# Compounding and derivational morphology in a finite-state setting

**Jonas Kuhn**
Department of Linguistics
The University of Texas at Austin
1 University Station, B5100
Austin, TX 78712-11196, USA
jonask@mail.utexas.edu

## Abstract

This paper proposes the application of finite-state approximation techniques on a unification-based grammar of word formation for a language like German. A refinement of an RTN-based approximation algorithm is proposed, which extends the state space of the automaton by selectively adding distinctions based on the parsing history at the point of entering a context-free rule. The selection of history items exploits the specific linguistic nature of word formation. As experiments show, this algorithm avoids an explosion of the size of the automaton in the approximation construction.

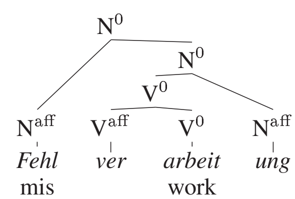
## 1 The locus of word formation rules in grammars for NLP

In English orthography, compounds following productive word formation patterns are spelled with spaces or hyphens separating the components (e.g., *classic car repair workshop*). This is convenient from an NLP perspective, since most aspects of word formation can be ignored from the point of view of the conceptually simpler token-internal processes of inflectional morphology, for which standard finite-state techniques can be applied. (Let us assume that to a first approximation, spaces and punctuation are used to identify token boundaries.) It makes it also very easy to access one or more of the components of a compound (like *classic car* in the example), which is required in many NLP techniques (e.g., in a vector space model).

If an NLP task for English requires detailed information about the structure of compounds (as complex multi-token units), it is natural to use the formalisms of computational *syntax* for English, i.e., context-free grammars, or possibly unification-based grammars. This makes it possible to deal with the bracketing structure of compounding, which would be impossible to cover in full generality in the finite-state setting.

In languages like German, spelling conventions for compounds do not support such a convenient split between sub-token processing based on finite-state technology and multi-token processing based on context-free grammars or beyond—in German, even very complex compounds are written without spaces or hyphens: words like *Verkehrswegeplanungsbeschleunigungsgesetz* ('law for speeding up the planning of traffic routes') appear in corpora. So, for a fully adequate and general account, the token-level analysis in German has to be done at least with a context-free grammar:[1] For checking the selection features of derivational affixes, in the general case a tree or bracketing structure is required. For instance, the prefix *Fehl-* combines with nouns (compare (1)); however, it can appear linearly adjacent with a verb, including its own prefix, and only then do we get the suffix *-ung*, which turns the verb into a noun.

(1)

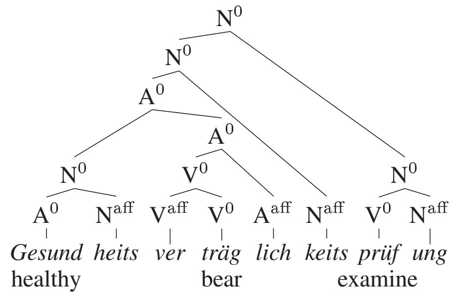



'misprocessing'

[1] For a fully general account of *derivational* morphology in English, the token-level analysis has to go beyond finite-state means too: the prefix *non-* in *nonrealizability* combines with the complex derived adjective *realizable*, not with the verbal stem *realize* (and *non-* could combine with a more complex form). However, since in English there is much less token-level interaction between derivation and compounding, a finite-state approximation of the relevant facts at token-level is more straightforward than in German.

Furthermore, context-free power is required to parse the internal bracketing structure of complex words like (2), which occur frequently and productively.

(2)




'check for health compatibility'

As the results of the DeKo project on derivational and compositional morphology of German show (Schmid et al. 2001), an adequate account of the word formation principles has to rely on a number of dimensions (or features/attributes) of the morphological units. An affix's selection of the element it combines with is based on these dimensions. Besides part-of-speech category, the dimensions include origin of the morpheme (Germanic vs. classical, i.e., Latinate or Greek[2]), complexity of the unit (simplex/derived), and stem type (for many lemmata, different base stems, derivation stems and compounding stems are stored; e.g., *träg* in (2) is a derivational stem for the lemma *trag(en)* ('bear'); *heits* is the compositional stem for the affix *heit*).

