Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-91-04

X2MORF:
A Morphological Component Based on Two-Level Morphology

Harald Trost

January 1991

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern und Saarbrücken is a non-profit organization which was founded in 1988 by the shareholder companies ADV/Orga, AEG, IBM, Insiders, Fraunhofer Gesellschaft, GMD, Krupp-Atlas, Mannesmann-Kienzle, Siemens-Nixdorf, Philips and Siemens. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

❏ Intelligent Engineering Systems
❏ Intelligent User Interfaces
❏ Intelligent Communication Networks
❏ Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

X2MORF: A Morphological Component Based on Two-Level Morphology

Harald Trost

DFKI-RR-91-04

X2MORF[1]:
A Morphological Component Based on Augmented Two-Level Morphology

Harald Trost
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Standort Saarbrücken
Stuhlsatzenhausweg 3, D-6600 Saarbrücken 11, FRG
phone: (+49 681) 302-5301
e-mail: htrost@dfki.uni-sb.de

Abstract

In this paper I describe X2MORF a language-independent morphological component for the recognition and generation of word forms based on a lexicon of morphs. The approach is an extension of two-level morphology. The extensions are motivated by linguistic examples 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 the 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 this 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] X2MORF is an acronym for eXtented 2-level MORphology with Filters.

Table of Contents

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 all 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 al. 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 I will present some examples from German and English 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.: *Haus* (house) ⇒ *Häuser,*
Wolf (wolf) ⇒ *Wölfe,*
Mutter (mother) ⇒ *Mütter..*

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 rule maps the lexical character to the umlaut (e.g., *U* to *ü*) 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 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, Görz & 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 phonological 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

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

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

acceptable. In some other words *e* epenthesis must not take place, e.g. for *piano* the only legitimate plural form is *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 suffixes starting with *s* or *t*. The following two-level rule captures that phenomenon:

(2) +:e ⇔ {d, t} _ {s, t} ;

This rule[4] correctly inserts 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. *hältst* (you hold), *rittst* (you rode), *sandtest* (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). The rule might be modified as follows:

(2a) +:e ⇔ {+, $} X* {d, t} _ {s, t} ;

where X ∈ ∑ \ {A:ä, E:i, O:ö, U:ü, ablaut-pairs}

One 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 wastful.

Even if we did this, we would still face cases that are not described correctly even by this extended rule. There exist forms such as *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. We are forced to view it as an idiosyncratic property of the paradigm position.

To deal with these problems in standard two-level morphology one has to create artificial phonological contexts by using additional 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 further divide 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*

[4] This rule (like all the others proposed 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 (1990b).

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 as well at the surface) level transferring it from the morphological to the phonological level (e.g., to trigger the umlaut rule for the plural *Mütter*).

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. In the following I will show in which ways X2MORF augments the standard model to enable a more adequate handling of these phenomena.

2 The architecture of X2MORF

To overcome the problems cited above, the standard model is augmented 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.

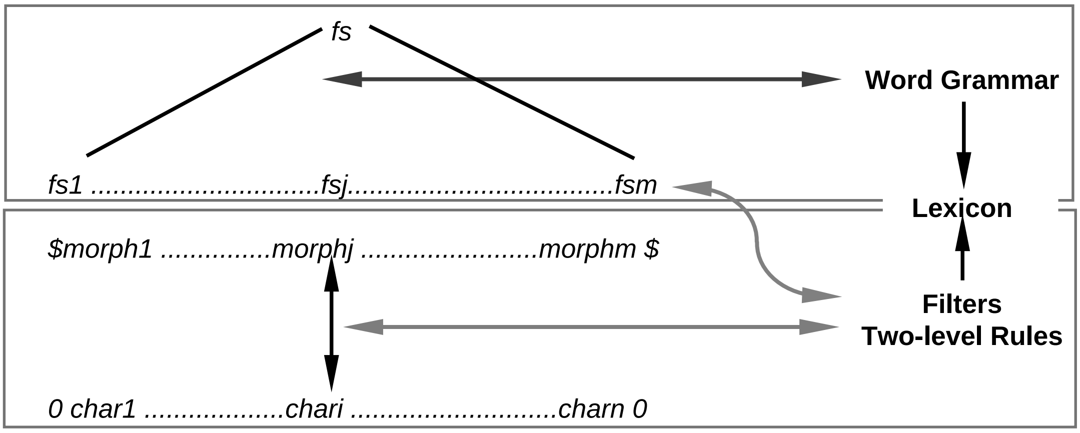
fs
Word Grammar
fs1 ....fsj....................................fsm
Lexicon
$morph1 ....morphj ....morphm $
Filters
Two-level Rules
0 char1 ....chari ....charn 0

