X2MORF:
A Morphological Component Based on Augmented Two-Level Morphology

Harald Trost
Deutsches Forschungszentrum fur Kunstliche Intelligenz (DFKI) GmbH
Standort Saarbriicken
Stuhlsatzenhausweg 3, D-6600 Saarbriicken 11, Germany
e-mail: htrost@dfki.uni-sb.de

Abstract

X2MORF[1] is a language independent morphological component for the recognition and generation of word forms based on a lexicon of morphs. The approach is based on two-level morphology. Extensions are motivated by linguistic data which call into question an underlying assumption of standard two-level morphology, namely the independence of morphophonology and morphology as exemplified by two-level rules and continuation classes. Accordingly, I propose a model which allows for interaction between these two parts.

Instead of using continuation classes, word formation is described in a feature-based unification grammar. Two-level rules are provided with a morphological context in the form of feature structures. Information contained in the lexicon and the word formation grammar guides the application of two-level rules by matching the morphological context against the morphs. I present an efficient implementation of that model where rules are compiled into automata (as in the standard model) and where processing of the feature-based grammar is enhanced using an automaton derived from that grammar as a filter.

1 Introduction

Recently there has been renewed interest in morphological analysis and synthesis. One widely used approach is two-level morphology which combines a fully declarative representation of morphological data with a non-directional processing model. Two-level morphology was originally proposed by [Koskenniemi, 1983] and has since been implemented in several systems, e.g. [Karttunen, 1983]. As the name suggests it assumes only two levels, namely lexical and surface level. Besides the normal characters (representing graphemes or phonemes) there are diacritics used at the lexical level describing morphophonologically relevant information, e.g. *'$'* to mark word boundary or '+' for morph boundary. By default ail characters map to themselves, diacritics to the 0 character. All other mappings between the two levels are governed by rules consisting of a substitution (a pair of characters), an operator and a left and a right context (regular expressions made up from such pairs). The substitution defines a mapping between lexical and surface level, where its application is restricted by the (phonological) contexts.

Word formation is handled very simply with so-called *continuation classes* which are non-disjoint sets of morphs. Every morph contains information about its potential continuations (a set of continuation classes).

In the following discussion basic familiarity of the reader with two-level morphology is assumed (a concise description can be found in, e.g., [Dalrymple et ah, 1987]).

The standard model of two-level morphology makes-at least implicitly-a number of assumptions:

a) Word formation is basically expressed by the concatenation of morphs,
b) the concatenation process can be (adequately) described by continuation classes, and
c) morphology and morphophonology are autonomous systems with no interdependencies.

For most cases these assumptions are justified. But none of them holds for the whole range of morphological phenomena encountered in inflecting languages. For every assumption stated above some examples from German and English shall serve to show where problems arise:

Concerning a), one can say that concatenation is the single most important phenomenon in word formation, but there are notable exceptions:

German umlaut[2] is an example for an originally phonological process which—over time—turned into a morphological one. Presently, umlaut expresses a variety of different morphological features, among them the plural of nouns, *e.g.:* ***Mutter*** **(mother)** $\Rightarrow$ ***Mütter*****,** ***Haus*** **(house)** $\Rightarrow$ ***Häuser; Wolf*** **(wolf)** $\Rightarrow$ ***Wölfe*****.**

As these examples show, umlaut occurs together with endings but it may also be the only morphological marker. One way to describe umlaut in two-level morphology is to assume a-phonologically underspecified-lexical character (e.g., *U)* which by default maps to the regular vowel (e.g., *u).* A two-level rule maps the lexical character to the umlaut (e.g., *U* to *ii)* in all cases where this is morphologically required. E.g., in our example it is the morphological feature *plural,* not any phonological context which triggers rule application.

As to b), right association can be adequately expressed by

[1] Work on this project began while the author was at the Austrian Research Institute for AI, Vienna. For fruitful discussions and support with the implementation of the system I would like to thank Ralph Flassig.

[2] The alternation of the stem vowels **a, i, o, u to ä, i, ö, ü** respectively.

the continuation class approach, but because of its left-to-right bias left association cannot and must be recoded into right association. Circumfixation, as e.g. in the German past participle, and infixation (e.g. German to-infinitive) must be expressed even more indirectly. A formalism which allows for a more natural description of such phenomena would be favourable.

A number of authors have proposed to replace continuation classes with a grammar based on feature structures describing the legal combination of morphs (e.g. [Bear, 1986; Carson, 1988; Gorz and Paulus, 1988].