Given these dimensions in the affix feature selection, we need a unification-based (attribute) grammar to capture the word formation principles *explicitly* in a formal account. A slightly simplified such grammar is given in (3), presented in a PATR-II-style notation:[3]

```
(3) a. X0 → X1 X2
       <X1 CAT>          = PREFIX
       <X0 CAT>          = <X1 MOTHER-CAT>
       <X0 COMPLEXITY>   = PREFIX-DERIVED
       <X1 SELECTION>    = X2
    b. X0 → X1 X2
       <X2 CAT>          = SUFFIX
       <X0 CAT>          = <X2 MOTHER-CAT>
       <X0 COMPLEXITY>   = SUFFIX-DERIVED
       <X2 SELECTION>    = X1
    c. X0 → X1 X2
       <X0 CAT>          = <X2 CAT>
       <X0 COMPLEXITY>   = COMPOUND
```

(4) *Sample lexicon entries*

```
    a. X0: intellektual-
       <X0 CAT>          = A
       <X0 ORIGIN>       = CLASSICAL
       <X0 COMPLEXITY>   = SIMPLEX
       <X0 STEM-TYPE>    = DERIVATIONAL
       <X0 LEMMA>        = 'intellektuell'
    b. X0: -isier-
       <X0 CAT>                = SUFFIX
       <X0 MOTHER-CAT>         = V
       <X0 SELECTION CAT>      = A
       <X0 SELECTION ORIGIN>   = CLASSICAL
```

Applying the suffixation rule, we can derive *intellektual.isier-* (the stem of 'intellectualize') from the two sample lexicon entries in (4). Note how the selection feature (SELECTION) of prefixes and affixes are unified with the selected category's features (triggered by the last feature equation in the prefixation and suffixation rules (3a,b)).

**Context-freeness** Since the range of all atomic-valued features is finite and we can exclude lexicon entries specifying the SELECTION feature embedded in their own SELECTION value, the three attribute grammar rewrite rules can be compiled out into an equivalent context-free grammar.

## 2 Arguments for a finite-state word formation component

While there is linguistic justification for a context-free (or unification-based) model of word formation, there are a number of considerations that speak in favor of a finite-state account. (A basic assumption made here is that a morphological analyzer is typically used in a variety of different system contexts, so broad usability, consistency, simplicity and generality of the architecture are important criteria.)

First, there are a number of NLP applications for which a token-based finite-state analysis is standardly used as the only linguistic analysis. It would be impractical to move to a context-free technology in these areas; at the same time it is desirable to include an account of word formation in these tasks. In particular, it is important to be able to break down complex compounds into the individual components, in order to reach an effect similar to the way compounds are treated in English orthography.

[2] Of course, not the true ethymology is relevant here; ORIGIN is a category in the synchronic grammar of speakers, and for individual morphemes it may or may not be in accordance with diachronic facts.

[3] An implementation of the DeKo rules in the unification formalism YAP is discussed in (Wurster 2003).

Second, inflectional morphology has mostly been treated in the finite-state two-level paradigm. Since any account of word formation has to be combined with inflectional morphology, using the same technology for both parts guarantees consistency and reusability.[4]

Third, when a morphological analyzer is used in a linguistically sophisticated application context, there will typically be other linguistic components, most notably a syntactic grammar. In these components, more linguistic information will be available to address derivation/compounding. Since the necessary generative capacity is available in the syntactic grammar anyway, it seems reasonable to leave more sophisticated aspects of morphological analysis to this component (very much like the syntax-based account of English compounds we discussed initially). Given the first two arguments, we will however nevertheless aim for maximal exactness of the finite-state word formation component.

