Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-93-27

Derivation Without Lexical Rules

Hans-Ulrich Krieger

June 1993

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-67608 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-66123 Saarbrücken, 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 and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, Siemens and Siemens-Nixdorf. 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
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

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.

Friedrich J. Wendl
Director

Derivation Without Lexical Rules

Hans-Ulrich Krieger

DFKI-RR-93-27

A version of this paper will be published in:
Constraints, Language and Computation, C.J. Rupp, M.A. Rosner & R.L. Johnson (eds.), Academic Press, 1993, and
IDSIA Working Paper No. 5, Lugano, November 1991.

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-9002 0).

Derivation Without Lexical Rules*

Hans-Ulrich Krieger
krieger@dfki.uni-sb.de
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, Germany

Abstract

In Krieger and Nerbonne (1992) we showed how to get rid of LEXICAL RULES for DERIVATION, as they are explicated by Pollard and Sag (1987) in HPSG I, Ch. 8.2. We proposed a treatment of derivation NOT by means of traditional lexical rules but instead in terms of PRINCIPLES, RULES, and LEXICAL ENTRIES entirely in the spirit of HPSG, together with unification-based inheritance of a very sophisticated kind. One major disadvantage of this approach was the employment of complex functions in certain principles. In this paper I first extend the old approach and then show how to eliminate these functional dependencies in the domain of derivational morphology by going back to simpler ones like `cons`, `first`, and `rest`. But this simplification is only achieved if we assume more complex feature structures than the ones described in Krieger and Nerbonne (e.g., by introducing two different SUBCAT features) and by proposing modified versions of the old Constituent Order Principle and the Subcategorization Principle for morphology. In addition, I postulate a hierarchy of affixes which is cross-classified, for instance, according to the effects these affixes contribute to the subcategorization information of a compound word.

The structure of the paper is as follows. We start with a very short introduction about the residence of word-formation rules in modern feature-based theories. After that we present our approach to derivational morphology which is distinguished in that it gives up the notion of lexical rule as a single entity (operator). We describe the structure of affixes and words (e.g., which attributes are appropriate?) and introduce the relevant principles and the rule schema of our approach to derivational morphology. The section shows how to reduce functional dependencies to a minimum at the cost of the size of our feature structures. We also present a technique which allows us to state relational dependencies as they are called by HPSG in a functional manner. In the next section we show how the whole treatment works by applying it to tough phenomena from prefixation and suffixation. The section presents many examples, which might serve as a *how to* guide to a practitioner. After that we explain the idea which will lead us to the affix hierarchy. We will see that the affix hierarchy is inspired by the work of HPSG on structured lexicons (i.e., by the hierarchy of lexical types). A lot of examples will again be given throughout this section. We finish the paper by summarizing our approach and by saying a few words about the topics which we will tackle next.

*This work grew out from discussions with John Nerbonne at the DFKI. Thanks are due to Susanne Riehemann and C.J. Rupp for their critique and numerous comments on earlier versions of this paper. The paper has also benefited from numerous people at various workshops where parts of it have been presented, in particular at the *ACQUILEX Workshop on Default Inheritance in the Lexicon* (Cambridge), the *ASL Workshop on DATR* (Bielefeld), the *Workshop on HPSG and German* (Saarbrücken), the *Workshop on Constraint Propagation, Linguistic Description, and Computation* (Lugano), and the *Sprachwissenschaftliches Kolloquium* (Univ. of Tübingen). This work was supported by a research grant from the German Bundesministerium für Forschung und Technologie to the DISCO project (FKZ ITW 9002 0).

Contents

1 Introduction **3**

2 The New Approach to Derivational Morphology **4**
2.1 The Structure of *word* and *affix* 6
2.2 Functional vs. Relational Dependencies 9
2.3 The Principles and the Rule Schema 10

3 Prefixation and Suffixation **17**
3.1 Suffixation 17
3.2 Prefixation 22

4 The Affix Hierarchy **25**
4.1 The `POS` Dimension 26
4.2 The `CAT` Dimension 26
4.3 The `SUBCAT` Dimension 26
4.4 The `SEM` Dimension 28
4.5 The `BIND` Dimension 28
4.6 More Examples 28

1 Introduction

This section treats the question of the RESIDENCE of lexical rules, viz., *where does word formation take place—within or without the lexicon?* WITHOUT means that the form of lexical rules is different from the structure of lexical entries (lexemes, also possibly morphemes). Although lexical rules in PATR-II [38], D-PATR [24], or in the ACQUILEX-LKB formalism [15] look like ordinary feature structure descriptions—actually they are represented via a collection of path equations using the distinguished attributes (metavariables) **IN** and **OUT**—their interpretation however is completely different from that of (normal) feature structure descriptions. The same is true if we move to other theories: f-structures differ in form and interpretation from lexical rules as they are sometimes stated in LFG (cf. the articles in [7]). This observation also holds for the Alvey tools project [36], for the early days of HPSG [20], for the work of Flickinger [19], and for Hoeksema's Categorial Morphology [22]. Lexical rules in HPSG ([34], Ch. 8.2) have the same property, in that feature structure descriptions and lexical rules (form: $\mathrm{AVM}_1 \longmapsto \mathrm{AVM}_2$) have nothing in common because they differ in form as well as in interpretation. This remark is supported by the following observation. Feature structures in HPSG are always typed, and these types can be (partially) ordered by means of subsumption. But this isn't true for lexical rules.[1] A lexical rule as a whole does not have a type and there's no way to relate it to other feature structures. If we assume for the moment that a lexical rule can in principle be typed and resides in the lexicon, this type ought to be a subtype of *lexical-sign* according to HPSG I and ought to have exactly the three top-level attributes **PHON**, **SYN**, and **SEM**—but this isn't the case, showing our assumption to be false.

By its nature, a lexical rule sets up a relation between two lexemes (actually, between classes of lexemes)—or, in the case of feature-based theories, between two feature structure descriptions. But specifying the exact meaning of this mapping is an open question—nearly all theories have different viewpoints when interpreting lexical rules:

- Are lexical rules functions or perhaps even relations?
- Do they take one argument or arbitrarily many?
- Are they unidirectional or bidirectional?
- Will they be interpreted declaratively (AVM_1 implies a corresponding AVM_2) or procedurally (lexical rule as an instruction to build AVM_2 out of AVM_1)?

Lexical rules in HPSG [34] for instance seem to be unary, unidirectional functions. A consequence of this approach to derivation is that every affix (prefixes as well as suffixes) introduces an additional lexical rule to the pool of existing lexical rules. In contrast to the HPSG treatment of derivation, Hoeksema's Categorial Morphology [22] only has a single (binary) rule for prefixation and a single one for suffixation. But in principle, his rules can take arbitrarily many arguments. We will later see that this approach only consists of a single rule (schema), our closest analogue to a lexical rule.

Derivational rules will be MODELED in our approach as feature structure descriptions, just as lexical entries are. This is aesthetically pleasing and has the further advantage of formal clarity because a lexical rule can now be described directly by means of the underlying (feature) logic (for instance Smolka's one [40, 41] or Carpenter's [11]). Perhaps most importantly, however, the fact that "lexical rules" and lexical entries are of the same formal (data) type allows one to liberate yet another level of linguistic structure from procedural considerations and therefore to interleave morphological and phrasal processing in a way that is otherwise prohibited. Thus a

[1] Of course, one can define a type *lexical-rule* (and also subtypes of this type) with two distinguished attributes **IN** and **OUT**. In fact, this is done, for instance, in the LKB formalism of Copestake et al. [15]. These types behave like other types with respect to the subsumption ordering. But as the authors noticed (p. 12), "They differ from normal types in that a lexical rule can be applied to a lexical entry to generate a new lexical entry ...", and this is unfamilar to normal lexical types. Another possibility to cope with the *type* of a lexical rule is by assigning it a functional type (*domain* $\longmapsto$ *range*)—therefore the type hierarchy (strictly speaking: the subsumption ordering) must be generalized to a hierarchy of function types (see for instance [9, 10]).

treatment like this one integrates these two stages because the feature structures directly serve as the interface between them. Another nice advantage in contrast to the HPSG treatment of derivation through lexical rules is that we are able to record the morphosyntactic tree structure of a complex compound word in terms of feature structure descriptions, i.e., it is possible to encode the smaller entities (free and bound morphemes), out of which the new word was built, as parts of the whole structure.

It is worth noting that there is no one-to-one correspondence between traditional lexical rules and objects (feature structures) in our approach—there's no distinguished (unique) element in the lexicon for a given lexical rule. Rather, the information represented in a lexical rule is DISTRIBUTED among lexical entries, principles, and a morphological dominance schema, the only analogue to a lexical rule in this approach.

There are only a few other approaches, which are also distinguished by the absence of traditional lexical rules, e.g., Kathol's work on passive in HPSG [25], Krieger and Nerbonne's treatment of inflection [29], and Russell *et al.*'s approach to inflection and derivation in the ELU system [37].[2]

2 The New Approach to Derivational Morphology

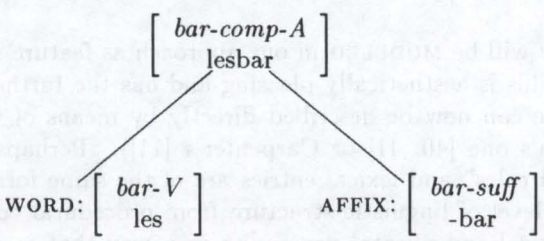
In HPSG I linguistic knowledge about word formation is encoded through a family of lexical rules (see the introductory remarks in [34], p. 209) which are not feature structures, but rather essentially external operators working on feature structures. This unsatisfactory view appears even more questionable given the view of most linguists that form and meaning are much harder to describe for sentences and phrases than for words. If this is the case, one may ask, why does HPSG treat word formation via external lexical rules rather than in a purely feature-based way? Why not formulate RULES and PRINCIPLES for a word grammar similar to those stated by Pollard and Sag in HPSG I for phrasal and sentential grammar? This general idea is the main working assumption of our approach. We think it is a promising task to approach DERIVATION purely in terms of feature structure descriptions—just in the spirit of HPSG. Recall the following equation from [34], p. 147.

(1) $$\textit{English} \;=\; P_1 \wedge \ldots \wedge P_{n+m} \wedge (L_1 \vee \ldots \vee L_p \vee R_1 \vee \ldots \vee R_q)$$

This fundamental equation defines an HPSG grammatical theory for phrases and sentences, and we propose a similar methodology to derivation, relying extensively on rules, principles, and (unification-based) inheritance (for an explanation of (1), see Pollard and Sag, [34], p. 147).

Treating DERIVATION (prefixation as well as suffixation) in our approach will lead to complex words consisting of a head daughter **AFFIX** and a complement daughter **WORD** (see example (2)) under the label **DTRS**. The task of the morphological daughters feature is to encode morphological structure, similarly to how **HEAD-DTR** and **COMP-DTRS** do this on the phrasal level (cf. [34]). This is in analogy to the HPSG formulation of phrase structure in features, yielding tree structures.[3]

(2)



[2] ELU treats inflection as well as derivation by means of pure (naive) inheritance. We are convinced however, that this approach is not strong enough for derivation (for an explanation, see below).

[3] Binary trees (possibly combined with unary ones for $\emptyset$ derivation) seem to suffice here for derivation. There are however problematic examples, for instance the German adjective *vier+bein+ig* (*has four legs*). At least four possible solutions are possible: (i) use a ternary tree (one head and two complements), (ii) still use a binary tree, therefore violate Aronoff's *Word Formation Hypothesis* [3] (*vierbein* and *beinig* are no legal words), (iii) assume that *-beinig* is a complex suffix, (iv) write a lexicon entry for *vierbeinig*.

We include (2) as an example of the hierarchical structure whose analysis is beyond the descriptive reaches of 'naive' inheritance (and of course beyond the power of lexical rules in HPSG which are unable to record such a structure). Here, NAIVE means that a word like for instance the German *weg+laufen* (*run away*) is defined by inheriting (unifying) all the properties from the separable prefix *weg-* and the verb *laufen*, as well as specifying additional idiosyncratic properties for the new complex lexeme, i.e.,

(3) $$weglaufen \;=\; [weg] \wedge [laufen] \wedge [\,\ldots\ldots\,].$$

ELU's treatment of derivation (cf. Russell et al. [37], p. 218) is done in such a way, but in general an approach like this one leads to several insurmountable problems:

- If we relied on naive inheritance as the sole descriptive means, it would seem impossible to explain how the iteration of derivational processes could ever lead to different results. If *anti-* (or take the German *Vor-* or *Ur-*) is a derivational prefix, and its effect on a stem is described via inheritance, then the effect of inheriting it should be the same, whether there are one, two, or more instances of the same prefix in a word because unification is IDEMPOTENT and our notion of inheritance is defined through unification. Thus a complex word like *anti-missile* (or *Vor+version—pre(liminary) version*) would be predicted to be the same as *anti-anti-missile* (or *Vor+vor+version*).[4] Likewise, such an approach is not capable of explaining the INDIRECT recursion occurring in complex compounds such as *institu+tion+al+isa+tion*.

