its use in a feature-based sentence-level processing system (see Trost and Matiasek (1994)). Second, it allows for interaction between two-level rules and word grammar facilitating the formulation of rules for non-concatenative morphotactics like umlaut.

3 The Integration Task

Although the main components to be integrated fulfill reusability requirements (FUF being a fairly general and modular generation engine, the HPSG grammar being a declaratively written resource), integration of these resources into a unified system could only be achieved after suitable adaptation. The morphological component of FUF is very restricted. Thus it needed to be replaced by X2MorF. The available German word level grammar of X2MorF was rewritten to conform to the feature structure notation employed by FUF. The two-level part stayed unchanged. More substantial changes were required to adapt the HPSG grammar. Not only syntactic adaptations to another feature formalism were needed, but also the operational characteristics of FUF had to be accounted for. Also some of the phrase structure information generalized in the form of principles could be "compiled" into phrase structure rules.

3.1 HPSG in FUF

First experiments to implement HPSG in FUF rather directly showed inefficient runtime behavior. Since most grammatical constraints in HPSG are expressed via structure sharing, and FUF uses pointers to indicate coreferences, most of the processing time was spent in following pointer chains through deeply nested feature structures. Thus the structures have considerably been flattened and some aspects (most notably SUBCAT and CONTent) have been encoded differently.

3.1.1 The Representation of Signs

The process of recasting the original HPSG structures in the FUF formalism can best be described by examples. In Fig. 2 the HPSG representation of the German verb *geht* (walks) and its representation in FUF is shown, exemplifying the following mappings of HPSG onto FUF:

- The subtyping of the HEAD is represented by the `cat` feature of FUF.
- SYNSEM|LOC|CAT|HEAD is mapped to `head`.
- SYNSEM|LOC|CONT|REL is mapped to `concept`.
- Instead of subcategorizing for *synsem* values as proposed in Pollard and Sag (1994) the convention of Pollard and Sag (1987) to subcategorize for signs is adopted.
- Instead of a list-valued SUBCAT feature the feature `args` is used. The correspondence between (syntactic) arguments and semantic roles is established by placing the constituent under a feature corresponding to its semantic role. Thus list manipulation is avoided and the structure corresponds more closely to the input specification (given in a language based on SPL (Kasper, 1989)).
- The NONLOCal feature is dropped. Slash extraction is handled differently.

It should be noted that this entry does not correspond exactly to the actual representation in the generator, it serves simply to illustrate the basic ideas underlying the transformation. The actual implementation additionally allows for

- the specification of arguments via `external` macros, accounting for a more principled treatment of case assignment, argument reduction and slash extraction;
- a differentiation between lexemes and stems to account for a treatment of inflection by the morphology component.

The representation of phrasal signs in HPSG parallels the one of lexical signs; an additional feature DTRS carries the subconstituents of the phrase. One of the daughters is the head of the phrase (HEAD-DTR), its head features are identical to the head features of the phrase (Head Feature Principle). The other daughter may be either a complement, an adjunct, a marker or a filler (realizing the slash feature of the head daughter). Each constituent structure is constrained by an associated set of dominance schemata and principles.

HPSG distinguishes between *substantive* categories (such as nouns or verbs) and *functional* categories (e.g., determiners). Since functional categories correspond to closed word classes, in the FUF implementation these categories are compiled into phrase structure rules.

The same approach, i.e. factoring subcategorization information into phrase structure rules, is taken with auxiliary and modal verbs and with phenomena which may well be regarded as the manifestation of a functional category, but which are not expressed by lexical items but by special constituent ordering (e.g., verb second position in declarative main clauses).

The treatment of adjunction in the FUF implementation reflects the representation of modifiers in the input language. The HPSG view of an adjunct as the semantic head selecting the sign it modifies, is changed to the view that adjuncts act as "optional" arguments of the syntactic head.

