FEATURE-BASED ALLOMORPHY*

Hans-Ulrich Krieger Hannes Pirker
German Research Center for
Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
W-66 Saarbrücken 11, Germany
{krieger,pirker}@dfki.uni-sb.de

John Nerbonne
Alfa Informatica, P.O.Box 716
Oude Kijk in 't Jatstraat 41
Rijksuniversiteit Groningen
NL 9700 AS Groningen, Holland
nerbonne@let.rug.nl

Abstract

Morphotactics and allomorphy are usually modeled in different components, leading to interface problems. To describe both uniformly, we define finite automata (FA) for allomorphy in the same feature description language used for morphotactics. Nonphonologically conditioned allomorphy is problematic in FA models but submits readily to treatment in a uniform formalism.

1 Background and Goals

ALLOMORPHY or MORPHOPHONEMICS describes the variation we find among the different forms of a morpheme. For instance, the German second person singular present ending *-st* has three different allomorphs, *-st, -est, -t*, determined by the stem it combines with:

	'say'	'pray'	'mix'
(1) 1sg pres ind	*sag+e*	*bet+e*	*mix+e*
2sg pres ind	*sag+st*	*bet+est*	*mix+t*
3sg pres ind	*sag+t*	*bet+et*	*mix+t*

MORPHOTACTICS describes the arrangement of morphs in words, including, e.g., the properties of *-st* that it is a suffix (and thus follows the stem it combines with), and that it combines with verbs. While allomorphy is normally described in finite automata (FA), morphotactics is generally described in syntax-oriented models, e.g., CFGs or feature-based grammars.

The present paper describes both allomorphy and morphotactics in a feature-based language like that of Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag 1987). The technical kernel of the paper is a feature-based definition of FA.[1] While it is unsurprising that the languages defined by FA may also be defined by feature description languages (FDL), our reduction goes beyond this, showing how the FA themselves may be defined. The significance of specifying the FA and not merely the language it generates is that it allows us to use FA technology in processing allomorphy, even while keeping the interface to other grammar components maximally transparent (i.e., there is NO interface—all linguistic information is specified via FDL).

Our motivation for exploring this application of typed feature logic is the opportunity it provides for integrating in a single descriptive formalism not only (i) allomorphic and morphotactic information but also (ii) concatenative and non-concatenative allomorphy. The latter is particularly useful when concatenative and non-concatenative allomorphy coexists in a single language, as it does, e.g., in German.

2 Finite Automata as Typed Feature Structures

An FA A is defined by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of STATES, Σ a finite INPUT ALPHABET, $\delta : Q \times \Sigma \rightarrow Q$ is the TRANSITION FUNCTION, $q_0 \in Q$ the INITIAL STATE, and $F \subseteq Q$ the set of FINAL STATES.[2] For reasons of simplicity and space, we only refer to the simplest form of FA, viz., DETERMINISTIC finite automata without ϵ-moves which consume exactly one input symbol at a time. This is of course not a restriction w.r.t. expressivity: given an arbitrary automaton, we can always construct a deterministic, equiva-

*This work was supported by research grant ITW 9002 0 from the German Bundesministerium für Forschung und Technologie to the DFKI DISCO project. We are grateful to an anonymous ACL reviewer for helpful comments.

[1] See Krieger 1993b for the details and several extensions.

[2] We assume a familiarity with automata theory (e.g., Hopcroft and Ullman 1979).

lent one which recognizes the same language (see Hopcroft and Ullman 1979). Fortunately, our approach is also capable of representing and processing directly non-deterministic FA with ϵ-moves and allows for edges which are multiple-symbol consumers.

Specifying an automaton in our approach means introducing for every state $q \in Q$ a possibly recursive feature type with the same name as q. We will call such a type a CONFIGURATION. Exactly the attributes **EDGE**, **NEXT**, and **INPUT** are appropriate for a configuration, where **EDGE** encodes disjunctively the outgoing edges of q, **NEXT** the successor states of q, and **INPUT** the symbols which remain on the input list when reaching q.[3] Note that a configuration does not model just a state of the automaton, but an entire description at a point in computation.

$$(2)\ \textit{proto-config} \equiv \begin{bmatrix} \textbf{EDGE}\ \textit{input-symb} \\ \textbf{NEXT}\ \textit{config} \\ \textbf{INPUT}\ \textit{list(input-symb)} \end{bmatrix}$$