Concerning c) one must in some cases assume an interference between lexical and/or paradigmatic features of morphs on the one hand and morphophonological rules on the other hand. I will provide two examples, one from English and one from German.

In English, an *e* must be inserted between noun stem and the plural morph *s* under certain orthographical conditions. One of these conditions is the stem ending in *o* (e.g. *potato* => *potatoes*). This can be expressed by the following rule[3]:

(1) **+:e ⇐ o _ s ;**

Unfortunately, there are exceptions to that rule: In some words, e.g *banjo,* epenthesis of *e* is optional, so both plural forms *banjos* and *banjoes* are acceptable. In some other words *e* epenthesis must not take place, e.g. *piano* ⇒ *pianos* (see [Bear, 1988]). To which of these three classes a stem belongs seems to be idiosyncratic.

In German, a schwa is inserted between stems ending in *d* or t and endings starting with *s* or t. The following two-level rule captures that fact:

(2) **+:e ⇔ {d, t} _ {s, t} ;**

This rule[4] will correctly insert a schwa in such forms as *badest* (you bath), *arbeitet* (you work), *leitetest* (you guided), etc. But at a closer look one identifies exceptions to the rule: e.g. *haltst* (you hold), *rittst* (you rode), *sandiest* (you sent). All these stems exhibit umlaut or ablaut. Therefore a possible explanation for these exceptions is that the alteration of the stem vowel inhibits the application of the rule (2). A modified rule would be:

(2a) **+:e ⇔ {+, $} X* {d, t} _ {s, t} ;**
where X ∈ Σ \ {A:ä, E:i, O:ö, U:ü, ablaut-pairs}

A problem with this solution is that it forces us to represent all alternations of the stem vowel (i.e. both umlaut and ablaut) as morphophonological phenomena. In the case of ablaut—which is fully lexicalized—this is both difficult and wasteful.

Even if we did this, we would still face cases which are not described correctly even by this extended rule. Namely we have *2nd person plural past tense* of verbs following *strong conjugation* like *tratet* (you kicked) or *hieltet* (you held) where schwa is inserted despite the occurrence of ablaut. No phonological context can be constructed to account for this exception. Again, we are forced to view this as an idiosyncratic property of the paradigm position.

To deal with these kinds of problems in standard two-level morphology one has to create artificial phonological contexts by using extra diacritics. I will show how this approach works using the English plural example explained before. Instead of a single plural morph *s* we have to assume two different ones, nameley *s* and *&s* (the pair *&:0* is added to the alphabet). Next we have to split up the continuation classes for noun stems: Stems which behave regularly (like *potato)* may continue with *s,* stems where epenthesis is blocked (e.g. *piano)* continue with &s, the ones with optional insertion of *e* take both ending as continuations. Application of rule (1) would then yield the desired results.

Analogous solutions can be found for the other problems cited above. There are some severe drawbacks though with this kind of solution:

- additional diacritics (e.g., the *&* in the above example) are needed which cannot be motivated phonologically,
- because of the artificial ambiguities created more morphs are needed (e.g. *s* and *&s)* which have to be organized in more continuation classes,
- the null morph must be explicitly represented at the lexical (and therefore also at the surface) level transferring it from the morphological to the phonological level (e.g., to trigger the umlaut rule for the plural *Mutter).*

Consequently, the use of that approach leads to both linguistically inadequate descriptions and-because of the ambiguities-to computational costs such as larger requirements of space and processing time.

2 The architecture of X2M0RF

To overcome the problems cited above X2MORF augments the standard model in two related respects. First, the continuation class approach is substituted by a feature-based unification grammar to describe word formation (feature structures may contain disjunction and negation). For every morph the lexicon contains a feature structure. Grammar rules and principles-also formulated in the form of feature structures-guide the combination of the morphs. A possible problem of the use of unification grammar is the higher complexity involved. I will show how compilation techniques can help to keep processing efficient