Figure 1: Architecture of X2MORF

Application of a rule is then 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 implementation. For a detailed discussion see Trost (1990b).

While a first implementation of X2MORF interpreted rules directly (Trost 1990a) I have now developed a more efficient implementation based on the original approach of Koskenniemi (1983) 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 our examples in the augmented formalism

I will now show how X2MORF overcomes the problems cited above. 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 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.

How are morphological contexts integrated into the automaton in figure 2? We want a rule to apply if both the phonological and the morphological

[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.

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 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.

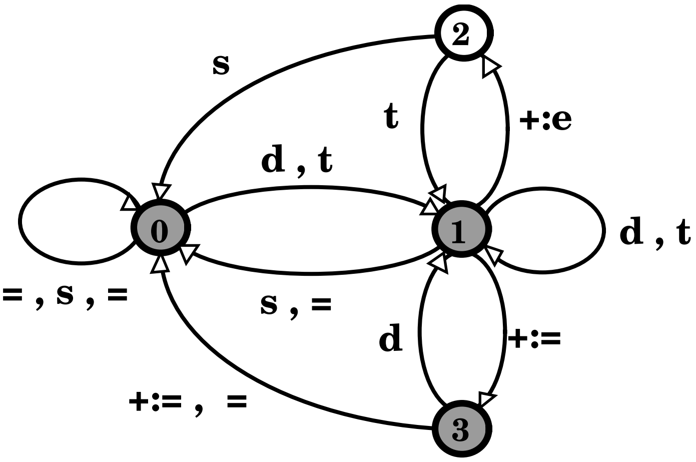

Figure 2: Automaton corresponding to rule (2)

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 the need to ensure that any morphological context used will either subsume the final feature structure of the morph to which it is applied or unify to FAIL with it. 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[6].

[6] The possibility to express optionality is in contrast to the phonological context where there is no such possibility. There are cases though where the linguistic data seem to be best expressed in terms of optionality.

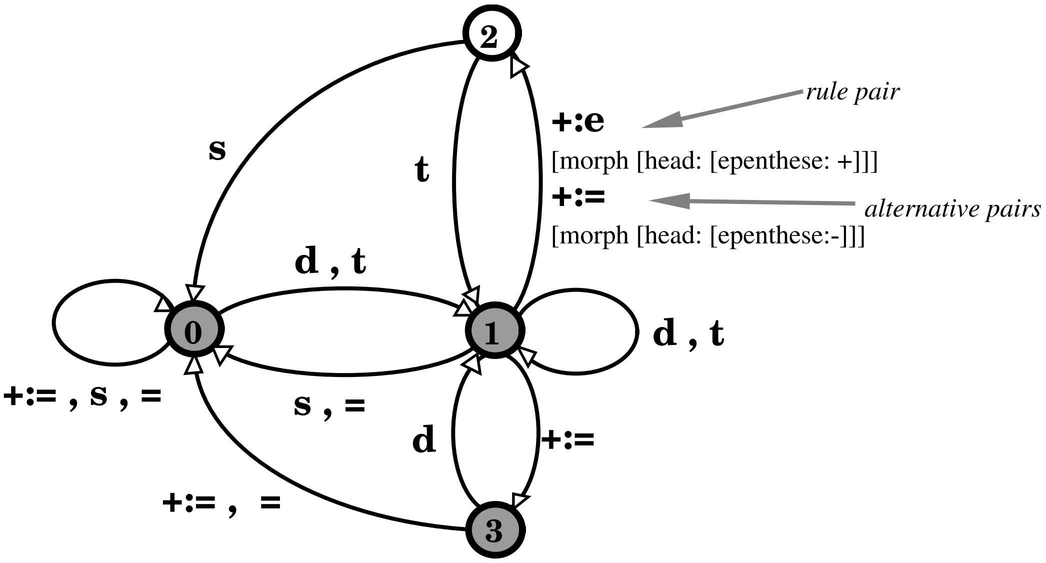

Figure 3: Augmented Automaton corresponding to rule (2b)

