Implementing HPSG in FUF 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

Keywords: Natural Language Understanding, Text Generation

[*]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.*

Implementing HPSG in FUF An Experiment in the Reusability of Linguistic Resources

Keywords: Natural Language Understanding, Text Generation

Abstract

In practical systems it is often required to reuse existing resources. Such an approach clearly has advantages: it speeds up the development process considerably, if one doesn't have to start from scratch. However, combining resources not designed to work together is not a trivial task.

An HPSG grammar of German has been implemented in FUF, an unification-based text generator. Although FUF is largely theory-neutral, some of its characteristics diverge from the processing requirements imposed by HPSG in its strict sense. The most prominent discrepancy is, that HPSG, being a lexically driven formalism lends itself best to a head-driven bottom-up processing strategy, whereas FUF, at least by default, uses a top-down, category-driven approach. FUF also lacks a morphological component able to deal with the rich German inflectional system. Therefore a two-level morphology component, X2MorF, has been added.

We describe the problems arising when integrating these three resources and the transformations and adaptations made to them, leading to a wide coverage tactical generator for German.

1 Introduction

The reuse of existing resources is a crucial demand when building natural language processing systems. However, this does not only save efforts but, to a hopefully much minor extent, creates also new tasks to be solved, i.e. the integration of resources not having been designed to work together.

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) and a morphology module were available inhouse. A LISP-based generator was available as public domain software, namely the FUF package [2].

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.

[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 [9, 10]

2 Characteristics of the available resources

2.1 The FUF Generator

FUF [2] implements a surface generator for natural language. It is based on the theory of functional unification grammar [5] and employs both phrase structure rules (being encoded by means of a special CATegory feature) and unification of feature descriptions. Input to FUF is a partially specified feature description which constrains the utterance to be generated. Output of FUF is a fully specified feature description (in the sense of the particular grammar) subsumed by the input structure, which is then linearized to yield a sentence.

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. Another special feature, `lex`, associates strings with lexical categories. The trivial grammar of Fig. 1 exemplifies the layout of grammar specifications within the FUF formalism. Pointers can be used to enforce identity of substructures 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. A further feature of FUF is the possibility of using `external` macros in the grammar. These macros can be defined by the user and return on demand a piece of grammar to be used in that place. This feature is heavily used in the German grammar and will be explained in more detail below.

2.1.2 Operational Characteristics

Generation with FUF starts from an input feature structure constraining the utterance to be produced. FUF unifies the grammar *into* the input structure, i.e.

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

Figure 1: A trivial FUF grammar

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, recursion on the subconstituents is performed. Every substructure of the enriched input structure which represents a category is recursively unified with the grammar. This process is repeated in a breadth first fashion until all constituents are leaves, i.e. have no subconstituents.

To determine which substructures correspond to constituents needing recursion, FUF employs two methods. The default strategy collects all substructures of the current level bearing the category feature `cat`. The second strategy requires explicit specification of the subconstituents in the grammar by means of the special feature `cset` (constituent set). If such a feature is present, FUF does not resort to its default strategy but only performs recursion on these explicitly given substructures. To illustrate this behavior, 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`. Furthermore, the `cset` feature provides control of the order in which constituents are unified and thus may be used to enhance efficiency.

In order to efficiently process a grammar, various methods of pruning the search tree are provided in FUF. The most simple one is to use the `:index <path>` keyword within a disjunction, telling FUF to use the value found in the input structure at `<path>` as an index and to ignore all alternatives in the grammar whose values do not match. Thus indexing forces a more deterministic processing and prevents a considerable amount of computations bound to fail.

Indexing does not help very much if the value of the index in the input structure is unknown. To avoid blind search in such a situation, the decision which branch to take can be delayed using the `:wait <path>` declaration. Then unification proceeds ignoring the disjunction at hand and puts the delayed choice point onto an agenda. Processing of the delayed disjunctions continues if either the value waited for is present or if nothing else is left to do.

FUF also provides special values, which may induce a nonmonotonic behavior of the generator. These values are `NONE, GIVEN` and `ANY`. The interpretation of `GIVEN` and `ANY` depends on the current status of the input structure. Thus these values have to be used with procedural considerations in mind.

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 must be 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 in order to produce correct results: `(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. FUF also provides a morphology component (for English only) taking care of producing the correct word forms including capitalization and punctuation.

