RECOGNITION AND GENERATION OF WORD FORMS FOR NATURAL LANGUAGE UNDERSTANDING SYSTEMS: Integrating Two-Level Morphology and Feature Unification

HARALD TROST
Deutsches Forschungszentrum fur Künstliche Intelligenz (DFKI), Standort Saarbrücken, Stuhlsatzenhausweg 3, D-6600 Saarbrücken 11, Germany

A language-independent morphological component for the recognition and generation of word forms is presented. Based on a lexicon of morphs, the approach combines two-level morphology and a feature-based unification grammar describing word formation. To overcome the heavy use of diacritics, feature structures are associated with the two-level rules. These feature structures function as filters for the application of the rules. That way information contained in the lexicon and the morphological grammar can guide the application of the two-level rules. Moreover, information can be transmitted from the two-level part to the grammar part. This approach allows for a natural description of some nonconcatenative morphological phenomena as well as morphonological phenomena that are restricted to certain word classes in their applicability. The approach is applied to German inflectional and derivational morphology. The component may easily be incorporated into natural language understanding systems and can be especially useful in medical, scientific, or technical languages where it can be used for the automatic processing of compound words.

INTRODUCTION

Until recently, little research effort was spent on morphology (and even less on morphonology) in knowledge-based natural language processing and computational linguistics. It was not regarded as a problem of great theoretical interest. Some natural language systems used full-form lexicons, eliminating the need for

morphological components altogether. At least for inflectional morphology a number of modules (for various natural languages) exist (for the German language, e.g., see Trost and Dorffner, 1987; Finkler and Neumann, 1988; Koch et al., 1989). These modules are based on the idea of grouping words into equivalence classes according to the set of endings they take (including morphonological alterations). Emphasis is on speed, efficiency, and portability, not on linguistic appropriateness. Derivation and compounding, on the other hand, are regarded as unwieldy for an algorithmic approach and left to the lexicon. Recently, however, there is a renewed interest in morphological components (reflected in the large number of contributions to recent conferences, e.g., five at COLING-88, four at European ACL-89). Why is this so?

The reasons are manifold. New grammar formalisms developed in the feature unification paradigm (cf. Shieber, 1986) provide for an integrated processing of the various linguistic levels which were traditionally treated in a strictly modular way. For the first time we can handle phenomena that cross the border between two (or more) linguistic levels in a principle-based way. This is a strong incentive to try to come up with a morphological component that is compatible with such formalisms. For, it is impossible to draw a clear-cut borderline between morphosyntax and syntax at the sentence level. The intuitive concept "word" defies an exact definition. Sometimes the same phenomenon might be described as a syntactic one or as a word formation process, depending on circumstances.

Take for example, to-infinitive formation in German. The usual way is to put the particle *zu* in front of the bare infinitive just like in English *(zu kaufen,* to buy) rendering a sequence of two words. For a certain class of verbs though (those having a separable prefix) a single word is formed (e.g., *einzukaufen,* to purchase). As a result, morphology must deal with the latter but not with the former. A very similar problem is posed by the omission of the stem when coordinating two words (e.g., *Bin- und Verkauf* for *Einkauf und Verkauf* purchase and sale). Without knowledge about word formation, such a phrase cannot be treated properly. To cope with such examples, a uniform processing of morphological and syntactical structures seems desirable.

The lexicon cannot be regarded as a closed set of entries. At least nouns, verbs, and adjectives are open word classes. Word formation is a very active process that leads to the continuous creation of new words. A majority of them are ad-hoc compounds that are invented on the spot and may never be used again. One cannot expect them to be included in any system's lexicon. Nevertheless, a language understanding system needs means to cope with them. A prerequisite is a system that is capable of analyzing such words in terms of the morphemes of which they are built.

There are more practical reasons, too. One is the development of natural

words combine with a large number of affixes encoding a wide range of morphosyntactic information. This leads to combinatoric explosion with regard to the number of possible word forms. A number of morphonological processes alter the surface forms depending on the phonological context. The resulting huge number of different word classes makes the use of a conventional classification-based morphological component technically impossible.

Another reason is knowledge acquisition. A lexicon containing the necessary vocabulary is a basic part of every natural language system. Lexical structures must be enlarged or changed all the time. The classification process used in current morphological components is hardly understandable to the user of a natural language system because it is used to encode all the morphonological rules. On the other hand, it is almost impossible to leave the adaptation of the lexicon solely to the expert. A partial remedy for this problem is the development of intelligent knowledge-based acquisition systems (e.g., Ballard, 1986; Trost and Buchberger, 1986). But all of these systems have to be developed for a specific classification scheme. None of these acquisition components could be used for a language other than the one for which it was made. Systems that allow the description of morphological (and morphonological) processes in a declarative way can do with a much more reasonable set of word classes and are, therefore, easier to work with.

From a more abstract point of view, there is a strong tendency in computational linguistics toward formalisms that allow for the declarative description of linguistic knowledge. The idiosyncrasies of a specific language can then be captured in the data while systems can be designed independent of language. Traditional morphological components do not fulfil these requests, because they also always encode language-specific information in the processes. Another advantage of the declarative approach is that the same linguistic knowledge base can be used for both analysis and generation. Bidirectional process models are currently a promising research area (cf. Shieber, 1988).

This article next introduces the most widely used of the new approaches to morphology, namely two-level morphology. Criticism brought forward against this approach and some proposals for remedies are discussed. On the basis of this discussion I introduce my ideas for a hybrid morphological system comprising a unification-based morphosyntactic grammar and the rule part of two-level morphology. The architecture of that system is explained in detail and I show how it can overcome many of the problems of standard two-level morphology by demonstrating its application to German morphology and morphonology.

Finally, I discuss some possible applications of such a system for practical natural language understanding systems. Besides the general advantages mentioned above, some specifically suited application areas are pointed out. It can be used to automatically analyze compounds with nonidiosyncratic semantics.

especially in medical expressions. One advantage over other approaches is the natural integration of that capability into my existing system.

TWO-LEVEL MORPHOLOGY

Rules for describing morphonological phenomena have been used in generative phonology for a long time. (For an introduction to the ideas of generative phonology, see for example, Schane, 1973.) There, the derivation of a word form from its lexical structure is performed by the successive application of phonological rules creating a multistep process involving several intermediate levels of representation. Such an approach is suited for generation but it leads to problems if applied to analysis. Since the ordering of rule application influences the result, it is difficult to reverse the process.

Two-level morphology is a an attempt to overcome these problems. It was originally proposed by Kimmo Koskenniemi (1983, 1984) and has since been implemented in a number of different systems (e.g., Karttunen, 1983). Starting with Finish (Koskenniemi, 1985), it has been applied to a wide range of natural languages. Among them are English (e.g., Karttunen and Wittenburg, 1983) and German (see "Describing German Morphology using Two-level Rules" below), but also quite exotic ones, like Accadian (Kataja and Koskenniemi, 1988).

As the name suggests, the main idea behind two-level morphology is that two levels suffice to describe the phonology (or orthography) of a natural language. These two levels are called lexical level and surface level. Each level has its own alphabet. On the surface level, words appear just as they are pronounced (or written) in ordinary language, with the important exception of the null character, which will be described later. On the lexical level, the representation of words includes special symbols—so-called diacritics—which are used to represent features that are not phonemes (or graphemes) but nevertheless constitute necessary phonological information. Two diacritics that are always used are + and *$*, which indicate morph boundary and word boundary, respectively.

The link between the two levels is a set of pairs of lexical and surface characters. These pairs constitute possible mappings between lexical and surface characters. Pairs are usually written as:

lexical character - colon - surface character (e.g., *a:a* or *+ :0)*

To any of these pairs, rules may be attached to restrict their applicability. Pairs with no attached rules can be applied as defaults. Rules serve to allow or enforce (depending on the rule operator) the application of a pair in a certain surrounding phonological context. In contrast to generative morphology, rules are not applied one after the other, but in parallel. They are viewed as constraints on

mappings between the surface and the underlying form of morphs. Since there is no ordering of the rules involved, this is a completely declarative way of description.

Rules consist of a substitution (one pair of characters), a left and a right context (regular expressions made up from such pairs), and an operator. The substitution indicates the affected character pair. The left and right context serve to define the phonological conditions for that substitution to take place. The operator defines the status of the rule. In the original version (Koskenniemi, 1984) there are four of them: <=, =>,<=>, and /<=.

The *context restriction operator* <= means that the substitution of the lexical character is obligatory in the context defined by that rule (other phonological contexts are not affected).

The *surface coercion operator* => means the substitution of the lexical character is allowed only in this context* (it may not occur anywhere else).

The <=> is a combination of the former two operators.

The fourth operator /<= states prohibitions (i.e., the substitution may not take place in this context).

As an example, let's look at a simple epenthesis rule:

```
+ :e <= {s:s x:x z:z[{s:s c:c} h:h]} _s:s ;          (1)
```

This rule specifies that a morph boundary on the lexical level (indicated by the '+') between *s, x, z, sh,* or *ch* on the left side and an 5 on the right side must correspond to an *e* on the surface level. It makes no statements about other contexts where ' + ' may map to an 'e'. This rule covers part of the cases where an *e* is inserted between stem and an inflectional morph starting with *s* (like the plural morpheme) in English. The idea is that, by default, the morph boundary maps to the null character, but that in some contexts it maps to *e.* Rule (1) would correctly produce, for example, *dishes* from *dish + s.* Of course, (1) does not capture all the cases where epenthesis of *e* occurs; for example, *spies* could not be produced from *spy + s.* A more complete rule would look like:

```
+e: o {s:s x:x z:z [{s:s: c:c} h:h] :v [C y:] 0:0} _s:s;          (2)
```

Rule (2) defines all contexts where + maps to an *e* (by use of the « operator). It makes use of some conventions for abbreviations. (Other conventions not used here are given in Dalrymple et al., 1987.) A colon followed by a character denotes the set of pairs with that surface character. Accordingly, a character

*Default character pairs could be viewed as such a kind of rules with an empty context.

followed by a colon means the set of pairs with that lexical character. Names may be given to arbitrary sets of characters that are globally defined. In rule (2), the C stands for the set of consonants (i.e., b:b, c:c, d:d,. . .). To cope with the spies example, we need another rule that controls the mapping from *y* to i.

$$\begin{array}{l} \text{y:i<=>C\_\{ + :e[+ :e:e]\};} \\ \quad \text{V C}^{+}\text{\_ + :C;} \end{array} \qquad (3)$$

Rule (3) differs from (2) in that it has two different contexts. If either of them is satisfied, the substitution must occur (i.e., contexts are OR connected). Contexts are separated by semicolons. In the second context the ' + ' operator is used. It indicates that at least one occurrence of the preceding sign is expected (the operator '*' has the reading *arbitrarily many occurrences).* V stands for the set of vowels. Rules (2) and (3) together would now correctly map *spies* to *spy + s.* Rule (2) handles some more cases like mapping *shelf + s* to *shelves* (if an appropriate rule for the pair f:v is available) *and potato + s to potatoes.*

A given pair of lexical and surface strings can map only if they are of equal length. There is no possibility of omitting or inserting a character in one of the levels. On the other hand, elision (deletion) and epenthesis (insertion) are common phonological phenomena. To cope with these, the null character (written as 0) is included in the surface alphabet. The null character is taken to be contained in the surface string for the purpose of mapping lexical to surface string and vice versa, but it does not show up in the output or input of the system. Diacritics are mapped to the null character by default. Any other mapping of a diacritic has to be licensed by a rule.

Assumption of the explicit null character is essential for the processing. A mapping between a lexical and a surface string presupposes that for every position there exists a character pair. This implies that both strings are of equal length (nulls are considered as characters in this respect). Rules can then be implemented as finite state transducers and these transducers can be applied in parallel to a given pair of strings. The use of finite state machinery allows for efficient implementation. An interesting aspect is the automatic compilation of two-level rules into finite state transducers. One example for such a compiler (named TWOL) is described in Dalrymple et al. (1987).