For example, imagine a stem not marked for the feature epenthesis at all. If the phonological context is present one could take both of the arcs connecting state 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-context$_1$ / morphological-context$_1$;
left-context _ right-context$_2$ / morphological-context$_2$;

When translating such a rule we would 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 follow 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. A diagram of the algorithm is given in figure 4.

This algorithm is more efficient than direct processing of the automata. It is also weaker because it cannot handle non-local dependencies. E.g., if there were different possibilities in two different positions the algorithm would always yield all the combinatorially possible combinations as results. The claim is that such non-local dependencies do not occur in the morphology of natural languages.

```
 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)
                                                    otherwise RETURN.
```

Figure 4: Algorithm for automata processing

5 A non-directional process model of 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 non-directional process 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 & Sag 1987) grammar. Other grammatical theories are also possible of course 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. Accordingly, the left boundary of the root node is set to zero and the right boundary to the maximal value occurring in the morph boundaries.

In case of generation the internal structure is fully specified, while every lexical entry is a potential leaf of the final structure. For efficiency reasons a necessary 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* containing their root form. Its value is a canonical form standing for the morpheme, e.g. all allomorphs of a verb stem would share the same root. Only 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 whose root form 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[7] 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

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

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.

```
PARSE-GENERATE (mother-of-tree, list-of-morphs)
   FOR-EVERY morph OF list-of-morphs:
      search for lexical heads by using head-info from mother-of -tree;
      heads-list := list of potential heads;
   FOR-EVERY head OF heads-list:
      max-structure := project head up to maximal structure;
      unfilled-complements := collect all open complement positions in max-structure;
      results-list := MAKE-LIST (UNIFY (max-structure, mother-of-tree));
      IF results-list = FAIL return failure.
      FOR-EVERY complement OF unfilled-complements:
         complements-list := PARSE-GENERATE (complement, list-of-morphs)

      IF EMPTY (complements-list)
            return failure
            ELSE return results-list := combine every element of results-list
                                              with every element of complements-list;
```

Figure 5: Algorithm of the parser-generator

The next step is performed in a bottom-up manner. 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 (figure 5 gives a descrition of the algorithm).

6 Partial compilation of the grammar to increase efficiency

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) 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 G_R producing languages L_R such that the following holds:

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

The task is to find such a grammar G_R which in the best case would be equivalent to G. But it suffices that L_R 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 up the lexicon into equivalence classes. On the basis of these classes and the grammar rules a regular grammar G_R 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 G_R 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 terminal feature structures standing for morphs have been eliminated by the filtering process.

7 Interaction with the word formation part

We are now in the position to take 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)
1) Activate all rules which are associated to any of the occurring lexical characters.
2) FOR-EVERY position:
      enter all surface characters which are potential mappings for the lexical character.
3) Initialize the transition tables of all active rules.
4) FOR-EVERY position i:
      FOR-EVERY pair with an associated rule:
         Unify morphological context of rule and 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.
5) Apply local constraint algorithm.
```

Let's now consider an example: The alphabet shall consist of the obvious pairs and the only two-level rule is (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 1-0 3-0	0-0 1-0 2-0	0-0 1-0 3-0	0-0 1-0 3-0	0-1 1-1 3-1	0-0 1-3 3-0	1-2	0-1 1-1 2-1	0-0 1-3 3-0	1-2	0-1 1-1 2-1	0-0 1-0 3-0	0 1 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 is removed and inserted for *+:0* the alternative at the position. 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 1-0 3-0	0-0 1-0 2-0	0-0 1-0 3-0	0-0 1-0 3-0	0-1 1-1 3-1	0-0 1-3 3-0 1-2	-	0-1 1-1 2-1	0-0 1-3 3-0	1-2	0-1 1-1 2-1	0-0 1-0 3-0	0 1 3

Next the local constraint algorithm is applied leading to the final situation from which the surface word *sandtet* can be generated.

	$	**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	-	2-1	-	1-2	2-1	1-0	0

Let's now turn to analysis[8]. 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

[8] For sake of simplicity we assume that all the null characters are already inserted in the surface string when we start the algorithm.

such a way that every different list of morphs is processed seperately.