We now define two natural subtypes of *proto-config*. The first one represents the non-final states $Q \setminus F$. Because we assume that exactly one input symbol is consumed every time an edge is taken, we are allowed to separate the input list into the first element and the rest list in order to structure-share the first element with **EDGE** (the consumed input symbol) and to pass the rest list one level deeper to the next state.

$$(3)\ \textit{non-final-config} \equiv \begin{bmatrix} \textit{proto-config} \\ \textbf{EDGE}\ \boxed{1} \\ \textbf{NEXT}|\textbf{INPUT}\ \boxed{2} \\ \textbf{INPUT}\ \langle\, \boxed{1} \cdot \boxed{2} \,\rangle \end{bmatrix}$$

The other subtype encodes the final states of F which possess no outgoing edges and therefore no successor states. To cope with this fact, we introduce a special subtype of $\top$, called *undef*, which is incompatible with every other type. In addition, successfully reaching a final state with no outgoing edge implies that the input list is empty.

$$(4)\qquad \textit{final-config} \equiv \begin{bmatrix} \textit{proto-config} \\ \textbf{EDGE}\ \textit{undef} \\ \textbf{NEXT}\ \textit{undef} \\ \textbf{INPUT}\ \langle\ \rangle \end{bmatrix}$$

[3]Note that **EDGE** is not restricted in bearing only atomic symbols, but can also be labeled with complex ones, i.e., with a possibly underspecified feature structure (for instance in the case of 2-level morphology—see below).

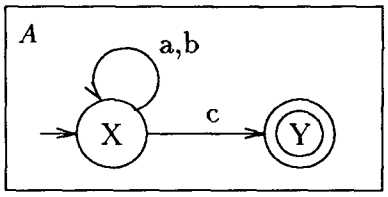


Figure 1: A finite automaton A recognizing the language $\mathcal{L}(A) = (a + b)^* c$.

Of course, there will be final states with outgoing edges, but such states are subtypes of the following DISJUNCTIVE type specification:

(5) *config* ≡ *non-final-config* ∨ *final-config*

To make the idea more concrete, let us study a very small example, viz., the FA A (see Figure 1). A consists of the two states X and Y, from which we define the types X and Y, where Y (7) is only an instantiation of *final-config*.

In order to depict the states perspicuously, we shall make use of DISTRIBUTED DISJUNCTIONS. Dörre and Eisele 1989 and Backofen et al. 1990 introduce distributed disjunctions because they (normally) allow more efficient processing of disjunctions, sometimes obviating the need to expand to disjunctive normal form. They add no expressive power to a feature formalism (assuming it has disjunction), but abbreviate some otherwise prolix disjunctions:

$$\begin{bmatrix} \textbf{PATH1}\ \{_{\$1}\ a \vee b\} \\ \textbf{PATH2}\ \{_{\$1}\ \alpha \vee \beta\} \\ \textbf{PATH3}\ [\ \ldots\ ] \end{bmatrix} =$$

$$\left\{ \begin{bmatrix} \textbf{PATH1}\ a \\ \textbf{PATH2}\ \alpha \\ \textbf{PATH3}\ [\ \ldots\ ] \end{bmatrix} \vee \begin{bmatrix} \textbf{PATH1}\ b \\ \textbf{PATH2}\ \beta \\ \textbf{PATH3}\ [\ \ldots\ ] \end{bmatrix} \right\}$$

The two disjunctions in the feature structure on the left bear the same name '$1', indicating that they are a single alternation. The sets of disjuncts named covary, taken in order. This may be seen in the right-hand side of the equivalence.[4]

We employ distributed disjunctions below (6) to capture the covariation between edges and

[4]Two of the advantages of distributed disjunctions may be seen in the artificial example above. First, co-varying but nonidentical elements can be identified as such, even if they occur remotely from one another in structure, and second, features structures are abbreviated. The amount of abbreviation depends on the number of distributed disjunctions, the lengths of the paths **PATH1** and **PATH2**, and—in at least some competing formalisms—on the size of the remaining structure (cf. **PATH3** [...] above).

their successor states: if a is taken, we must take the type X (and vice versa), if b is used, use again type X, but if c is chosen, choose the type Y.

$$X \equiv \begin{bmatrix} \textit{non-final-config} \\ \texttt{EDGE}\ {}_{\$1}\{\texttt{a} \vee \texttt{b} \vee \texttt{c}\} \\ \texttt{NEXT}\ {}_{\$1}\{X \vee X \vee Y\} \end{bmatrix} \tag{6}$$

$$Y \equiv [\,\textit{final-config}\,] \tag{7}$$

Whether an FA A ACCEPTS the input or not is equivalent in our approach to the question of FEATURE TERM CONSISTENCY: if we wish to know whether w (a list of input symbols) will be recognized by A, we must EXPAND the type which is associated with the initial state q_0 of A and say that its INPUT is w. Using the terminology of Carpenter 1992: (8) must be a TOTALLY WELL-TYPED feature structure.