- Sole reliance on naive inheritance leaves little opportunity to explain the hierarchical structure often found in morphology, e.g., the difference in bracketing one finds in complex words containing at least two affixes, e.g., [*un-* [*do -able*]] as opposed to [[*un- do*] *-able*]. Because inheritance is ASSOCIATIVE and MONOTONIC (in the absence of overwriting), other mechanisms must be at play. Naive inheritance seems incapable of accounting for any structure, let alone ambiguous hierarchical structure.

- Simple examination of derivational results suggests that treating all of them via naive inheritance from a single lexeme will lead to unwieldy lexicons: a form such as German *Ableit+bar+keit* (*derivability*) would seem to require that verbal, adjectival, and nominal paradigms be found as heirs of the single lexeme (recall that we dealt with this above by modeling it via mapping from lexeme to lexeme).

- It also turns out that there are technical problems connected with the treatment of derivation as inheritance. These may be summarized, albeit cryptically, in the following way: we should prefer that the result of a category-changing derivational process, e.g., the process which derives *derive+able* from *derive* and *-able*, is a full-fledged member of the target category (of the derivational process)—in this case, the class of adjectives. Now, if the derivational process is modeled by naive inheritance only, then *derive+able* ought to inherit from the class of verbs (through *derive*), as well. It is easy to continue this line of reasoning further (consider *derive+abil+ity*) to see how this sort of explanation leads one to the postulation of lexemes of dubious lineage, inheriting from too many ancestors.

The objections to the description of derivation in terms of naive inheritance do not apply here, since, e.g., tree adjunction is not idempotent—so that, e.g., *Vor+version* may be distinguished from *Vor+vor+version*; tree adjunction generates hierarchical structures (evident here), and, as we shall see, it distinguishes inheritance (sharing properties) from the requirements that sublexemes come from particular word classes or types (so that the tree structure for *lesbar* above cannot be interpreted to mean that the adjective *lesbar* is in any sense a verb of the same type as *lesen* or a suffix of the same type as *-bar*).

[4] Permitting iteration of derivational prefixes only to a certain depth (which seems *prima facie* plausible since, e.g., words such as German *Vor+vor+vor+vor+version* are questionable), will solve this problem, if every element of the finite set of complex prefixes is coded as a lexical entry. But this attempt of repair is (i) theoretically extremely unsatisfying and (ii) incomplete, because the depth of composition is a subjective measure.

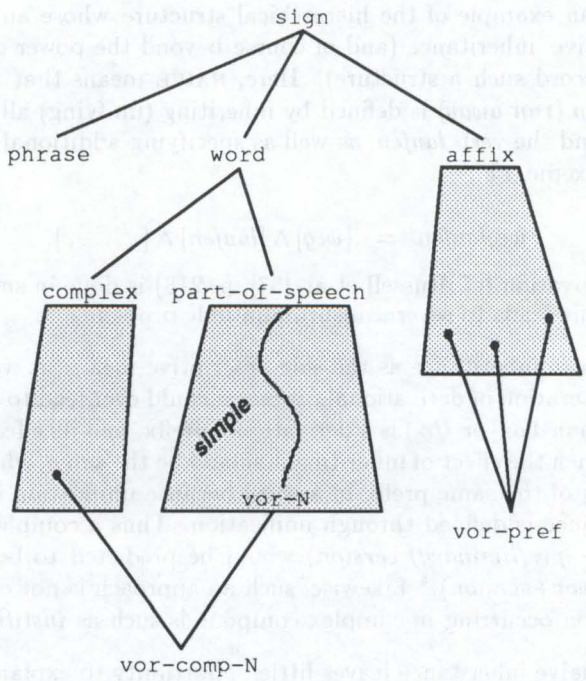


Figure 1: The overall structure of the type lattice for derivation.

Before we're going to apply our approach to derivational morphology and see how the whole treatment works, we must first of all say how the internal structure of a *word* or an *affix* looks like, i.e., which attributes are appropriate for a given type and which type is appropriate for a given attribute. In addition, we have to specify the relevant principles and the single rule schema of our approach to derivational morphology.

2.1 The Structure of *word* and *affix*

Our approach assumes the following top-level attributes for instances of type *word* (or *lexical-sign* according to HPSG I; direct subtypes are *part-of-speech* and *complex*, cf. Fig. 1):

(4)
$$\begin{bmatrix} \textit{word} \\ \texttt{MORPH}\ \textit{word-morphology} \\ \texttt{SYN}\ \textit{word-syntax} \\ \texttt{SEM}\ \textit{word-semantics} \\ \texttt{CORD}\ \textit{list(part-of-speech} \vee \textit{affix)} \\ (\texttt{DTRS}\ \textit{affix-word-struct}) \end{bmatrix}$$

The structure of **MORPH** resp. the type *word-morphology* expects certain attributes like **STEM**, **FORM**, **ENDING**, **PARADIGM** etc.—this is analogous to our old approach presented in [29].[5]

The value of **SEM** is in general a predicate-argument structure (see examples). Representing the propositional semantics of *word*s in such a way is along the lines of HPSG and the situation schemata framework (see for instance [18]).

The attribute **DTRS** is only appropriate for the type *complex* (and of course for subtypes of this type; cf. Fig. 1). **DTRS** expects exactly two (typed) attributes, viz., **AFFIX** and **WORD**.

[5]It might be useful to proclaim an additional attribute **SUBCAT** under **MORPH** which is typed to *affix*. Why so? The next subsection will present TWO alternative subcategorization principles, one (27) having the same formal power as the subcat principle of the old approach, and a stronger one (28) which however assumes a **MORPH|SUBCAT** attribute in instances of type *word*. The intention behind the second subcategorization principle is fairly simple—a free word and an affix only come together if they subcategorize each other.

(5)
$$\begin{bmatrix} \textit{affix-word-struct} \\ \texttt{AFFIX}\ \textit{affix} \\ \texttt{WORD}\ \textit{part-of-speech} \end{bmatrix}$$

The structure of *word-syntax* (SYN) is different to the one given in the old approach with respect to the value of SUBCAT under path SYN|LOC. Instead of proclaiming a subcategorization LIST like HPSG does (type: *list(sign)*), we propose a different type, called *subcat-info*, to be appropriate for SYN|LOC|SUBCAT, which must have exactly the four attributes SUBJ (subject), OBJ (object), OBJ2 (second object), and COMPS (the rest of the complements, the oblique objects).[6]

(6)
$$\begin{bmatrix} \textit{subcat-info} \\ \texttt{SUBJ}\ \textit{sign} \vee \textit{null} \\ \texttt{OBJ}\ \textit{sign} \vee \textit{null} \\ \texttt{OBJ2}\ \textit{sign} \vee \textit{null} \\ \texttt{COMPS}\ \textit{list(sign} \vee \textit{null)} \end{bmatrix}$$

This feature structure replaces the well known subcategorization list of HPSG. The motivation for the new structure is primarily a technical one—affixes and the objects they subcategorize are easier to describe and depict if we assume a keyword approach to grammatical relations instead of using a single list (see examples for more details). Note that there are other remarkable HPSG-oriented proposals which also give up the original subcategorization list in favour of a keyword approach (see [5], [33], [25]).

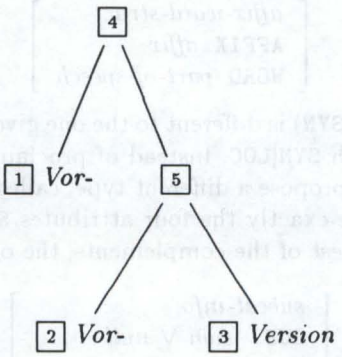
The value of the attribute CORD (Constituent ORDer) in (4) is always a non-empty list of feature structures which are either of type *part-of-speech* or of type *affix*. This list directly reflects the actual linear surface order of the parts of a complex word (cf. the next subsection to see how CORD works together with the Constituent Order Principle (20)).

Note that, in contrast to CORD, the attribute DTRS only states the hierarchical structure of the new compound but NOT the linear order. Let's take an example to clarify the distinction between CORD and DTRS—the German word *Vor+vor+version* (cf. the section on *Vor* prefixation). The analysis tree and the overall structure of *Vorvorversion*, concerning only the attributes MORPH|FORM, CORD, and DTRS, is depicted in (8) and (7). Don't be confused about the cycle [3] in the feature structure of *Version* (the explanation follows when we deal with the Constituent Order Principle).

(7)
$$\boxed{4}\begin{bmatrix} \textit{vor-comp-N} \\ \texttt{MORPH|FORM}\ \texttt{"Vorvorversion"} \\ \texttt{SYN|LOC|LEX}\ + \\ \texttt{CORD}\ \langle \boxed{1}, \boxed{2}, \boxed{3} \rangle \\ \texttt{DTRS}\ \begin{bmatrix} \textit{affix-word-struct} \\ \texttt{AFFIX}\ \boxed{1}\begin{bmatrix} \textit{vor-pref} \\ \texttt{MORPH|FORM}\ \texttt{"Vor-"} \end{bmatrix} \\ \texttt{WORD}\ \boxed{5} \end{bmatrix}, \begin{bmatrix} \textit{vor-comp-N} \\ \texttt{MORPH|FORM}\ \texttt{"Vorversion"} \\ \texttt{SYN|LOC|LEX}\ + \\ \texttt{CORD}\ \langle \boxed{2}, \boxed{3} \rangle \\ \texttt{DTRS}\ \begin{bmatrix} \textit{affix-word-struct} \\ \texttt{AFFIX}\ \boxed{2}\begin{bmatrix} \textit{vor-pref} \\ \texttt{MORPH|FORM}\ \texttt{"Vor-"} \end{bmatrix} \\ \texttt{WORD}\ \boxed{3} \end{bmatrix}, \begin{bmatrix} \textit{vor-N} \\ \texttt{MORPH|STEM}\ \texttt{"Version"} \\ \texttt{SYN|LOC|LEX}\ + \\ \texttt{CORD}\ \langle \boxed{3} \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

[6] Our treatment of subcategorization expects a special type *null*, whose extension is the unique constant NIL. We will use NIL only when we have to state that a grammatical role isn't filled.

(8)



After having inspect the form of words, we must now describe how affixes are internally structured and in what the type *word* (4) differs from the type *affix* (9). *affix* makes use of a new attribute called **COND** (CONDition; see below), but does neither contain the attribute **CORD** nor the attribute **DTRS**. This structural divergence relies on two arguments: (i) affixes do not occur freely and (ii) they are never complex in our approach. Therefore, instances of type *affix* (cf. Fig. 1) must have exactly the following four attributes:

(9)
$$\left[\begin{array}{l} \textit{affix} \\ \textbf{MORPH}\ \textit{affix-morphology} \\ \textbf{SYN}\ \textit{affix-syntax} \\ \textbf{SEM}\ \textit{affix-semantics} \\ \textbf{COND}\ (\top^* \longmapsto \{\top, \bot\}) \end{array}\right]$$

The structure of *affix-morphology* (10) is different to the one of *word-morphology*—*affix-morphology* expects exactly four attributes, viz., **FORM**, **POS**, **ACAT**, and **SUBCAT**. Although a **FORM** attribute is assumed, there's neither a **STEM** nor an **ENDING** attribute as in *word-morphology*.[7] Because of the binary tree property in derivation, the affix takes via its **SUBCAT** attribute only ONE argument, a free word it subcategorizes for of type *part-of-speech*. **ACAT** (Affix CATegory) states whether the affix is classified as a **Prefix** or as a **Suffix**. **POS** (POSition) corresponds to the 'position' of the affix on the surface level under the left-to-right order. Note that **left**, **right**, **Prefix**, and **Affix** are atoms, i.e., they are no further structured.

(10)
$$\left[\begin{array}{l} \textit{affix-morphology} \\ \textbf{FORM}\ \textit{string} \\ \textbf{POS}\ \texttt{left} \vee \texttt{right} \\ \textbf{ACAT}\ \texttt{Prefix} \vee \texttt{Suffix} \\ \textbf{SUBCAT}\ \textit{part-of-speech} \end{array}\right]$$

In languages like English, German, French, or Italian, **left** of course directly corresponds to **Prefix** and **right** to **Suffix**. So we might state the only two LP constraints (11) for derivational morphology which however can be COMPILED INTO the Constituent Order Principle (20) as we will show in the subsection concerning the principles and the rule schema.

(11)
$$\begin{array}{l} \text{AFFIX}[\textbf{POS left}] < \text{WORD} \\ \text{AFFIX}[\textbf{POS right}] > \text{WORD} \end{array}$$

The structure of *affix-syntax* under the attribute **SYN** in (9) looks pretty much the same as the one for *word-syntax*, i.e., there is, for instance, a **HEAD** and a **SUBCAT** attribute under the path

[7] Suffixes like the German *-chen* and *-lein* in words like *Lämmchen/Lämmlein* (*little lamb*), which differ only in their **FORM** value but behave identical in everything else, will be represented in our approach as instances of the same suffix type but with a different surface realization. However, if we had proposed a **STEM** attribute for affixes, it would make sense to proclaim a single abstract stem, both for *-chen* and *-lein*.

SYN|LOC. This is due to the fact that the Head Feature Principle (24) and the Subcategorization Principle (27) in our approach treat the affix as the HEAD, therefore the SUBCAT and the HEAD values of the affix under path SYN|LOC are coindexed with the same attributes of the mother. Of course, the value for SYN|LOC|SUBCAT is no longer a list—only feature structures of type *subcat-info* (6) are appropriate (this type is also the appropriate one for SYN|LOC|SUBCAT in *word*).