2.2 The HPSG Grammar for German

In HPSG [9, 10], the fundamental objects of linguistic analysis are signs modeled by typed feature structures and constrained by global principles (e.g. the Head Feature Principle). The basic attributes for signs include PHON for phonological information and SYNSEM for syntactic and semantic information. SYNSEM in turn is highly structured including LOCal and NONLOCal features. LOCal features comprise CONTent, containing semantic information and the CATegory complex, which includes the HEAD features and the SUBCAT list to model subcategorization information. NONLOC features are used to model nonlocal dependency constructions such as topicalization, questions and relative clauses.

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 [10] rather strictly, deviating from it 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 (occurring, e.g., in passivization) can be captured in a principled fashion (see [3]).
- Verb second position is handled by a mechanism resembling the notion of head movement of GB-theory.

The German grammar and the processing modules for parsing and generation have been implemented in a direction-independent manner using a CLP extension of SICStus Prolog (the implementation techniques used are described in [7, 8]).

2.3 X2MorF

X2MorF [11] is a morphological component based on two level morphology [6]. The basic idea behind two-level morphology is the treatment of morphophonology by means of rules that map between the lexical representation of a word and its surface form (as it appears in text). Rules can be applied in parallel (either directly or in the form of finite automata) leading to a very efficient implementation. Morphology proper on the other hand is viewed as a simple concatenation process governed by a regular grammar, implemented using a continuation class mechanism.

X2MorF augments standard two level morphology in two ways. First, it replaces the continuation class lexicon with a feature-based word grammar and lexicon. This is an important requisite for its use as a morphological component in a feature-based sentence-level processing system (see [12]). Second, it allows for interaction between the two-level rules and the word grammar which makes it easier to formulate rules for non-concatenative morphotactics like umlaut. Up to now the system has been used to describe German inflectional and derivational morphology and Italian and Turkish inflectional morphology.

3 The Integration Task

Integration of the existing resources into a unified system could only be achieved after suitably adapting each of these resources. The FUF system itself needed a replacement of its morphological component by the appropriate calls to the two-level rules of X2MorF and a revision of the linearizer. The word level grammar of X2MorF was rewritten to use FUF as unification engine.

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. In order to make the system as efficient as possible, the grammar had to respect the processing strategies of FUF. 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 in a rather direct way, preserving the deeply structured shape of feature structures indicated, that – although possible – this strategy led to inefficient runtime behavior. Since most grammatical constraints in HPSG are expressed by means of structure sharing, and FUF uses pointers to indicate such 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 the SUBCAT list and the 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 lexical entry of the German verb *geht* (walks) as it appeared in the original HPSG grammar is shown.

$$
{}_{word}\left[\begin{array}{ll}
\text{PHON} & \textit{"geht"} \\
\text{SYNSEM} & {}_{synsem}\left[\begin{array}{ll}
\text{LOC} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & {}_{verb}\left[\begin{array}{ll}
\text{VFORM} & \textit{finit} \\
\text{TENSE} & \textit{present} \\
\text{PERSON} & \textit{third} \\
\text{NUM} & \textit{sg}
\end{array}\right] \\
\text{SUBCAT} & \left\langle {}_{synsem}\left[\begin{array}{ll}
\text{LOC} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & {}_{noun}\left[\begin{array}{ll}
\text{CASE} & \textit{nom} \\
\text{PERSON} & \textit{third} \\
\text{NUM} & \textit{sg}
\end{array}\right] \\
\text{SUBCAT} & \langle\rangle
\end{array}\right] \\
\text{CONT} & {}_{nom\_obj}\left[\text{INDEX}\ \boxed{1}\right]
\end{array}\right]
\end{array}\right] \right\rangle
\end{array}\right] \\
\text{CONT} & {}_{psoa}\left[\begin{array}{ll}
\text{REL} & \textit{walk} \\
\text{ACTOR} & \boxed{1}
\end{array}\right]
\end{array}\right] \\
\text{NONLOC} & \textit{nonloc}
\end{array}\right]
\end{array}\right]
$$

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

The following decisions regarding the mapping onto FUF have been made:

- The subtyping of the HEAD is represented by the `cat` feature of FUF.
- SYNSEM|LOC|CAT|HEAD is mapped to a toplevel feature `head`, similarly the feature SYNSEM|LOC|CONT|REL is mapped to `concept`.
- Instead of subcategorizing only for *synsem* values as proposed in [10] the convention of [9] is adopted and the whole sign is subcategorized for.
- The constituents subcategorized for are placed under a feature `args` at the toplevel of the FUF structure and no longer constitute a list. 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 [4]).
- The NONLOCal feature is dropped. Slash extraction is handled differently.