## 3 Previous strategies of addressing compounding and derivation

Naturally, existing morphological analyzers of languages like German include a treatment of compositional morphology (e.g., Schiller 1995). An overgeneration strategy has been applied to ensure coverage of corpus data. Exactness was aspired to for the inflected head of a word (which is always right-peripheral in German), but not for the non-head part of a complex word. The non-head may essentially be a flat concatenation of lexical elements or even an arbitrary sequence of symbols. Clearly, an account making use of morphological principles would be desirable. While the internal structure of a word is not relevant for the identification of the part-of-speech category and morphosyntactic agreement information, it *is* certainly important for information extraction, information retrieval, and higher-level tasks like machine translation.

[4]An alternative is to construct an interface component between a finite-state inflectional morphology and a context-free word formation component. While this can be conceivably done, it restricts the applicability of the resulting overall system, since many higher-level applications presuppose a finite-state analyzer; this is for instance the case for the Xerox Linguistic Environment (http://www.parc.com/istl/groups/nltt/xle/), a development platform for syntactic Lexical-Functional Grammars (Butt et al. 1999).

An alternative strategy—putting emphasis on a linguistically satisfactory account of word formation—is to compile out a higher-level word formation grammar into a finite-state automaton (FSA), assuming a bound to the depth of recursive self-embedding. This strategy was used in a finite-state implementation of the rules in the DeKo project (Schmid et al. 2001), based on the AT&T Lextools toolkit by Richard Sproat.[5] The toolkit provides a compilation routine which transforms a certain class of regular-grammar-equivalent rewrite grammars into finite-state transducers. Full context-free recursion has to be replaced by an explicit cascading of special category symbols (e.g., N1, N2, N3, etc.).

Unfortunately, the depth of embedding occurring in real examples is at least four, even if we assume that derivations like *ver.träg.lich* (‘compatible’; in (2)) are stored in the lexicon as complex units: in the initially mentioned compound *Verkehrs.wege.planungs.beschleunigungs.gesetz* (‘law for speeding up the planning of traffic routes’), we might assume that *Verkehrs.wege* (‘traffic routes’) is stored as a unit, but the remainder of the analysis is rule-based. With this depth of recursion (and a realistic morphological grammar), we get an unmanagable explosion of the number of states in the compiled (intermediate) FSA.

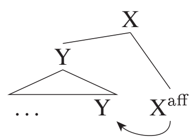
## 4 Proposed strategy

We propose a refinement of finite-state *approximation* techniques for context-free grammars, as they have been developed for syntax (Pereira and Wright 1997, Grimley-Evans 1997, Johnson 1998, Nederhof 2000). Our strategy assumes that we want to express and develop the morphological grammar at the linguistically satisfactory level of a (context-free-equivalent) unification grammar. In processing, a finite-state approximation of this grammar is used. Exploiting specific facts about morphology, the number of states for the constructed FSA can be kept relatively low, while still being in a position to cover realistic corpus example in an *exact* way.

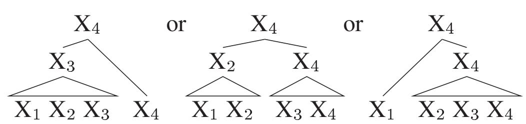
The construction is based on the following observation: Intuitively, context-free expressiveness is not needed to constrain *grammaticality* for most of the

[5]Lextools: a toolkit for finite-state linguistic analysis, AT&T Labs Research; http://www.research.att.com/sw/tools/lextools/

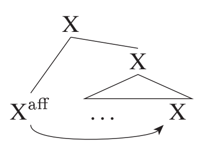
word formation combinations. This is because in most cases, either (i) morphological feature selection is performed between string-adjacent terminal symbols, or (ii) there are no categorial restrictions on possible combinations. (i) is always the case for suffixation, since German morphology is exclusively right-headed.[6] So the head of the unit selected by the suffix is always adjacent to it, no matter how complex the unit is:

(5)



(i) is also the case for prefixes combining with a simple unit. (ii) is the case for compounding: while affix-derivation is sensitive to the mentioned dimensions like category and origin, no such grammatical restrictions apply in compounding.[7] So the fact that in compounding, the heads of the two combined units may not be adjacent (since the right unit may be complex) does not imply that context-freeness is required to exclude impossible combinations:

(6)



The only configuration requiring context-freeness to exclude ungrammatical examples is the combination of a prefix with a complex morphological unit:

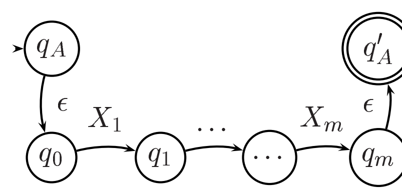
(7)



As (1) showed, such examples do occur; so they should be given an exact treatment. However, the depth of recursive embeddings of this particular type (possibly with other embeddings intervening) in realistic text is limited. So a finite-state approximation keeping track of prefix embeddings in particular, but leaving the other operations unrestricted seems well justified. We will show in sec. 6 how such a technique can be devised, building on the algorithm reviewed in sec. 5.

## 5 RTN-based approximation techniques

A comprehensive overview and experimental comparison of finite-state approximation techniques for context-free grammars is given in (Nederhof 2000). In Nederhof's approximation experiments based on an HPSG grammar, the so-called RTN method provided the best trade-off between exactness and the resources required in automaton construction. (Techniques that involve a heavy explosion of the number of states are impractical for non-trivial grammars.) More specifically, a parameterized version of the RTN method, in which the FSA keeps track of possible derivational histories, was considered most adequate.

The RTN method of finite-state approximation is inspired by recursive transition networks (RTNs). RTNs are collections of sub-automata. For each rule $A \to X_1 \dots X_m$ in a context-free grammar, a sub-automaton with $m + 3$ states is constructed:

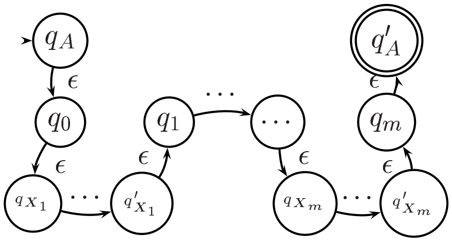
(8)



As a symbol is processed in the $A$ automaton (say, $X_1$), the RTN control jumps to the respective sub-automaton's initial state (so, from $q_0$ in (8) to a state $q_{X_1}$ in the sub-automaton for $X_1$), keeping the return address on a stack representation. When the sub-automaton is in its final state ($q'_{X_1}$), control jumps back to the next state in the $A$ automaton: $q_1$.

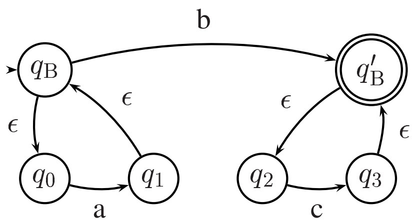
In the RTN-based finite-state approximation of a context-free grammar (which does not have an unlimited stack representation available), the jumps to sub-automata are hard-wired, i.e., transitions for non-terminal symbols like the $X_1$ transition from $q_0$ to $q_1$ are replaced by direct $\epsilon$-transitions to the initial state and from the end state of the respective sub-automata: (9). (Of course, the resulting non-deterministic FSA is then determinized and minimized by standard techniques.)

[6] This may appear to be falsified by examples like *ver-* ($V^{aff}$) + *Urteil* (N, 'judgement') = *verurteilen* (V, 'convict'); however, in this case, a noun-to-verb conversion precedes the prefix derivation. Note that the inflectional marking is always right-peripheral.

[7] Of course, when speakers disambiguate the possible bracketings of a complex compound, they can exclude many combinations as implausible. But this is a defeasible world knowledge-based effect, which should not be modeled as strict selection in a morphological grammar.