The last attribute, COND, defaults to $\top$ (this is allowed as the result of the Kleene star in (9)) and is restricted to the FUNCTION TYPE ($\top^* \longmapsto \{\top, \bot\}$), i.e., the value of COND leads either to a unification failure, because $[\ldots,$ COND $\bot, \ldots] \doteq \bot$, or leaves the whole structure untouched as a consequence of the fact that $\top$ always unifies with everything else. The intention behind COND might sound at the moment a little bit mystical—it is possible to couch relational dependencies purely functionally.

2.2 Functional vs. Relational Dependencies

RELATIONAL DEPENDENCIES in HPSG can indeed be (re-)formulated functionally. A relational dependency, which is associated with a specific feature structure (type), must hold for the entire feature structure to be a legal one. But if the condition fails, the feature structure isn't licensed by that constraint. Thus, if the unification machinery forces us to state only functional constraints (dependencies) but not relational ones (which is the case in our DISCO project), we must represent relational dependencies in a different way. The technique which we will present here, allows us to embed a restricted relational subset, viz. the set of predicates (in constrast to constructors like **cons**), in a feature-value logic consisting of function symbols and the propositional connectives $\wedge$, $\vee$, and $\neg$. Note that the class of functions to which COND is typed ($\top^* \longmapsto \{\top, \bot\}$), correponds to the class of CHARACTERISTIC FUNCTIONS, known from the theory of computation. To give a flavor how the whole mechanism works, take, for instance, the simplified version of rule 4 (head-adjunct rule) in Pollard and Sag [34], p. 161.

(12)
$$\left[\text{DTRS}\ \left[\begin{array}{l}\text{HEAD-DTR|SYN|LOC|HEAD|ADJUNCTS}\ \{\ldots,\boxed{1},\ldots\}\\ \text{ADJ-DTR|SYN}\ \boxed{1}\end{array}\right]\right]$$

(12) can be rewritten to an equivalent feature structure (13), where the membership of $\boxed{1}$ is represented as a functional constraint under attribute COND and **member** is a function yielding either the value $\top$ or $\bot$.

(13)
$$\left[\begin{array}{l}\text{DTRS}\ \left[\begin{array}{l}\text{HEAD-DTR|SYN|LOC|HEAD|ADJUNCTS}\ \boxed{2}\\ \text{ADJ-DTR|SYN}\ \boxed{1}\end{array}\right]\\ \text{COND}\ \texttt{member}(\boxed{1},\boxed{2})\end{array}\right]$$

Moving back to the structure of *affix* (9), everything works fine (is monotonic) in all subtypes of the type *affix* when inheriting the properties of their supertypes because COND should be given the value $\top$ if no functional constraint is needed and the COND attributes of *affix* and all its subtypes must be typed to functions which either yield $\top$ or $\bot$ (and of course $\top \wedge \top \doteq \top$ and $\top \wedge \bot \doteq \bot$ is the case).[8]

The COND attribute will gain importance when we're going to formulate affix types. We will see that the affix hierarchy consists of types which cannot be described by a 'normal' feature logic without functions or relations (see section on the affix hierarchy). Without functions, we're not able to specify known generalizations about affixes in a proper and convenient way, or in other words, the type definitions turn out to be not specific enough, if we simply omit existing functional dependencies in a feature structure.

The functions under the COND attribute serve as PREDICATES as we have seen above. Besides those predicates we need other functions as well to build up new structures (CONSTRUCTORS) or to access specific elements (ACCESSORS). This approach currently employs only **cons**, **first**, and **rest** in certain affix types and in the Constituent Order Principle. Applying them only

[8] In our context, inheriting functions which act as predicates means, to conjoin their results. We expect functions to be RESIDUATED [2, 42] if there's not enough information present to construct the 'full' answer $\top$ or $\bot$.

leads to the following nice advantages: (i) the functions are simple in the sense of having a LOW COMPUTATIONAL COMPLEXITY—actually **cons**, **first**, and **rest** are at least needed to work on lists, and (ii) the constructor **cons** is one-to-one and therefore reversible in the sense that there exists an inverse function and not only a relation (in contrast to **append**), i.e., if we take a list, the first element of this list is uniquely determined, as is the rest. Equation (14) always holds for every list l.

(14) $$cons(first(l), rest(l)) \doteq l$$

first and **rest** might be real functions, but they can also be regarded as macro facilities to abbreviate the internal structure of lists. Take for instance the two element list **<1,2>**. This list might be a shorthand for the more elaborated feature structure representation **[FIRST 1, REST [FIRST 2, REST NIL]]**, where **FIRST**, **REST**, and **NIL** are distinguished symbols. Therefore, the following two feature structures are equivalent in terms of the above remark:

(15)
$$\begin{bmatrix} \texttt{a}\ \boxed{1} \\ \texttt{b}\ \texttt{first}(\boxed{1}) \\ \texttt{c}\ \texttt{rest}(\boxed{1}) \end{bmatrix} \equiv \begin{bmatrix} \texttt{a}\ [\texttt{FIRST}\ \boxed{1}, \texttt{REST}\ \boxed{2}] \\ \texttt{b}\ \boxed{1} \\ \texttt{c}\ \boxed{2} \end{bmatrix}$$

The same argumentation holds for the constructor **cons** which we will use in (20)—**cons** too can be seen as a macro. Take for instance the following two equivalent feature structures:

(16)
$$\begin{bmatrix} \texttt{a}\ \boxed{1} \\ \texttt{b}\ \boxed{2} \\ \texttt{c}\ \texttt{cons}(\boxed{1},\boxed{2}) \end{bmatrix} \equiv \begin{bmatrix} \texttt{a}\ \boxed{1} \\ \texttt{b}\ \boxed{2} \\ \texttt{c}\ [\texttt{FIRST}\ \boxed{1}, \texttt{REST}\ \boxed{2}] \end{bmatrix}$$

Our approach and the examples to come expect that the following equations always hold for **first**, **rest**, and **NIL**. Note that **NIL** is overloaded with respect to its type, i.e., **NIL** denotes the empty list **< >** as well as a special symbol.

(17)
$$\begin{aligned} &eq(<>, NIL) \doteq \top, \\ &first(<>) \doteq NIL, \\ &rest(<>) \doteq NIL. \end{aligned}$$

2.3 The Principles and the Rule Schema

After we have given an overview about the general structure of affixes and words, it's now the right time to present the principles and the single rule schema. In analogy to the rules stated in HPSG I, [34], Ch. 6, we postulate a very general immediate dominance schema *MAWR* (Morphological Affix-Word Rule) which is responsible for licensing complex words (see the feature structure (7) of *Vorvorversion* as a good example for a multiple application of (18)).

(18)
$$MAWR = \begin{bmatrix} \textit{complex} \\ \texttt{SYN|LOC|LEX}\ + \\ \texttt{DTRS} \begin{bmatrix} \textit{affix-word-struct} \\ \texttt{AFFIX}\ \textit{affix} \\ \texttt{WORD}\ \textit{part-of-speech} \end{bmatrix} \end{bmatrix}$$

or informally as an ID rule

(19) $$\text{M[LEX +]} \longrightarrow \text{A, W.}$$

Note that the value of **SYN|LOC|LEX** is **+**, i.e., the whole structure is classified as a (legal) word. This is necessary because **LEX** is NOT a **HEAD** feature and will therefore not be percolated via the Head Feature Principle (24) to the mother. Note too that **DTRS|WORD|SYN|LOC|LEX** is also **+** as the result of typing **DTRS|WORD** to *part-of-speech*.

Just as HPSG proclaim universal as well as language-specific principles, we will now define and motivate FIVE MORPHOLOGICAL PRINCIPLES, which are consistent with our linguistic data and specified as typed implications: (i) a morphological CONSTITUENT ORDER PRINCIPLE *MCOP*, (ii) a morphological SURFACE REALIZATION PRINCIPLE *MSRP*, (iii) a morphological HEAD FEATURE PRINCIPLE *MHFP*, (iv) a morphological SEMANTICS PRINCIPLE *MSP*, and (v) a morphological SUBCATEGORIZATION PRINCIPLE *MSCP*.

Our old Constituent Order Principle (cf. [29]) employed a functional dependency to determine the **MORPH|FORM** value of the new complex word.[9] One of our main goals in this paper, however, is to reduce the NUMBER and COMPLEXITY of functional dependencies to a minimum, making the INHERENT KNOWLEDGE of such constraints explicit in feature structure descriptions and using functions only if the problem we want to describe can no longer be stated in a feature logic without functional constraints. Encoding linguistic knowledge in this way is at least interesting from three different viewpoints:

- We have to make our knowledge explicit in a high level language (feature logic) and cannot rely on 'low level' functions (maybe written in Lisp). We cannot longer say *OK, here's a function which does the job for me*—this was the case in our old approach when we used the function `order-constituents`, but didn't say how `order-constituents` really works.
- The whole unification machinery becomes much slender because it is set free from handling functional constraints, at least from complex ones.
- Leaving out functional constraints or restricting oneself to reversible functions becomes important in the light of reversibility—only ONE grammar is needed, both for parsing and generation.

In contrast to the constituent order principle of HPSG which must 'only' capture the linear precedence of the parts of a sentence, the old Constituent Order Principle however was concerned with two jobs—MORPHOTACTICS and MORPHOPHONEMICS (allomorphy).[10] We will now show how to split the functionality of the old principle into two new principles.

MCOP (20) does the LINEARIZATION part of the old principle, i.e., it determines in what order words and affixes occur on the surface level of the new compound. This (ordered) sequence is then stored under the attribute **CORD** (see above). But *MCOP* only applies the very simple function **cons** in contrast to `order-constituents`, stated before.

MSRP (23) is concerned with the SURFACE REALIZATION of the new word. The actual realization of the word is assigned to the attribute **MORPH|FORM**. Of course, *MSRP* applies the function `realize-surface`, but this function is set free from the whole morphotactics and handles 'only' the morphophonemics. It MIGHT directly correspond to a call of a two-level morphological component [26], i.e., as a foreign function interface to a system outside of the apparatus of feature logic. We will have nothing to say about this here, although we are convinced that allomorphy can be handled entirely in a feature unification-based framework (see for instance the novel work of Bird on finite-state phonology in HPSG [4] and Krieger *et al.* [30] on feature-based allomorphy, showing for the first time how to encode and process two-level automata within typed feature formalisms; cf. [27] for the general framework).[11]

[9] The form of the old *Constituent Order Principle* was mainly inspired by HPSG ([34], p. 169):

$$\begin{bmatrix} \textit{complex} \\ \textbf{DTRS}\ \textit{affix-word-struct} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{complex} \\ \textbf{MORPH|FORM}\ \texttt{order-constituents}\langle \boxed{1} \rangle \\ \textbf{DTRS}\ \boxed{1} \end{bmatrix}$$

[10] Linear precedence in HPSG is given by a set of linear precedence (LP) constraints (actually, there are two kinds of LP statements, viz., $X < Y$ and $X \ll Y$; for a detailed explanation, see [34], Ch. 7). This sort of linear precedence roughly corresponds to morphotactics, and in fact, we have given above the only two LP constraints for derivational morphology (11) which we will compile into the new principle and the feature structure descriptions for affixes. However, there is no pendant to morphophonemics on the sentence level (except perhaps the behaviour of clitics).

[11] The general rule of thumb for derivation in German is to concatenate the affix and the stem or the former constructed complex word in case of multiple-affix application. However, there are many exceptions to this rule, for example *ess+bar* → *eßbar* (*edible*) or *entschuldig+bar* → *entschuldbar* (*excusable*).

A treatment of such a kind allows us to reduce the complexity originally expressed in the old constituent order principle by dividing the whole work into two parts. This is only possible because we make a clear distinction between morphotactics and morphophonemics. Note that *MAWR* (18) and *MCOP* (20) are responsible for the WHOLE morphotactics.

(20)
$$MCOP = \left\{ \begin{bmatrix} complex \\ \texttt{DTRS}\ affix\text{-}word\text{-}struct \end{bmatrix} \Longrightarrow \begin{bmatrix} complex \\ \texttt{DTRS} \begin{bmatrix} \texttt{AFFIX}\ \boxed{1}\,[\texttt{MORPH|POS}\ \boxed{3}] \\ \texttt{WORD|CORD}\ \boxed{2} \end{bmatrix} \\ \texttt{CORD}\ \texttt{cons}(\boxed{1},\boxed{2},\boxed{3}) \end{bmatrix} \right\}$$

The 3-place **cons** in *MCOP* is a generalized version of the binary function **cons**—the third argument (the value of **POS**) determines whether the first argument will be the first (**[POS left]**) or the last element (**[POS right]**) of the new list. Another possibility to do things right is to use a binary, polymorphical version of **cons** which determines with respect to the type of its first argument (type of the value of the the **AFFIX** attribute) whether to cons left or right—in this case, the attribute **POS** can be omitted.

(21)
$$\begin{aligned} cons(first, rest, left) &= < first\ .\ rest > \text{ vs.} \\ cons([prefix], rest) &= < [prefix]\ .\ rest > \\ cons(first, rest, right) &= < first\ .\ rest^{-1} >^{-1} \text{ vs.} \\ cons([suffix], rest) &= < [suffix]\ .\ rest^{-1} >^{-1} \end{aligned}$$

The incorporation of the two LP constraints (11) via **cons** into the feature structure description of *MCOP* (20) has the nice advantage that neither additional principles nor new rule schemata must be introduced. From a processing point of view, this compilation is also interesting because we are no longer forced to say when to evaluate the LP constraints for morphology.