```
ANALYZE (surface-string)
1) FOR-EVERY surface character: enter all potentially mapping lexicical characters.
2) Check with lexicon to eliminate all pairs which do not lead to a 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:
     Activate all rules which are associated to any of the occurring lexical characters.
     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 local constraint algorithm.
     FOR-EVERY position j: TEST-MORPH-CONTEXT (pair, morph, j, rules).
     IF an arc has been removed because of failed test:  apply local constraint algorithm.
```

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 in step 5). Or the morphological context does not apply: In that case we need the corresponding arc for MORPH-TEST to decide.

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

```
TEST-MORPH-CONTEXT (x:y, morph, arc-sets, rules)
FOR-EVERY rule i:
 1)   arc-set := all arcs of rule i for x:y at position j:
 2)   test-structure := disjunction of all tests associated to arcs in arc-set;
 3)   IF y of pair <> y of rule-pair: test-structure := negated test-structure;
 4)   result := UNIFY (test-structure,  feature of morph)
 5)   IF result = FAIL: remove pair x:y at position j in all rules;
            ELSE: feature structure of morph := result;
```

In step 5) 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.

One can easily understand that behaviour when looking at the semantics of feature structures. Every feature structure denotes a set of elements. Further specification of a feature structure narrows down that set. Unification yields the intersection of two sets. Therefore it may happen that further specification of a morph's feature structure yields a subset which is disjoint to the intersection. On the other hand, subsumption denotes a subset relation and intersection would therefore yield the whole subset as result.

I will demonstrate the algorithm using the same example. Now we start from the surface form *sandtet* (you sent). After the initialization of the 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 1-0 3-0	0-0 1-0 2-0	0-0 1-0 3-0	0-0 1-0 3-0	0-1 1-1 3-1	0-0 1-3 3-0	0-1 1-1 2-1	0-0 1-2 1-0 3-0	0-1 1-1 2-1	0-0 1-0 3-0	0 1 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 +t +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 1-0 3-0	0-0 1-0 2-0	0-0 1-0 3-0	0-0 1-0 3-0	0-1 1-1 3-1	0-0 1-3 3-0 1-2	0-1 1-1 2-1	1-2	0-1 1-1 2-1	0-0 1-0 3-0	0 1 3

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

	$	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	1-2	2-1	1-2	2-1	1-0	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 +t +t* (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: 2]]]]

8 Conclusion and further work

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 realized 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 implemented in the form described in this paper. It is programmed in CommonLisp currently running on a Mac II fx. Work on the compiler for the automated extraction of a finite state automaton from the feature-based unification grammar is currently in progress.

X2MORF has been applied to German morphology. The fragment used at the moment covers all of inflectional morphology and parts of derivation. It will be extended deal with composition as well. The system is currently integrated into the core natural language system developed in the project DISCO.

References:

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

Bear J. (1986): A Morphological Recognizer with Syntactic and Phonological Rules, COLING-86, Bonn, BRD.

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

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

Dalrymple M., Kaplan R.M., Karttunen L., Koskenniemi K., Shaio S., Wescoat M. (1987): Tools for Morphological Analysis, Stanford Univ., Report No. CSLI-87-103, Stanford, Calif.

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

Görz G., Paulus D. (1988): A Finite State Approach to German Verb Morphology, COLING-88, Budapest, 212-215.

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

Koskenniemi K. (1983): Two-level Model for Morphological Analysis, IJCAI-83, Karlsruhe, BRD, 683-685.

Pollard C.J., Sag I.A. (1987): Information-Based Syntax and Semantics, Vol. 1: Fundamentals, CSLI Lecture Notes No.13, Chicago: Univ. of Chicago Press.

Shieber S., Noord G. van, Pereira F., Moore R. (1990): Semantic-Head-Driven Generation, Computational Linguistics 16(1)1990, 30-42.