(9) 


The technique is approximative, since on jumping back, the automaton "forgets" where it had come from, so if there are several rules with a right-hand side occurrence of, say $X_m$, the automaton may non-deterministically jump back to the wrong rule. For instance, if our grammar consists of a recursive production B → a B c for category B, and a production B → b, we will get the following FSA:

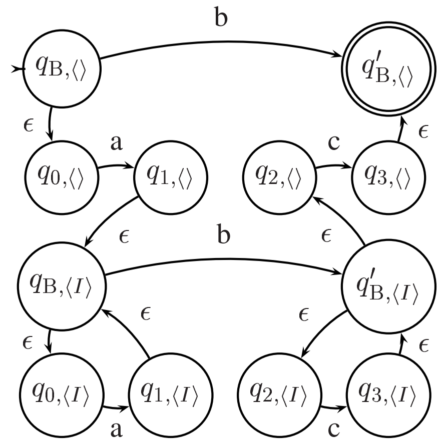
(10) 


The approximation loses the original balancing of a's and c's, so "abcc" is incorrectly accepted.

In the **parameterized version of the RTN method** that Nederhof (2000) proposes, the state space is enlarged: different copies of each state are created to keep track of what the derivational history was at the point of entering the present sub-automaton. For representing the derivational history, Nederhof uses a list of "dotted" productions, as known from Earley parsing. So, for state $q_{\mathrm{B}}$ in (10), we would get copies $q_{\mathrm{B},\langle\rangle}$, $q_{\mathrm{B},\langle[\mathrm{B}\to\mathrm{a}\bullet\mathrm{Bc}]\rangle}$, etc., likewise for the states $q_0, q_1, \ldots$ The $\epsilon$-transitions for jumping to and from embedded categories observe the laws for legal context-free derivations, as far as recorded by the dotted rules.[8] Of course, the window for looking back in history is bounded; there is a parameter (which Nederhof calls $d$) for the size of the history list in the automaton construction. Beyond the recorded history, the automaton's approximation will again get inexact.

(11) shows the parameterized variant of (10), with parameter $d = 2$, i.e., a maximal length of one element for the history ($I$ is used as a short-hand for item $[\mathrm{B} \to \mathrm{a} \bullet \mathrm{Bc}]$). (11) will not accept "abcc" (but it will accept "aabccc").

[8]For the exact conditions see (Nederhof 2000, 25).

(11) 


The number of possible histories (and thus the number of states in the non-deterministic FSA) grows exponentially with the depth parameter, but only polynomially with the size of the grammar. Hence, with parameter $d = 2$ ("RTN2"), the technique is usable for non-trivial syntactic grammars. Nederhof (2000) discusses an important additional step for avoiding an explosion of the size of the intermediate, non-deterministic FSA: before the described approximation is performed, the context-free grammar is split up into **subgrammars of mutually recursive categories** (i.e., categories which can participate in a recursive cycle); in each subgrammar, all other categories are treated as non-terminal symbols. For each subgrammar, the RTN construction and FSA minimization is performed separately, so in the end, the relatively small minimized FSAs can be reassembled.

## 6 A selective history-based RTN-method

In word formation, the split of the original grammar into subgrammars of mutually recursive (MR) categories has no great complexity-reducing effect (if any), contrary to the situation in syntax. Essentially, all recursive categories are part of a single large equivalence class of MR categories. Hence, the size of the grammar that has to be effectively approximated is fairly large (recall that we are dealing with a compiled-out unification grammar). For a realistic grammar, the parameterized RTN technique is unusable with parameter $d = 3$ or higher. Moreover, a history of just two previous embeddings (as we get it with $d = 3$) is too limited in a heavily recursive setting like word formation: recursive embeddings of depth four occur in realistic text.