3.1.2 Encoding of Principles

Many constraints expressed in HPSG by means of principles (e.g., dominance schemata) are already built into the phrase structure rules compiled out of the original grammar. There remain, however, the most central HPSG principles constraining all phrases and ensuring the proper information sharing between mother and head daughter. These are inserted into the grammar at the level `(cat phrasal-category)`. The branches dispatching to particular phrase types are specified later in an embedded disjunction.

```
(defparameter *phrasal-principles*
  '(;;; HEAD FEATURE PRINCIPLE
    (head    {^ head-dtr head})
    ;;; SEMANTICS PRINCIPLE:
    (concept {^ head-dtr concept})
    (args    {^ head-dtr args})
    (index   {^ head-dtr index})
    ;;; SLASH INHERITANCE PRINICIPLE:
    (slash   {^ head-dtr slash})))
```

Figure 3: HPSG Principles in FUF

However, one important principle of HPSG, the Subcategorization Principle ensuring the proper relationship between the arguments subcategorized for and the constituent structure of the phrase still needs to be accounted for. How this constraint is met will be discussed next.

3.1.3 Control Strategy

FUF employs a top-down processing scheme driven by the syntactic category of the mother. This control strategy is inadequate when the constituent structure is specified lexically by the head and thus unknown until the head is expanded. HPSG lends itself best to head-driven, bottom-up processing, at least for generation. Since the control regime of FUF cannot be changed in principle (only delay methods are available), the grammar itself has to account for adequate processing characteristics. This means, that the lexicon driven approach has to be emulated within the grammar, based on the operational behavior of FUF.

The basic idea for realizing head driven processing behavior is to use the `cset` and `pattern` special attributes of FUF in an asymmetrical fashion. Generation of a phrase starts by realizing its `head-dtr`. Therefore only the head daughter is specified in the constituent set of the phrase. Once the lexical head of the phrase is generated, its argument list is activated using the default recursion strategy of FUF (since no `cset` attribute is present). The lexically specified arguments are now generated in a (virtually) bottom up fashion. Structure sharing percolates the `args` upwards to the phrasal level, where they are then realized via the pattern feature. The basic mechanism of en-

```
((cat      phrase)
 (head-dtr ((cat lex-cat) ... ))
 ;; percolate arguments
 (args     {^ head-dtr args})
 ;; recursion only on head daughter
 (cset     (head-dtr))
 ;; realize head and arguments
 (pattern  (args head-dtr)))
```

Figure 4: Head driven generation in FUF

coding this processing strategy in the grammar is given in Fig. 4. If functional categories are present in a phrase, then the appropriate slots have to be specified and added to `cset` and `pattern`.

Thus the shape of the resulting phrase largely depends on the kind of arguments its lexical head admits. In order to realize its arguments, every word able to act as the head of a phrase has to provide a syntactic and semantic specification of its arguments. This specification also has to account for long distance phenomena, i.e. extraction of an argument (e.g., wh-movement). Furthermore, variations of case assignment (e.g., in passivization) have to be accounted for.

3.1.4 Argument Structure Encoding

Although a large amount of information has to be stored in the lexicon, a compact and easily maintainable structure of the lexicon is a crucial requirement. Therefore extensive use has been made of FUFs `external` macros.

Fig. 5 shows the actual encoding of the lexical entry for *"warten"* ("wait"), subcategorizing for an actor and a patient. Syntactic restrictions on the argument are given by macros. `pp-auf-acc` expands to a PP with preposition *auf* and accusative case, the realization of the structural argument `np-ext-da` depends on whether argu-

```
((cat     lex-verb)
 (lxm     "wart")
 (concept wait)
 (args   ((actor   #(external np-ext-da))
          (patient #(external pp-auf-acc)))))
```

Figure 5: Lexical Entry for *"warten"* in FUF

ment reduction (i.e. passivization) has to be performed or not (for a theoretical background see Heinz and Matiasek (1994)). In active contexts it becomes the subject and receives nominative case, in passive contexts it may be optionally realized as a PP_{von} (see Fig.6).

```
((alt (((({^ ^ reduction} no)
        (cat np)
        ;; promote to subject
        ({^ actor} {^ subj}))
        ;; passivization
       (({^ ^ reduction} yes)
        (alt (;; optional pp(von)
              ((concept GIVEN)
               (cat pp)(adpos ((lxm "von"))))
              ((concept NONE)
               (cat     NONE))))))))
```