Up to now, we have described only the rule part of two-level morphology that is responsible for taking care of morphonological phenomena. It is complemented by a partitioned lexicon of morphs (or words) that takes care of word formation by affixation. The underlying idea is that of partitioning the whole lexicon into (nondisjunctive) continuation classes. The members of every continuation class form their own sublexicon. For every morph, a set of legal continuation classes is stored. The continuation class found with a morph determines

which sublexicon must be searched for continuations. The class of morphs that can begin a word is stored in the so-called *init lexicon.*

The whole process can be viewed as the stepping through a finite automaton. A successful match can be taken as a move from some state *x* of the automaton to some other state *y.* Lexical entries can be thought of as arcs of the automaton: a sublexicon is a collection of arcs having a common *from* state. The lexicon in two-level morphology is used for two purposes: one is to describe which combinations of morphs are legal words of the language, the other one is to act as a filter whenever a surface word form shall be mapped to a lexical form. Its use for the second task is crucial because otherwise there would be no way to limit the insertion of the null character.

To enable fast access, lexicons are organized in the form of a letter trie (Fredkin, 1960). Such a structure is well suited for an incremental (letter-by-letter) search because at every point in the trie exactly those continuations leading to legal morphs are available. With every node that represents a legal morph its continuation classes are stored. In recognition we can now make use of that structure. Search starts at the root of the trie. Each character that is proposed must be matched against the lexicon. Only if that character is a legal continuation at that node in the trie may it be considered as a possible mapping.

Related Formalisms

Some proposals have been made to extend or change the two-level framework as proposed by Koskenniemi. Bear (1986) notes the difficulty in expressing optional rules with the standard rule set. As an example he gives *e* insertion between stems ending in *o* and flectional *s* in English. In some cases (e.g., *potatoes)* the *e* is obligatory, in some others (e.g., *banjoes* or *banjos)* it is optional, and in some (e.g., *pianos}* it may not occur. To cope with optionality, he proposes a system with a slightly altered set of operators.

In particular, Bear's system uses the three operators—'allowed', 'disallowed' and 'requested'—instead of the standard ones. As in the standard formalism there is a set of default character pairs. 'Allowed' rules serve to restrict the occurrence of a character pair to certain contexts, 'disallowed' rules, on the other hand prohibit the occurrence of a certain character pair in the given context. The 'required' rule is a combination of an 'allowed' rule with a corresponding 'disallowed' rule for all the other character pairs having the same lexical character. Internally, 'required' rules are represented using the other two operators. Unfortunately, Bear's proposal solves the above-stated problem only partially. One can now state rules in a way that the *e* is always optional, thereby overgenerating for all words where *e* insertion is required or forbidden.

There is a correspondence to Koskenniemi's operators. The 'required' context restriction corresponds to the <=> operator, 'allowed' to =>. The <= has a

more complicated correspondence. The rule *+:e <= s:s__s:s ;* in the Koskenniemi formalism would correspond to *+:0 disallowed in s:s__s:s* iff *+:0* and *+:e* are the only pairs with '+' as lexical character.

Bear also proposes a different implementation of the two-level rules. Instead of their transformation to finite state transducers, the rules are interpreted directly. While working from left to right over the input string, the system keeps track of the left contexts of all rules. If a left context is found to be satisfied, the rule becomes applicable, that is, for every such *allowed* rule a branch (representing a possible partial mapping) is created (if it is not wed out by a 'disallowed' rule). The right context of this rule must now be watched. If it turns out to be fulfilled (and if no disallowed rule becomes true) a successful match has been reached.

One subtle difference between direct rule interpretation and transducers occurs in the repeated application of the same rule to one string. The transducer implicitly extends the phonological context to the whole string. It must therefore explicitly take care of overlapping right and left contexts [e.g., in (1), the pair *s:s* constitutes both a left and right context]. With direct interpretation, a new instance of the rule is activated every time the left context is found in the string and overlapping must not be treated explicitly.

Black et al. (1987) note the inelegance of Koskenniemi's formalism for describing a phonological (or orthographic) change that affects not a single character but a sequence of characters. In Koskenniemi's algorithm, this requires the definition of a separate rule for every character involved, the interaction of which is not easy to understand.

As a remedy, Black et al. (1987) propose a slightly different form of rules. Their rules consist of a surface string (called LHS for left-hand side), an operator (<= or =>) and a lexical string (called RHS for right-hand side). LHS and RHS must be of equal length. Surface-to-lexical rules (=>) request that there exists a partition of the surface string so that each partition is the LHS of a rule and the lexical string is a concatenation of the corresponding RHSs. Lexical-to-surface rules (<=) request that any substring of a lexical string which equals a RHS of a rule must correspond to the surface string of the LHS of the same rule. The rules in (4) are equivalent to rule (1).

ses ⇒ s+s ses ⇐ s+s

xez ⇒ x+s xes ⇐ x+s

zes ⇒ z+s zes ⇐ z+s

shes ⇒ sh+s shes ⇐ sh+s

ches => ch + s ches <= ch + s (4)

These rules collapse context and substitution into one indistinguishable unit. Instead of regular expressions, only strings are allowed. Using this approach, phonological (orthographical) changes affecting more than one character can be described easily.

A drawback is the fact that surface-to-lexical rules may not overlap. If two different changes happen to occur close to each other, they must be captured in a single rule. It remains unclear how this could be done in a situation where the same rule must be applied to an unknown number of instances. Take, for example, a phonological process like vowel harmony in Turkish. The stem vowel determines the vowel in the surface form of all suffixes. So-called large vowel harmony enforces *i* after stem *e* and *i, u* after ö and ü (we disregard the rest of the Turkish vowels here). Consequently, the lexical form *bil+ Vr+ Vm** would render the surface form *bilirim* (I speak), while *gör +Vr+ Vm* yields *görürüm* (I see). Such a process could easily be captured in the standard formalism with rules like

V:i <=> {:e :i} X* _; (where X is the set of all characters besides the vowels) (5)

but it cannot be described in the same generality in the rule formalism described by Black et al. (1987).

Ruessink (1989) tries to remove this problem by introducing contexts again. Both LHS and RHS may come with a left and right context. He also allows LHS and RHS to be of different length, doing away with the null character. He gives no account of the resulting complexity of his algorithm though. One can suspect that it is, in general, less constrained than the Koskenniemi system.

Recently, some alternative proposals for morphological systems have been made in computational linguistics. They include so-called paradigmatic morphology described in Calder (1989) and the DATR system (Evans and Gazdar, 1989a, 1989b). Common to both is the idea to introduce some default mechanism that makes it possible to define a hierarchically structured lexicon where general information is stored at a very high level. Lower in the hierarchy this information can be overwritten. Both systems seem to be more concerned with morphosyntax than with morphonology. It is an open question if these approaches could somehow be combined with two-level rules.

*The V stands for a lexically underspecifled vowel. Its surface form is determined by the preceding stem vowel and the associated vowel harmony process.

Criticism against Two-Level Morphology

Criticism has been directed against various aspects of two-level morphology. We will start with some results by Barton (1986) and Barton et al. (1987, Chapter 5) concerning the complexity of two-level morphology. The claim of supporters of two-level morphology is that because of the compilation of rules into finite state transducers and because of the use of continuation classes for morphosyntax, these systems are inherently fast and efficient.

In contrast to that claim, Barton shows that—despite the use of finite state machinery—the problem is at least NP-complete with respect to the number of rules. This is demonstrated by using two-level morphology to solve arbitrary constraint-satisfaction problems which leads to combinatorial search. At the same time he argues that such complexity is probably not required by the problems posed by the morphology of natural languages. He says that morphology can do with a more constrained algorithm than the standard mechanism used in two-level morphology. He proposes the use of constraint algorithm with local information flow and restricted interactions between different automata. This less complex algorithm is based on an algorithm for the interpretation of line drawings described by Waltz (1975).

Koskenniemi and Church (1988) argue that Barton—while his results are formally correct—misses the point, because inherent properties of natural languages inhibit the creation of rule sets that would lead to exponential complexity. More specifically, they argue that the only critical phonological processes are the vowel harmony processes like (5) and that existing languages exhibit, at most, two different vowel harmony processes. Still, it seems promising to try to restrict the complexity of two-level morphology by using a weaker algorithm. It is interesting to note in this respect that Ritchie (1989) proves that arbitrary finite state transducers allow the expression of problems that cannot be expressed in two-level rules. He does not show though if his results have any impact on the complexity considerations of Barton.

Apart from complexity and efficiency considerations, linguistically motivated criticism against the two-level approach has also been brought forward. In Koskenniemi's original proposal, morphosyntax is handled with the use of continuation classes. The use of continuation classes is probably insufficient for the description of more complicated morphological phenomena. Even if it were sufficient, it makes it very difficult to state the word grammar in a linguistically adequate way. It also goes against the tendency to integrate morphology and syntax more closely as suggested in the introduction. As a solution, the replacement of these continuation classes with a grammar based on feature structures describing the legal combinations of morphemes has been proposed in several papers (e.g., Bear, 1986; Ritchie et al., 1987; Carson, 1988; Pulman et al., 1988; Görz and Paulus, 1988). Of course, unification of feature structures with

disjunction and negation is NP-complete (Smolka, 1988). Therefore, compared to continuation classes, one gets a higher degree of complexity. But feature structures are easily integrated into a sentence grammar that is based on such feature structures as well. Therefore, their use in morphology seems to be well motivated.

Another point is the liberal use of diacritics at the lexical level. The original intention was to describe phonological features apart from the phonemes like morph boundaries and stress markers. But, mainly because there is no way to restrict the application of phonological rules by the morphosyntax (captured in the continuation classes), diacritics are used to express purely morphological features as well. Several authors have criticized this point. Bear (1988b, p. 29), for example, notes that

> *. . . diacritics are put into the lexical representation of a word in order to allow the linguist to write a phonological rule that applies in some words and not others according to the presence or absence of the diacritic.*

This points at a serious lack of expressivity of the formalism which forces grammar writers to use available features in a nonintended way.

One typical problem with two-level morphology is that some morphonological rules apply only to certain subclasses of the vocabulary. As an example, Bear (1988b) cites the plural forms of English nouns ending in *-o* like *potato* and *piano* which are *potato-e-s* and *piano-s* respectively. Emele (1988) gives some examples for German, where *schwa* (an unmarked neutral vowel, in German orthographically represented by e) epenthesis depends on word category. The widely used solution in such cases is the use of a diacritic to ensure correct rule application, that is, a diacritic is inserted in the lexical representations belonging to that subclass and the corresponding rule expects that diacritic in its context.

A related problem is the fact that morphosyntax makes use not only of affixation but also of a range of nonconcatenative processes. One such example is umlaut in German. Umlaut is used to express a range of morphosyntactic features (see the next section). It coincides with certain affixes (one of which happens to be the null morph). In standard two-level morphology, this can again be handled by marking these affixes with a certain diacritic which is then put into the right context of umlaut rules. The only use of that diacritic is to encode morphosyntactic information at the phonological level. Such a use of diacritics should be reexamined. Diacritics have their place in two-level morphology if they are justified for phonological reasons, but they should not be used to transfer information or to encode purely morphological features.

More generally stated, a major problem with the standard two-level formalism is the lack of interaction between the two-level rules and the word formation nert (i.e. the continuation classes). When replacing continuation classes with a

feature-based unification grammar, one gets the ability to express all morphological features there. What is needed is a way to transfer information from the unification grammar to the two-level rules and vice versa. I will show that this can be accomplished by associating feature structures—called filters—with the two-level rules.

One final remark regarding the application of two-level morphology to written language: Two-level rules have been developed as a means to deal with morphonology. Nevertheless, most applications deal with written language, hence, one should apply them to orthography. This may lead to some complications because rules of orthography are slow to adapt to changes in the language, and many orthographic conventions can be explained only diachronically.

German Morphology and Morphonology

German is an inflecting language with a comparatively rich morphology. Basically, inflection is expressed by affixation. As usual in inflecting languages, such affixes may carry more than one morphosyntactic feature (portmanteau morphs), and the same morpheme may be realized by different affixes in different inflection paradigms. The unmarked case, that is, the nonoccurrence of an inflectional affix is also quite common (e.g., case marking for nouns is rudimentary, at most, genitive singular and dative plural are explicitly marked by endings).

The use of a prefix for morphosyntactic making is restricted to two cases. One is the particle *ge-*, which indicates past participle together with the appropriate ending *sag* (say) => *ge-sag-t].* The other one is the particle *zu* which helps forming to-infinitive just like it does in English.