$$\begin{bmatrix} q_0 \\ \texttt{INPUT}\ w \end{bmatrix} \tag{8}$$

Coming back to our example (see Figure 1), we might ask whether abc belongs to $\mathcal{L}(A)$. We can decide this question, by expanding the type X with [INPUT ⟨a,b,c⟩]. This will lead us to the following consistent feature structure which moreover represents, for free, the complete recognition history of abc, i.e., all its solutions in the FA.

(9)
$$\begin{bmatrix} X \\ \texttt{EDGE}\ \boxed{1}\ \texttt{a} \\ \texttt{NEXT} \begin{bmatrix} X \\ \texttt{EDGE}\ \boxed{3}\ \texttt{b} \\ \texttt{NEXT} \begin{bmatrix} X \\ \texttt{EDGE}\ \boxed{5}\ \texttt{c} \\ \texttt{NEXT} \begin{bmatrix} Y \\ \texttt{EDGE}\ \textit{undef} \\ \texttt{NEXT}\ \textit{undef} \\ \texttt{INPUT}\ \boxed{6}\ \langle\ \rangle \end{bmatrix} \\ \texttt{INPUT}\ \boxed{4}\ \langle \boxed{5} \cdot \boxed{6} \rangle \end{bmatrix} \\ \texttt{INPUT}\ \boxed{2}\ \langle \boxed{3} \cdot \boxed{4} \rangle \end{bmatrix} \\ \texttt{INPUT}\ \langle \boxed{1} \cdot \boxed{2} \rangle \end{bmatrix}$$

Note that this special form of type expansion will always terminate, either with a unification failure (A does not accept w) or with a fully expanded feature structure, representing a successful recognition. This idea leads us to the following ACCEPTANCE CRITERION:

$$w \in \mathcal{L}(A) \iff \exists \begin{bmatrix} q_0 \\ \texttt{INPUT}\ w \\ (\texttt{NEXT})^* \begin{bmatrix} f \\ \texttt{INPUT}\ \langle\ \rangle \end{bmatrix} \end{bmatrix} \neq \bot, \quad \text{where } f \in F \tag{10}$$

Notice too that the acceptance criterion does not need to be checked explicitly—it's only a logical specification of the conditions under which a word is accepted by an FA. Rather the effects of (10) are encoded in the type specifications of the states (subtypes of *final-config*, etc.).

Now that we have demonstrated the feature-based encoding of automata, we can abbreviate them, using regular expressions as "feature templates" to stand for the initial states of the automaton derived from them as above.[5] For example, we might write a feature specification [MORPH|FORM $(a + b)^*c$] to designate words of the form accepted by our example automaton.

As a nice by-product of our encoding technique, we can show that unification, disjunction, and negation in the underlying feature logic directly correspond to the intersection, union, and complementation of FA. Note that this statement can be easily proved when assuming a classical set-theoretical semantics for feature structures (e.g., Smolka 1988). To give the flavor of how this is accomplished, consider the two regular expressions $\mathcal{L}_1 = ab^*c$ and $\mathcal{L}_2 = a^*bc$. We model them via six types, one for each state of the automata. The initial state of $\mathcal{L}_1$ is A, that of $\mathcal{L}_2$ is X. The intersection of $\mathcal{L}_1$ and $\mathcal{L}_2$ is given by the unification of A and X. Unifying A and X leads to the following structure:

(11)
$$\begin{bmatrix} A \\ \texttt{EDGE a} \\ \texttt{NEXT}\ B \end{bmatrix} \wedge \begin{bmatrix} X \\ \texttt{EDGE}\ {}_{\$1}\{\texttt{a} \vee \texttt{b}\} \\ \texttt{NEXT}\ {}_{\$1}\{X \vee Y\} \end{bmatrix} = \begin{bmatrix} A \wedge X \\ \texttt{EDGE a} \\ \texttt{NEXT}\ B \wedge X \end{bmatrix}$$

Now, testing whether w belongs to $\mathcal{L}_1 \cap \mathcal{L}_2$ is equivalent to the satisfiability (consistency) of

$$A \wedge X \wedge [\texttt{INPUT}\ w], \tag{12}$$

where type expansion yields a decision procedure. The same argumentation holds for the union and complementation of FA. It has to be noted that the intersection and complementation of FA via unification do not work in general

[5] 'Template' is a mild abuse of terminology since we intend not only to designate the type corresponding to the initial state of automaton, but also to suggest what other types are accessible.

for FA with ϵ-moves (Ritchie et al. 1992, 33-35). This restriction is due to the fact, that the intersected FA must run "in sync" (Sproat 1992, 139-140).