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

Trost H. (1990b): Recognition and Generation of Word Forms for Natural Language Understanding Systems: Integrating Two-Level Morphology and Feature Unification, Applied Artificial Intelligence 4(4)1990 (in print).

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
67608 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymem ftp von ftp.dfki.uni-kl.de (131.246.241.100) unter pub/Publications bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or per anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) under pub/Publications.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-92-43
Christoph Klauck, Jakob Mauss: A Heuristic driven Parser for Attributed Node Labeled Graph Grammars and its Application to Feature Recognition in CIM
17 pages

RR-92-44
Thomas Rist, Elisabeth André: Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP
15 pages

RR-92-45
Elisabeth André, Thomas Rist: The Design of Illustrated Documents as a Planning Task
21 pages

RR-92-46
Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, Wolfgang Wahlster: WIP: The Automatic Synthesis of Multimodal Presentations
19 pages

RR-92-47
Frank Bomarius: A Multi-Agent Approach towards Modeling Urban Traffic Scenarios
24 pages

RR-92-48
Bernhard Nebel, Jana Koehler:
Plan Modifications versus Plan Generation: A Complexity-Theoretic Perspective
15 pages

RR-92-49
Christoph Klauck, Ralf Legleitner, Ansgar Bernardi:
Heuristic Classification for Automated CAPP
15 pages

RR-92-50
Stephan Busemann:
Generierung natürlicher Sprache
61 Seiten

RR-92-51
Hans-Jürgen Bürckert, Werner Nutt:
On Abduction and Answer Generation through Constrained Resolution
20 pages

RR-92-52
Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul: PHI - A Logic-Based Tool for Intelligent Help Systems
14 pages

RR-92-53
Werner Stephan, Susanne Biundo:
A New Logical Framework for Deductive Planning
15 pages

RR-92-54
Harold Boley: A Direkt Semantic Characterization of RELFUN
30 pages

RR-92-55
John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, Stephan Oepen: Natural Language Semantics and Compiler Technology
17 pages

RR-92-56
Armin Laux: Integrating a Modal Logic of Knowledge into Terminological Logics
34 pages

RR-92-58
Franz Baader, Bernhard Hollunder:
How to Prefer More Specific Defaults in Terminological Default Logic
31 pages

RR-92-59
Karl Schlechta and David Makinson: On Principles and Problems of Defeasible Inheritance
13 pages

RR-92-60
Karl Schlechta: Defaults, Preorder Semantics and Circumscription
19 pages

RR-93-02
Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist:
Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-03
Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi:
An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages

RR-93-04
Christoph Klauck, Johannes Schwagereit:
GGD: Graph Grammar Developer for features in CAD/CAM
13 pages

RR-93-05
Franz Baader, Klaus Schulz: Combination Techniques and Decision Problems for Disunification
29 pages

RR-93-06
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux: On Skolemization in Constrained Logics
40 pages

RR-93-07
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux: Concept Logics with Function Symbols
36 pages

RR-93-08
Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer: COLAB: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-09
Philipp Hanschke, Jörg Würtz:
Satisfiability of the Smallest Binary Program
8 Seiten

RR-93-10
Martin Buchheit, Francesco M. Donini, Andrea Schaerf: Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-11
Bernhard Nebel, Hans-Juergen Buerckert:
Reasoning about Temporal Relations:
A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-12
Pierre Sablayrolles: A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-13
Franz Baader, Karl Schlechta:
A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-14
Joachim Niehren, Andreas Podelski,Ralf Treinen: Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-15
Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster: PLUS - Plan-based User Support
Final Project Report
33 pages

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz: Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-17
Rolf Backofen:
Regular Path Expressions in Feature Logic
37 pages

RR-93-18
Klaus Schild: Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-20
Franz Baader, Bernhard Hollunder:
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-22