This version of the Morphological Constituent Order Principle *MCOP* assumes that the **CORD** attribute is also appropriate for *simple* words (words having no internal constituent structure, i.e. no attribute **DTRS**) and not only for *complex* ones.[12] This is due to the fact that the second argument of **cons** in *MCOP* (20) has to be a LIST and this argument is coindexed with the value of **DTRS|WORD|CORD** via $\boxed{2}$, which might be a simple word. In the case of *simple* words, the value of **CORD** is a list containing the WHOLE feature structure ITSELF as the only element! Notice that a feature structure like (22) is forbidden according to HPSG I, p. 37.

(22)
$$\boxed{1} \begin{bmatrix} vor\text{-}N \\ \texttt{MORPH|STEM "Version"} \\ \ldots\ldots \\ \texttt{CORD} < \boxed{1} > \end{bmatrix}$$

A more general example, viz. the analysis tree (8) and the feature structure (7) of *Vorvorversion*, was given in the previous subsection. With assistance of the new constituent order principle and the assumption that simple words refer to themselves via **CORD**, we can now verify that (7) depicts indeed the legal structure for *Vorvorversion*. Because *Vor-* is classified as a prefix, we know that the value of its **POS** attribute must be **left**, i.e., the **cons** in *MCOP* is enforced to cons *Vor-* left and this new sequence is then stored under the **CORD** attribute of the mother.

However, we're NOT directly concerned with the morphological surface realization of the new compound—that's the main reason why we propose the functional dependency **realize-surface** in the Surface Realization Principle *MSRP* (23). Because *MSRP* is liberated from the whole morphotactics, **realize-surface** is in fact less complex than the function **order-constituents** of the old constituent order principle.

[12] Using TYPE NEGATION, one can easily define a type *simple* abbreviating a certain sublattice in Fig. 1 which contains only simple words (note the marked region in Fig. 1): $simple := part\text{-}of\text{-}speech \wedge \neg complex$.

(23)
$$MSRP = \left\{ \begin{array}{l} \left[\begin{array}{l} \textit{complex} \\ \texttt{DTRS}\ \textit{affix-word-struct} \end{array} \right] \Longrightarrow \\ \left[\begin{array}{l} \textit{complex} \\ \texttt{MORPH|FORM realize-surface}(\boxed{1},\boxed{2}) \\ \texttt{CORD}\ \boxed{1} \\ \texttt{DTRS}\ \boxed{2} \end{array} \right] \end{array} \right.$$

Likewise for derivation, the formulation of the Morphological Head Feature Principle was taken over from HPSG ([34], p. 58)—only certain attributes and type names were altered. Among other things, *MHFP* is responsible for deducing the category of a new word from the category of the head daughter.

(24)
$$MHFP = \left\{ \begin{array}{l} \left[\begin{array}{l} \textit{complex} \\ \texttt{DTRS}\ \textit{affix-word-struct} \end{array} \right] \Longrightarrow \\ \left[\begin{array}{l} \textit{complex} \\ \texttt{SYN|LOC|HEAD}\ \boxed{1} \\ \texttt{DTRS|AFFIX|SYN|LOC|HEAD}\ \boxed{1} \end{array} \right] \end{array} \right.$$

The Morphological Semantics Principle corresponds to the simplest version of HPSG I ([34], p. 99): the semantics of the mother is equal to the semantics of the head daughter.

(25)
$$MSP = \left\{ \begin{array}{l} \left[\begin{array}{l} \textit{complex} \\ \texttt{DTRS}\ \textit{affix-word-struct} \end{array} \right] \Longrightarrow \\ \left[\begin{array}{l} \textit{complex} \\ \texttt{SEM}\ \boxed{1} \\ \texttt{DTRS|AFFIX|SEM}\ \boxed{1} \end{array} \right] \end{array} \right.$$

In order to eliminate the complex function **construct-subcat** in the subcategorization principle of the old approach [29], we make a distinction between MORPHOLOGICAL and SYNTACTIC subcategorization (see example (26)): an affix looks for the right feature structure to bind (morphological subcategorization), but the subcategorization information of the new complex word (its sentential subcategorization) directly comes from the syntactic subcategorization of the affix by means of structure sharing.[13] We will see that the new Morphological Subcategorization Principle *MSCP* makes no reference to a function any longer—we use equality (structure sharing) only. To make this approach a bit more concrete, we take a look on the almost regular suffix *-bar* (cf. the subsection on *bar* suffixation) and then move straight on to the new Subcategorization Principle.

Feature structure (26) classifies *-bar* as a suffix via the **ACAT** (resp. **POS**) attribute, but syntactically *-bar* must be regarded, as a consequence of the head feature principle, as an adjective (because of **[SYN|LOC|HEAD|MAJ A]**). Note that it subcategorizes for a *bar* verb via **MORPH|SUBCAT**. The structure sharing between attributes under **MORPH|SUBCAT|SYN|LOC|SUBCAT** and **SYN|LOC|SUBCAT** guarantees that *MSCP* transports the right subcat information (the one under **SYN|LOC|SUBCAT**) from the head (the affix) to the mother. Note too that we are sometimes forced to state that the value of **OBJ** or **OBJ2** is **NIL**, or to assign the empty list (< >) to **COMPS** (this can be technically achieved via subtypes of (a more general) *subcat-info*).

[13] The form of the old subcategorization principle for morphology differs from the sentential subcat principle presented by Pollard and Sag in [34], p. 71:

$$\left[\begin{array}{l} \textit{complex} \\ \texttt{DTRS}\ \textit{affix-word-struct} \end{array} \right] \Longrightarrow \left[\begin{array}{l} \textit{complex} \\ \texttt{SYN|LOC|SUBCAT construct-subcat}(\boxed{1}) \\ \texttt{DTRS}\ \boxed{1} \left[\begin{array}{l} \texttt{AFFIX|SYN|LOC|SUBCAT}\ \boxed{2} \\ \texttt{WORD}\ \boxed{2} \end{array} \right] \end{array} \right]$$

(26) *bar* =

$$\left[\begin{array}{l} \textit{bar-suff} \\ \textbf{MORPH} \left[\begin{array}{l} \textit{affix-morphology} \\ \textbf{POS right} \\ \textbf{FORM "-bar"} \\ \textbf{ACAT Suffix} \\ \textbf{SUBCAT} \left[\begin{array}{l} \textit{bar-V} \\ \textbf{SYN|LOC|SUBCAT} \left[\begin{array}{l} \textit{subcat-info} \\ \textbf{OBJ } \boxed{1} \\ \textbf{COMPS } \boxed{2} \end{array}\right] \end{array}\right] \boxed{3} \end{array}\right] \\ \textbf{SYN|LOC} \left[\begin{array}{l} \textit{local} \\ \textbf{HEAD|MAJ A} \\ \textbf{SUBCAT} \left[\begin{array}{l} \textit{subcat-info} \\ \textbf{SUBJ } \boxed{1} \\ \textbf{OBJ NIL} \\ \textbf{OBJ2 NIL} \\ \textbf{COMPS } \boxed{2} \end{array}\right] \end{array}\right] \\ \textbf{SEM} \left[\begin{array}{l} \textit{affix-semantics} \\ \textbf{OPERATOR } \diamond \\ \textbf{SCOPE } \boxed{3} \end{array}\right] \end{array}\right]$$

The following Morphological Subcategorization Principle *MSCP* takes care that the AFFIX will bind the right WORD via **DTRS|AFFIX|MORPH|SUBCAT** (value: single feature structure of type *part-of-speech*) and additionally determines the right subcategorization information of the new COMPOUND word through **DTRS|AFFIX|SYN|LOC|SUBCAT** (typed to *subcat-info*):

(27) *MSCP* =

$$\left\{\begin{array}{l} \left[\begin{array}{l} \textit{complex} \\ \textbf{DTRS } \textit{affix-word-struct} \end{array}\right] \Longrightarrow \\ \left[\begin{array}{l} \textit{complex} \\ \textbf{SYN|LOC|SUBCAT } \boxed{1} \\ \textbf{DTRS} \left[\begin{array}{l} \textbf{AFFIX} \left[\begin{array}{l} \textbf{SYN|LOC|SUBCAT } \boxed{1} \\ \textbf{MORPH|SUBCAT } \boxed{2} \end{array}\right] \\ \textbf{WORD } \boxed{2} \end{array}\right] \end{array}\right] \end{array}\right.$$

The reason for the simplicity of *MSCP* arises from the new structure of the affixes: the distinction between morphological and syntactic/sentential subcategorization and the complex structure sharing of information between *subcat-info* attributes under **MORPH|SUBCAT|SYN|LOC|SUBCAT** and **SYN|LOC|SUBCAT** guarantees that *MSCP* works proper.[14]

However, it might be useful to have even a STRONGER version of the Subcategorization Principle (28). Such a stronger principle is often useful to handle SUBREGULARITIES and EXCEPTIONS in word class definitions—it helps to avoid the definition of additional subtypes of a given type or the introduction of non-monotonic extensions to the type system resp. to the unifier. We will study an example in the next section where the more restricted Subcategorization Principle is in fact applied to exceptions occuring in the context of *bar* suffixation. (28) presumes that the affix AND the word will SUBCATEGORIZE EACH OTHER. As we mentioned above, this version also assumes that **MORPH|SUBCAT** is appropriate for the type *part-of-speech*, i.e., a free word (the lexeme) also possesses two **SUBCAT** attributes.

[14] Moreover, the binary tree property we assume for derivation here, allows us to get rid of the function append used by Pollard and Sag for sentential subcategorization (we need only structure-sharing). This approach is also capable of working with non-binary trees (introduce a list of complements) like HPSG recommends for the sentence level, because a list append must not be seen necessarily as a function/relation but can be simulated through the difference list technique used in the logic programming community [32] or via a recursive type specification, as Aït-Kaci has shown in [1] (note that we won't stick to functions or even relations as long as possible).

(28) $$MSCP^{*} = \left\{ \begin{array}{l} \left[\begin{array}{l} \textit{complex} \\ \texttt{DTRS}\ \textit{affix-word-struct} \end{array} \right] \Longrightarrow \\ \left[\begin{array}{l} \textit{complex} \\ \texttt{SYN|LOC|SUBCAT}\ \boxed{1} \\ \texttt{DTRS} \left[\begin{array}{l} \texttt{AFFIX}\ \boxed{3} \left[\begin{array}{l} \texttt{SYN|LOC|SUBCAT}\ \boxed{1} \\ \texttt{MORPH|SUBCAT}\ \boxed{2} \end{array} \right] \\ \texttt{WORD}\ \boxed{2} \left[\texttt{MORPH|SUBCAT}\ \boxed{3} \right] \end{array} \right] \end{array} \right] \end{array} \right.$$

Although Pollard and Sag strictly type the attributes of feature structures in general, they do not explicitly state that PRINCIPLES as well as RULES may also be regarded as types. But we may interpret them as types which have to satisfy the SUBSUMPTION relation only.[15] In taking this step, one has to INTEGRATE them CONSISTENTLY into the subsumption lattice (cf. the types $MHFP_c$, $MSCP_c$, MSP_c, $MCOP_c$, and $MSRP_c$ in Fig. 2).

With respect to equation (1), we extend the set of principles and the set of rules by adding *MCOP*, *MSRP*, *MHFP*, *MSP*, *MSCP*, and *MAWR* to them. Finally, in typing the antecedents of the implications, we must take care, since not every principle can be combined with every rule or lexical entry. Because only morphological affix-word structures are examined in this paper, equation (1) allows us to unify the feature structure descriptions associated with (the right-hand sides of) *MCOP*, *MSRP*, *MHFP*, *MSP*, *MSCP*, and *MAWR* (call the result *AWR&Ps*), and to regard this feature structure as a RESTRICTION (a constraint, a filter) for all feature structures belonging to this new type. All complex words containing the attribute DTRS must satisfy this type restriction, i.e., must be of the type *AWR&Ps*, or equivalently, *AWR&Ps* states what is common to ALL morphologically complex words.

(29) $$AWR\&Ps = MHFP_c \wedge MSCP_c \wedge MSP_c \wedge MCOP_c \wedge MSRP_c \wedge MAWR$$

(30) $$AWR\&Ps = \left[\begin{array}{l} \textit{complex} \\ \texttt{MORPH|FORM realize-surface}(\boxed{4},\boxed{5}) \\ \texttt{SYN|LOC} \left[\begin{array}{l} \textit{local} \\ \texttt{LEX} + \\ \texttt{HEAD}\ \boxed{6} \\ \texttt{SUBCAT}\ \boxed{8} \end{array} \right] \\ \texttt{SEM}\ \boxed{7} \\ \texttt{CORD}\ \boxed{4}\ \texttt{cons}(\boxed{1},\boxed{2},\boxed{3}) \\ \texttt{DTRS}\ \boxed{5} \left[\begin{array}{l} \textit{affix-word-struct} \\ \texttt{AFFIX}\ \boxed{1} \left[\begin{array}{l} \textit{affix} \\ \texttt{MORPH} \left[\begin{array}{l} \textit{affix-morphology} \\ \texttt{POS}\ \boxed{3} \\ \texttt{SUBCAT}\ \boxed{9} \end{array} \right] \\ \texttt{SYN|LOC} \left[\begin{array}{l} \textit{local} \\ \texttt{HEAD}\ \boxed{6} \\ \texttt{SUBCAT}\ \boxed{8} \end{array} \right] \\ \texttt{SEM}\ \boxed{7} \end{array} \right] \\ \texttt{WORD}\ \boxed{9} \left[\begin{array}{l} \textit{part-of-speech} \\ \texttt{CORD}\ \boxed{2} \end{array} \right] \end{array} \right] \end{array} \right]$$

Trying to encode rules and principles explicitly as elements of a type subsumption lattice (inheritance network) along the lines of HPSG ([34], Ch. 8), requires a 'rewriting' step. Because of their implicative nature, we cannot state principles DIRECTLY as types in a distributive lattice (or

[15] Principles CONSTRAIN existing types and therefore must be interpreted as supertypes. In translating a principle—usually expressed as a conditional—into a type, we only use the the right side of the conditional, the consequent (for a motivation, see below). Note the similarity between principles here and *completion rules* in the Alvey tools project [36].