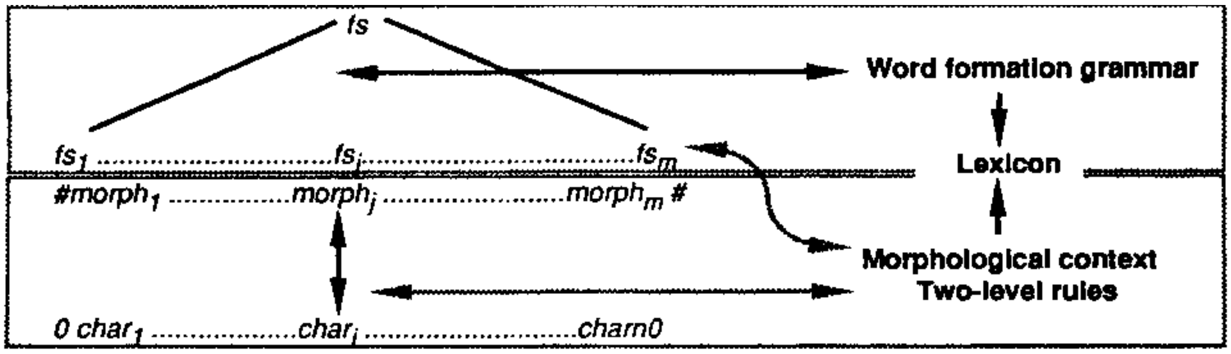
Secondly, two-level rules are provided with a *morphological context* in addition to the phonological one. This is accomplished by associating a feature structure with the rule. This feature structure is checked against the feature structure of the morph to which the substitution pair of the rule belongs. Checking means unifying the two: If the result of unification is FAIL, the morphological context of the rule is not present. If it succeeds the resulting feature structure is associated with the morph.

Application of a rule is now dependent of the presence of both the phonological and morphological context. Similar ideas have been proposed by [Bear,1988] and [Emele, 1988] but neither author came up with a correct algorithm. For a detailed discussion see [Trost, 1991].

While a first implementation of X2MORF interpreted rules directly [Trost, 1990] I have now developed a more

3 For the exact meaning of the operators in two-level rules see Koskennicmi (1983).

This rule (like all others used in this paper) is a simplified version of what is really needed for a morphological account of schwa epenthesis in German. But for the purposes of this paper it suffices. For more detail see Trost (1991).



fs
Word formation grammar
fs_1 ... fs_i ... fs_m
Lexicon
#$morph_1$ ... $morph_i$ ... $morph_m$ #
Morphological context
Two-level rules
0 $char_1$ ... $char_i$ ... $char_n$0

Figure 1: Global arcitecture of X2MORF

efficient implementation based on the original approach of [Koskenniemi, 1983J of compiling rules into automata. Figure 1 gives a sketch of the overall architecture. Like the standard model X2MORF consists of two parts. One translates from a string of characters or phonemes (the surface level) to a list of morphs (the lexical level). This is the morphophonological component. The other one combines the feature structures associated with every one of these morphs with a feature structure describing the word form (a lexeme plus morphosyntactic information). This is the morphological component, the word formation grammar.

3 Describing the example data in the augmented formalism

I will now show how X2MORF overcomes the problems cited in chapter 1. Obviously, the more powerful mechanism of unification grammar allows one to describe both left and right association, circumfixation and infixation in an adequate way.

How about the interaction between morphology and morphophonology? Let's return to our example of German schwa epenthesis. All morphs where schwa epenthesis should rightfully apply are marked with *[morph [head [epenthesis: +]]]* and all others where it must be blocked are marked with *[morph [head [epenthesis: -]]]* (this being the negation). An augmented rule incorporating the morphological context would then look like:

(2b) +:e ⇔ {d, t}_{s, t} / [morph [head [epenthesis: +]]] ;

The morphological context will then achieve the required results of restricting the application of schwa epenthesis in contrast to rule (2).

Rule (2b) comes with both a phonological and a morphological context. In general, rules need not have both contexts specified. Many phonological rules require no interaction with morphology, and there are also rules where the application is only morphologically restricted. An example for such a purely morphological rule is umlaut.

Again we must start by providing morphs with the necessary features: But now we want to link umlaut with the plural of nouns. This is described in grammar rule (3a) which states the interdependence between umlaut and noun plural and (3b) relating the absence of umlaut to singular:

(3a) [morph [head [umlaut: +]]] ⇔ [morph [head [cat: noun, number: plural]]]

(3b) [morph [head [umlaut: -]]] ⇔ [morph [head [cat: noun, number: sing]]]

We can then formulate rule (4). This rule would produces an umlaut in the surface form in all cases where the morph is marked accordingly and where the vowel U occurs.

(4) U:ü ⇔ _ / [morph [head [umlaut: +]]] ;

As these examples show, X2MORF is capable of representing data in a linguistically adequate way which pose problems for standard two-level morphology. We will now turn to the question of how to implement the system in an efficient way.