The resulting FUF representation of the same verb is given in Fig. 3. It should be noted, however, that this entry does not correspond exactly to the way it is actually represented in the generator, it serves simply to illustrate the basic ideas underlying the transformation of HPSG to FUF. The main differences are

- the specification of arguments via `external` macros, accounting for a more principled treatment of case assignment, argument reduction and movement;
- a differentiation between lexemes and stems to account for a treatment of inflection by the morphology component (Fig. 3 corresponds to a full form lexicon).

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

Figure 3: Lexical Entry for *"geht"* in FUF

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 token-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 kind of 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 way in which modifiers are represented 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.,the dominance schemata) are already built into the phrase structure rules compiled out of the original grammar. There remain, however, the most central principles of HPSG constraining all phrases, which ensure the proper sharing of information between the mother and the head daughter. They are inserted into the grammar at the level `(cat phrasal-category)`. The branches dispatching to the particular phrase types are specified in an embedded disjunction afterwards.

However, one important principle of HPSG, the Subcategorization Principle

```
 (defparameter *phrasal-principles*
   '(;;; HEAD FEATURE PRINCIPLE
     (head    {^ head-dtr head})
     ;;; SEMANTICS PRINCIPLE: (syntactic head is always the semantic head)
     (concept {^ head-dtr concept})
     (args    {^ head-dtr args})
     (index   {^ head-dtr index})
     ;;; SLASH INHERITANCE PRINICIPLE: percolate slash of head-dtr
     (slash   {^ head-dtr slash})))
```

Figure 4: HPSG Principles in FUF

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 in the next section.

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, taking the operational behavior of FUF into account.

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, i.e. `(cset (head-dtr))`. 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 (e.g., `(pattern (args head-dtr))`). The basic mechanism of encoding this processing strategy in the grammar is given in Fig. 5. If functional categories are present in a phrase, then the appropriate slots

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

Figure 5: Head driven generation in FUF

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 possible 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 for a practical system. Therefore extensive use has been made of the `external` macros provided by FUF, which are expanded only on demand.

Fig. 6 shows the actual encoding of the lexical entry for *"warten"* ("wait"), subcategorizing for an actor and a patient. Syntactic restrictions on the argument

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

Figure 6: Lexical Entry for *"warten"* in FUF

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 argument reduction (i.e. passivization) has to be performed or not (for a theoretical background see [3]). 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.7).

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. 8). At the phrasal level the argument which has to be extracted (e.g., in wh-questions the constituent asked for) has to be

```
((alt ((({^ ^ reduction} no)
        (cat np)
        ({^ actor} {^ subj}))   ; promote to subject
       (({^ ^ reduction} yes)
        ;; optional role when passivized, realize only on demand
        (alt (((concept GIVEN)  (cat pp)(adpos ((lxm "von"))))
              ;; --- otherwise no cat to prevent realization
              ((concept NONE)   (cat NONE))))))))
```

Figure 7: Expansion of `#(external np-ext-da)`

```
  ((alt (;; try to fill slash by unifying it with argument
         (({^ <slot>}  {^ slash}))
         ;; argument does not unify --> just add pattern
         (({^ pattern} (... <slot> ...))))))
```

Figure 8: Slash extraction (slightly simplified)

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 Verb Second and a Sentence 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 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, i.e. with the finite verb in sentence-final position. To account for V1 and V2 phenomena, a mechanism resembling the GB notion of head movement is implemented. 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 not generated in place but instead 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

```
((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)  ; case assignment and agreement
                   (num    {^ ^ ^ head num})
                   (person {^ ^ ^ head person})))))
 (args     ((subj    {^ subj})))
 ;; force extraction of one constituent (defaulting to subject)
 (alt      (((focus  GIVEN)    (focus  {^ args slash}))
            ((focus  {^ subj}) (subj   {^ args slash}))))
 (cset     (head-dtr))
 (pattern  (focus v2 args head-dtr)))
```

Figure 9: FUF Grammar for Declarative Main Clause (slightly simplified)

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. 7). 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. If no constituent to be topicalized is specified in the input, it defaults to `subj`. This extraction uses the slash mechanism described above (see Fig. 8). The interaction between top down category driven and "bottom up" lexicon driven processing is illustrated in Fig. 10, showing also the effects of the two slash extraction mechanisms.