Apart from affixation there are some nonconcatenative morphological phenomena as well. The most prominent one is umlaut, where the stem vowel changes in a systematic manner. Transformations are *a* => *a*, *o* => *o*, *u* => *u*, and, in some cases, *e* => *i.* While the umlaut can historically be explained as a phonological phenomenon, it has attained the status of morphological feature* in modern German. Umlaut realizes quite different features. Most important is its use in inflection. With nouns it can mark the plural either by itself (e.g., *Mutter* => *Mutter)* or in combination with a some plural endings (e.g., *Mann* => *Manner),* depending on the inflection paradigm. With adjectives, it is used to mark some comparative forms *(gro@* => *grower* => *am größten),* for verbs following strong conjugation, it marks the subjunctive II *(gabst* =* *gäbst)* and second- and third-person singular of the indicative present tense *(gebe* => *gibst).* It also

*One fact supporting this claim is that it is still productive in some cases. When forming a diminutive form using the derivative affix *-chen,* the stem may carry an umlaut even if its a foreign one (e.g., *Atom* => *Atömchen).*

occurs in some forms of modal verbs like the plural forms of indicative present tense.

Umlaut also occurs in derivation in combination with a number of derivational particles, for example *-lich (klagen => kläglich)*. In contrast to its use in inflection, umlautung provides for no extra morphosyntactic information in derivational forms. Lastly, it appears in compounding in combination with some "Fugenelement" (joining element) (e.g., *Männerchor,* male chorus). (Alternatively, one could interpret these forms as plurals, because they always coincide morphologically with the plural form.)

To have the potential for umlaut is an idiosyncratic property of stems, at least viewed synchronically. Stems having an umlaut in inflection usually also take one in derivation and vice versa. Unfortunately, there are some exceptions to the rule. The plural of *Hand* (hand) is *Hände* while the derived adjective *handlich* (handy) shows no umlaut. The plural of the noun *Hund* (dog) is *Hunde* (although the plural morph *-e* triggers umlaut, e.g., *Wolf => Wölfe),* while the derivation with suffix *-in* is *Hündin* (female dog).

There are two common ways to cope with umlautung in conventional morphological components for German. One is to treat all forms created by umlautung as suppletions, that is, to enter these forms explicitly into the lexicon. This is linguistically inadequate, because it obscures the morphological (and phonological) similarity between the two forms. The other solution is a special function replacing (and interpreting) or generating the umlaut in all stems which are marked for umlautung required by the context. This makes umlautung a special case neglecting its status as a regular means of morphosyntactic marking.

Besides umlaut there are other vowel changes as well, but they are lexicalized and not recognized as conforming to a rule anymore. The so-called ablaut is used to form the different tensed forms of modal verbs and verbs following strong conjugation. For instance, the verb *schwimmen* (to swim) has the allomorphs *schwimm, schwamm, schwomm* for the formation of present tense, past tense, and past participle, respectively. Stems formed with ablaut are probably best treated as suppletive forms, because rules governing the vowel change would need a quite idiosyncratic phonological context (for the stem vowel *i* there is, e.g., also the possibility *sing, sang, sung)* and to contemporary speakers of German, the formation of these forms from a common stem is no more transparent.

Another particular feature of German morphology are separable verb prefixes, for example, *einkaufen* (to buy), but *ich kaufe ein* (I buy) when used in a main clause. These particles force the past participle prefix *ge- (ein-ge-kauft)* as well as infinitival *zu (ein-zu-kaufen)* into an infix position.

There are a number of morphonological rules. The most interesting ones concern the occurrence (or absence) of schwa at morph boundaries (between a

stem and an inflectional ending) and inside some stems (which would otherwise not be phonologically well formed). According to current literature (Rennison, 1980; Giegerich, 1987; Wiese, 1988), this phenomenon is best explained as epenthesis (i.e., insertion) of the schwa. The application of epenthesis rules in some cases seems to depend on the morphological class, compare for example the two different past tense forms of *senden* (to send) *sand-te* vs. *send-e-te,* where schwa epenthesis occurs only with the weak inflexional paradigm.

We will now go through some of the examples in more detail to show the problems in developing a coherent theory of schwa epenthesis in German word forms. Let's start with schwa epenthesis inside stems. Roughly, the concerned stems are those ending with a consonant cluster the last consonant of which is /, *r, m* or *n.*

stem		schwa	no schwa
handl	(trade)	*handelst*	*handle*
niedr	(low)	*nieder*	*niedrig*
atm	(breath)	*Atem*	*Atmung*
trockn	(dry)	*trocken*	*Trockner*

(6)

The rule for stem epenthesis is roughly the following: For all those stems, schwa is inserted if the morph is followed by a word boundary (e.g., *Trockeneis,* dry ice) or a derivative morph starting with a consonant (e.g., *atemlos,* breathless). Those stems ending in / and *r* additionally insert schwa if followed by any morph starting with a consonant (e.g., *handelst,* you bargain).

We will now turn to schwa epenthesis at the morph boundary. A pervasive phenomenon is schwa epenthesis before the ending *n.* This is true for most of the different morphosyntactic uses, with verbs *(geh + n => gehen),* nouns *(Fleck + n => Flecken)* and adjectives *(alt + n => alten).* There is some interdependence between schwa inside the stem and this epenthesis rule. We will discuss this problem later on.

verb stems ending with a final *d* or *t* also trigger schwa epenthesis before inflectional *s* and *t* (e.g., *bad + st => badest, arbeit + t => arbeitet).* For verbs following strong conjugation this rule does not apply in all cases (e.g., *tritt + st => trittst, tritt + t => tritt, trat + st => tratst).*

An interesting problem is posed by subjunctive formation. To indicate subjunctive, a schwa is always inserted between stem and endings. In cases where epenthesis takes place with indicative as well, forms are of course indistinguishable. According to these data, one could suppose a morpheme for subjunctive realized by the morph *-e* which is attached to the stem. There is one counterex-

ample though. For stems ending in *el* and *er* (coming with a schwa inserted in the stem) no subjunctive *-e* appears. If we do not want any schwa elision rules, we cannot assume such a morph. Instead, we are forced to explain the data with an epenthesis rule which is triggered by subjunctive.

A last case includes word forms where the schwa indicates the ending, for example, first-person singular *(geh-e,* I go; *sag-t-e,* I said). According to Wiese (1988) this can be explained by assuming a kind of "empty" morpheme which cannot be realized at the surface but nevertheless triggers schwa epenthesis.

A different view on schwa in German is expressed by Issatschenko (1974). He assumes two kinds of schwa. One which he calls "schwa mobile" can be present or absent, depending on the phonological context. The other one called "schwa constans" is always present. Schwa at the end of a word would be an example of the schwa constans. Another example is the schwa that comes with adjective endings. According to his theory, one would assume all of these endings to start with *e (e, em, en, er, es).* This assumption gets the data straight. Adjectives with a stem ending in *el* and *er* behave differently from verbs with such stems if we assume the flectional affix to come without schwa (e.g., *-ri)* in both cases: for the adjective *teuer* (expensive) affixation with *n* yields the word form *teuren;* for the verb with stem *wander* (to hike), the form *wandern.*

Other morphonological phenomena (e.g., the elision of *s* or *t* in certain contexts) pose no problems. They can be explained by post-lexical rules straightforwardly without considering morphological features.

Finally, something should be said about orthography as opposed to phonology. A phoneme is sometimes represented not by a single character but by a combination of several characters (in German such combinations are *sch, ch,* and *ng,* which represent the phonemes \, *x,* and , respectively). Another problem is the optional representation of the length of a vowel. In German, an *h* following a vowel indicates a long vowel *(ah, eh, ih, oh, uh).* A different representation is *ie* for a long 'i'. Both methods may be combined yielding *ieh* to represent the same phoneme. Some vowels may simply be geminated to express length *(aa, ee, oo).* Short vowels can be indicated by a gemination of the following consonant.

In itself, these things are not necessarily problems for two-level morphology. They may become problems only when different word forms make use of different conventions. Take, for example, the German verb *sehen* (to see). The second-person singular is formed by umlaut transforming *e* to *i.* Thus, one would expect the form to be *du sihst* (you see), but the length of the vowel is indicated differently, producing the actual form *du siehst.* Such problems render some forms as irregular that would otherwise fall into the regular classes.

A possible solution to such problems with orthography is to represent combinations of characters representing a single phoneme as a single surface (and corresponding lexical) character. For example, *ch* in German can never indicate

two phonemes, so one might as well treat them as a single character. With *sch,* it is a bit more difficult. It can indicate a single phoneme (e.g., *haschen,* to snatch) or a concatenation of *s* and *ch* (e.g., *Haschen,* bunny). This leads to an ambiguity in the creation of the surface string.

DESCRIBING MORPHONOLOGY AND MORPHOLOGY USING TWO-LEVEL MORPHOLOGY

How can two-level rules be used to describe morphonological phenomena? Apparently, the most important application of two-level rules is to describe the phonological (or orthographical) changes that occur in the process of concatenating morphs. There are three types of possible alterations:

a. A lexical character may change to a different surface character [rule (3) is an example for a change from lexical *y* to surface i].

b. An elision rule may prevent a lexical character from appearing in the surface string. In two-level morphology, this involves a rule that maps that character to the null character (while the default is a mapping to the corresponding surface character).

c. The third possibility is epenthesis. There a character is inserted in the surface string that has no correspondence in the lexical string. The null character is not included in the lexical alphabet; therefore, a direct encoding of this phenomenon is impossible. Instead we must use a more indirect way. We need a lexical character that maps to the null character by default (like diacritics). An epenthesis rule is a rule that defines a mapping of that lexical character to a surface character different from the null character. An example is rule (2), which maps lexical ' + ' to surface *e.*

More difficult to handle are situations where more than one character is involved in a phonological (or orthographic) change. We must then define different rules for each character involved. Moreover, one must make sure that these rules interact in the expected way. Again, a simple example would be rules (2) and (3). Their contexts are defined in such a way that the expected orthographic changes take place. Rule (2) maps ' + ' to *e* after a lexical *y.* Rule (3), on the other hand, maps *y* to *i* in front of a ' + ' mapped to *e.*

Basically, two-level morphology has been developed to deal with morphonological phenomena that are related to the process of affixation. Historically, this can be explained because affixation is the single most important morphological process in agglutinating languages like Finnish, the first language to which two-level morphology was applied.

Morphosyntax is handled with the use of continuation classes (see "Two-Level Morphology," above). Again, the emphasis is on affixation. Continuation

classes handle "rightward" subcategorization straightforwardly in that one can define, for every morph, the class of morphs that may follow to complement it. "Leftward" subcategorization can be dealt with only indirectly. If, for example, a derivative suffix requests a certain subclass of adjectives, this cannot be stated with the suffix. Instead, it must be encoded in the continuation classes of the adjectives concerned.

More complex concatenation rules, where information must be transmitted over some distance, can hardly be described in a natural way using continuation classes. Take, for example, prefixes that go only with a certain class of endings (examples in German are the past participle formed by the prefix *ge-* and a suffix and to-infinitive of verbs with separable prefix). Grammar rules are much better suited for this task.

Besides affixation, there is a wide range of nonconcatenative morphological phenomena as well. An attempt to apply a two-level device to such phenomena as well is described in Kay (1987). His aim is to describe Arabic word formation. In Arabic, root forms are triples of consonants. Their internal ordering and vocalization in a specific word form carries the morphosyntactic information. Kay proposes to use transducers running in parallel on four instead of two tapes. One is the surface tape as usual. The other ones are for the consonant triple (resembling the root), the vowels for vocalization, and the pattern of vowel-consonant distribution (describing the morphosyntactic status), respectively. Morphological knowledge is encoded in the quadruples of characters—called frames—responsible for the mapping of the tapes. Although it is a very interesting approach, it has not been pursued further in the literature. Approaches to deal with nonconcatenative morphosyntax in German (e.g., umlaut) are described in the next section.

We have now reached the point where we can formulate some requirements a two-level morphological component must meet if it shall be applied to the full range of morphological phenomena described up to here.

- The integration of morphosyntax into a more general framework, including sentence level syntax on the one hand and the more natural description of rightward reaching subcategorization requirements on the other hand, ask for the use of a (feature-based) word grammar instead of continuation classes.
- To handle morphosyntactic phenomena involving the change of the internal structure of a stem (e.g., German umlaut), we need a means to transfer information between two-level rules and the word grammar. The indirect treatment with the use of diacritics is inappropriate.
- Because the application of some morphonological rules is restricted to certain morphosyntactic environments (e.g., word class) it must be possible to restrict the application of two-level rules using filters that are checked against morphosyntactic information.