Figure 6: Expansion of `#(external np-ext-da)`

A mechanism common to all arguments and thus incorporated into every macro expanding to an argument specification is the extraction mechanism required to handle movement (see Fig. 7). At the phrasal level the argument which has to

```
((alt (;; try to fill slash by unification
       (({^ <slot>}  {^ slash}))
       ;; does not unify --> add pattern
       (({^ pattern} (... <slot> ...))))))
```

Figure 7: Slash extraction (slightly simplified)

be extracted (e.g., in wh-questions the constituent asked for) has to be specified as the `slash` feature of the `args`. Each argument must be checked during generation if it is unifiable with the `slash` specification, and, if so, it has to be made coreferential with `slash`. Otherwise, an appropriate `pattern` feature has to be produced to ensure the realization of the argument at the `args` level.

3.1.5 V2 and a Generation Example

German is commonly regarded as an SOV language. However, the standard word order – a sentence final verbal complex with the finite verb as the last element – is encountered only in subordinate clauses. In declarative sentences and wh-questions the finite element of the verbal complex occupies the second position in the sentence. Sentence initial position of the finite verb is encountered in imperative clauses and yes-no questions.

In our grammar, the verbal complex is always generated in the standard order. To account for V1 and V2 phenomena, a mechanism resembling the GB notion of head movement is imple-

```
((cat s)
 (s-type  declarative)
 (head-dtr((cat        vk)
           (head       ((vform fin)))
           (head-slash ((cat lex-verb)))))
 (v2      {^ head-dtr head-slash})
 (subj    ((head ((case nom)
                  (num  {^ ^ ^ head num})
                  (pers {^ ^ ^ head pers})))))
 (args    ((subj {^ subj})))
 ;; force extraction of one constituent
 (alt     (((focus  GIVEN)
            (focus  {^ args slash}))
           ((focus  {^ subj})
            (subj   {^ args slash}))))
 (cset    (head-dtr))
 (pattern (focus v2 args head-dtr)))
```

Figure 8: Declarative Main Clause in FUF

mented. This mechanism functions analogously to the slash mechanism presented above. If a feature `head-slash` is passed to the verbal complex, the finite verb is extracted, allowing the governing phrase to realize it in first or second position. The morphology component ensures that separable prefixes are left in place.

The verbal complex is generated top down. The arguments of the main verb are generated lexicon driven, once the lexical head of the phrase has been established.

Subject-verb agreement and nominative case assignment is handled via the `subj` slot which is coreferential with `args:subj` and – after argument generation – contains the subject of the sentence (cf. Fig. 6). Verb second position can only be ensured, if the constituent in sentence initial position is nonempty. The slot `focus` is designed to hold that constituent. The constituent to be topicalized or, if not specified in the input, the `subj` is extracted via the slash mechanism (cf. Fig. 7). The interaction between top down category driven and "bottom up" lexicon driven processing is illustrated in Fig. 9, showing also the effects of the two slash extraction mechanisms.

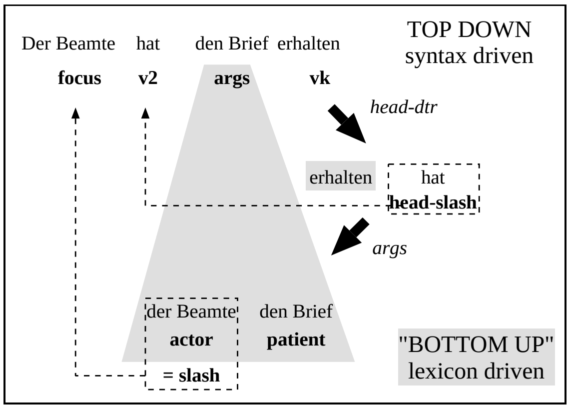

Figure 9: Generating a Declarative Main Clause

3.2 X2MorF in FUF

For the integration of X2MorF into FUF the unification engine used in X2MorF was replaced by FUF itself, and the existing word grammar and morph lexicon were reformulated in the FUF formalism, and the word form generation task is now performed by FUF itself. The two-level rules could be taken over in their original form, only the morphological filters had to be translated.