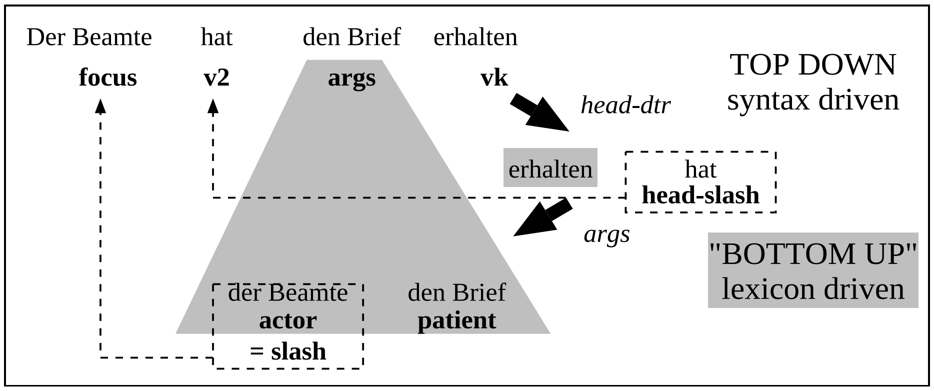

Figure 10: Generating a Declarative Main Clause

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 had to be reformulated in the FUF formalism. While X2MorF is used for both parsing and generation, only the generating task is relevant here, thus the whole word form generation task is now performed by the standard FUF procedure. Two-level rules could be taken over in their original form from the X2MorF implementation, only the morphological filters had to be translated.

The structure of phrases within the morph grammar is much simpler than in the sentence level grammar. A simple functor/argument scheme is sufficient. What types of combinations between argument categories and functor categories are permitted is determined by the phrase structure rules of the morph grammar. The affixes (being the functors) may further restrict the arguments they may be applied to. Fig. 11 shows an example of morphological categories responsible for nominal inflection. A noun stem has to be followed by a case suffix which determines the head features of the resulting noun form. The head features of

```
((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 11: Nominal Inflection

the argument are made available to the `functor` via the `arghead` feature, thus enabling the functor to lexically restrict the argument it is applied to (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"*.

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 (such as `case`, `person`, `num` etc.) 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.

4 Conclusion

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

- 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.
- 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.

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

References

[1] Buchberger, E., E. Garner, W. Heinz, J. Matiasek, and B. Pfahringer. 1991. VIE-DU—Dialogue by Unification. In H. Kaindl, editor, *7. Österreichische Artificial-Intelligence Tagung*, pages 42–51, Berlin. Springer.

[2] Elhadad, M. 1991. FUF: The Universal Unifier User Manual, Version 5.0. Technical report, Dept.of Computer Science, Columbia University.

[3] Heinz, W. and J. Matiasek. 1994. Argument Structure and Case Assignment in German. In John Nerbonne, Klaus Netter, and Carl Pollard, editors, *German in Head-Driven Phrase Structure Grammar.* CSLI Publications, Stanford, pages 199–236.

[4] 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.

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

[6] Koskenniemi, K. 1983. Two-Level Model for Morphological Analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann.

[7] Matiasek, J. and W. Heinz. 1993. A CLP Based Approach to HPSG. Technical Report TR-93-26, Austrian Research Institute for Artificial Intelligence.

[8] Matiasek, Johannes. 1994. Conditional Constraints in a CLP-based HPSG Implementation. In H. Trost, editor, *Konvens 94*, pages 230–239. Springer.

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

[10] Pollard, C. and I. Sag. 1994. *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago and CSLI Publications, Stanford, CA.

[11] Trost, Harald. 1991. X2MORF: A Morphological Component Based on Augmented Two-Level Morphology. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia.

[12] Trost, Harald and Johannes Matiasek. 1994. Morphology with a Null-Interface. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, Kyoto, Japan, August 5-9.