The following closure properties are demonstrated fairly directly.

Let $A_1 = (Q_1, \Sigma_1, \delta_1, q_0, F_1)$ and $A_2 = (Q_2, \Sigma_2, \delta_2, q_0', F_2)$.

- $A_1 \cap A_2 \doteq q_0 \wedge q_0'$
- $A_1 \cup A_2 \doteq q_0 \vee q_0'$
- $\overline{A_1} \doteq \neg q_0$

In addition, a weak form of functional uncertainty (Kaplan and Maxwell 1988), represented through recursive type specifications, is appropriate for the expression also concatenation and Kleene closure of FA. Krieger 1993b provides proofs using auxiliary definitions and apparatus we lack space for here.

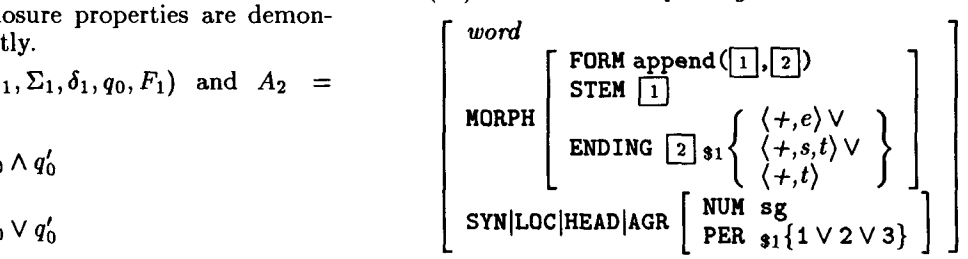
3 Allomorphy

The focus of this section lies in the illustration of the proposal above and in the demonstration of some benefits that can be drawn from the integration of allomorphy and morphotactics; we eschew here the discussion of alternative theories and concentrate on inflectional morphology. We describe inflection using a word-and-paradigm (WP) specification of morphotactics (Matthews 1972) and a two-level treatment of allomorphy (Koskenniemi 1983). We also indicate some potential advantages of mixed models of allomorphy—finite state and other.[6]

3.1 *WP* Morphotactics in FDL

Several WORD-GRAMMARS use FDL morphotactics (Trost 1991, Krieger and Nerbonne 1992 on derivation); alternative models are also available. Krieger and Nerbonne 1992 propose an FDL-based *WP* treatment of inflection. The basic idea is to characterize all the elements of a paradigm as alternative specifications of abstract lexemes. Technically, this is realized through the specification of large disjunctions which unify with lexeme specifications. The three elements of the paradigm in (1) would be described by the distributed disjunction in (13).

(13) *weak-paradigm* $\equiv$



This treatment provides a seamless interface to syntactic/semantic information, and helps realize the goal of representing ALL linguistic knowledge in a single formalism (Pollard and Sag 1987).

Nevertheless, the model lacks a treatment of allomorphy. The various allomorphs of *-st* in (1) are not distinguished in the FDL, and Krieger and Nerbonne 1992 foresaw an interface to an external module for allomorphy. It would be possible—but scientifically poor—to distinguish all of the variants at the level of morphotactics, providing a brute-force solution and multiplying paradigms greatly.[7] The characterization in Section 2 above allows us to formulate WITHIN FDL the missing allomorphy component.

3.2 *Two-Level* Allomorphy

Two-level morphology has become popular because it is a declarative, bidirectional and efficient means of treating allomorphy (see Sproat 1992 for a comprehensive introduction). In general, two-level descriptions provide constraints on correspondences between underlying (lexical) and surface levels. We shall use it to state constraints between morphemic units and their allomorphic realizations. Because two-level automata characterize relations between two levels, they are often referred to (and often realized as) transducers. The individual rules then represent constraints on the relation being transduced.