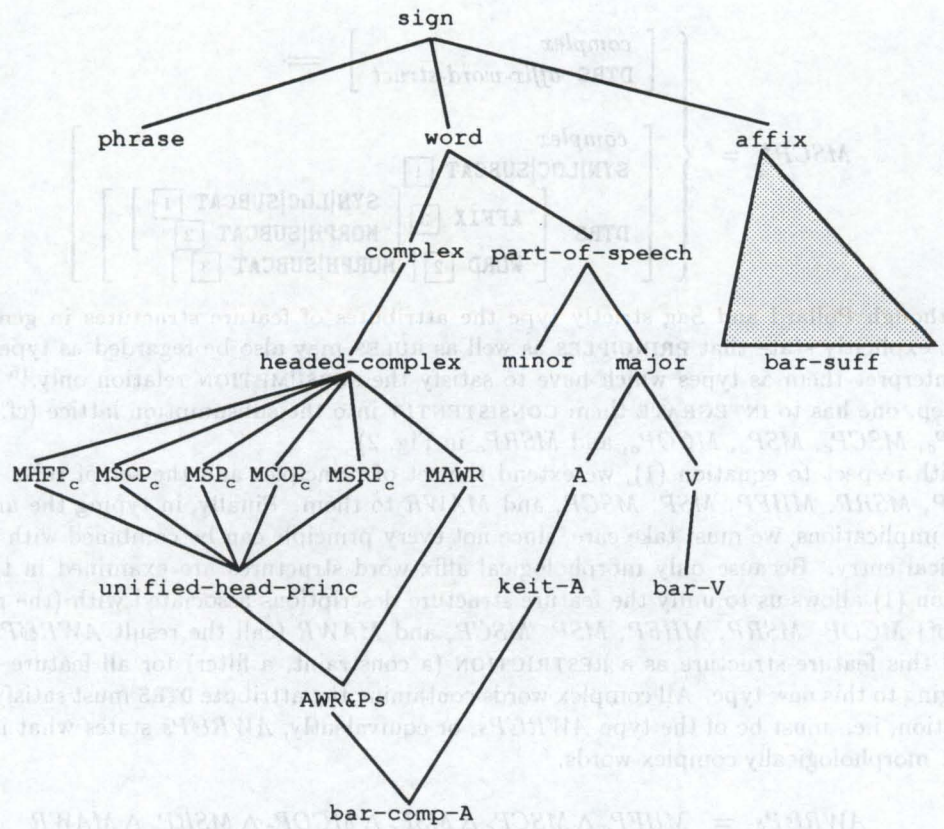


Figure 2: Structure of the inheritance network for *bar* suffixation (see next section), regarding the principles and the rule. Note that we additionally impose LOCAL constraints on certain classes, especially on *bar-comp-A*; for motivation, see text. Note further, that although the class of adjectives formed using *-bar* inherits from *keit-A* (*keit* adjectives) and from *AWR&Ps*—it does NOT inherit from either of its component morphs, *bar-V* or *bar-suff*.

even in a more weaker topology)—we must rewrite them. REWRITING means first, that only the right side (the consequent) of an implication will be regarded as a type. Second, in order to obtain the force of the antecedent, the type associated with the conjunctive feature structure representing the consequent has to be integrated into the 'right' position in the lattice (cf. *AWR&Ps* in Fig. 2), where RIGHT is determined by *taking care that the subsumption relation holds*.[16] Even an equation like (1), containing lots of implications, can then be compiled to form an inheritance hierarchy, consisting only of conjunctive feature types.[17] The idea of reducing implications to conjunctive types will lead us directly to the structure of the type/class subsumption lattice (cf. Fig. 2).

In the following, we will further motivate and exemplify our approach to derivation by applying it to examples from SUFFIXATION and PREFIXATION.

[16] In general, there's only one right position—*the most general position at which the subsumption relation holds.* But this is only true if we assume a (subsumption) lattice where subsumption is strict, i.e., where it is not possible to have two different types standing in a subsumption relation, even though their denotation is the same.

[17] The rewriting step is subtle in that it moves information from object-language implicational statements into restrictions in the type hierarchy which forms the skeleton of the interpretation. On the one hand, because of general laws of interpretation for feature logics, we have the following inference for the feature structures *Ante* and *Conseq*: from the principle $Ante \Rightarrow Conseq$, we know that $Ante \sqsubseteq Conseq$. On the other hand, the principles always add information to a feature structure description to which they are applied, so that *Ante* always subsumes *Conseq*, i.e., $Ante \sqsupseteq Conseq$. This leads to an effective identification of *Ante* and *Conseq* which is realized in the type hierarchy.

3 Prefixation and Suffixation

This section is intended to show how the principles, the single rule schema, and the lexical entries fit together by applying them to concrete examples. We show that the approach presented up to here is capable of handling really tough phenomena from the domain of derivational morphology. In addition, we argue that subregularities and exceptions to a certain degree can be handled by the stronger Subcategorization Principle *MSCP** (28). In general, capturing such non-regular data through a non-monotonic device might be the better way.[18] The non-monotonic mechanism we would like to employ during this section is termed SINGLE LINK OVERWRITING (SLO).[19]

3.1 Suffixation

The treatment of German *bar* suffixation is interesting from different points of view and presents severe problems, which can however be adequately solved in our approach.[20]

- **sporadic applicability** *-bar* suffixes many verbs but not all,
- **partial regularity** many *bar* derivatives have regular forms but irregular semantics or irregular forms with regular syntax and semantics,
- **category change** *bar* suffixation changes (syntactic) category (TV⇝A),
- **subcategorization change** the subcategorization information of V+*bar* changes, the semantic argument positions in the scope of *-bar*, on the other hand, do not change.

Starting with a verb like the German *lesen* (*to read*), where *bar* suffixation is perfectly regular, we may construct a possible lexicon entry with respect to the inheritance network of Fig. 2.

(31)

$$
\textit{lesen} = \boxed{3}\left[\begin{array}{l}
\textit{bar-V} \\
\texttt{MORPH|STEM "les"} \\
\texttt{SYN|LOC}\left[\begin{array}{ll}
\texttt{LEX} & + \\
\texttt{HEAD|MAJ} & \texttt{V} \\
\texttt{SUBCAT} & \left[\begin{array}{l}
\textit{subcat-info} \\
\texttt{SUBJ}\ \textit{NP}\boxed{1} \\
\texttt{OBJ}\ (\textit{NP})\boxed{2} \\
\texttt{OBJ2 NIL} \\
\texttt{COMPS}\ <\ldots>
\end{array}\right]
\end{array}\right] \\
\texttt{SEM}\left[\begin{array}{l}
\texttt{RELN read'} \\
\texttt{SOURCE}\ \boxed{1} \\
\texttt{THEME}\ \boxed{2}
\end{array}\right] \\
\texttt{CORD}\ <\boxed{3}>
\end{array}\right]
$$

Notice that although *lesen* is syntactically classified as a verb (V), it is an instance of the class *bar-V* (verbs that may combine with *-bar*). Note also that we employ here the LEXEME *lesen* rather than, e.g., the infinitive in *lesen*'s paradigm—this is compatible with the fact that only the stem *les-* is found in the derived word.

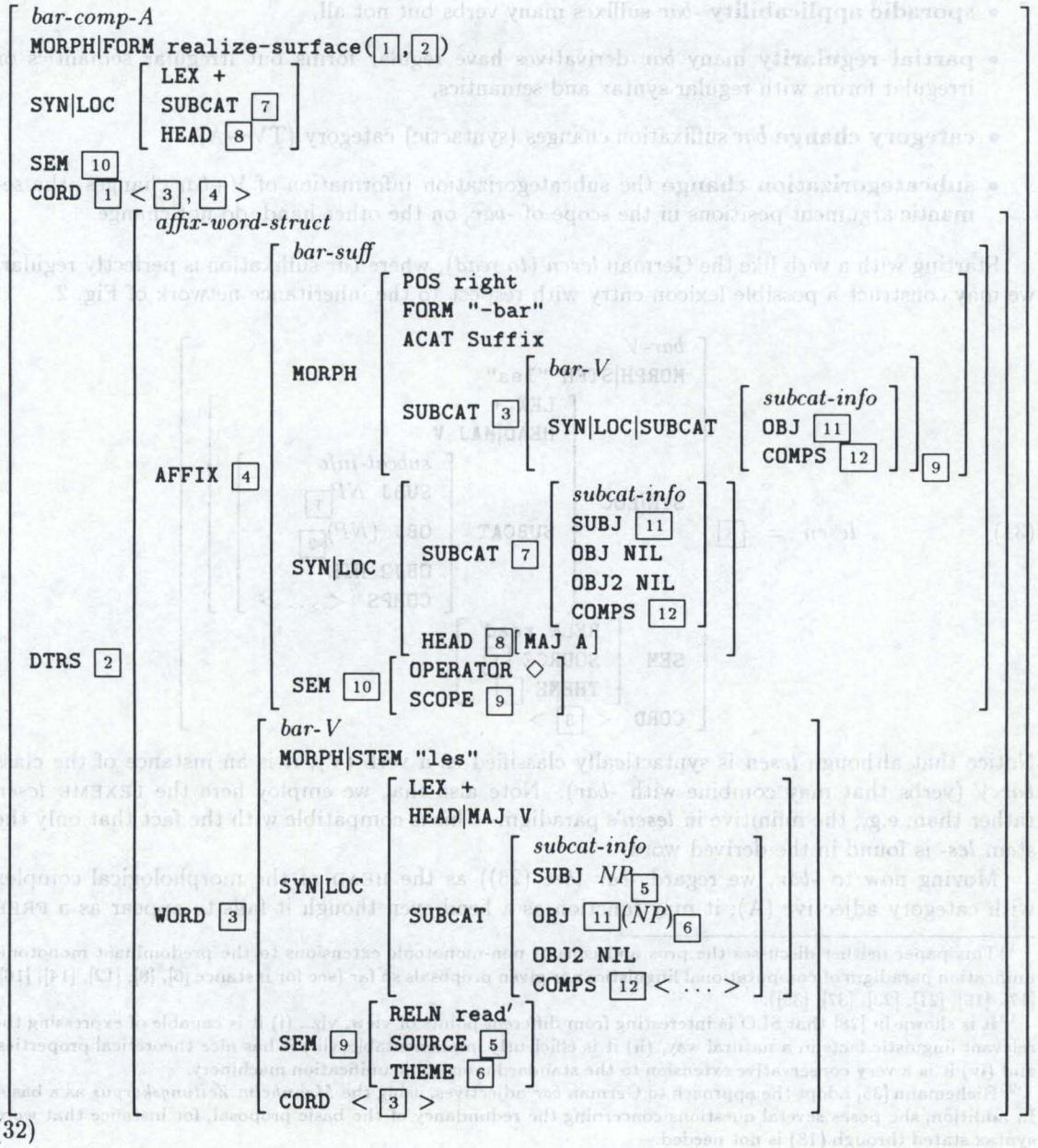
Moving now to *-bar*, we regard *-bar* (see (26)) as the HEAD of the morphological complex with category adjective (A); it may function as a head even though it fails to appear as a FREE

[18] This paper neither discusses the pros and cons of non-monotonic extensions to the predominant monotonic unification paradigm of computational linguistics nor given proposals so far (see for instance [6], [8], [12], [14], [16], [17], [19], [21], [23], [37], [39]).

[19] It is shown in [28] that SLO is interesting from different points of view, viz., (i) it is capable of expressing the relevant linguistic facts in a natural way, (ii) it is efficiently implementable, (iii) it has nice theoretical properties, and (iv) it is a very conservative extension to the standard monotonic unification machinery.