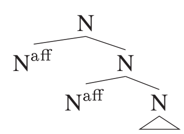
However, we can exploit more effectively the "mildly context-free" characteristics of morphological grammars (at least of German) discussed in sec. 4. We propose a refined version of the parameterized RTN-method, with a selective recording of derivational history. We stipulate a distinction of two types of rules: "historically important" h-rules (written $A \stackrel{h}{\to} X_1 \dots X_m$) and non-h-rules (written $A \stackrel{nh}{\to} X_1 \dots X_m$). The h-rules are treated as in the parameterized RTN-method. The non-h-rules are not *recorded* in the construction of history lists; they are however taken into account in the determination of *legal histories*. For instance, $[\mathrm{A} \stackrel{h}{\to} \bullet\mathrm{Bc}]$ will appear as a legal history for the sub-automaton for some category D only if there is a derivation $\mathrm{B} \stackrel{nh}{\Rightarrow}^{*} \alpha\, \mathrm{D}\, \beta$ (i.e., a sequence of rule rewrites making use of non-h-rules). By classifying certain rules as non-h-rules, we can concentrate record-keeping resources on a particular subset of rules.

In sec. 4, we saw that for most rules in the compiled-out context-free grammar for German morphology (all rules compiled from (3b) and (3c)), the inexactness of the RTN-approximation does not have any negative effect (either due to head-adjacency, which is preserved by the non-parametric version of RTN, or due to lack of category-specific constraints, which means that no context-free balancing is checked). Hence, it is safe to classify these rules as non-h-rules. The only rules in which the inexactness may lead to overgeneration are the ones compiled from the prefix rule (3a). Marking these rules as h-rules and doing selective history-based RTN construction gives us exactly the desired effect: we will get an FSA that will accept a free alternation of all three word-formation types (as far as compatible with the lexical affixes' selection), but stacking of prefixes is kept track of. Suffix derivations and compounding steps do not increase the length of our history list, so even with a $d = 2$ or $d = 3$, we can get very far in exact coverage.

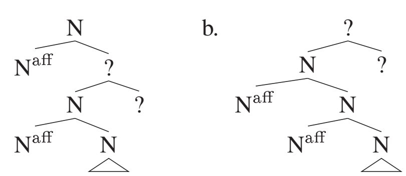
## 7 Additional optimizations

Besides the selective history list construction, two further optimizations were applied to Nederhof's (2000) parameterized RTN-method: First, Earley items with the same remainder to the right of the dot were collapsed ($[A \stackrel{h}{\to} \alpha_1 \bullet \beta]$ and $[A \stackrel{h}{\to} \alpha_2 \bullet \beta]$). Since they are indistinguishable in terms of future behavior, making a distinction results in an unnecessary increase of the state space. (Effectively, only the material to the right of the dot was used to build the history items.) Second, for immediate right-peripheral recursion, the history list was collapsed; i.e., if the current history has the form $\langle [A \stackrel{h}{\to} \dots \bullet A], \dots \rangle$, and the next item to be added would be again $[A \stackrel{h}{\to} \dots \bullet A]$, the present list is left unchanged. This is correct because completion of $[A \stackrel{h}{\to} \dots A\bullet]$ will automatically result in the completion of all immediately stacked such items.

Together, the two optimizations help to keep the number of different histories small, without losing relevant distinctions. Especially the second optimization is very effective in a selective history setting, since the "immediate" recursion need not be literally immediate, but an arbitrary number of non-h-rules may intervene. So if we find a noun prefix $[\mathrm{N} \stackrel{h}{\to} \mathrm{N}^{\mathrm{aff}} \bullet \mathrm{N}]$, i.e., we are looking for a noun, we need not pay attention (in terms of coverage-relevant history distinctions) whether we are running into compounds or suffixations: we know, when we find another noun prefix (with the same selection features, i.e., origin etc.), one analysis will always be to close off both prefixations with the same noun:

(12)


Of course, the second prefixation need not have happened on the right-most branch, so at the point of having accepted $\mathrm{N}^{\mathrm{aff}}\ \mathrm{N}^{\mathrm{aff}}\ \mathrm{N}$, we may actually be in the configuration sketched in (13a):

(13) a. b.


Note however that in terms of grammatically legal continuations, this configuration is "subsumed" by (13b), which *is* compatible with (12) (the top '?' category will be accessible using $\epsilon$-transitions back from a completed N—recall that suffixation and compounding is not controlled by any history items).