The different forms of the suffix in 2nd person singular in (1) are predictable given the phonological shape of the stem, and the alternations can be described by the following (simplified) two-level rules (we have abstracted away from inessential restrictions here, e.g., that (strong) verbs with *i/e*-umlaut do not show epenthesis):

[6] The choice of two-level allomorphy is justified both by the simplicity of two-level descriptions and by their status as a "lingua franca" among computational morphologists. Two-level analyses in FDLs may also prove advantageous if they simplify the potential compilation into a hybrid two-level approach of the kind described in Trost 1991.

[7] Tzoukermann and Libermann 1990 show that multiplying paradigms need not degrade performance, however.

(14) *e*-epenthesis in the *bet-* case
$+:e \Leftrightarrow \{d,t\}\_\_\{s,t\}$
s-deletion in the *mix-* case
$s:\emptyset \Leftrightarrow \{s,x,z,ch\}+:\emptyset\_\_t$

The colon ':' indicates a correspondence between lexical and surface levels. Thus the first rule states that a lexical morph boundary + must correspond to a surface *e* if it occurs after *d* or *t* and before *s* or *t*. The second specifies when lexical *s* is deleted (corresponds to surface $\emptyset$). Two-level rules of this sort are then normally compiled into transducers (Dalrymple et al. 1987, p.35-45).

3.3 FDL Specification of Two-Level Morphology

Two-level descriptions of allomorphy can be specified in FDLs straightforwardly if we model not transducers, but rather two-level acceptors (of strings of symbol pairs), following Ritchie et al. 1992. We therefore employ FA over an alphabet consisting of pairs of symbols rather than single symbols.[8]

The encoding of these FA in our approach requires only replacing the alphabet of atomic symbols with an alphabet of feature structures, each of which bears the attributes LEX and SURF. A pair of segments appearing as values of these features stand in the lexical-surface correspondence relation denoted by ':' in standard two-level formalisms. The values of the attributes STEM and ENDING in (13) are then not lists of symbols but rather lists of (underspecified) feature structures. Note that the italicized *t* etc. found in the sequences under MORPH|ENDING (13) denote types defined by equations such as (16) or (17). (To make formulas shorter we abbreviate 'alphabet' etymologically as '$\alpha\beta$'.)

(15) $\alpha\beta \equiv$

$$\begin{bmatrix} \text{LEX} & \$_1\{\text{"a"} \vee \ldots \text{"s"} \vee \text{"s"} \vee \text{"+"} \vee \text{"+"}\} \\ \text{SURF} & \$_1\{\text{"a"} \vee \ldots \text{"s"} \vee \emptyset \ \vee \text{"e"} \vee \emptyset\} \end{bmatrix}$$

(16) $t \equiv \alpha\beta \wedge [\text{LEX "t"}] = \begin{bmatrix} \alpha\beta \\ \text{LEX "t"} \\ \text{SURF "t"} \end{bmatrix}$

(17) $+ \equiv \alpha\beta \wedge [\text{LEX "+"}] = \begin{bmatrix} \alpha\beta \\ \text{LEX "+"} \\ \text{SURF "e"} \vee \emptyset \end{bmatrix}$

[8] Since our formalisation of FA cannot allow ϵ-transitions without losing important properties, we are in fact forced to this position.

It is the role of the collection of FA to restrict underspecified lexical representations to those obeying allomorphic constraints. This is the substance of the allomorphy constraint (18), which, together with the Acceptance Criterion (10), guarantees that the input obeys the constraints of the associated (initial states of the) FA.

(18) *allomorphy* $\equiv \begin{bmatrix} \text{MORPH|FORM} & \boxed{1} \\ \text{INPUT} & \boxed{1} \end{bmatrix}$

Rules of the sort found in (14) can be directly compiled into FA acceptors over strings of symbol pairs (Ritchie et al. 1992, p.19). Making use of the regular expression notation as templates (introduced in Section 2 above), (19-21) display a compilation of the first rule in (14). Here the composite rule is split up into three different constraints. The first indicates that epenthesis is obligatory in the environment specified and the latter two that each half of the environment specification is necessary.[9]

(19) *epenth-1* $\equiv$

$$\begin{bmatrix} \textit{allomorphy} \\ \text{MORPH}\,[\text{FORM}\,(\overline{\pi^*\ \{\text{t,d}\}\ \text{+:}\emptyset\ \ \{\text{s,t}\}\ \pi^*})] \end{bmatrix}$$

(20) *epenth-2* $\equiv$

$$\begin{bmatrix} \textit{allomorphy} \\ \text{MORPH}\,[\text{FORM}\,(\overline{\pi^*\ \text{+:e}\ \overline{\{\text{s,t}\}}\ \pi^*})] \end{bmatrix}$$