A simple functor/argument scheme is sufficient for the word grammar. The possible combinations are given by the phrase structure rules of the morph grammar. The affixes (functors) may further restrict the arguments they may be applied to. Fig. 10 shows an example of morphological categories responsible for nominal inflection. A noun stem has to be followed by a case suffix which determines case and number of the resulting noun form. The head features of the argument are made available to the `functor` via the `arghead` feature, thus enabling the functor to subcategorize for its argument (e.g., by requiring a certain inflection paradigm). One of the possible case suffixes is a null morph inducing plural in a certain class of nouns with `(noun-paradigm null)`. It applies in all cases except dative[3] setting the `umlaut` feature, which triggers the two level rule forcing umlaut. An example is *"Garten"* with plural *"Gärten"*.

```
((cat noun-form)
 (functor ((cat     case-suffix)
           (head    {^ ^ head})
           (arghead {^ ^ arg head})))
 (arg     ((cat     noun-stem)
           (stem    {^ ^ stem})))
 (cset    (arg functor))
 (pattern (arg functor)))
```

```
((cat case-suffix)
 ((lex "")
  (head    ((umlaut aou-umlaut)
            (case   not-dat)
            (num    pl)))
  (arghead ((noun-paradigm null)))))
```

Figure 10: Nominal Inflection

The interface between syntactic and word level processing is provided by the lemma lexicon. It contains the argument structure of the lexemes and links them to (possibly prefixed) stems. The required syntactic features of a particular word form are determined by the sentence level syntactic generation. The lemma lexicon passes these features to the morphological level and the word level grammar takes care of selecting the appropriate affixes. During the final linearization the extended two level rules map the concatenated stems and affixes to the appropriate surface strings.

[3]The boolean combinations of certain features have been spelled out in the type hierarchy.

4 Conclusion

We have shown how existing resources can be adapted to new applications thereby saving considerably on development efforts. We have demonstrated integration tasks on two different levels:

- Integration of software systems: by combining FUF with X2MorF we have extended the functionality of FUF. While the original morphology component of FUF is geared towards English only, X2MorF can be used with a wide range of languages.
- Adaptation of linguistic resources to processing requirements: by adapting our existing HPSG grammar for German to FUF we have shown that a declaratively written linguistic resource can be used in a new processing environment with modest effort.

This is an important step in bringing natural language processing techniques closer to real-world applications, where the minimizing of adaptation cost and the maximal use of existing resources is crucial for success.

References

Buchberger, E., E. Garner, W. Heinz, J. Matiasek, and B. Pfahringer. 1991. VIE-DU—Dialogue by Unification. In H. Kaindl, ed, *7. Österr. AI Tagung*, pp 42–51, Berlin. Springer.

Elhadad, M. 1991. FUF: The Universal Unifier User Manual, Version 5.0. Technical report, Dept.of Comp. Sc., Columbia University.

Heinz, W. and J. Matiasek. 1994. Argument Structure and Case Assignment in German. In J. Nerbonne, K. Netter, and C. Pollard, eds, *German in Head-Driven Phrase Structure Grammar*. CSLI Publications, Stanford, pp 199–236.

Kasper, R. T. 1989. A flexible interface for linking applications to Penman's sentence generator. In *Proceedings of the DARPA Speech and Natural Language Workshop*, Philadelphia.

Kay, Martin. 1979. Functional Grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society.* Berkeley Linguistics Society, Berkeley, CA.

Koskenniemi, K. 1983. Two-Level Model for Morphological Analysis. In *Proc. IJCAI-83*, Los Altos, CA. Morgan Kaufmann.

Pollard, C. and I. Sag. 1987. *Information-Based Syntax and Semantics, Vol. 1: Fundamentals.* CSLI Lecture Notes 13. CSLI, Stanford, CA.

Pollard, C. and I. Sag. 1994. *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago.

Trost, Harald. 1991. X2MORF: A Morphological Component Based on Augmented Two-Level Morphology. In *Proc. IJCAI-91*, Sydney.

Trost, Harald and Johannes Matiasek. 1994. Morphology with a Null-Interface. In *Proc. COLING-94*, Kyoto, Japan, August 5-9.