Describing German Morphology Using Two-Level Rules

Many morphonological phenomena in German can be described straightforwardly by two-level rules. There are, for example, elision rules for s and *t* at morph boundaries. A lexical string like *$ras + st$* must render the surface string *rast, $tritt + t$* the surface sting *tritt.* The following two rules will take care of this:

$$s{:}0 \Leftrightarrow \{s{:}s\ x{:}x\ z{:}z\} + {:}0\_\_; \qquad (7)$$

$$t{:}0 \gg t{:}t + {:}0\_;$$

Two phenomena require close attention: umlaut and schwa epenthesis. Two different solutions have already been proposed for umlaut in the two-level paradigm. Both solutions rely on the idea of representing stem vowels that exhibit umlautung with special diacritics* (e.g., *A)* at the lexical level. Both are also concerned only with umlaut in verb conjugation and disregard the other uses of umlaut in German morphology.

Görz and Paulus (1988) extend the two-level formalism. To trigger the appropriate substitution, they use a separate data structure which they call vowel table. Stems contain only a *"generic"* vowel. The vowel table indicates for each word form which vowel is to appear at the surface level for the generic vowel. Using this vowel table they can handle ablaut as well.

Schiller and Steffens (1990) stay in the two-level paradigm. In stems that exhibit umlaut, a diacritic represents that stem vowel. By default, this diacritic is mapped to the original vowel (e.g., *A* => a). A rule is used to produce umlaut. To trigger this rule, they introduce still another diacritic symbol. Flectional endings triggering umlautung start with the diacritic *$* (realized as 0 at the surface level). (This *$* should not be confused with the diacritic for word boundary in the standard formalism. In the system of Schiller and Steffens a different symbol is used for that purpose.) The context to the right of the substitution of all umlaut rules requires the occurrence of exactly that *$.* Therefore, the umlaut rule would fail if no such affix follows the stem. (As a consequence, the null morph must be explicitly represented by *$* in lexical strings where morphosyntactic information is expressed by umlautung only.)

Although both solutions work, they provide no clean and general solution for the integration of umlautung in the framework of two-level morphology. The use of a separate data structure is contrary to the intuition that umlautung is a regular phenomenon of German morphology, the treatment of which should

*One can regard these stem vowels as being underspecified lexically, which leads to different representa-

require no extra mechanism. And the use of the diacritic *$* is one that has been rightly criticized in the literature (see "Related Formalisms," above). It places a burden on morphonology which clearly belongs to morphosyntax.

The approach by Schiller and Steffens (1990) would also run into problems if it were to be extended to derivational morphology (currently the system deals only with inflection). As mentioned, umlaut in inflection does not always imply umlaut in derivation and vice versa. What we do need in this case is some explicit flow of information between the morphosyntactic part and the two-level rules proper, as proposed above.

For schwa epenthesis inside stems, we use a separate diacritic e. The corresponding default pair is e:0. If we adopt the idea of a "schwa constans" we can define an epenthesis rule for e:e that is applicable across the whole lexicon. For schwa epenthesis at the morph boundary, the situation is different. We need to restrict the application of some of the various contexts attached to the + :e epenthesis rule. Probably the most obvious example is that subjunctive mode triggers schwa epenthesis (see "German Morphology and Morphonology," above). (Of course, the solution to introduce a diacritic is also open. But as with umlaut this solution seems inappropriate.)

A remark is in order with regard to the use of continuation classes vs. more elaborate grammar rules. Unlike agglutinating languages like Finnish, inflecting languages like German use different inflectional paradigms for, say, declination and conjugation but usually have only a quite limited vocabulary of endings. Almost every ending has therefore different interpretations, that is, a certain ending may stand for different morphemes in different contexts. This leads to many different continuation classes (or sublexicons) and, consequently, to inefficiency. Furthermore, a single ending sometimes encodes two or more morphemes in parallel (e.g., number and case, number and person). Another point against the standard formalism are unmarked forms where a null morph is assumed. An explicit word grammar makes it possible to treat null morphs exclusively in the grammar part. They no longer have to show up in the lexical string given to the the two-level part anymore.*

EXTENSIONS TO THE TWO-LEVEL FORMALISM

A number of proposals have been made to counter the criticism against two-level morphology formulated in the section "Related Formalisms," above. To us, the most important drawback seems to be the lack of a clear distinction between morphology and morphonology. The concatenative part of morphology

*Note that the use of diacritics for the triggering of rules (e.g., to handle umlaut) entails that at least those null morphs must show up in the lexical string which contain that diacritic (although it always translates

has been separated from the two-level rules proper where all other phenomena are handled. The only—very indirect—connection is via the use of diacritics. This unmotivated use of diacritics has been rightly criticized.

A first step to overcome these problems is to replace continuation classes by some other formalism more appropriate for the description of a word grammar. Most alternatives up to now make use of some form of feature-unification-based formalism (e.g., Bear, 1986; Ritchie et al., 1987; Carson, 1988; Görz and Paulus, 1988). My system follows this line, too. The word grammar is exclusively expressed in feature structures.

For the integration of word grammar and two-level rules according to the demands expressed in the previous section, Bear (1988b) and Emele (1988) suggest making use of the feature structure part to restrict rule application. Bear (1988b) introduces the notion of what he calls "negative rule features." The algorithm he proposes for their processing rests on the direct interpretation of rules in his system as described uner "Related Formalisms," above. In the course of recognition, all two-level rules that are applied must be collected. "Allowed"-type rules must also be assigned to the character pair they support. Whenever a morph boundary is reached, the algorithm must search the set of rules. All those rules where the rule filter does not unify with the feature structure of the morph must be discarded. A character pair that is left without a supporting rule indicates that the solution found so far must be discarded as a whole.

There seem to be flaws in this algorithm. "Disallowed" rules block the use of a character pair whenever their morphological context is found. If they are provided with negative rule features, they would still block in the same cases. Therefore, the morph that would exclude the application of that rule will never be regarded as a possibility. In generation, of course this problem would not appear. A second disadvantage is that the algorithm takes into account only such features that are lexically linked to the morph where the rule applies.

Emele (1988) proposes allowing arbitrary feature structures as filters for rules. His proposed rules are of the form *IF feature structure*, THEN *two-level rule.* As an example, he gives a rule for umlaut with nouns. In our notation,* this rule looks like:

IF [number: pi] THEN A:a <=>_; (8)

Although the basic idea behind this proposal seems plausible, there are several problems. The possibility of umlaut is an idiosyncratic property of stems. Num-

*The original rule makes no use of the diacritic A Instead it uses the additional feature [umlaut +] in the rule All stems must be marked with either [umlaut +] or [umlaut -] and the phonological context must make sure that only the stem vowel is changed Besides an obvious loss in efficiency little seems to depend on that

ber, on the other hand, is indicated by the inflectional ending. One must, therefore, check the feature [number: p1] against the ending. Since, in general, it depends on the grammar to which morph the needed information (in our example [number: p1]) is attached, this is not a viable solution. Or we must make sure that the relevant features are associated with the structure of the word as a whole. This leads to difficulties, since the relevant information may be only local and not be applicable to the whole word.

A second problem is that this filter works only with noun inflection. As we have seen, umlaut is a much more pervasive feature of German morphology. To integrate the morphosyntactic interpretation of umlaut into the rule filter should be avoided. It would lead to separate rules for every different morphosyntactic feature that is expressed by umlaut. Instead, we would like to transfer from the two-level rule to the word grammar only the information that umlaut has taken place. The interpretation of the umlaut is then left to morphosyntax.

In conclusion, we can say that both proposals recognize the problem, but that their solution is not wholly appropriate and, also, that the algorithms proposed contain some subtle errors. The rest of the paper is devoted to a thorough discussion of my system. I take up the idea of attaching feature structures to the two-level rules but the algorithm avoids the pitfalls pointed out in this section.

SYSTEM ARCHITECTURE

Now I can describe the architecture of my morphological component. It is a further development along the lines suggested above. As we have seen in the last section, there are two sources of problems for an approach restricting the application of the two-level rules with the use of filters:

- The interdependence between the definition of the rule operators and filter application must be considered. Both are means to restrict the applicability of a rule and may therefore interfere with each other.
- The feature structure attached to a morph may be complete only when the word grammar has been successfully applied to the whole string. When a filter is checked against a feature structure during processing, the possible incompleteness of that structure must be regarded.

The general architecture of the system consists of two parts. One is a feature-based word grammar. This word grammar interprets the lexical string as a sequence of morphs with attached feature structures. It integrates all these feature structures into one feature structure representing the whole word form. The other part is a set of two-level rules. They map between a surface and a lexical string. Both parts of the system make use of a common lexicon of morphs. Morph boundary and word boundary are part of the lexical representa-

tion of morphs, that is, prefixes end with ' + ', suffixes start with ' + ', Accordingly, every legal lexical string must purely be a concatenation of morphs.

To provide for the necessary interaction between word grammar and two-level rules, it is possible to attach arbitrary feature structures (so-called filters) to the two-level rules. These filters are used to restrict the application of the rules. The purpose of the rules is to license a certain character pair. A rule may be applied only if its filter unifies with the feature structure of the morph to which the lexical character of that pair belongs. Since a lexical string is just a concatenation of morphs, that morph is always defined.

Figure 1 gives a sketch of the architecture. The two parts operate largely independently. Two-level rules operate on the characters, the word grammar on the feature structures. They are coupled via the lexicon which contains all legal morphs. Morphs are seen as sequences of characters by the two-level part and as bundles of feature structures by the word grammar. Explicit interaction between the two parts is made possible by the filters. They allow the transformation of information directly from the two-level part into the feature structures and the other way round.

If an obligatory rule and an optional rule are applicable at the same time, the obligatory rule wins. Two (or more) applicable optional rules show a (local) ambiguity, two different applicable obligatory ones mean a failure (this could happen only if the two rules have nonexclusive contexts). The drawback with the "obligatory" operator is just that possibility of conflict. (Conflicts may of course also arise in the Koskenniemi, 1983, system. For a discussion, see Dalrymple et al., 1987.) The advantage is that we can avoid the problem with Bear's 'disallowed' operator with respect to the rule filters.

The mapping from one string (lexical or surface) to the other one(s) is organized in a breadth-first way. Every rule with a satisfied (or empty) left context is active. Those active rules where a fitting character pair is found constitute one hypothesis, that is, at every point in that process there are as many hypotheses as there are different character pairs licensed by rules. One can view this process as spanning a search tree. Right contexts are used as filters for new hypotheses. If no more continuation can be found, that branch of the search tree must be discarded. A special case is obligatory rules. If the application of an obligatory rule leads to at least one successful solution, all the optional branches starting at the same position are removed.

Morphs are organized in a letter-trie-structured lexicon. Each morph has an associated feature graph describing its morphological and morphosyntactic properties. This lexicon is also accessed by the complementary unification grammar part. That unification grammar encodes the word formation rules. The next subsection will give a detailed description. For the moment, it suffices to say that it governs the combination of stems with prefixes and suffixes while at the same time delivering a morphosyntactic interpretation. To handle nonconcatena-

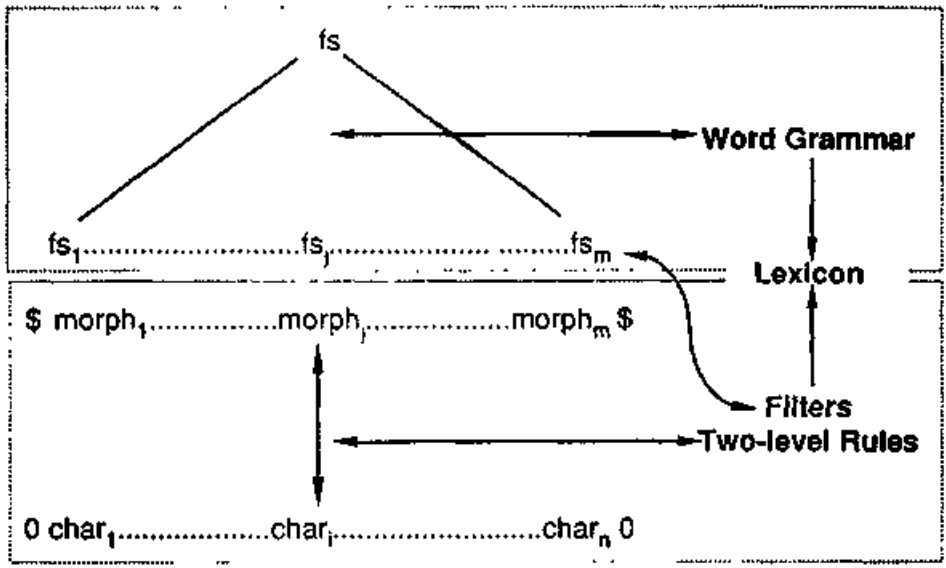


FIGURE 1. Architecture of the system.

tive morphology (i.e., umlaut) and to be able to restrict two-level rules to certain morphological contexts, we can provide two-level rules with *filters* connecting the two-level rules with the unification grammar.