[20] Riehemann [35] adopt the approach to German *bar* adjectives, using the *Mannheim Zeitungskorpus* as a basis. In addition, she poses several questions concerning the redundancy of the basic proposal, for instance that word syntax stated through (18) is not needed.

word. Instead, it occurs only as a BOUND morpheme (instance of the class *bar-suff*; cf. Fig. 2). As a result of the Head Feature Principle the mother automatically obtains the category of the head daughter—and this is exactly what we want, since *les+bar* (*readable*) is an adjective. The Affix-Word Rule, the Subcategorization Principle and the specification of **MORPH|SUBCAT** to be an (underspecified) instance of type *bar-V* additionally guarantee that *-bar* only combines with *bar* verbs. Semantically, *-bar* functions as a modal operator, working on the propositional content of *lesen* (*read*) to create a proposition, asserting the possibility of reading. We note here the co-specification between the semantics of the subcategorized element under **MORPH|SUBCAT** and the value of the **SCOPE** attribute in the modal proposition. These assumptions has led us to the structure of *-bar* which is depicted in (26). The entries for *lesen* and *-bar* together with the Affix-Word Rule and the Morphological Principles permit us therefore to construct a well-formed feature structure for *les+bar*, and also to reject ill-formed feature structures, so that we can show that (32) is the predicted structure and (2) the corresponding analysis tree. This meshing of mechanisms ensures that *lesbar* has the right internal structure.



(32)

In the same way, we might construct an entry for the complex noun *Lesbar+keit* (*readability*) by using the complex adjective *les+bar* which we had already built—all we have to do further is to specify the feature structure (33) of the suffix *-keit*, saying that it will subcategorize for objects belonging to the same category *lesbar* belongs to. The Principles and the Affix-Word Rule then ensure that the feature structure for *Lesbarkeit* will have the right form. The suffix *-keit* (33) subcategorizes for a *keit* adjective *keit-A* but *lesbar* (32) is classified as a complex *bar* adjective *bar-comp-A*. However, this causes no problem when assuming a type hierarchy like the one depicted in Fig. 2, because *bar-comp-A* is a SUBTYPE of *keit-A*, and *-keit* is of course allowed to bind a more specific type than *keit-A*.[21]

(33)
$$keit = \begin{bmatrix} keit\text{-}suff \\ \texttt{MORPH} \begin{bmatrix} affix\text{-}morphology \\ \texttt{FORM "-keit"} \\ \texttt{SUBCAT } keit\text{-}A \end{bmatrix} \\ \texttt{SYN|LOC|HEAD|MAJ N} \end{bmatrix}$$

Among *bar* verbs such as *lesen*, having perfectly regular *bar* adjectives (i.e., complex adjectives, containing *-bar* as their head, e.g., *lesbar*), there are others whose derived adjectives are partially irregular, for example with respect to their form (e.g., *sichtbar, kündbar*) or their semantics. As an additional complication, some *bar* adjectives of these verbs are provided with an additional regular, but non-standard reading. Take for instance the German verb *essen* (*to eat*):

(34)
$$essen = \begin{bmatrix} bar\text{-}V \\ \texttt{MORPH|STEM "ess"} \\ \texttt{SYN|LOC|SUBCAT} \begin{bmatrix} \texttt{SUBJ } NP\boxed{1} \\ \texttt{OBJ } (NP)\boxed{2} \end{bmatrix} \\ \texttt{SEM} \begin{bmatrix} \texttt{RELN eat'} \\ \texttt{SOURCE } \boxed{1} \\ \texttt{THEME } \boxed{2} \end{bmatrix} \end{bmatrix}$$

The non-standard (semantically regular) reading of *eßbar* can be built in a regular way by means of the mechanisms described above, taking *essen* and *-bar* to form a complex word:

(35)
$$eßbar^{non-stand} = \begin{bmatrix} bar\text{-}comp\text{-}A \\ \texttt{SEM } \boxed{1} \begin{bmatrix} \texttt{OPERATOR } \Diamond \\ \texttt{SCOPE|RELN eat'} \end{bmatrix} \\ \texttt{DTRS|AFFIX|SEM } \boxed{1} \end{bmatrix}$$

The standard reading of *eßbar* on the other hand is *edible* (the property of an object which can SAFELY be eaten). Constructing the standard reading (with irregular semantics) for *eßbar* can be done in our approach in two different ways:

1. We do NOT regard *eßbar* as an instance of the class *bar-comp-A*; instead, *eßbar* is entered separately as a whole into the lexicon and belongs to a different word class, say *?-A*. A treatment of this kind leads us to the question whether the feature structure (36) actually will have a **DTRS** attribute—since no use need to be made of the structure.

2. The semantics of (35) (the entry which was built regularly) is modified by using a non-monotonic device to enforce the standard reading. In this case, *eßbar* (37) still belongs to the class *bar-comp-A* and all other properties remain the same. The mechanism we want

[21] The type *keit-A* is used to characterize adjectives that can be bound by the suffix *-keit* but do not necessarily end in the suffix *-bar*, e.g., the German adjective *heilig* (*holy*) which might be bound by *-keit* to form the word *Heiligkeit* (*holiness*).

to employ during the DEFINITION of subregularities and exceptions is a special form of overwriting, so called Single Link Overwriting.[22]

(36)
$$eßbar^{stand} = \begin{bmatrix} ?\text{-}A \wedge \neg bar\text{-}comp\text{-}A \\ \texttt{SEM|RELN safely-eat}' \\ \texttt{DTRS ???} \end{bmatrix}$$

(37)
$$eßbar^{stand'} := eßbar^{non-stand} \;\&\; ![\texttt{SEM|RELN safely-eat}']$$
$$= \begin{bmatrix} bar\text{-}comp\text{-}A \\ \texttt{SEM} \boxed{1} \begin{bmatrix} \texttt{RELN safely-eat}' \\ \texttt{SOURCE} \ldots \\ \texttt{THEME} \ldots \end{bmatrix} \\ \texttt{DTRS|AFFIX|SEM} \boxed{2} \\ \boxed{1} \neq \boxed{2} \end{bmatrix}$$

The advantage of the second approach is that regular properties of partially regular derivations need not be specified redundantly, as would be the case in the first approach. The use of default specifications thus obtains the same advantages in DERIVATION that Flickinger et al. [20] and Evans and Gazdar [17] have shown in word-class definitions. Defaults, together with the possibility of overwriting defaults in more specific definitions may turn out to be even more important in connection with the analysis of derivational relationships, since these are notoriously irregular in morphological form, syntactic feature assignment, and semantics.

There are also linguistically motivated examples where the stronger Subcategorization Principle *MSCP** comes into play and might be prefered, rather than to employ Single Link Overwriting. Take for instance the following example. *lesen* (*to read*) and *sehen* (*to see*) are closely related transitive verbs (*TV*), having (nearly) the same morphological, syntactical, and semantical properties. However, it turns out that *bar* suffixation would predict a wrong morphological form for *sehen*, viz., *sehbar* which is out, but only *sichtbar* is perfect. This and similar facts can be captured by *MSCP** (28) and by assuming an additional morphological subcategorization feature for objects of type *part-of-speech*, which is typed to *affix*. To represent these facts we encode *-bar* (38) as before, saying that it subcategorizes for transitive verbs:

(38)
$$bar = \begin{bmatrix} bar\text{-}suff \\ \texttt{MORPH|SUBCAT } TV \end{bmatrix}$$

Now, if one wants to state that *lesen* (39) is a transitive verb *TV* and regular with respect to the above given subcategorization whereas *sehen* (40), although belonging to *TV*, fails to form a REGULAR *bar* derivative, this is easy to encode:

(39)
$$lesen = \begin{bmatrix} TV \\ \texttt{MORPH|SUBCAT } bar\text{-}suff \end{bmatrix}$$

(40)
$$sehen = \begin{bmatrix} TV \\ \texttt{MORPH|SUBCAT } \neg bar\text{-}suff \end{bmatrix}$$

The typed approach to *bar* suffixation also allows us to prevent ill-formed *bar* adjectives; e.g., we have to rule out the combination of *haben* (*to have*) together with *-bar*. This is very easy to

[22] The term SINGLE LINK OVERWRITING is used to emphasize the fact that OVERWRITING takes place via a SINGLE INHERITANCE LINK, i.e., defining a new word class using SLO forces us to specify a single class from which we inherit and saying which VALUES we would like to overwrite. Therefore inheritance conflicts (*which value to choose?*) will never occur because the more specific information always wins (there is always only ONE supertype). Note that we are allowed to OVERWRITE COREFERENCE CONSTRAINTS; this was done during the definition of *eßbar* in (37) because we want to state that the Semantics Principle *MSP* of course does not hold under this exception. In our special case the syntax is of the following form: *new-class* := *old-class* & !*overwrite-info*. SLO will never fail during the definition of a new class under the assumption that the *overwrite-info* is consistent. However, the closely related notion of SLO UNIFICATION (see [28]) which we would like to apply during processing, might fail—this is in contrast to other proposals which never yield $\perp$, for instance Bouma's Default Unification [6].

achieve under the assumption that *haben* doesn't belong to the *bar* verb class *bar-V*, but instead to another class (say *?-V*), thus preventing *haben* from combining with *-bar*—therefore *hab+bar* is disallowed.

(41) $$haben = \left[\begin{array}{l} ?\text{-}V \wedge \neg bar\text{-}V \\ \ldots\ldots \end{array} \right]$$

It is nevertheless possible to construct *handhab+bar* (*manageable*) out of *handhaben* (*to handle, to manage*), since *haben* and *handhaben* are distinct lexemes. By explicitly encoding *handhaben* as an entry of type *bar-V*, we can move to a legal description of *handhabbar*.

(42) $$handhaben = \left[\begin{array}{l} bar\text{-}V \\ \ldots\ldots \end{array} \right]$$

(43) $$handhabbar = \left[\begin{array}{ll} bar\text{-}comp\text{-}A & \\ \texttt{DTRS} & \left[\begin{array}{l} \texttt{AFFIX|MORPH|FORM "-bar"} \\ \texttt{WORD|MORPH|STEM "handhab"} \end{array} \right] \end{array} \right]$$

The structure of the class hierarchy (cf. Fig. 2) leads us to a treatment of suffixation (and also of prefixation in general), where the whole process can be described within the framework of inheritance reasoning over feature structure descriptions. On what grounds are we allowed to state such a thesis? At first sight, this statement seems to stand in contrast with the claim made in the beginning, that *naive* inheritance is not enough. But we do NOT rely on naive inheritance as the ONLY mechanism. So we turn now to an examination of why this is so. We noted earlier that *les+bar* and *Les+bar+keit* are legal lexemes because they (and their complex parts) satisfy all principles whose left sides they match (implying that they have to meet the right sides too), and because they are composed out of lexicon entries by means of rules. In doing realistic parsing or generation, we might assume an additional CONTROL MACHINERY outside of the grammar/lexicon, which uses principles and rules to accept or reject, or alternatively, to generate well-formed complex phrases. However because we regard principles as well as rules as types, equation (1) allows us to employ the laws of feature algebras to construct new types (call them PRECATEGORIES), which are subsumed by all principles having a more general left side and by at least one rule (cf. *AWR&Ps* (29) and (30)). Complex words like *lesbar* on the other hand will then be subsumed by such precategories.

It is now easy to see that the processes described up to now can be represented entirely via inheritance of a sophisticated kind: it is possible to define new legal complex word classes by inheriting from precategories as well as from simple lexical categories (cf. the subtypes of *part-of-speech* in Fig. 2) and by stating additional local constraints for the class in question. Looking at Fig. 2, *bar-comp-A* is such a complex class. *bar-comp-A* inherits from *keit-A* and from the precategory *AWR&Ps*, but also enforces idiosyncratic constraints which have to be satisfied by words that are members of this class:

(44) $$bar\text{-}comp\text{-}A = AWR\&Ps \wedge keit\text{-}A \wedge \left[\texttt{DTRS} \left[\begin{array}{l} \texttt{AFFIX } bar\text{-}suff \\ \texttt{WORD } bar\text{-}V \end{array} \right] \right]$$

It's very important to constrain **AFFIX** to *bar-suff* and **WORD** to *bar-V* in order to get the right feature structure for *bar-comp-A*. Since furthermore *AWR&Ps* (30) is also associated with a feature structure, it's not difficult to construct the prototypical feature structure for *bar-comp-A* by unifying all the information. But once this is achieved, we may construct an entry for *lesbar* by instantiating the class *bar-comp-A* with complement daughter *lesen*, i.e., **WORD** must have as its value a feature structure equal to that of the lexeme *lesen* (31).

(45) $$lesbar = bar\text{-}comp\text{-}A \wedge [\,\texttt{DTRS|WORD } lesen\,]$$

Notice that the feature structure for (45) corresponds to the one of *les+bar* (32) provided earlier. In entirely the same fashion, we then let create feature structures for new words, e.g., for *Les+bar+keit* (47).

(46) $$\textit{keit-comp-N} = \textit{AWR\&Ps} \wedge \textit{CN} \wedge \left[\text{DTRS} \left[\begin{array}{l} \text{AFFIX } \textit{keit-suff} \\ \text{WORD } \textit{keit-A} \end{array} \right] \right]$$

(47) $$\textit{Lesbarkeit} = \textit{keit-comp-N} \wedge [\,\text{DTRS|WORD } \textit{lesbar}\,]$$

3.2 Prefixation

What prefixes and suffixes have in common is that they serve as HEADS in our simple head-complement approach, although they differ in many ways.[23]

This section investigates for further depth the German prefix *Vor-*, showing how the relevant data about *Vor* prefixation can be captured properly in our approach presented so far. Some known facts about *Vor* prefixation:

- **sporadic applicability** *Vor-* prefixes many nouns but not all,
- **partial regularity** many *Vor* derivatives have regular forms and irregular semantics,
- **category constant** the syntactic category of the *Vor* derivative does not change,
- **subcategorization constant** the subcategorization information of the derived complex word *Vor*+N is taken over completely from the complement,
- **iterability** the prefix *Vor-* can be applied iteratively.