4 Processing augmented two-level rules

Similar to the standard model the augmented two-level rules of X2MORF are translated into automata table. Let's look at a sample rule to demonstrate that: Rule (2) would yield the automaton[5] shown in figure 2. But, as we have seen, rule (2) must be augmented to rule (2b) by a morphological context to guarantee correct application.

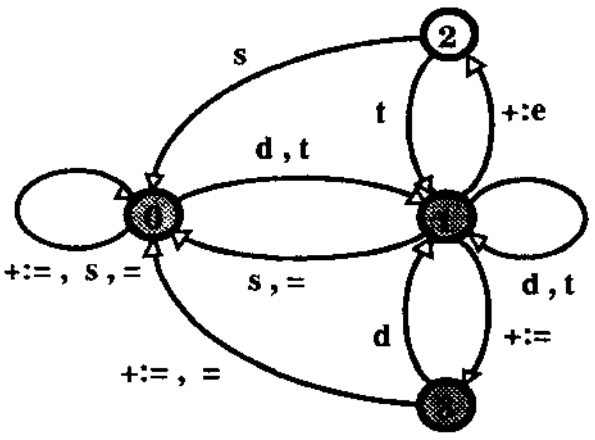


Figure 2: Automaton corresponding to rule (2)

How are morphological contexts integrated into the automaton in figure 2? We want a rule to apply if both the phonological and the morphological context apply. The automaton in figure 2 checks for the phonological context. The morphological context is to be checked only when the substitution pair actually occurs. This is equivalent to the

[5] Concerning the notation, shaded circles mean terminal nodes, the = stands for all characters not explicitly mentioned in the set of labels, i.e. +:= stands for all pairs with lexical + except +:e. With respect to the assumed alphabet that is the pair +:0.

situation that the arc labeled with the substitution pair is taken. Consequently, morphological contexts can be realized as tests on those arcs which are labeled with the substitution pair of the two-level rule.

But what happens if the phonological context is present, but the test returns failure? Then the arc may not be taken and the automaton in figure 2 would block. But of course it would also block for all alternative pairs (e.g. +:0). This is clearly wrong. To handle that situation correctly we must insert an extra arc labeled with all alternative pairs to the substitution pair (i.e. all pairs with the same lexical but a different surface character). Of course, this new arc may only be taken if the test on the original arc returns failure. In all other cases it should block. To produce that behaviour we have to associate a test to it as well. This test is the negation of the original test. The result of that augmentation is the automaton shown in figure 3.

A consequence of that realization of our morphological contexts is that we have to make sure that any morphological context used will either subsume the final feature structures of the morphs to which it is applied or unify to FAIL. If this is not the case the application of the rule is optional for that morph with respect to the morphological context, i.e the rule may or may not be applied.

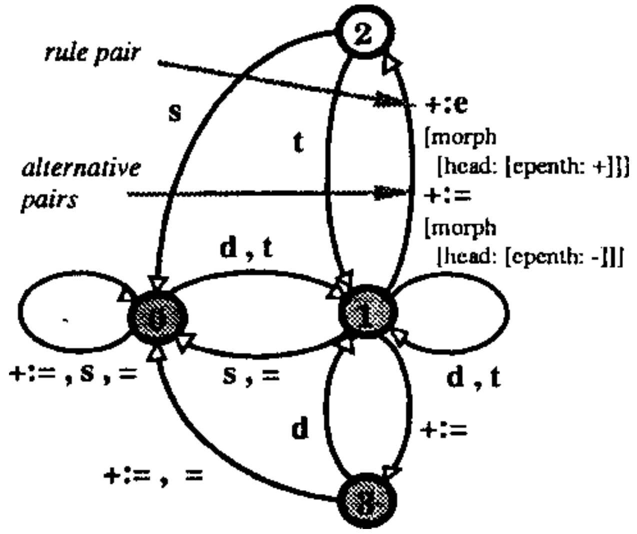


Figure 3: Augmented Automaton corresponding to rule (2b)

For example, imagine a stem not marked for the feature *epenthese* at all. If the phonological context is present one could take both of the arcs connecting stale 1 and 2 because both tests would yield a positive result. In some cases such optionality might be wanted. Remember the example of the plural of words like banjo, where both forms *banjos* and *banjoes* are correct

There is another consequence to associating tests with arcs. It might lead to indeterminism in the automaton which cannot be reduced. Look at the following rule:

(5) x:y ⇔ left-context_right-contexti / morph-contexti;
left-context_right-context2 / morph-context2;

When translating such a rule we have to accept an indeterminism because we need two arcs labeled x:y with the different tests (morphological contexts) attached.