Two different morphological components incorporating these ideas have been implemented. A first implementation (Trost, 1990) was based on Bear (1988a, 1988b). A short overview of this implementation that follows will give us an opportunity to look at the problems of using rule filters in more detail. The rest of this section describes in detail the second implementation of the hybrid morphological component where the two-level part comes closer to the original system of Koskenniemi.

A First Implementation

In a first implementation, I followed the approach of Beer (1988a, 1988b). Rules are not translated into finite state transducers but are interpreted directly. A slightly different interpretation of the operators helped to avoid the problem with rule filters that Bear's system encounters. Rules can be either 'obligatory' (i.e., if the phonological context is present, the only possible mapping for the lexical character is defined by the substitution pair) or 'optional' (i.e., the rule may be applied if applicable; this is of course, equivalent to Bear's "allowed" operator).

A rule may be applied only if its filter unifies with the feature structure of

the morph actually processed (i.e., the morph that contains the lexical character of the substitution pair of the rule). This is a relevant specification which is missing in both Emele (1988) and Bear (1988b). It commits us to including the morph boundary in the lexical representation of the morphs. We can now always be sure which feature structure we have to check the filter against.

In generation, filter checking is easy because all the information is already available with the feature structures of the morphs. Before generating a surface character, unification of the filter of the licensing rule with the feature structure of the morph is attempted. In case of failure the application of that rule is blocked. With recognition the task is more difficult. At the time of the selection of the character pair, the morph will not yet be recognized. The filters must be stored and are checked against each candidate morph. Only those where unification succeeds may be considered as possible candidates (if the right context of the two-level rule applies too).

It can easily be seen that this procedure also handles the task of transmitting information from the two-level part to the word grammar. Filters contain information that is unified with the feature structures of the morphs. This influences the working of the unification grammar. Therefore, in recognition, the relevant information is transported from the two-level part to the grammar. In generation, on the other hand, filters are used to trigger those rules effecting nonconcatenative morphological phenomena.

Because filters are tested against the feature structure of the actual morph, they also make computation easier. While the use of diacritics to restrict the application of rules makes a wide context necessary, this does not hold for filters. There the propagation of the necessary information is performed by the unification process, and the need for contexts is reduced to true phonological cases.

We will now look at two examples from German morphology. The first one is concerned with umlaut. What we need to do is to mark all stems that may take umlaut. This is done by using a diacritic for the concerned (phonologically underspecified) stem vowels (say, e.g., *A* for an *a).* We then need a feature *[umlaut: +]* for all suffixes triggering umlaut, and *[umlaut:* -] for all others. The following two rules are needed:

A:a <=> _ ; optional

A:ä <=>_; obligatory; filter: *[umlaut +]*

The feature *[umlaut: +]* makes up the filter of the (obligatory) rule mapping lexical *A* to surface *a.* The (optional) default rule for *A:a* needs no filter. Note that we need no context at all for these rules.

Let's now take the plural form of *Garten* (garden) which is formed with

umlaut: *Garten.* The morph found in the lexicon is *gArten.* For generation, it is necessary to provide it with the feature *[umlaut: +]* to trigger the correct rule. This is done by the unification grammar which combines the stem *gArten* with the null morph making plural. The feature structure associated to the null morph contains *[umlaut: +]* which is unified with the feature structure of *gArten.* As a result, the rule for *A:ä* applies, yielding the surface string *gärten.* In recognition the rule unifies its filter with the features of the morph. This blocks all affixes in the unification grammar which have the feature *[umlaut: —]*, as for example, the singular morpheme.

As a second example, take schwa epenthesis. Let's look at the past form of *senden* (to send), which can be either *sendete* or *sandte.* The two lexical strings are *$send + t + e$* and *$sand + t + e$,* respectively. To generate the correct forms we need two rules for ' + ':

```
+ :0 <=> _ ; optional

+ :e<=>{d, t} _{s, t, n}; obligatory
```

These two rules would help to generate *sendete* correctly, but would also generate *sandete,* which is wrong. We need to prevent the application of the second rule by providing it with an appropriate filter which restricts its application to weak verb stems. Verb stems are marked in the lexicon with either *{paradigm : weak]* or [*paradigm : strong].* The appropriate filter is therefore *[paradigm : weak].* Now this rule can no longer be applied to *$sand + t + e$,* and application of the optional default rule yields *sandte.*

Although that implementation of the system does work, there were some drawbacks. It proved extremely difficult to directly interpret arbitrary regular expressions as contexts (most difficult were contexts of unbounded length). Also, the search through the tree of possible mappings can be computationally very costly. These considerations led to a complete reimplementation of the system which came closer to the original proposal in Koskenniemi (1983, 1984). The rest of this section is devoted to a thorough description of that new implementation.

Word Grammar and Morph Lexicon

The word grammar is the part of the morphological component that governs the rules for combining morphs to legal word forms. It relies on a lexicon of morphs containing all the necessary morphological and morphosyntactic information. As stated before, the lexicon is organized as a letter trie. At every node that represents a legal morph, information is stored in the form of an extended feature structure.

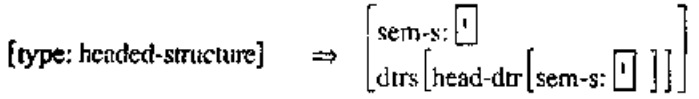
In our system, we use an approach that relies solely on feature structures for both the morph lexicon and the word grammar without using a context-free backbone for the rules of the latter. The distinction between grammar and lexicon is blurred in such an approach. This view of the grammar is influenced by head-driven phase structure grammar (HPSG) as described in Pollard and Sag (1987). There is a distinction in HPSG between grammar principles on the one hand and language specific grammar rules and lexical entries on the other hand. Principles are applied to every legal feature structure. They should capture information that is independent of the specific language and grammar. Language-specific rules and lexical entries are more idiosyncratic. The basic idea, then, is to apply principles in conjunction, and grammar rules and lexical entries in disjunction.

A prerequisite for such an approach is a feature unification formalism which can deal with disjunctice and implicative (or negative) feature structures. A number of algorithms have been proposed and some systems have been implemented that meet these requirements. Disjunctive feature structures were proposed quite early (e.g., Karttunen, 1984). A formal semantics was given by Kasper and Rounds (1986). First implementations based on the use of disjunctive normal form proved to be inefficient. Recently, algorithms have been proposed that promise to be much more efficient in the average case (Kasper, 1987; Eisele and Dörre, 1988). A new generation based on feature logic (Ait-Kaci, 1984) may perform even better (e.g., Eisele and Dorre, 1989).

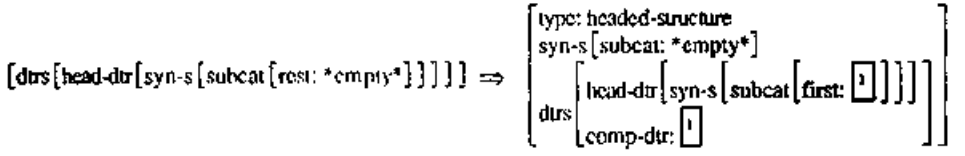
We use a feature unification formalism developed and implemented at the Austrian Research Institute for Artificial Intelligence (Matiasek, 1990), which builds on the algorithm proposed in Eisele and Dorre (1988) for handling disjunction. It also allows for implication (as proposed in Kasper, 1988), with a built-in modus-ponens inference rules. Using this feature unification system, we can write grammar principles in the form of implicative feature structures. In our morphosyntactic grammar there are very few principles needed. The most important one is a head feature convention (cf. Pollard and Sag, 1987, p. 58):

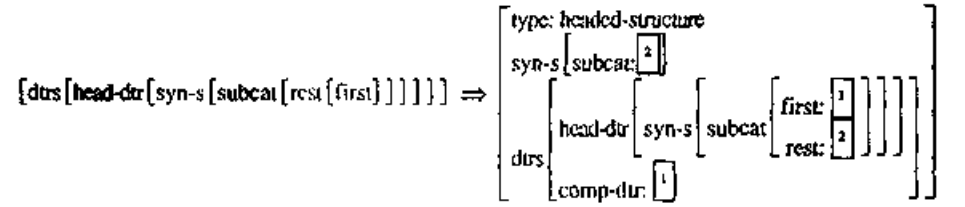
$$\text{[type: headed-structure]} \quad \Rightarrow \quad \begin{bmatrix} \text{syn-s} & [\text{head:}\boxed{1}] \\ \text{dtrs} & [\text{head-dtr}\,[\text{syn-s}\,[\text{head:}\boxed{1}]]] \end{bmatrix}$$

Head feature convention simply pushes the information up from the head daughter. Next there is a rudimentary form of the semantics principle. Currently its only task is to push root information up the projection path. For the semantic interpretation of compounds one would, of course, need a more elaborate form of that principle.

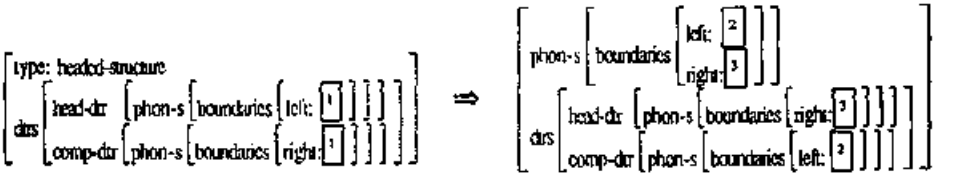


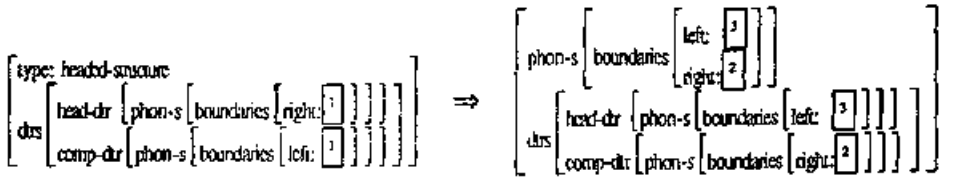
Projection is very simple. Some lexical entries subcategorize for other items. This is indicated by a nonempty subcategorization list (this is the equivalent of minimal bar level in Xbar theory). The subcategorization principle consists of projection rules that are responsible for creating a structure where the original structure is the head daughter, and the complement daughter is joined with the first element of its subcategorization list. The rest of the subcategorization list is just pushed "upward". One projection rule creates a headed structure with an empty subcategorization list. The complement daughter path is linked to the first element of the subcategorization list of the head daughter.





The last principle is concerned with the linear ordering of the constituents. There is one rule for left and another one for right complementation. The purpose of these rules is to get the boundary information correct.





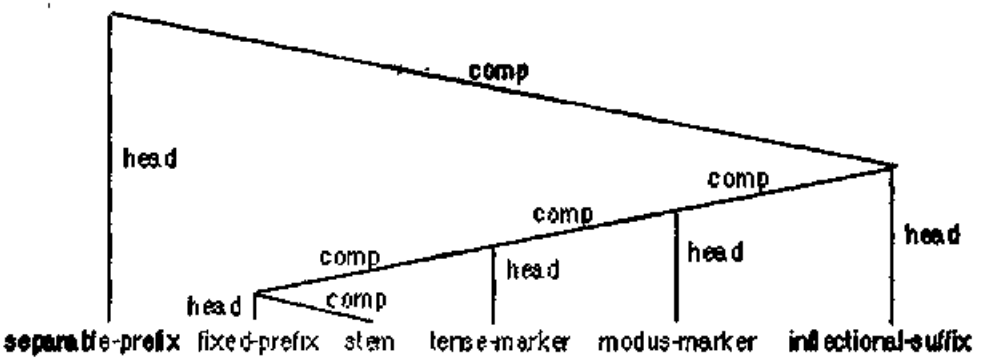


FIGURE 2. Maximal structure of finite verb forms.