(21) *epenth-3* $\equiv$

$$\begin{bmatrix} \textit{allomorphy} \\ \text{MORPH}\,[\text{FORM}\,(\overline{\pi^*\ \overline{\{\text{t,d}\}}\ \text{+:e}\ \pi^*})] \end{bmatrix}$$

3.4 Limits of Pure FA Morphology

Finite-state morphology has been criticized (i) for the strict finite-stateness of its handling of morphotactics (Sproat 1992, 43-66); (ii) for making little or no use of the notion of inflectional paradigms and inheritance relations between morphological classes (Cahill 1990); and (iii) for its strict separation of phonology from morphology—i.e., standard two-level rules can only be sensitive to phonological contexts (including word and morpheme boundaries), and apply to all forms where these contexts hold. In fact, allomorphic variation is often "fossilized", having outlived its original phonological motivation. Therefore some allomorphic rules

[9] π^* denotes the Kleene closure over alphabet π and $\overline{A}$ the complement of A with respect to π.

are restricted in nonphonological ways, applying only to certain word classes, so that some stems admit idiosyncratic exceptions with respect to the applicability of rules (see Bear 1988, Emele 1988, Trost 1991).

To overcome the first difficulty, a number of researchers have suggested augmenting FA with "word grammars", expressed in terms of feature formalisms like PATR II (Bear 1986) or HPSG (Trost 1990). Our proposal follows theirs, improving only on the degree to which morphotactics may be integrated with allomorphy. See Krieger and Nerbonne 1992 for proposals for treating morphotactics in typed feature systems.

We illustrate how the FDL approach overcomes the last two difficulties in a concrete case of nonphonologically motivated allomorphy. German epenthesizes schwa (< e >) at morph boundaries, but in a way which is sensitive to morphological environments, and which thus behaves differently in adjectives and verbs. The data in (22) demonstrates some of these differences, comparing epenthesis in phonologically very similar forms.

	free, adj super	frei+st	freiest
(22)	*free*, v 2s pres	be+frei+st	befreist
	woo, v 2s pres	frei+st	freist

While the rule stated in (14) (and reformulated in (19)-(21)) treats the verbal epenthesis correctly, it is not appropriate for adjectives, for it does not allow epenthesis to take place after vowels. We thus have to state different rules for different morphological categories.

The original two-level formalism could only solve this problem by introducing arbitrary diacritic markers. The most general solution is due to Trost 1991, who associated two-level rules with arbitrary filters in form of feature structures. These feature structures are unified with the underlying morphs in order to check the context restrictions, and thus serve as an interface to information provided in the feature-based lexicon. But Trost's two-level rules are a completely different data structure from the feature structures decorating transitions in FA.

We attack the problem head on by restricting allomorphic constraints to specific classes of lexical entries, making use of the inheritance techniques available in structured lexicons. The cases of epenthesis in (22) is handled by defining not only the rule in (19-21) for the verbal cases, but also a second, quite similar rule for the more liberal epenthesis in adjectives.[10] This frees the rule from operating on a strictly phonological basis, making it subject to lexical conditioning. This is illustrated in Figure 2.

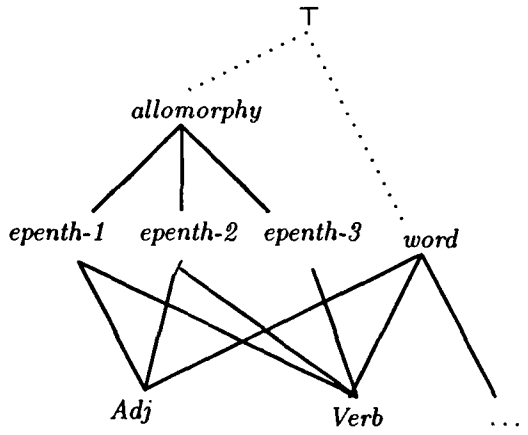


Figure 2: **Nonphonological Conditioning** of allomorphy is achieved by requiring that only some word classes obey the relevant constraints. Adjectives inherit from two of the epenthesis constraints in the text, and verbs (without i/e umlaut) satisfy all three. This very natural means of restricting allomorphic variation to selected, nonphonologically motivated classes is only made available through the expression of allomorphy in type hierarchy of the FDL. (The types denote the initial states of FA, as explained in Section 2.)

But note that this example demonstrates not only how feature-based allomorphy can overcome the strictly phonological base of two-level morphology (criticism (iii) above), but it also makes use of the inheritance structure in modern lexicons as well.