Instead of processing the automata the usual way I realize a proposal by [Barton, 1986] to use a local constraint algorithm instead. One initializes the process by associating to each character pair of a given mapping all arcs which are labeled with that pair. The algorithm then proceeds by marking all possible paths eliminating dead ends.

```
PATH-FINDING
1) Initialization
  FOR-EVERY position:
    FOR-EVERY character pair:
      FOR-EVERY rule: Enter all arcs labeled with that pair.
2) Forward Scan
  FOR-EVERY rule:
    From left to right FOR-EVERY position i:
      Remove all arcs with no predecessor in position i-1
      IF no arc left for a pair at position i:
        eliminate that pair throughout all rules.
      IF no pair left at position i: RETURN with FAILURE.
3) Backward Scan
  FOR-EVERY rule:
    From right to left FOR-EVERY position i:
      Remove all arcs with no successor at position i+1
      IF no arc left for a pair at position i:
        eliminate that pair throughout all rules.
      IF no pair left at position i: RETURN with FAILURE.
4) IF any arc was deleted in step 2) or 3): GOTO step 2)
      ELSE RETURN.
```

This algorithm cannot handle non-local dependencies. The claim is that such dependencies do not occur in the morphology of natural languages.

5 Processing the word formation grammar

Because non-directionality is one of the advantages of the two-level model the replacement of continuation classes by a feature-based grammar should keep this property. Accordingly, a parser-generator had to be developed which is able to match lists of morphs and its internal representation (lexeme plus morphosyntactic information).

The parser-generator of X2MORF uses an algorithm oriented on the ideas of [Shieber et al., 1990]. It is based on the notion of heads. To be compatible with that algorithm the word formation grammars must fulfil the following requirements:

- Structure is defined via head daughters and complement daughters, i.e. every non-lexical item consists of a mother, a head daughter and (one or more) complement daughters.
- Head Feature Convention must be obeyed, i.e. head information is shared by the mother and their head daughter, and
- all complements are defined via a subcategorization list (in particular, there are no optional elements like modifiers).

These requirements led to an HPSG-style [Pollard and Sag, 1987] grammar. Other grammatical theories are of course also possible as long as the above requirements are met.

The algorithm maps internal structures (lexemes plus morphosyntactic information) to the feature structures associated to a list of morphs. In the process of this mapping a complex feature structure is created where the morphs form the leaves and the internal structure the root of the tree created by the head and complement daughter structure.

In the case of parsing the internal structure (i.e. the root node) comes with only those features shared by all word forms. The morphs (i.e. the set of possible leaves) on the other hand are fully determined and already ordered, i.e. boundary information is present. Therefore we have one additional constraint for parsing: The final structure is legal only if it spans the whole length of the morph list as specified by the boundary information.

In case of generation the internal structure is fully specified, while at the morphs' side we have (potentially) the whole lexicon. A first step is the collection of only the relevant lexical entries. Here we make use of the fact that all lexical entries have a feature *root*. Its value is a canonical form standing for the morpheme, e.g. all allomorphs of a verb stem would share the same root. Inflectional endings have an empty *root* feature because they do not contribute to the root form of a particular word form.

Because of compounding and derivation the root feature of word forms may be a list of entries. All lexical entries whose root form contains a member of that list or is empty are collected. They make up the set of relevant morphs in the case of generation.

Of course, a number of intermediate possibilities for specifying the arguments of the algorithm exist. If more specific information is available as to which word form to expect[6] the analysis can be restrained. If on the other hand the internal structure is not fully specified when generating, the algorithm will produce all the corresponding morph lists. This can be very useful in testing a certain set of linguistic data.

Let's now turn to the description of the core algorithm. Its task is to create a feature structure which combines root element and (a subset of) the lexical elements. This is accomplished by applying a mixed-mode approach. First, head information of the mother is projected onto the lexical elements to find potential heads. Every element thus selected is taken as a potential candidate.

The next step is bottom-up. The lexical element is projected up to its maximal projection (where its subcategorization list is empty). Places in the tree where complement daughters are to be added are collected. The created structure is unified with the root.