These principles suffice for our purposes. It should be pointed out here why a word grammar is much simpler than a sentence grammar:

- There are no optional constituents.
- There is always exactly one complement. Instead of head and complement, we could as well use a simple functor-argument structure.
- The order of the constituents is fixed.
- Long-distance phenomena occur only in a very restricted form.

The lexicon contains an entry for every morph. I will shortly give some information about the structure of the entries in the current lexicon of German morphs. Note that the actual form of the lexicon entries must be viewed separately from the overall design of the morphological component which would of course also do well with a differently defined set of lexical entries and grammar rules.

Figure 2 shows the maximal structure for a finite verb form. The occurrence of prefixes is optional. This is no contradiction to the statement above that there are no optional constituents in word grammar because prefixes categorize for other constituents, not the other way round. All other constituents shown in Fig. 2 must occur, but adjacent ones may be collapsed into a single constituent. For example, all strong verbs have different present and past tense stems, that is stem and tense-maker are collapsed into a tensed stem.

With the exception of prefixes, heads are in the right position and complements in the left. It is unclear whether inflectional-suffix and modus-marker should not generally be collapsed into a single unit because the modus marker is consistently realized by the null morph. The *e* that appears in some subjunctive forms is best explained as schwa epenthesis (see "German Morphology and Morphonology," above). Finite and infinite verb forms would then have an

almost equal basic structure comprising prefixes, (tensed) stem, and modus-marker. Using that head-complement structure, nouns and adjectives can be described in an analogous way. Adjective stems combine with comparison marker and inflectional suffix, noun stems with number and case markers.

To provide for the necessary information, morphs must have information that is needed by the two level part, for example, an umlaut feature for endings. The grammar must also provide for means to share these features with other morphs where the information is needed. In the case of the umlaut feature this means the stem for which the ending subcategorizes.

Two-Level Part

Our two-level rule formalism follows quite closely Koskenniemi's (1984) proposal and the more detailed description of the KIMMO system given in Dalrymple et al. (1987). Rules are expressed in the same notation and have the same interpretation as proposed there. The important difference is the addition of filters, which will be described later. Rules are translated into transition tables of finite state transducers. Processing of the two-level rules is organized along the lines proposed in Barton et al. (1987, Chapter 6). The idea is not to run through the finite state automata but to use a local constraint algorithm (Waltz, 1975) instead.

A (very simplified) rule for schwa epenthesis in German verbs will help to demonstrate our approach. The morphonological rule is as follows: An *e* must be inserted between stems with final *d* or *t* and inflectionals starting with *s* or *t*. Such a rule can be easily expressed in the formalism:

$$+:e <=> \{d:dt:t\}\_\{s:st:t\}; \qquad (9)$$

Let's assume that this is our only rule and that the lexical alphabet comprises all the regular German characters plus the diacritics ' +' and *'$'* with their usual interpretation. The surface alphabet comprises all German characters too, plus the '0'. Allowed character pairs are all pairs where a lexical character is mapped to its direct surface counterpart (i.e., *a:a, b:b,* . . .) plus the pairs *+ :0, + :e,* and *$:0.*

Now our system containing rule (9) and the alphabet of pairs defined above is applied to present tense forms of the two German verbs *sagen* (to say) and *senden* (to send):

lexical level: $ s a g + e $ $ s a g + s t $ $ s e n d + e $ $ s e n d + s t $

surface level: 0 s a g 0 e 0 0 s a g 0 s t 0 0 s e n d 0 e 0 0 s e n d e s t 0

(10)

As expected, the *e* is inserted only in case of the verb form *sendest* (you send). In all other forms, rule (9) did block and the default pair *+ :0* was taken instead. No other pairings could be found for either the lexical or the surface forms than the ones given in (10). We will now expand our example to past tense forms of the same verbs. Past tense of regular verbs (like the ones from our example) form past tense by adding a t to their stem, for example, present tense *sage (I* say) has the corresponding past tense *sagte* (I said).

lexical level: $ s a g + t + e $ $ s e n d + t + e $ $ s e n d + t + s t $

surface level: O s a g O t O e O O s e n d e t O e O O s e n d e t e s t O

(11)

Again, the rule functions just the way we expect it to. An *e* is inserted between the stem *send* and the affix *t,* but not between *sag* and the *t.* Again, no other pairings would fit. But *senden* is one of those few regular verbs that come with two different stems for past tense. The second stem is *sand.* Applying our rules to this stem leads to the pairings:

lexical level: $ s a n d + t + e $ $ s a n d + s t $

surface level: O s a n d e t O e O O s a n d e t e s t O ' '

Unfortunately, the surface forms are wrong this time. Instead of *sandete* and *sandetest* one expects *sandte* and *sandiest.* That means that the epenthesis rule should not apply in this case. Clearly, there is no morphonological reason for this. The behavior is due to the word class, because all other verbs in this class (e.g., *wenderi)* exhibit the same behavior. What we do need is a means of restricting the application of two-level rules to certain classes of words. This is done by providing rules with filters. In our example, that means attaching information about the paradigm to all stems. The feature structure *[paradigm: strong]* is attached to the lexical entry of all past tense stems behaving like *sand.* We must then attach the filter *[paradigm: weak]* to rule (9), resulting in a new rule:

+ :e & {d:d t:t}___{s:s t:t}, *[paradigm: weak];* (13)

Before applying this new rule (13), we must now unify its filter with the feature structure associated with the affected morph. If unification is successful, the rule must be applied; if it fails, this rule must not be considered. The only problem that remains is to decide which morph is affected An intuitive solution is to say

that the affected morph is the one to which the lexical character belongs whose substitution is governed by the rule.

In our example, that lexical character is the '+.' Up to now, we treated morph boundaries as not belonging to any morph. There are two solutions to this. One is to add the morph boundary to all prefixes and affixes (i.e., *sand + t + st* is made up of the morphs *sand, +t* and *+st).* The other solution is to assume that filters that are applied to '+' must be checked against either the preceding or the following morph. Adding boundaries to the morphs has the additional benefit that it helps to discriminate between stems, prefixes, and suffixes. Therefore this solution is the one adopted in my system.

Implementing the Rules

Because of efficiency considerations, we did not want to interpret rules directly. Instead, we are translating them into transducer tables. These tables are used to apply a local constraint mechanism (Waltz, 1975). We first show informally the construction of these tables from the rules. At the moment, this rule compilation process is done manually. A compiler is under development that will eventually perform this task automatically.

Going back to rule (13), we start by defining equivalence classes over our pair alphabet with respect to the rule. One class is clearly formed by the substitution pair *+:e.* A next class contains the set of all other pairings for lexical '+,' which is in our example only the pair +:0. As a next step, we are looking at the pairs occurring in the contexts. The left context is a simple set of character pairs. It forms the class {d:d t:t}. Analogously, the right context forms the class {s:s t:t}. Next, we exclude all pairs that are in the intersection of these sets, namely t:t. This pair must be excluded from both sets and forms a set of its own. This leaves us with the three classes: {s:s}, {d:d}, and {t:t}. The remaining pairs of the alphabet form together one more class. This class is designated by =: =. On the basis of these equivalence classes, we can now construct our transducer. Transition table (14) describes a transducer with four states which is a translation of rule 9.

	1.	2.	3	4.
+:e	-	3	-	-
+:0	1	4	-	1
d:d	2	2	-	2

t:t	2	2	2	-	
=:=	1	1	-	1	(14) (*Cont.*)

The header line of diagram (14) denotes the four states. Periods following state numbers indicate a terminal state. The columns give the successor states for each transition. Dashes indicate that no transition starting from that node is possible for that pair.

Let's now come back to our example (11). We must make a distinction between recognition and generation. We will start with generation because it is easier to describe. The input word shall be the lexical string $send + t + st$. As a first step in our algorithm, we now search through the alphabet of pairs and find for each lexical character possible surface characters. For most of our lexical characters, we come up with exactly one pair. For '+' we find the two pairs *+:0* and *+:e*. This yields the following structure:

lexical level:	$ s e n d + t + s t $
surface level:	O s e n d G e t O e s t O

(15)

Next, we have to activate the appropriate subset from our rules. The selection criterion is the set of pairs used in (15). Rules associated to any of these pairs are to be activated. For all activated rules with filters we must check for applicability. This check is done against the feature structure of the morph to which the pair that led to the activation of the rule belongs. In our example, the only selected rule is (13). Since rule (13) has an associated filter, we have to check whether it may be applied. For that purpose, we must examine the relevant morphs, which are *+t* and *+st*. The feature structures of both unify with the filter; therefore, the rule must be applied with both occurrences of lexical '+.' We must now look through the transition table and write down all the possible transitions for every pair (this has to be done for every applicable rule). We also have to write down the start symbol in front of the first pair and the terminal nodes after the last one.

Start node:	1
$ 0	1-1 2-1 4-1
s s	1-1 2-1 3-1
e e	1-1 2-1 4-1

n	n	1-1 2-1 4-1	
d	d	1-2 2-2 4-2	
+	0	1-1 2-4 4-1	
	e	2-3	
t	t	1-2 2-2 3-2	(16)
+	0	1-1 2-4 4-1	(Cont.)
	e	2-3	
s	s	1-1 2-1 3-1	
t	t	1-2 2-2 3-2	
$	0	1-1 2-1 4-1	
Terminal nodes:		1 2 4	

We can now apply our constraint mechanism. It is searching for contingent paths from the start node to one of the terminal nodes. It starts from the left, canceling out all transitions that have no fitting predecessor. The start node is 1. At the next position all transitions starting with a node different from 1 must be eliminated. This leaves us with transition 1-1. The only continuation node is 1 again. We now have to apply this procedure iteratively until the terminal position is reached.

If any transitions have been eliminated in the course of this procedure, we must check whether the elimination of a successor has left a predecessor without a continuation. Therefore, we have to go backward again, applying the same strategy. Then the procedure goes forward again, and so on, until finally no more canceling takes place, ending with the situation described in (17). If a character pair has no more transition attached, it must be excluded from the list of possibilities. If that leaves a position in the lexical string without any mapping surface character, no legal mapping exists.

Start node:		1	
$	0	1-1	(17)

s	s	1-1	
e	e	. 1 - 1	
n	n	1-1	
d	d	1-2	
+	0		
	e	2-3	
t	t	3-2	
+	o	-	(17) *(Cont.)*
	e	2-3	
s	s	3-1	
t	t	1-2	
$	0	2-1	
Terminal nodes:		1	

The solution found is correct. Both ambiguities were resolved because in both positions the pair *+ :0* has no more transition attached to it. All other pairs are still valid mappings. This leaves the string *sendetest* as result.

The whole procedure becomes slightly more complicated if more than one rule is involved. We must then transmit to all other rules the fact that a rule has sorted out a possible pair. This is quite simple. We must cancel all possible transitions at that position in all other rules. Then we apply our local constraint algorithm again.

Let's now turn to an example where rule application should be blocked by the filter. As we know, this is the case with the lexical string *$sand + t + st$*. Testing the rule filter with the morphs again, we find that it should be applied in the second, but not the first case. How are we to proceed? The solution that springs to mind first is to omit the rule altogether if the filter block obviously doesn't work, because then we would get no epenthesis in the second position as well.