Let's examine the German noun *Version* (*version*) that may combine with *Vor-* in order to form a complex noun.[24]

(48) $$\textit{Version} = \boxed{1} \left[\begin{array}{l} \textit{vor-N} \\ \text{MORPH|STEM "Version"} \\ \text{SYN|LOC} \left[\begin{array}{l} \text{LEX +} \\ \text{HEAD|MAJ N} \\ \text{SUBCAT} \left[\begin{array}{l} \textit{subcat-info} \\ \ldots \end{array} \right] \end{array} \right] \\ \text{SEM} \left[\text{PRED version}' \right] \\ \text{CORD} \langle \boxed{1} \rangle \end{array} \right]$$

In order to construct *Version*, *Vorversion*, *Vorvorversion*, ..., the prefix *Vor-* (49) must contain at least the following information.[25]

(49) $$\textit{Vor} = \left[\begin{array}{l} \textit{vor-pref} \\ \text{MORPH} \left[\begin{array}{l} \text{POS left} \\ \text{FORM "Vor-"} \\ \text{ACAT Prefix} \\ \text{SUBCAT} \left[\begin{array}{l} \textit{vor-N} \\ \text{SYN|LOC|SUBCAT } \boxed{1} \end{array} \right]_{\boxed{2}} \end{array} \right] \\ \text{SYN|LOC} \left[\begin{array}{l} \text{HEAD|MAJ N} \\ \text{SUBCAT } \boxed{1} \end{array} \right] \\ \text{SEM} \left[\begin{array}{l} \text{OPERATOR vor}' \\ \text{SCOPE } \boxed{2} \end{array} \right] \end{array} \right]$$

[23] One might argue that only suffixes can be regarded as heads and prefixes should be given the status of a modifier (the syntactic category of the compound word is determined by the free word which is the head in this case). However, under this assumption, we have two work with two ID rule schemata, one for prefixes and one for suffixes.

[24] For expository purposes, the semantics of *Version* is stated as simple as possible, but in general a more complex one must be employed, e.g., the proposal given by Pollard and Sag [34], Ch. 4, for (unsaturated) common nouns.

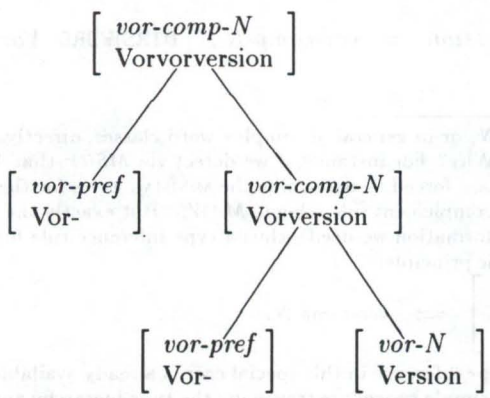
[25] As we mentioned above, we have taken the prefix *Vor-* as an example to show how certain phenomena can be handled in our approach. The assumption that *Vor-* functions semantically as an operator, working on the semantics of the noun is of course not an in-depth analysis and may not be useful in real applications. There are other prefixes like *Anti-* or *Ur-* having similar properties, but their semantics is more complicated.

Note the structure sharing [1] between the sentential subcategorization of the morphologically subcategorized element *vor-N* and the syntactic subcategorization of *Vor-*. This is due to the fact that the Subcategorization Principle *MSCP* coindexes the value of **SYN|LOC|SUBCAT** in affix and mother. In contrast to *-bar*, *Vor-* behaves different to how it contributes to the syntactic subcategorization of the new compound word: the prefix *Vor-* takes a noun and yields a complex noun which bears the same grammatical relations. By means of the Head Feature Principle *MHFP*, the mother of the morphological phrase will be assigned the same category, the prefix *Vor-* bears. That's the main reason why RECURSION is possible—the new word *Vor+Version* (50) will again be classified as a noun and could then combine with a new *Vor-* to form a more complex word, viz., *Vor+Vor+Version* in the same way as described before (see analysis tree (51) and the partially depicted feature structure (7) of *Vorvorversion*). The potential multiple application of *Vor-* is only allowed because Fig. 3 specifies *vor-comp-N* as a SUBTYPE of *vor-N*, and *Vor-* morphologically subcategorizes for objects of type *vor-N*, therefore nouns of the subtype *vor-comp-N* are especially welcomed.

(50)

$$\left[\begin{array}{l} \textit{vor-comp-N} \\ \texttt{MORPH|FORM "Vorversion"} \\ \texttt{SYN|LOC} \left[\begin{array}{l} \texttt{LEX +} \\ \texttt{HEAD } \boxed{1} \\ \texttt{SUBCAT } \boxed{2} \end{array}\right] \\ \texttt{SEM } \boxed{3} \\ \texttt{CORD} < \boxed{4}, \boxed{5} > \\ \texttt{DTRS} \left[\begin{array}{ll} \texttt{AFFIX } \boxed{4} & \left[\begin{array}{l} \textit{vor-pref} \\ \texttt{MORPH} \left[\begin{array}{l} \texttt{FORM "Vor-"} \\ \texttt{SUBCAT } \boxed{5}\texttt{[SYN|LOC|SUBCAT } \boxed{2}\texttt{]}_{\boxed{6}} \end{array}\right] \\ \texttt{SYN|LOC} \left[\begin{array}{l} \texttt{HEAD } \boxed{1}\texttt{[MAJ N]} \\ \texttt{SUBCAT } \boxed{2} \end{array}\right] \\ \texttt{SEM } \boxed{3} \left[\begin{array}{l} \texttt{OPERATOR vor}' \\ \texttt{SCOPE } \boxed{6} \end{array}\right] \end{array}\right] \\ \texttt{WORD } \boxed{5} & \left[\begin{array}{l} \textit{vor-N} \\ \texttt{MORPH|STEM "Version"} \\ \texttt{SYN|LOC|SUBCAT } \boxed{2} \\ \texttt{SEM } \boxed{6} \\ \texttt{CORD} < \boxed{5} > \end{array}\right] \end{array}\right] \end{array}\right]$$

(51)



We described *bar* suffixation above by means of type unification and inheritance over feature terms; once we assume an inheritance network such as Fig. 3, we may analyze *Vor* prefixation

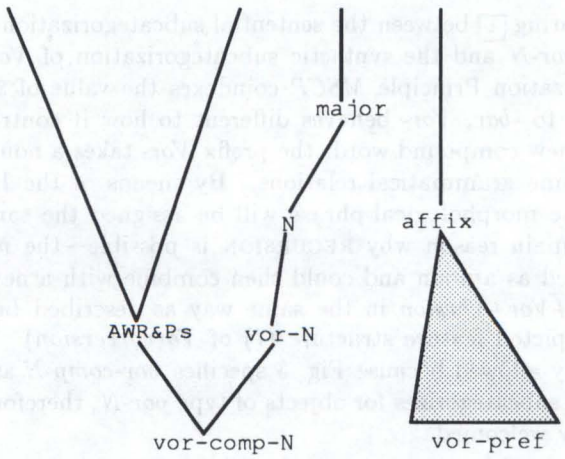


Figure 3: Structure of the inheritance network in case of *Vor* prefixation, regarding the principles and the rule. Note that we additionally impose LOCAL constraints on certain classes, especially on *vor-comp-N*.

(including recursion) similarly. Analogous to the type *bar-comp-A* (44), we may state the right definition for *vor-comp-N* (52) with respect to Fig. 3.[26]

$$(52) \qquad \textit{vor-comp-N} \;=\; \textit{AWR\&Ps} \wedge \textit{vor-N} \wedge \left[\, \texttt{DTRS} \left[\begin{array}{l} \texttt{AFFIX}\ \textit{vor-pref} \\ \texttt{WORD}\ \textit{vor-N} \end{array}\right] \right]$$

The iterated application of *Vor-* will be guaranteed by using the RECURSIVE type definition of *vor-comp-N*. But how do we block infinite recursion in cases of concrete words? Only feature structures whose (minimal) type is *vor-N* will stop parsing or generation of complex word forms, because those instances don't have any internal constituent structure (no **DTRS** attribute), i.e., there's no way to expand them further.

Constructing an entry for *Vorversion/Vorvorversion* (53/7) is done in a trivial way by instantiating *vor-comp-N* and by imposing an additional restriction, namely that the complement daughter **WORD** must hold a feature structure representing the word *Version/Vorversion*.

$$(53) \qquad \textit{Vorversion} \;=\; \textit{vor-comp-N} \wedge [\, \texttt{DTRS|WORD}\ \textit{Version}\,]$$

$$(54) \qquad \textit{Vorvorversion} \;=\; \textit{vor-comp-N} \wedge [\, \texttt{DTRS|WORD}\ \textit{Vorversion}\,]$$

[26] The definition of *vor-comp-N*, or in general of complex word classes, directly leads us to the TYPE INFERENCE RULES we need during parsing. Why? For instance, if we detect via *MSRP* that *Vorversion* consists of the prefix *Vor-* and the noun *Version*, we are forced to determine the MINIMAL TYPE for the feature structure of *Vorversion* which was licensed by the head-complement rule scheme *MAWR*. But exactly the LOCAL constraints we impose on *vor-comp-N* represent all the information we need—thus a type inference rule for *vor-comp-N* might be encoded by the following language-specific principle:

$$\left[\begin{array}{l} \textit{complex} \\ \texttt{DTRS} \left[\begin{array}{l} \texttt{AFFIX}\ \textit{vor-pref} \\ \texttt{WORD}\ \textit{vor-N} \end{array}\right] \end{array}\right] \Longrightarrow \textit{vor-comp-N}.$$

But the information for doing type inference in this special case is already available, viz., in the hierarchy of lexical types and can be extracted by a simple procedure traversing the type hierarchy and yielding a set of type inference rules. However, its a matter of choice whether principles such the one above should be stated explicitly as typed implications in the style of HPSG and processed completely by the underlying unification machinery or should be factored out as context-free rewrite rules, i.e., as phrasal constraints, which can then be applied independently by a special purpose machinery to build up phrase structure trees (see [31] on the advantages of separating phrasal and functional constraints).

4 The Affix Hierarchy

It is well known that HPSG was the first linguistic theory of the unification paradigm which incorporated the ideas on STRUCTURED OBJECTS and INHERITANCE, well known from computer science, semantic networks, and terminological knowledge representation systems, to state linguistic knowledge in a distributed way and to structure the lexicon highly modularly, i.e., to capture regularities and to avoid redundant information. Other approaches like LFG or GPSG didn't focus on the lexicon very much—they have either little to say about it or seem to assume a flat lexicon structure.

Because we approach derivation in the same way HPSG handle sentential phenomena, it might be interesting to explore whether the HPSG-style of defining lexical hierarchies can be applied to represent AFFIXES in the same way.[27] And in fact, the arguments presented by Pollard and Sag for lexical types do hold for affixes as well, so we adopt their notation and cross-classify ALL affixes along certain relevant dimensions at once. For instance, the feature structure of the suffix *-bar* depicted in (26) is a fully expanded instance of the type *bar-suff*. However, the IDIOSYNCRATIC PROPERTIES of *bar-suff* only concern the morphological form, parts of the semantics and the type of the subcategorized object.

Currently FIVE DIMENSIONS contribute to the classification of affixes, but there might be additional generalizations as well:

- **POS**—is the affix classified as a prefix or as a suffix?
- **CAT**—does the new word undergo a category change?
- **SUBCAT**—how does the subcategorization information of the new word look like?
- **BIND**—which object does the affix morphologically subcategorize for?
- **SEM**—which form does the semantics of the affix have?

To give an impression how such a hierarchy might look like and how it helps to shorten the definition of a specific affix, we study this mechanism by taking again the suffix *-bar* (26) as an example. With Fig. 4 in mind, we can express what is idiosyncratic to *bar-suff* and what must be represented in the types from which *bar-suff* inherits:

$$\begin{aligned} \textit{bar-suff} \;=\; & \textit{suffix} \wedge \textit{V-to-A} \wedge \textit{drop-subject} \wedge \textit{1-place-operator} \wedge \textit{bind-TV} \\ & \wedge \left[\begin{array}{l} \texttt{MORPH} \left[\begin{array}{l} \texttt{FORM "-bar"} \\ \texttt{SUBCAT}\ \textit{bar-V} \end{array}\right] \\ \texttt{SEM|OPERATOR}\ \Diamond \end{array}\right] \end{aligned} \tag{55}$$

(55) classifies *bar-suff* as a suffix (*suffix*) which triggers a category change from verb to adjective (*V-to-A*) and leads to a set of grammatical relations in the mother with fewer elements than the complement consists of by throwing away the subject (*drop-subject*); in addition, *bar-suff* morphologically subcategorizes for a transitive verb (*bind-TV*) and the semantics is characterized roughly as an operator-scope structure (*1-place-operator*). Although *bar-suff* inherits from *bind-TV*, (55) forces **MORPH|SUBCAT** to be of type *bar-V*, since not all combinations of a transitive verb with the suffix *-bar* are legal; so we assume that *bar* verbs are a natural subclass of the transitive verbs, i.e., $\textit{bar-V} \sqsubset \textit{TV}$.