We then recursively apply the algorithm to fill in the complements. Whenever the algorithm fails to return a complement structure the whole structure has to be discarded. More than one returned complement structure means an ambiguity. The following diagram gives a more exact picture of the algorithm.

```
PARSE-GENERATE (root-of-tree, list-of-morphs)
  FOR-EVERY morph OF list-of-morphs:
    search for lexical heads by using head-info from root-of -tree;
    heads-list := list of potential heads;
  FOR-EVERY head OF heads-list:
    max-struct := project head up to maximal structure;
    unfilled-comps := collect all open comps in max-struct;
  results-list := LIST (UNIFY (max-struct, root-of-tree));
  IF results-list = FAIL: RETURN failure.
  FOR-EVERY comp OF unfilled-comps:
    comps-list := PARSE-GENERATE (comp, list-of-morphs)
    IF EMPTY (comps-list): RETURN failure
      ELSE RETURN
        results-list := combine every element of results-list
                        with every element of comps-list;
```

6 If X2MORF is embedded in a full-fledged system then the sentence level parser e.g., could provide for expectations concerning morphosyntax.

6 Partial compilation of the grammar

While the algorithm described so far is very general and powerful its expressive power is greater than what is needed for describing morphology. At least most of the data could be expressed by means of a regular grammar (or a finite state automaton of course) which would lead to a much more efficient implementation.

On the other hand we want to keep the possibility of describing word formation in the elegant and adequate way offered by unification grammar. And-for ease of interaction with other parts of a complete natural language processing system—the description in the form of feature structures is desirable.

It would lead to a great enhancement if it was possible to compile a finite state automaton from the word formation grammar which can then be used as a filter to rule out most of the possible combinations before the unification grammar is actually applied.

The word formation grammar G works on the alphabet of morphs Σ producing a language L. Of course, there is a set of regular grammars GR producing languages LR such that the following holds:

$$\Sigma^* \supset L_R \supseteq L$$

The task is to find such a grammar GR which in the best case would be equivalent to G. But it suffices that LR is at least much smaller than Σ^*. How can one arrive at such a grammar? The grammar writer must define a relevant subset of features. This subset is used to split the lexicon up into equivalence classes. On the basis of these classes and the grammar rules a regular grammar GR is constructed which accepts at least all the legal words in L. At the moment this compilation process is done by hand, but work on such a compiler is in progress.

As a next step an automaton equivlalent to GR is built. This automaton is then used as a filter. In parsing it is applied to the morphs found in the lexicon weeding out most of the spurious readings. In generation it is applied to the original set of lexical entries proposed and thereafter again in any recursive step of the algorithm. This filtering speeds up processing considerably because most of the unifications which would eventually lead to failure anyway do not come up in the first place because the morphs have been eliminated by the filtering process.

7 Interaction with the word formation part

We are now in the position to have a close look at the interaction between parser-generator and two-level rules. What makes this interaction complex is the fact that the morphological context must be tested against a feature structure which might still be incomplete.

To make things easier to understand I will start with generation where the the feature structures associated to the morphs are already fully specified by the parser-generator before the two-level rules are applied. The algorithm consists of the following steps:

```
GENERATE (lexical-string)
active-rules:= all rules which are associated to any of the
                     occurring lex-chars;
FOR-EVERY position i:
  enter all surface-chars potentially mapping to the lex-char at
     position i;
INITIALIZE the transition tables of all active rules;
FOR-EVERY position i:
     FOR-EVERY pair with an associated rule:
        UNIFY(morphological context of rule,
                     feature structure of the morph);
        IF unification succeds: OK
           ELSE remove arc for that pair at position i
              enter the same arc for all alternative pairs at position i.
Apply PATH-FINDING.
```

Let's consider an example: The alphabet shall consist of the obvious pairs and the only two-level rule shall be (2b). The list of morphs created by the word formation grammar is *(sand* +t *+t).* After initialization of the transition table of (2b) we encounter the following situation:

	$	s	a	n	d	+		t	+		t	$	
	0	s	a	n	d	0	e	t	0	e	t	0	
0	0-0	0-0	0-0	0-0	0-1	0-0	1-2	0-1	0-0	1-2	0-1	0-0	0
	1-0	1-0	1-0	1-0	1-1	1-3		1-1	1-3		1-1	1-0	1
	3-0	2-0	3-0	3-0	3-1	3-0		2-1	3-0		2-1	3-0	3

Testing of the morphological context will then take place at positions where +:e is found. In the first case the test fails, the arc 7-2 is removed and inserted for *+:0* which is the only alternative at that position. Since no arc is left for +;e this pair is ruled out. In the second case the test succeeds.