Start node:		1				1
$	0	1-1	2-1	4-1	$	0 1-1
s	s	1-1	2-1	3-1	s	s 1-1
a	a	1-1	2-1	4-1	a	a 1-1
n	n	1-1	2-1	4-1	n	n 1-1
d	d	1-2	2-2	4-2	d	d 1-2
+	0	1-1	2-4	4-1	+	0 -
	e	2-3				e 2-3
t	t	1-2	2-2	3-2	t	t 3-2
+	0	1-1	2-4	4-1	+	0 -
	e	2-3				e 2-3
s	s	1-1	2-1	3-1	s	s 3-1
t	t	1-2	2-2	3-2	t	t 1-2
$	0	1-1	2-1	4-1	$	0 2-1
Terminal nodes:		1	2	4		1

,10,

What we really need is to locally change the automaton. That can be done by eliminating the transition for + :e. Unfortunately this is not sufficient. Now the rule would block altogether, because we have designed the automaton in a such a way as to prevent it from allowing the default pair if the specified context is present. We must now enable the automaton to continue from the state that it reaches by processing + :0 with those continuations which are normally allowed only as continuations of + :e. This leads to the following automaton:

	1.	2.	3	4.
+:e	-	-	-	-

(19)

+ :0	**1**	**4**		**1**
d:d	**2**	**2**		**2**
s:s	**1**	**1**	**1**	**1**
t:t	**2**	**2**	**2**	**2**
= . =	**1**	**1**		**1**

(19) *(Cont.)*

In this table, state 3 is useless because it cannot be reached anymore. Instead we can continue with s:s and t:t from state 4. That is exactly what we want in the situation where the filter blocks. We must now find a way to simulate this sort of behavior just in the positions where it is needed.

In our constraint algorithm, we can simulate that behavior with some minor adaptations. The column corresponding to + :e must be eliminated. Now the only possible path is via the alternative + :0. But there would be no continuations. We now check for the next pair(s). In our example this is the t:t. We then insert the path 4-2 at that position. Now the algorithm will find a contingent path while correctly blocking the rule.

Start node:	1		1
$ 0	1-1 2-1 4-1	$ 0	1-1
s s	1-1 2-1 3-1	s s	1-1
a a	1-1 2-1 4-1	a a	1-1
n n	1-1 2-1 4-1	n n	1-1
d d	1-2 2-2 4-2	d d	1-2
+ 0	1-1 2-4 4-1	+ 0	2-4
e	-	e	-
t t	1-2 2-2 3-2 4-2	t t	4-2
+ 0	1-1 2-4 4-1	+ 0	-

(20)

	e	2-3			e	2-3	
s	s	1-1	2-1	3-1	s	s	3-1
t	**t**	**1-2**	**2-2**	**3-2**	**t**	**t**	**1-2**
$	0	1-1	2-1	4-1	$	0	2-1
Terminal nodes:		1	2	4			1

(20) *(Cont)*

Let us now reconsider the algorithm for effecting the local change in the behavior of the automaton. First, we must always be able to know to which context of the rule the filter is attached. Therefore we must always have a uniquely identifiable edge licensing the substitution under a certain context. We may never collapse two such edges. Instead, we would accept a nondeterminism that poses no problem for the constraint algorithm. Second, we must know the set of edges that must be inserted into the automaton to allow it to accept the default pair (or any other pairing) at the respective position. That set can easily be calculated. It is the set of edges leaving the node where the substitution edge ends, but we have to change the start node into the node where the edge of the other pair ends.

Looking at (14) we see that the substitution edge (for the pair + :e) is 2-3. The only alternative edge (for the pair + :0) is 2-4. The set of edges starting from node 3 is 3-1 (for s:s) and 3-2 (for t:t). That makes the set of replacement edges 4-1 (for s:s) and 4-2 (for t:t). It is important to remember the label (pair) of the edge. We can now describe the final form of our automaton:

	1.	2.	3	4.
+ :e	–	3	–	–
+ :0	1	4	–	1
d:d	2	2	–	2
s:s	1	1	1	–
t:t	2	2	2	
	1	1		1

(21)

(filter: [paradigm: weak] substitution: 2-3 replacement: (s:s 4-l)(t:t 4-2))

This automaton is completely equivalent to rule (13). Rules with more than one context may of course have more filters and associated edge sets. The actual insertion of the replacement edge(s) must be done according to the character pair actually present in the string pair processed [e.g., edge 4-2 in (20)].

Up to now we have taken the feature unification part for granted. We will now have a closer look and describe how it works. We will then finally be able to show how the interaction between these two parts is organized.

Processing of the Word Grammar

The implementation of the feature unification part was designed in a bidirectional way to fit with the two-level part. It was developed in the spirit of the approach proposed in Shieber et al. (1989). Since morphology is inherently less complex than syntax on the sentence level, the system could be kept quite simple.

Central to the processing are the ideas expressed in HPSG (Polland and Sag, 1987) where all grammatical and lexical information comes in the form of extended feature structures. Ideally, processing is extremely simple. To every evolving structure, the conjunction of the grammar principles and the disjunction of the lexicon is applied. The projection goes along head and complement daughter paths. Also, if two structures are to be combined, their phonological (graphemic) strings must be in adjacent position. Projection goes along head daughters. The structures that are subcategorized become complement daughters. Every structure that is not lexical contains exactly one head and one complement daughter. Instead of the notion of head and complement, we could have also used the terms functor and argument. What distinguishes our approach from categorial grammar, though, is the use of rich feature structure instead of using just categories.

Generation

Input to the generation is made up of the root form of the verb plus a number of syntactic features. From a certain point of view, this root form can be regarded as similar to the semantic description of a sentence that is input into a sentence generator. A root form may actually be a list of simpler root forms to describe a compound word, whereas lexical entries (which represent single morphs) by definition contain only simple root forms. [Morphs that do not contribute to the semantics of a word form (i.e., endings) have no root form and do therefore not show up in the input root form.] An example for an input to the generator is (22) which should be generated either as *$send + t + st$* or *$sand*

+ *t* + *st$*. These forms would then be input to the two-level part which should map them to the surface strings *sendetest* and *sandtest.*

```
[ [sem-s [root [first: send] ] ]
  [syn-s [head [cat: verb]
               [person: 2]
               [modus: indicative]
               [number: sg]
               [tempus: past]
               [well-formed: +] ] ] ]                    (22)
```

The generation part starts by selecting entries from the lexicon that are potentially applicable to the actual generation task. For this task, we select all entries that unify with the root information of the input structure (i.e., [[sem-s [root [first: send]]). In our example, this would yield all different stems (allomorphs) of *send* (i.e., the entries for the morphs *send* and *sand)* plus all the lexical entries with an empty root form. These are all lexical entries that do not contribute tot he root form of a word (e.g., inflectional particles). For the rest of the generation process, only this pool of lexical entries is regarded, while the remainder of the lexicon may safely be ignored.

As a first step, all those lexical entries are extracted from this pool that can be unified with the head features of the input because, according to the head feature principle, the input structure must be a maximal projection of a lexical entry with the same head structure (in cases where a word is made up of a single morph the lexical entry itself will already be that maximal projection).

We then build the actual structure in accordance with the projection rules and the grammar principles. The structure we get applying this procedure can then be unified with the original input. If we have created any complement daughter slots during this procedure, we must now fill them in. This is done by a recursive application of the process. We extract the head information from the complement daughter slot (which is shared with the subcategorization slot) and start the selection of lexical entries from our pool again. This procedure is repeated until no more open complement daughters are left.

In the case of our example, we would in the first step select the lexical entry for the morph '+st'. This entry unified with the head features of the input structure gives

```
[[sem-s *pointer* syn-s subcat first sem-s]
 [syn-s [head [cat: v]
        [modus: indicative]
        [number: sg]
        [paradigma *pointer* syn-s subcat first syn-s head paradigma]
        [person: 2]
        [tempus: past]
        [umlaut ^pointer* syn-s subcat first syn-s head umlaut]
        [well-formed:  +]]


    [subcat [first [syn-s [head [cat: v]


                                  [modus: modus-]
                                  [tempus
                                   *pointer* syn-s subcat first syn-s head tempus]
                                  [well-formed: -]]
                                  [prefix [trenn: -]]
                           [subcat: *empty*]]]
          [rest: *empty*]]]
 dtrs: *empty*]]                                                    (23)
```

ic subcategorization list contains one element (according to Fig. 2, a tensed rb stem). Application of the projection principle leads to a structure where 3) becomes the head daughter. Because the subcategorization list of that new ucture is empty, the maximal projection level is reached. At this point, (22) is ified with that structure. We have now one complement daughter left to fill, king the head features attached to the complement daughter feature, we search

the pool of relevant lexical entries. This search delivers the feature structure of the tense marker ' + t'.

That feature structure in turn has a one-element subcategorization list. Recursive application of the process gets two lexical entries for the morphs *send* and *sand*. These two have both an empty subcategorization list. Therefore, the recursion stops here and we get the expected solutions *$send + t + st$* or *$sand + t + st$* which can be given to the two-level part to generate the corresponding surface forms.

Analysis

Basically, analysis works the same way as generation. For efficiency, a chart is used where the lexical entries are entered. This helps in two ways. First, it quite naturally defines the part of the lexicon that has to be considered, and second, it makes the ordering constraints explicit.

One remark about a difference between analysis and generation: While for the generation process, every fully expanded structure constitutes a legal solution, the same is not true for analysis. The criterion of having reached a maximal projection (i.e., a structure with empty subcategorization list) does not entail that this structure is a legal word form. We need to explicitly mark forms as being regular words of the language and not just parts of complete word forms.

Let's look at the string *$sag + t$*, for example. This string consists of the two morphs *sag* and +t which are input to the parser. The former is a regular verb stem which means it may function either as untensed stem or as a present tense stem. The latter can be either an inflectional ending or a derivational suffix transforming untensed stems into past tense stems. If we combine the reading present tense stem with the inflectional reading of +t we arrive at a correct word form, namely *present tense third-person singular*. Taking the other possible combination, we arrive at a past tense stem. This structure is a maximal projection, too. But in itself it is no legal word form because number/person information is missing. What we need is the equivalent of the start symbol in a more conventional grammar. Our solution is to add a head feature *[well-formed: +]* to all lexical entries the maximal projection of which will qualify as legal word forms, *[well-formed: -]* to all others. Solutions are only those spanning edges containing the feature *[well-formed: +]*.

Let's now return to our example from generation. Suppose the two-level part comes up with the lexical string *$sand + t + st$* which consists of the sequence of morphs *sand*, *+ t*, and *+ st*. The input to the parser is a chart where the lexical entries corresponding to these morphs are entered (these structures possibly are already altered because of filter application in the two-level part). The chart also contains edges for all possible occurrences of null morphs.

To all entries with a nonempty subcategorization list, the projection principles are applied (this can be viewed as a prediction step). The grammar principles are applied to the resulting structures which are then merged with neighboring structures (the completion step) whenever possible (i.e., if these structures have an empty subcategorization list and fit into the complement daughter). This procedure continues until no more progress can be made. All spanning structures with the feature *[well-formed: +]* are legal solutions.

Interaction between the Two Parts

We are now in the position to describe the interaction between the word grammar and the two-level part more closely. Let us examine once again what the rule filters are used for technically. They are a means to block the application of a rule on morphological grounds, although it would be applicable because of the phonological (orthographic) context. (The transmission of information between word grammar and two-level rules arises as a by-product which needs no special effort.) Inhibiting the application of a rule allows other pairings (most notably the default pairing) to be applied again.

One can also view this from the other side. Application of the default pairing implies that no rule with that lexical character in its substitution pair is applicable. If the phonological context precludes that, then no further action has to be taken. Otherwise the rule filters must be taken into consideration. A further complication is that filters are attached to single contexts rather than a rule as a whole. Therefore, the interaction of different contexts must also be regarded.

Interaction between the two components of our morphology is slightly different for generation and recognition. With generation the interaction is rather simple. The word form generator produces a fully expanded feature structure from its input. From this structure—with its implicit ordering constraints—a list of the morphs is created. To each morph, the relevant part of the feature structure is associated. This list is given to the two-level part.

From that point on, processing continues as described above under "Implementing the Rules." Whenever it turns out that a rule filter fails to unify with the relevant feature structure, the local change procedure is applied. The substitution edge is eliminated, replacement edges are inserted. Besides that, processing continues the regular way.

For recognition, the task is more subtle. Here processing starts with the two-level part. Rules without filters function the usual way. Whenever a filter rule is applied (i.e., the relevant substitution edge has been consumed), we must store the filter. Of course, only the filters of those contexts that are actually (or potentially, if the right context is still open) present are relevant. As soon as a

morph is found in the lexicon, we must check our filters. For every filter there are three possible results:

- Unification fails. The substitution edge must be removed and the replacement edges inserted. If there leaves no contingent path for that rule, the morph must be discarded.
- The filter subsumes the feature structure of the morph. In this case, we are safe and the rule must be applied.
- The filter unifies with the feature structure of the morph but does not subsume it. In this case, we do not know if the parsing process will add features and values that may lead to a feature clash. We can apply the rule anyway because if that happens the parser will discard that solution. A problem here is that another context of the rule may be applicable the filter of which would still unify. We guarantee a correct solution in the following way: a disjunctive filter structure is built from the filters of all applicable contexts of the rule.