Manfred Meyer, Jörg Müller:
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy:
Comparative Study of Connectionist Simulators
20 pages

RR-93-24
Rainer Hoch, Andreas Dengel:
Document Highlighting —
Message Classification in Printed Business Letters
17 pages

RR-93-25
Klaus Fischer, Norbert Kuhn: A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-26
Jörg P. Müller, Markus Pischel: The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-27
Hans-Ulrich Krieger:
Derivation Without Lexical Rules
33 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker: Feature-Based Allomorphy
8 pages

RR-93-29
Armin Laux: Representing Belief in Multi-Agent Worlds viaTerminological Logics
35 pages

RR-93-33
Bernhard Nebel, Jana Koehler:
Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-34
Wolfgang Wahlster:
Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.):
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt: Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-38
Stephan Baumann: Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf:
Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-41
Winfried H. Graf: LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-42
Hubert Comon, Ralf Treinen:
The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-45
Rainer Hoch: On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

DFKI Technical Memos

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Busemann: Prototypical Concept Formation An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang: Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh:
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben:
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

TM-93-01
Otto Kühn, Andreas Birk: Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

TM-93-02
Pierre Sablayrolles, Achim Schupeta:
Conlfict Resolving Negotiation for COoperative Schedule Management
21 pages

TM-93-03
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages

DFKI Documents

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokument-analyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-22
Werner Stein: Indexing Principles for Relational Languages Applied to PROLOG Code Generation
80 pages

D-92-23
Michael Herfert: Parsen und Generieren der Prolog-artigen Syntax von RELFUN
51 Seiten

D-92-24
Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25
Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26
Enno Tolzmann: Realisierung eines Werkzeugauswahlmoduls mit Hilfe des Constraint-Systems CONTAX
28 Seiten

D-92-27
Martin Harm, Knut Hinkelmann, Thomas Labisch: Integrating Top-down and Bottom-up Reasoning in COLAB
40 pages

D-92-28
Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

D-93-01
Philipp Hanschke, Thom Frühwirth: Terminological Reasoning with Constraint Handling Rules
12 pages

D-93-02
Gabriele Schmidt, Frank Peters, Gernod Laufkötter: User Manual of COKAM+
23 pages

D-93-03
Stephan Busemann, Karin Harbusch(Eds.): DFKI Workshop on Natural Language Systems: Reusability and Modularity - Proceedings
74 pages

D-93-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1992
194 Seiten

D-93-05
Elisabeth André, Winfried Graf, Jochen Heinsohn, Bernhard Nebel, Hans-Jürgen Profitlich, Thomas Rist, Wolfgang Wahlster: PPP: Personalized Plan-Based Presenter
70 pages

D-93-06
Jürgen Müller (Hrsg.): Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz Saarbrücken 29.-30. April 1993
235 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-07
Klaus-Peter Gores, Rainer Bleisinger: Ein erwartungsgesteuerter Koordinator zur partiellen Textanalyse
53 Seiten

D-93-08
Thomas Kieninger, Rainer Hoch: Ein Generator mit Anfragesystem für strukturierte Wörterbücher zur Unterstützung von Texterkennung und Textanalyse
125 Seiten

D-93-09
Hans-Ulrich Krieger, Ulrich Schäfer: TDL ExtraLight User's Guide
35 pages

D-93-10
Elizabeth Hinkelman, Markus Vonerden,Christoph Jung: Natural Language Software Registry
(Second Edition)
174 pages

D-93-11
Knut Hinkelmann, Armin Laux (Eds.): DFKI Workshop on Knowledge Representation Techniques — Proceedings
88 pages

D-93-12
Harold Boley, Klaus Elsbernd, Michael Herfert, Michael Sintek, Werner Stein: RELFUN Guide: Programming with Relations and Functions Made Easy
86 pages

D-93-14
Manfred Meyer (Ed.): Constraint Processing – Proceedings of the International Workshop at CSAM'93, July 20-21, 1993
264 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).