4 Conclusions

In this section we examine our proposal vis-à-vis others, suggest future directions, and provide a summary.

4.1 Comparison to other Work

Computational morphology is a large and active field, as recent textbooks (Sproat 1992 and Ritchie et al. 1992) testify. This impedes the identification of particularly important predecessors, among whom nonetheless three stand out. First, Trost 1991's use of two-level morphology in combination

[10] In fact, the rules could be specified so that the verbal rule inherited from the more general adjectival rule, but pursuing this here would take us somewhat afield.

with feature-based filters was an important impetus. Second, researchers at Edinburgh (Calder 1988, Bird 1992) first suggested using FDLs in phonological and morphological description, and Bird 1992 suggests describing FA in FDL (without showing how they might be so characterized, however—in particular, providing no FDL definition of what it means for an FA to accept a string).

Third, Cahill 1990 posed the critical question, viz., how is one to link the work in lexical inheritance (on morphotactics) with that in finite-state morphology (on allomorphy). This earlier work retained a separation of formalisms for allomorphy (MOLUSC) and morphotactics (DATR). Cahill 1993 goes on to experiment with assuming all of the allomorphic specification into the lexicon, in just the spirit proposed here.[11] Our work differs from this later work (i) in that we use FDL while she uses DATR, which are similar but not identical (cf. Nerbonne 1992); and (ii) in that we have been concerned with showing how the standard model of allomorphy (FA) may be assumed into the inheritance hierarchy of the lexicon, while Cahill has introduced syllable-based models.

4.2 Future Work

At present only the minimal examples in Section 2 above have actually been implemented, and we are interested in attempting more. Second, a compilation into genuine finite state models could be useful. Third, we are concerned that, in restricting ourselves thus far to acceptors over two-level alphabets, we may incur parsing problems, which a more direct approach through finite-state transducers can avoid (Sproat 1992, p.143). See Ritchie et al. 1992, 19-33 for an approach to parsing using finite-state acceptors, however.

4.3 Summary

This paper proposes a treatment of allomorphy formulated and processable in typed feature logic. There are several reasons for developing this approach to morphology. First, we prefer the GENERALITY of a system in which linguistic knowledge of all sorts may be expressed—at least as long as we do not sacrifice processing efficiency. This is an overarching goal of HPSG (Pollard and Sag 1987)—in which syntax and semantics is described in a feature formalism, and in which strides toward descriptions of morphotactics (Krieger 1993a, Riehemann 1993, Gerdemann 1993) and phonology (Bird 1992) have been taken. This work is the first to show how allomorphy may be described here. The proposal here would allow one to describe segments using features, as well, but we have not explored this opportunity for reasons of space.

Second, the uniform formalism allows the exact and more transparent specification of dependencies which span modules of otherwise different formalisms. Obviously interesting cases for the extension of feature-based descriptions to other areas are those involving stress and intonation—where phonological properties can determine the meaning (via focus) and even syntactic well-formedness (e.g., of deviant word orders). Similarly, allomorphic variants covary in the style register they belong to: the German dative singular in *-e, dem Kinde*, belongs to a formal register.

Third, and more specifically, the feature-based treatment of allomorphy overcomes the bifurcation of morphology into lexical aspects—which have mostly been treated in lexical inheritance schemes—and phonological aspects—which are normally treated in finite-state morphology. This division has long been recognized as problematic. One symptom of the problem is seen in the treatment of nonphonologically conditioned allomorphy, such as German *umlaut*, which (Trost 1990) correctly criticizes as *ad hoc* in finite-state morphology because the latter deals only in phonological (or graphemic) categories. We illustrated the benefits of the uniform formalism above where we showed how a similar nonphonologically motivated alternation (German schwa epenthesis) is treated in a feature-based description, which may deal in several levels of linguistic description simultaneously.

[11] Cf. Reinhard and Gibbon 1991 for another sort of DATR-based allomorphy

References

Backofen, R., L. Euler, and G. Görz. 1990. Towards the Integration of Functions, Relations and Types in an AI Programming Language. In *Proc. of GWAI-90.* Berlin. Springer.

Bear, J. 1986. A Morphological Recognizer with Syntactic and Phonological Rules. In *Proc. of COLING*, 272–276.

Bear, J. 1988. Morphology with Two-Level Rules and Negative Rule Features. In *Proc. of COLING*, 28–31.

Bird, S. 1992. Finite-State Phonology in HPSG. In *Proc. of COLING*, 74–80.