We must now examine the situation where we assume the nonapplicability of a rule. If we use a default pair, all the rules attached to the lexical character of that pair become active. If none of these rules blocks because of the phonological context, everything is okay. If a rule blocks, the phonological context would require the surface character licensed by the rule instead of the one actually present. In that case the applicability of the filter must be checked. Again there are three possible results:

- Unification fails, in which case we may safely ignore the rule.
- The filter subsumes the feature structure of the morph, which means the rule should have been applied and we must discard the morph we are currently working on.
- The filter unifies with the feature structure of the morph but does not subsume it. Again we are in a situation where no final decision is possible. At this point, the filter applies, but it might be the case that failure comes later on because of additional information supplied by the word grammar.

This last case is the only really problematic one because we can hold up our assumption only if we take for granted that a failure will happen in the future course of processing. What we do is to unify the feature structure of the morph with a negation of the filter. This leads to correct results if we can be sure that the filter will be approved or disproved in the final feature structure of the word form. Therefore, we have to pose this restriction on the form of the filters with resnect to the word grammar.

APPLICATIONS

I have so far presented a morphological component that recognizes and generates word forms based on a morph lexicon, a word grammar, and a set of two-level rules. It seems appropriate at this point to show potential applications of such a system in practical environments. An advantage of our component is that it is up to the system's designer if the component is used to deal only with inflection—like conventional morphological modules for natural language understanding systems—or also with some subset of derivation. Promising areas are scientific and technical languages, and especially medical language.

In medical language, we find a large set of complex derivatives and compounds with clear compositional semantics. Instead of including all these words into the lexicon, one can instead analyze them with the use of our component. Since semantic interpretation follows a compositional pattern, one can use the same mechanism for semantic interpretation as on the sentence level. Such an approach would also make sure that, whenever a new technical expression is entered into the lexicon, the full range of possible derivational and compound forms is available. This facilitates knowledge acquisition and guarantees consistency.

The range of possible applications goes from the (partial) analysis of unrestricted natural language text to less complex retrieval systems. For example, the system WAREL (Dorda, 1990) retrieves data from a large base of unformatted texts on the basis of keywords a user enters. A keyword may occur in different inflected forms, be used as the basis for a derived word, and so on. The described morphological component could be used to create all legal forms in which that keyword could appear.

CONCLUSION

In this article I described a system for the morphological recognition and generation of words that combines two-level rules with a unification-based word grammar. Instead of just substituting continuation classes with feature structures, I propose a different solution. Filters in the form of feature structures are added to the two-level rules. These filters fulfill two different purposes. They restrict the application of two-level rules to certain classes of morphs, thereby eliminating the need to encode morphological information with diacritics on the lexical level. At the same time, they allow the transfer of morphological information from the two-level rules to the grammar component. That way aspects of nonconcatenative morphology like umlaut can be handled in a linguistically satisfying way.

In the design of the two-level part, I adhered to the original notation of Koskenniemi (1983, 1984[8]). The imolementation though is different. T use a local

constraint algorithm as proposed by Barton et al. (1987, chapter 6) which enables the integration of the processing of the filters into the algorithm. Sources of possible problems with these filters are discussed and solutions are shown. The word grammar is represented in the form of feature structures without a context-free backbone. Further work will be necessary to restrict the computational power of that part in a linguistically motivated way. The approach has been successfully applied to the inflectional morphology and some aspects of the derivational morphology of German. It remains to be shown that it can be applied to the whole of derivational morphology including compounding.

ACKNOWLEDGMENT

I would like to thank my colleagues Ralph Flassig, Hans Haugeneder, Wolfgang Heinz, John Nerbonne, Günter Neumann, and Hans Uszkoreit for fruitful discussions and comments on earlier versions of this paper. Work described in this paper was begun while I worked at the Department of Medical Cybernetics and Artificial Intelligence at the University of Vienna, Austria.

REFERENCES

Ait Kaci, H 1984 A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures PhD thesis, University of Pennsylvania

Ballard, B W 1986 User Specification of Syntactic Case Frames in TELI, A Transportable, User Customized Natural Language Processor *Proc 11th COLING*, Bonn, Germany

Barton, G E 1986 Constraint Propagation in KIMMO Systems *Proc 24th ACL Conf*, New York

Barton, G E , Berwick, R C , and Ristad, E S 1987 *Computational Complexity and Natural Language*, Cambridge, Mass MIT Press

Bear, J 1986 A Morphological Recognizer with Syntactic and Phonological Rules, *Proc 11th COLING*, Bonn, Germany

Bear, J 1988a Generation and Recognition of Inflectional Morphology in Trost H (ed), *Proc 4 Oesterreichische Artificial-Intelligence Tagung*, ed H Trost, Berlin Springer-Verlag

Bear, J 1988b Morphology and Two level Rules and Negative Rule Features *Proc 12th COLING*, pp 28-32, Budapest

Black, A W, Ritchie, G D , Pulman, S G , and Russell, G J 1987 Formalisms for Morphographemic Description *Proc 3rd European ACL* pp 11-18, Kopenhagen

Calder, J 1989 Paradigmatic Morphology *Proc 4th European ACL*, pp 58-65, Manchester

Carson, J 1988 Unification and Transduction in Computational Phonology *Proc COLLING-88*, pp 106-111, Budapest

Dalrymple, M , Kaplan, R M , Karttunen, L , Koskenmemi, K , Shaio, S , and Wescoat, M 1987 Tools for Morphological Analysis, Center tor the Study of Language and Information, Stanford University, Report No CSLI 87-103, Stanford, Calif

Dorda, W 1990 Data-Screening and Retrieval of Medical Data by the System WAREL *Methods Inf Med* , 293-11

Emele, M 1988 Uberlegungen zu emer Two-Level Morphologic fur das Deutsche In *Proc 4 Oesterreichische Artificial Inteligence-Tagung*, ed H Trost, pp 156-163 Berlin Springer Verlag

Eisele, A , and Dorre, J 1988 Unification of disjunctive feature descriptions *Proc 26th ACL Conf,* pp 286-294, Buffalo, N Y

Eisele, A , and Dorre, J 1989 Determining Consistency of Feature Terms with Distributed Disjunctions *Proc GWAI-89* Berlin Sprmger-Verlag

Evans, R , and Gazdar, G 1989a The Semantics of DATR In *Proc of AISB-89, ed* A Cohn London Pitman

Evans, R , and Gazdar, G 1989b Inference in DATR *Proc 4th European ACL,* pp 66-71, Manchester

Finkler W, and Neumann, G 1988 MORPHIX—A Fast Realization of a Classification Based Approach to Morphology *In Proc 4 Österreichische Artificial-Intelligence-Tagung* ed H Trost, pp 11-19, Berlin Springer Verlag

Fredkm, E 1960 Trie Memory *Communication* ACM.* 3 490-499

Giegench, H 1987 Zur Schwa Epenthese im Standarddeutschen *Linguist Ber,* 112449-469

Gorz, G , and Paulus, D 1988 A Finite State Approach to German Verb Morphology *Proceedings COLING 88* pp 212-215, Budapest

Issatschenko, A 1974 Das "schwa mobile" und "schwa constans" im Deutschen In *Sprachsystem und Sprachgebrauch Festschrift fur Hugo Moser zum 65 Geburtstag,* ed U Engel and P Grebe, pp 142-171 Dusseldorf Schwann

Rasper, R 1987 A Unification Method for Disjunctive Feature Descriptions *Proc 25th ACL Conf,* pp 235 242, Stanford, Calif

Kasper, R 1988 Conditional Description in Functional Unification Grammar Proc *25th ACL,* pp 233-240 Buffalo, N Y

Kasper, R , and Rounds W 1986 A Logical Semantics for Feature Structures *Proc 24th ACL, pp* 257-266, New York

Karttunen, L 1983 KIMMO A General Morphological Processor *Tex Linguist Forum,* 22 167 186

Karttunen, L 1984 Features and Values *Proc 10th COLING* Stanford, Calif

Karttunen, L , and Wittenburg, K 1983 A Two-level Morphological Analysis of English *Tex Linguis Forum,* 22 217-228

Kataja, L , and Koskennierm, K 1988 Finite-State Description of Semitic Morphology A Case Study of Ancient Accadian *Proc 12th COLING,* pp 313-315, Budapest

Kay, M 1987 Nonconcatenative Finite-State Morphology *Proc 3rd European ACL* pp 2-10, Kopenhagen

Koch S , Kustner A , Menzel, W, and Rudiger, B 1989 Heuristic Morphological Analysis of German *Prague Bull Math Linguist* , 52 5-24

Koskennierm, K 1983 Two-Level Model for Morphological Analysis *Proc 8th IJCAI,* Los Altos, Calif Morgan Kaufmann

Koskennierm, K 1984 A General Computational Model for Word-Form Recognition and Production *Proc 10th COLLING,* Stanford, Calif

Koskennierm, K 1985 An Application of the Two level Model to Finnish In *Computational Morphos\ntax,* pub no 13 ed F Karlsson Helsinki Dept of Linguistics, University of Helsinki, pp 19-42

Koskennierm, K , and Church, K W 1988 Complexity, Two-Level Morphology and Finnish *Proc 12th COLING* pp 335-340, Budapest

Matiasek, J 1990 FUN *An Extended Feature Unification Formalism* Austrian Research Institute for Artificial Intelligence TR-90-10, Vienna, Austria

Polland, C J , and Sag, I A 1987 *Information Based Syntax and Semantics,* vol 1 Fundamentals, CSLI Lecture Notes No 13 Chicago University of Chicago Press

Pulman, S G , Russell, G J , Ritchie, G D , and Black, A W 1988 Computational Morphology of English *Linguistics* 26 545-560

Renmson, J 1980 What Is schwa in Austrian German? The Case for Epenthesis and its Consequences *Wien Ling Gaz* 24 33-42

Ritchie, G D 1989 On the Generative Power of Two-Level Morphological Rules *Proc 4th European ACL Conf,* pp 51-57, Manchester

Ritchie, G , Pulman, S , Black, A , and Russell, G 1987 A Computational Framework for Lexical Description *Computat Linguist* 3 4(13) 290-307

Ruessmk, H 1989 *Two-Level Formalisms, Working Papers in Natural Language Processing 5* Rijksumversiteit Utrecht

Schane, S 1973 *Generative Phonology* Englewood Cliffs, N J Prentice-Hall

Schiller, A , and Steffens, P 1990 *A Two-Level Morphology for a German Natural Language Understanding System* Stuttgart IBM (unpublished)

Shieber, S 1986 *An Introduction to Unification-Bailed Approaches to Grammar* CSLI Lecture Notes No 4, Chicago University of Chicago Press

Shieber, S 1988 A Uniform Architecture for Parsing and Generation *Proc 12th COLING,* pp 614-619, Budapest

Shieber, S , van Noord, G , Moore, R , and Pereira, F 1989 A Semantic Head-Driven Generation Algorithm for Unification-Based Formalisms *Proc 27th ACL Conf* pp 7-17, Vancouver, B C

Smolka, G 1988 Feature logic with Subsorts *Proc Workshop Unification Formalisms Syntax, Semantics, and Implementation* Titisee, Germany

Trost, H 1990 The application of two-level morphology to non-concatenative German morphology *Proc COLING-90,* Helsinki

Trost, H , and Buchberger, E 1986 Towards the Automatic Acquisition of Lexical Data *Proc IIth COLING,* Bonne, FRG

Trost, H , and Dorffner, G 1987 A System for Morphological Analysis and Synthesis of German Texts In *New Developments in Computer-Assisted Language Learning,* ed D Hamline London Crooms Helm

Waltz, D 1975 Understanding Line Drawings of Scenes with Shadows In *Psychology of Computer Vision,* ed P Winston, pp 19-92 New York McGraw-Hill

Wiese, R 1988 Silbische und lexikalische Phonologic—Studien zum chmesischen und Deutschen, pp 140-175, Tubingen Niemeyer