So we can note that the only examples for which the approximating FSA is inexact are those where the stacking depth of *distinct* prefixes (i.e., selecting

| | | # diff. pairs of categ./hist. list | interm. non-deterministic fsa # states | # $\Sigma$-trans. | # $\epsilon$-trans. | minimized fsa # states | # trans. |
|---|---|---|---|---|---|---|---|
| plain | $d=2$ | 169 | 1,118 | 640 | 963 | 2 | 16 |
| parameterized | $d=3$ | 1,861 | 13,149 | 7,595 | 11,782 | 11 | 198 |
| RTN-method | $d=4$ | 22,333 | | | | | |
| selective | $d=2$ | 229 | 2,934 | 1,256 | 4,000 | 14 | 361 |
| history-based | $d=3$ | 2,011 | 26,343 | 11,300 | 36,076 | 14 | 361 |
| RTN-method | $d=4$ | 18,049 | | | | | |

Figure 1: *Experimental results for sample grammar with 185 rules*

for a different set of features) is greater than our parameter $d$. Thanks to the second optimization, the relatively frequent case of stacking of two verbal prefixes as in *vor.ver.arbeiten* 'preprocess' counts as a single prefix for book-keeping purposes.

## 8 Implementation and experiments

We implemented the selective history-based RTN-construction in Prolog, as a conversion routine that takes as input a definite-clause grammar with compiled-out grounded feature values; it produces as output a Prolog representation of an FSA. The resulting automaton is determinized and minimized, using the FSA library for Prolog by Gertjan van Noord.[9] Emphasis was put on identifying the most suitable strategy for dealing with word formation taking into account the relative size of the FSAs generated (other techniques than the selective history strategy were tried out and discarded).

The algorithm was applied on a sample word formation grammar with 185 compiled-out context-free rules, displaying the principled mechanism of category and other feature selection, but not the full set of distinctions made in the DeKo project. 9 of the rules were compiled from the prefixation rule, and were thus marked as h-rules for the selective method.

We ran a comparison between a version of the non-selective parameterized RTN-method of (Nederhof 2000) and the selective history method proposed in this paper. An overview of the results is given in fig. 1.[10] It should be noted that the optimizations of sec. 7 were applied in *both* methods (the non-selective method was simulated by marking all rules as h-rules).

As the size results show, the non-deterministic FSAs constructed by the selective method are more complex (and hence resource-intensive in minimization) than the ones produced by the "plain" parameterized version. However, the difference in exactness of the approximizations has to be taken into account. As a tentative indication for this, note that the minimized FSA for $d = 2$ in the plain version has only two states; so obviously too many distinctions from the context-free grammar have been lost.

In the plain version, all word formation operations are treated alike, hence the history list of length one or two is quickly filled up with items that need not be recorded. A comparison of the number of different pairs of categories and history lists used in the construction shows that the selective method is more economical in the use of memory space as the depth parameter grows larger. (For $d = 4$, the selective method would even have fewer different category/history list pairs than the plain method, since the patterns become repetitive. However, the approximations were impractical for $d = 4$.) Since the selective method uses non-h-rules only in the determination of legal histories (as discussed in sec. 6), it can actually "see" further back into the history than the length of the history list would suggest.

What the comparison clearly indicates is that in terms of resource requirements, our selective method with a parameter $d_0$ is much closer to the $d_0$-version of the plain RTN-method than to the next higher $d_0+1$ version. But since the selective method focuses its record-keeping resources on the crucial aspects of the finite-state approximation, it brings about a much higher gain in exactness than just extending the history list by one in the plain method.

We also ran the selective method on a more fine-

[9] FSA6.2xx: Finite State Automata Utilities; http://odur.let.rug.nl/~vannoord/Fsa/

[10] The fact that the minimized FSAs for $d = 2, 3$ are identical for the selective method is an artefact of the sample grammar.

grained morphological grammar with 403 rules (including 12 h-rules). Parameter $d = 2$ was applicable, leading to a non-deterministic FSA with 7,345 states, which could be minimized. Parameter $d = 3$ led to a non-deterministic FSA with 87,601 states, for which minimization could not be completed due to a memory overflow. It is one goal for future research to identify possible ways of breaking down the approximation construction into smaller subproblems for which minimization can be run separately (even though all categories belong to the same equivalence class of mutually recursive categories).[11] Another goal is to experiment with the use of transduction as a means of adding structural markings from which the analysis trees can be reconstructed (to the extent they are not underspecified by the finite-state approach); possible approaches are discussed in Johnson 1996 and Boullier 2003.