Cahill, L. J. 1990. Syllable-Based Morphology. In *Proc. of COLING*, 48–53.

Cahill, L. J. 1993. Morphonology in the Lexicon. In *Proc. of the 7th European ACL*, 87–96.

Calder, J. 1988. Paradigmatic Morphology. In *Proc. of the 5th European ACL.*

Carpenter, B. 1992. *The Logic of Typed Feature Structures.* No. 32Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press.

Dalrymple, M., R. Kaplan, L. Karttunen, K. Koskenniemi, S. Shaio, and M. Wescoat. 1987. Tools for Morphological Analysis. Technical Report CSLI-1987-108, CSLI, Stanford University.

Dörre, J., and A. Eisele. 1989. Determining Consistency of Feature Terms with Distributed Disjunctions. In *Proc. of GWAI-89 (15th German Workshop on AI)*, ed. D. Metzing, 270–279. Berlin. Springer-Verlag.

Emele, M. 1988. Überlegungen zu einer Two-Level Morphologie für das Deutsche. In *Proc. der 4. Österreichischen Artificial-Intelligence-Tagung und des WWWS*, ed. H. Trost, 156–163. Berlin: Springer. Informatik-Fachberichte 176.

Gerdemann, D. 1993. Complement Inheritance as Subcategorization Inheritance. In *German Grammar in HPSG*, ed. J. Nerbonne, K. Netter, and C. Pollard. Stanford: CSLI.

Hopcroft, J. E., and J. D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Reading, Massachusetts: Addison-Wesley.

Kaplan, R., and J. Maxwell. 1988. An Algorithm for Functional Uncertainty. In *Proc. of Coling 1988*, 303–305. Budapest.

Koskenniemi, K. 1983. Two-Level Model for Morphological Analysis. In *Proc. of IJCAI*, 683–685.

Krieger, H.-U. 1993a. Derivation Without Lexical Rules. In *Constraint Propagation, Linguistic Description and Computation*, ed. R. Johnson, M. Rosner, and C. Rupp. Academic Press.

Krieger, H.-U. 1993b. Representing and Processing Finite Automata Within Typed Feature Formalisms. Technical report, Deutsches Forschungsinstitut für Künstliche Intelligenz, Saarbrücken, Germany.

Krieger, H.-U., and J. Nerbonne. 1992. Feature-Based Inheritance Networks for Computational Lexicons. In *Default Inheritance within Unification-Based Approaches to the Lexicon*, ed. T. Briscoe, A. Copestake, and V. de Paiva. Cambridge: Cambridge University Press. Also DFKI Research Report RR-91-31.

Matthews, P. 1972. *Inflectional Morphology: A Theoretical Study Based on Aspects of Latin Verb Conjugation.* Cambridge, England: Cambridge University Press.

Nerbonne, J. 1992. Feature-Based Lexicons—An Example and a Comparison to DATR. In *Beiträge des ASL-Lexikon-Workshops, Wandlitz (bei Berlin)*, ed. D. Reimann, 36–49. also DFKI RR-92-04.

Pollard, C., and I. Sag. 1987. *Information-Based Syntax and Semantics, Vol.I.* Stanford: CSLI.

Reinhard, S., and D. Gibbon. 1991. Prosodic Inheritance and Morphological Generalizations. In *Proc. of the 6th European ACL*, 131–137.

Riehemann, S. 1993. Word Formation in Lexical Type Hierarchies. A Case Study of *bar*-Adjectives in German. Master's thesis, Eberhard-Karls-Universität Tübingen, Seminar für Sprachwissenschaft.

Ritchie, G. D., G. J. Russell, A. W. Black, and S. G. Pulman. 1992. *Computational Morphology: Practical Mechanisms for the English Lexicon.* Cambridge: MIT Press.

Smolka, G. 1988. A Feature Logic with Subsorts. Technical Report 33, WT LILOG-IBM Germany.

Sproat, R. 1992. *Morphology and Computation.* Cambridge: MIT Press.

Trost, H. 1990. The Application of Two-Level Morphology to Non-concatenative German Morphology. In *Proc. of COLING*, 371–376.

Trost, H. 1991. X2MORF: A Morphological Component Based on Augmented Two-Level Morphology. Technical Report RR-91-04, DFKI, Saarbrücken, Germany.

Tzoukermann, E., and M. Libermann. 1990. A Finite-State Morphological Processor for Spanish. In *Proc. of COLING*, Vol. 3.