	$	s	a	n	d	+		t	+		t	$	
	0	s	a	n	d	0	e	t	0	e	t	0	
0	0-0	0-0	0-0	0-0	0-1	0-0	-	0-1	0-0	1-2	0-1	0-0	0
	1-0	1-0	1-0	1-0	1-1	1-3		1-1	1-3		1-1	1-0	1
	3-0	2-0	3-0	3-0	3-1	3-0		2-1	3-0		2-1	3-0	3
						1-2							

Next the local constraint algorithm is applied. By removing all arcs for +:0 in the second case it rules out that pair. The final situation looks as follows:

	$	s	a	n	d	+		t	+		t	$	
	0	s	a	n	d	0	e	t	0	e	t	0	
0	0-0	0-0	0-0	0-0	0-1	1-2	-	0-1	-	1-2	0-1	0-0	0

Both potential ambiguities have been resolved and the surface word *sandtet* can be generated.

We shall now turn to analysis[7]. There we make use of the morph lexicon to constrain the possibilities for lexical mappings of surface characters. If ambigous mappings remain we split up the resulting pairings in step 3) in such a way that every different list of morphs is processed seperately.

```
ANALYZE (surface-string)
1) FOR-EVERY position i:
  enter all lex-chars potentially mapping to the surface-char at
     position i;
2) Check with lexicon to eliminate all pairs which do not lead to a
     legal string of morphs;
3) word-forms := a list of all possible pairings
                     (one pair at every position);
4) FOR-EVERY word-form OF word-forms:
        active-rules:= all rules which are associated to any of the
                     occurring lex-chars;
     INITIALIZE the transition tables of all active rules;
     FOR-EVERY pair x:y1 where a rule is associated to a pair x:y2
                     with y1 <> y2: insert the tested arcs;
     Apply PATH-FINDING;
     FOR-EVERY position j: MORPH-TEST (pair, morph, j, rules);
     IF an arc has been removed because of failed test:
                     apply PATH-FINDING.
```

Why are tested arcs inserted in step 4) of the algorithm? One can assume an alternative pair only under the hypothesis that the rule does not apply at this position. This holds if either the phonological context does not apply: Then the local constraint algorithm will remove the newly inserted arc anyway. Or the morphological context does not apply: In that case we need the corresponding arc for MORPH-TEST which will then decide.

The testing of the morphological context consists of the following steps:

```
MORPH-TEST (x:y, morph, j, rules)
FOR-EVERY rule i:
  arc-set := all arcs of rule i for x:y at position j;
  test-structure := a disjunctive feature structure of all tests
                        associated to arcs in arc-set;
IF NOT-EQUAL(test-structure, NIL)
  IF y of pair <> y of rule-pair of rule i:
        test-structure := NOT(test-structure);
  result := UNIFY (test-structure, feature structure of morph)
  IF result = FAIL: remove pair x:y at position j in all rules;
                  ELSE: feature structure of morph := result;
```

There are two different possibilities for success. If the test structure subsumes the morph's feature structure, the morphological context is granted. Otherwise, grammar processing might add information which proves the morphological context wrong. Since the test structure is unified into the morph's feature structure this will correctly lead to failure.

I will demonstrate the algorithm for analysis using the same example as before. This time we start from the surface form *sandtet* (you sent). After the initialization of the

7 For sake of simplicity we assume that all the null characters are already inserted in the surface string when the algorithm starts.

transition tables we encounter the following situation:

	$	s	a	n	d	+	t	e +	t	$	
	0	s	a	n	d	0	t	e	t	0	
0	0-0	0-0	0-0	0-0	0-1	0-0	0-1	0-0 1-2	0-1	0-0	0
	1-0	1-0	1-0	1-0	1-1	1-3	1-1	1-0	1-1	1-0	1
	3-0	2-0	3-0	3-0	3-1	3-0	2-1	3-0	2-1	3-0	3

Now the lexicon has to be consulted. It will rule out the pair *e:e* because no morph + *tet* is found. We are therefore left with a single list of morphs, namely *(sand* +f *+t).* As a next step tested arcs are inserted at +:0 positions because a rule is associated with lexical +:

	$	s	a	n	d	+	t	+	t	$	
	0	s	a	n	d	0	t	e	t	0	
0	0-0	0-0	0-0	0-0	0-1	0-0	0-1	1-2	0-1	0-0	0
	1-0	1-0	1-0	1-0	1-1	1-3	1-1		1-1	1-0	1
	3-0	2-0	3-0	3-0	3-1	3-0	2-1		2-1	3-0	3

As a next step the local constraint algorithm is applied. The result is a single continous path:

	$	s	a	n	d	+	t	+	it	$	
	0	s	a	n	d	0	t	e	t	0	
0	0-0	0-0	0-0	0-0	0-1	1-2	2-1	12	0-1	0-0	0

Now the tests have to be performed. Both tests succed, i.e. unification yields a result different from FAIL. The feature structures associated to the morphs *sand* +/ +r (including the information transferred by filter testing) are input to the word formation grammar which will eventually come up with a feature structure comprising the information:

[[root: send]

[morph [head [cat:verb, tense:past, num:plural, person: 2J]J]

8 Conclusion

I have presented the system X2MORF, a morphological component based on two-level morphology. In contrast to the standard two-level model, X2MORF provides for interaction between the word formation part and the two-level rules. I have shown that such an interaction provides for a linguistically more adequate and a computationally feasible description of morphology.

Interaction is provided in the form of feature structures associated with the two-level rules. Unification of these feature structures with the feature structures of the morphs of which the pair associated to the rule is a part is used as a test to restrict the application of the two-level rules on morphological grounds.

With that augmentation X2MORF can provide two-level rules with an extra morphological context. This context is used for two different purposes:

- non-concatenative morphological phenomena like German umlaut can be expressed by two-level rules, inducing the necessary information transfer between two-level rules and word formation, and
- the morphological context can restrict the application of morpho-phonological rules.

I have shown an efficient implementation of X2MORF by compiling the two-level rules into finite state automata and by extracting a regular grammar from the feature-based unification grammar which is used as a filter sharply reducing the inherent combinatorial complexity of the unification grammar.

The system is currently running in CommonLisp on a Mac II fx. It has been used to describe German inflectional and derivational morphology. Currently it is being integrated with a lexicon structure containing lexeme-specific syntactic and semantic information.

References:

[Barton, 1986] Barton G.: Constraint Propagation in KIMMO Systems, ACL-86, New York.

[Bear, 1986] Bear J.: A Morphological Recognizer with Syntactic and Phonological Rules, COLING-86, Bonn.

[Bear, 1988] Bear J.: Morphology and Two-level Rules and Negative Rule Features, COLING-88,28-32, Budapest.

[Carson, 1988] Carson J.: Unification and transduction in Computational Phonology, COLING-88, 106-111, Budapest.

[Dalrymple et al., 1987] Dalrymple M., Kaplan R.M., Karttunen L., Koskenniemi K., Shaio S., Wescoat M.: Tools for Morphological Analysis, Stanford Univ., Report No. CSLI-87-103.

[Emele, 1988] Emele M.: Uberlegungen zu einer Two-Level Morphologie fur das Deutsche, in Trost H.(ed.), 4. Oesterreichische Artificial-Intelligence-Tagung, 156-163, Springer, Berlin.

[Gorz and Paulus, 1988] Gorz G., Paulus D.: A Finite State Approach to German Verb Morphology, COLING-88, 212-215, Budapest

[Karttunen, 1983] Karttunen L.: KIMMO: A General Morphological Processor, Texas Linguistic Forum 22, 167-186.

[Koskenniemi, 1983] Koskenniemi K.: Two-level Model for Morphological Analysis, IJCA1-83, 683-685, Karlsruhe.

[Pollard and Sag, 1987] Pollard CJ., Sag I.A.: Information-Based Syntax and Semantics, Vol. 1: Fundamentals, CSLI Lecture Notes No. 13, Univ. of Chicago Press, Chicago.

[Shieber et al., 1990] Shieber S., Noord G. van, Pereira F., Moore R.: Semantic-Head-Driven Generation, *Computational Linguistics,* 16(l):30-42.

[Trost, 1990] Trost H.: The application of two-level morphology to non-concatenative German morphology, COLING-90, Vol.II 371-376, Helsinki.

[Trost, 1991] Trost H.: Recognition and Generation of Word Forms for Natural Language Understanding Systems: Integrating Two-Level Morphology and Feature Unification, *Applied Artificial Intelligence,* 5(4)1991 (in print).