Inspection of the longest few hundred prefix-containing word forms in a large German newspaper corpus indicates that prefix stacking is rare. (If there are several prefixes in a word form, this tends to arise through compounding.) No instance of stacking of depth 3 was observed. So, the range of phenomena for which the approximation is inexact is of little practical relevance. For a full evaluation of the coverage and exactness of the approach, a comprehensive implementation of the morphological grammar would be required. We ran a preliminary experiment with a small grammar, focusing on the cases that might be problematic: we extracted from the corpus a random sample of 100 word forms containing prefixes. From these 100 forms, we generated about 3700 grammatical and ungrammatical test examples by omission, addition and permutation of stems and affixes. After making sure that the required affixes and stems were included in the lexicon of the grammar, we ran a comparison of exact parsing with the unification-based grammar and the selective history-based RTN-approximation, with parameter $d = 2$ (which means that there is a history window of one item). For 97% of the test items, the two methods agreed; 3% of the items were accepted by the approximation method, but not by the full grammar. The approximation does not lose any test items parsed by the full grammar. Some obvious improvements should make it possible soon to run experiments with a larger history window, reaching exactness of the finite-state method for almost all relevant data.

[11] A related possibility pointed out by a reviewer would be to expand features from the original unification-grammar only where necessary (cf. Kiefer and Krieger 2000).

## 9 Acknowledgement

I'd like to thank my former colleagues at the *Institut für Maschinelle Sprachverarbeitung* at the University of Stuttgart for invaluable discussion and input: Arne Fitschen, Anke Lüdeling, Bettina Säuberlich and the other people working in the DeKo project and the IMS lexicon group. I'd also like to thank Christian Rohrer and Helmut Schmid for discussion and support.

## References

Boullier, Pierre. 2003. Supertagging: A non-statistical parsing-based approach. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT'03), Nancy, France*.

Butt, Miriam, Tracy King, Maria-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. Number 95 in CSLI Lecture Notes. Stanford, CA: CSLI Publications.

Grimley-Evans, Edmund. 1997. Approximating context-free grammars with a finite-state calculus. In *ACL*, pp. 452–459, Madrid, Spain.

Johnson, Mark. 1996. Left corner transforms and finite state approximations. Ms., Rank Xerox Research Centre, Grenoble.

Johnson, Mark. 1998. Finite-state approximation of constraint-based grammars using left-corner grammar transforms. In *COLING-ACL*, pp. 619–623, Montreal, Canada.

Kiefer, Bernd, and Hans-Ulrich Krieger. 2000. A context-free approximation of head-driven phrase structure grammar. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT'00), February 23-25*, pp. 135–146, Trento, Italy.

Nederhof, Mark-Jan. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics* 26:17–44.

Pereira, Fernando, and Rebecca Wright. 1997. Finite-state approximation of phrase-structure grammars. In Emmanuel Roche and Yves Schabes (eds.), *Finite State Language Processing*, pp. 149–173. Cambridge: MIT Press.

Schiller, Anne. 1995. DMOR: Entwicklerhandbuch [developer's handbook]. Technical report, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Schmid, Tanja, Anke Lüdeling, Bettina Säuberlich, Ulrich Heid, and Bernd Möbius. 2001. DeKo: Ein System zur Analyse komplexer Wörter. In *GLDV Jahrestagung*, pp. 49–57.

Wurster, Melvin. 2003. Entwicklung einer Wortbildungsgrammatik fuer das Deutsche in YAP. Studienarbeit [Intermediate student research thesis], Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.