Before we go on in representing more affixes in the way above, we have to say a few words about the different dimensions and partitions (types), which can be found in Fig. 4. As a main result, we will show that a feature logic without functions or relations is too WEAK to characterize the **SUBCAT** dimension.

[27] See Pollard and Sag [34] for a motivation and many examples and [43] or [13] for the logic underlying those hierarchies.

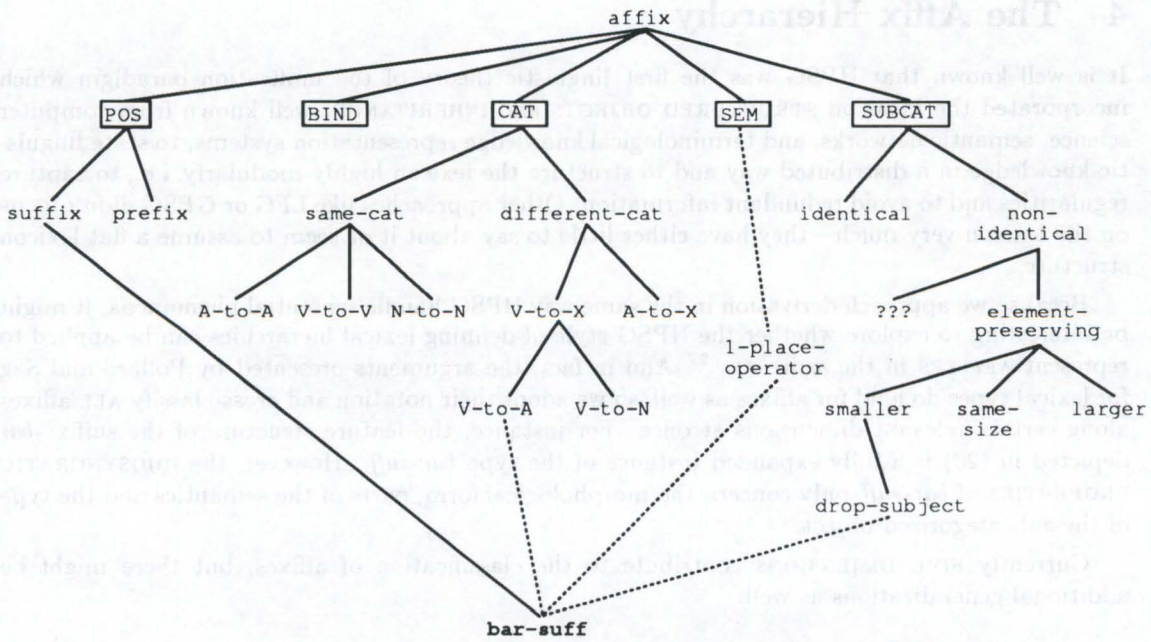


Figure 4: Structure of the cross-classified affix hierarchy with respect to the special suffix type *bar-suff*. Dashed lines should indicate that there might be additional types which are however not shown in the picture. Note that we omit the whole **BIND** dimension and also essential parts of the **SEM** dimension.

4.1 The POS Dimension

The **POS** dimension (see Fig. 4) simply states whether the affix is classified as a *prefix* (56) or a *suffix*.

(56) $$prefix \;=\; affix \wedge \left[\, \texttt{MORPH} \left[\begin{array}{l}\texttt{POS left}\\ \texttt{ACAT Prefix}\end{array}\right]\right]$$

4.2 The CAT Dimension

The **CAT** dimension (see Fig. 4) expresses whether the syntactic category of the mother differs from the one of the complement or is identical.

(57) $$same\text{-}cat \;=\; affix \wedge \left[\begin{array}{l}\texttt{MORPH|SUBCAT|SYN|LOC|HEAD|MAJ}\ \boxed{1}\\ \texttt{SYN|LOC|HEAD|MAJ}\ \boxed{1}\end{array}\right]$$

(58) $$different\text{-}cat \;=\; affix \wedge \left[\begin{array}{l}\texttt{MORPH|SUBCAT|SYN|LOC|HEAD|MAJ}\ \boxed{1}\\ \texttt{SYN|LOC|HEAD|MAJ}\ \boxed{2}\\ \boxed{1} \neq \boxed{2}\end{array}\right]$$

(59) $$V\text{-}to\text{-}X \;=\; different\text{-}cat \wedge [\ \texttt{MORPH|SUBCAT|SYN|LOC|HEAD|MAJ V}\ ]$$

(60) $$V\text{-}to\text{-}A \;=\; V\text{-}to\text{-}X \wedge [\ \texttt{SYN|LOC|HEAD|MAJ A}\ ]$$

4.3 The SUBCAT Dimension

Because the subcategorization information of the mother is mainly determined by the complement, the free word, and because of the special nature of the Subcategorization Principle *MSCP* (27),

we introduce the **SUBCAT** dimension (see Fig. 4) to state the relevant generalizations about the sententially subcategorized elements in mother and complement. The first question is whether they are identical (61) or not (62).

(61) $$\textit{identical} = \textit{affix} \wedge \begin{bmatrix} \texttt{MORPH|SUBCAT|SYN|LOC|SUBCAT}\ \boxed{1} \\ \texttt{SYN|LOC|SUBCAT}\ \boxed{1} \end{bmatrix}$$

(62) $$\textit{non-identical} = \textit{affix} \wedge \begin{bmatrix} \texttt{MORPH|SUBCAT|SYN|LOC|SUBCAT}\ \boxed{1} \\ \texttt{SYN|LOC|SUBCAT}\ \boxed{2} \\ \boxed{1} \neq \boxed{2} \end{bmatrix}$$

If the sentential subcategorization information in mother and complement isn't identical, the question arises whether all subcat elements of the complement can be found in the mother OR all elements of the mother occur as parts of the complement (63). However, to formulate such a constraint, we have to use a feature logic that allows us to state functional/relational constraints. Without functions/relations, we cannot express the linguistic generalizations properly, i.e., it turns out that certain type definitions would be too GENERAL, therefore allowing us to predict non-legal descriptions.

(63) $$\left\{ \begin{array}{l} \textit{element-preserving} = \textit{non-identical} \wedge \begin{bmatrix} \texttt{MORPH|SUBCAT|SYN|LOC|SUBCAT}\ \boxed{1} \\ \texttt{SYN|LOC|SUBCAT}\ \boxed{2} \end{bmatrix} \\ \\ \text{CONDITION: } (\texttt{subcat-elements}(\boxed{1}) \subseteq \texttt{subcat-elements}(\boxed{2})) \vee \\ \qquad\qquad\quad (\texttt{subcat-elements}(\boxed{1}) \supseteq \texttt{subcat-elements}(\boxed{2})) \end{array} \right\}$$

Exactly the CONDITION part of (63) and (64) (and also of other **SUBCAT** types; see Fig. 4) cannot be represented otherwise. Why? Because the constraint is formulated as a condition that must hold on parts/elements of a set (set inclusion and set membership) and this is outside the expressiveness of (propositional) feature logic. The same is true if we have to formulate that two arbitrary list of unknown length (see the **COMPS** attribute in type *subcat-info*) must have the same elements, i.e., are equal modulo permutations.

(64) $$\left\{ \begin{array}{l} \textit{smaller} = \textit{element-preserving} \wedge \begin{bmatrix} \texttt{MORPH|SUBCAT|SYN|LOC|SUBCAT}\ \boxed{1} \\ \texttt{SYN|LOC|SUBCAT}\ \boxed{2} \end{bmatrix} \\ \\ \text{CONDITION: } \texttt{subcat-elements}(\boxed{1}) \supset \texttt{subcat-elements}(\boxed{2}) \end{array} \right\}$$

Note that **subcat-elements** is defined as

(65) $$\begin{array}{l} \texttt{subcat-elements}\left(\left[\texttt{SUBJ}\ \boxed{1}, \texttt{OBJ}\ \boxed{2}, \texttt{OBJ2}\ \boxed{3}, \texttt{COMPS}\ \boxed{4}\right]\right) := \\ \quad \texttt{singleton}(\boxed{1}) \cup \texttt{singleton}(\boxed{2}) \cup \texttt{singleton}(\boxed{3}) \cup \texttt{list-to-set}(\boxed{4}). \end{array}$$

drop-subject is the **SUBCAT** type from which *bar-suff* directly inherits (cf. (55)). It is responsible that *bar*, and therefore all complex *bar* adjectives will get the right sentential subcategorization frame (see subsection on suffixation and cf. the feature structure of *bar* (26)).

(66) $$\textit{drop-subject} = \textit{smaller} \wedge \begin{bmatrix} \texttt{MORPH|SUBCAT|SYN|LOC|SUBCAT} & \begin{bmatrix} \textit{subcat-info} \\ \texttt{OBJ}\ \boxed{1} \\ \texttt{OBJ2}\ \boxed{2} \\ \texttt{COMPS}\ \boxed{3} \end{bmatrix} \\ \texttt{SYN|LOC|SUBCAT} & \begin{bmatrix} \textit{subcat-info} \\ \texttt{SUBJ}\ \boxed{1} \\ \texttt{OBJ}\ \boxed{2} \\ \texttt{OBJ2 NIL} \\ \texttt{COMPS}\ \boxed{3} \end{bmatrix} \end{bmatrix}$$

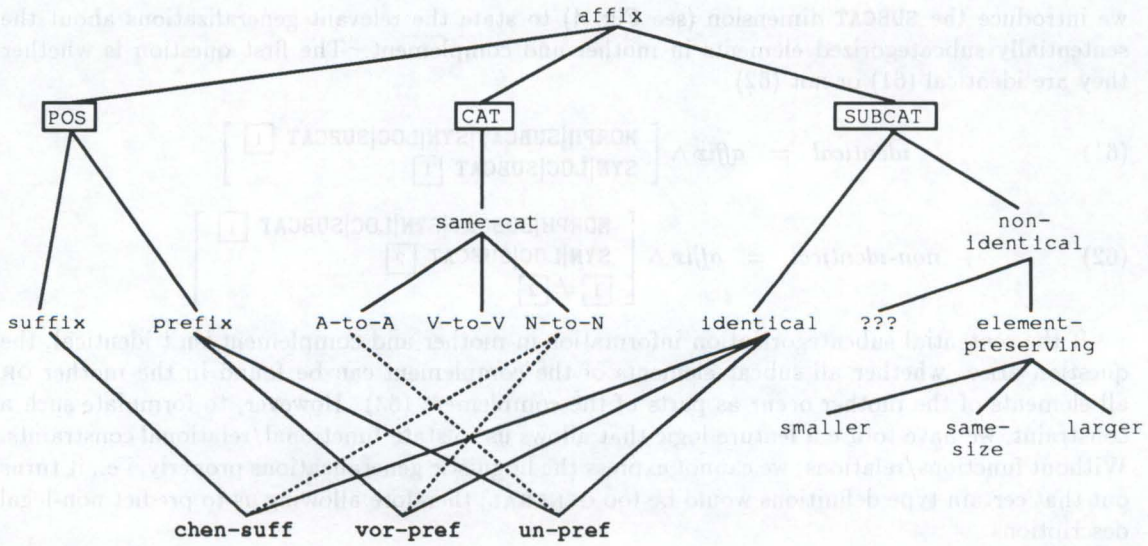


Figure 5: Structure of the cross-classified affix hierarchy with respect to the **SUBCAT** partition *identical*.

4.4 The SEM Dimension

The **SEM** dimension specifies how the semantics of the affix looks like. It is useful to distinguish in general between *predicate-argument* and *operator-scope* structures. In addition, a differentiation with respect to the *arity* of predicates/operators does make sense. However, we will give only the definition of the type *1-place-operator* which can be found in the definition of *bar-suff* (55).

$$(67) \qquad \textit{1-place-operator} \;=\; \textit{affix} \wedge \left[\begin{array}{ll} \textbf{MORPH|SUBCAT|SEM} & \boxed{1} \\ \textbf{SEM} & \left[\begin{array}{l} \textit{operator-scope-struct} \\ \textbf{OPERATOR} \\ \textbf{SCOPE}\ \boxed{1} \end{array}\right] \end{array}\right]$$

4.5 The BIND Dimension

The last dimension **BIND** states which sort of objects an affix morphologically subcategorizes for. It is worth noting that different affixes might share the same **BIND** dimension, e.g., the German suffixes *-chen* and *-lein* morphologically subcategorize for common nouns. The **BIND** dimension of the type *bar-suff* (55) above is given by the following feature structure:

$$(68) \qquad \textit{bind-TV} \;=\; \textit{affix} \wedge \left[\ \textbf{MORPH|SUBCAT}\ \textit{TV}\ \right]$$

4.6 More Examples

In the same way we might cross-classify other affixes (see Fig. 5), for instance the prefix *Vor-* (49), we have studied in the subsection on prefixation.

$$(69) \qquad \begin{array}{l} \textit{vor-pref} \;=\; \textit{prefix} \wedge \textit{N-to-N} \wedge \textit{identical} \wedge \textit{1-place-operator} \wedge \textit{bind-CN} \\ \qquad\qquad \wedge \left[\begin{array}{ll} \textbf{MORPH} & \left[\begin{array}{l} \textbf{FORM "Vor-"} \\ \textbf{SUBCAT}\ \textit{vor-N} \end{array}\right] \\ \textbf{SEM|OPERATOR}\ \textbf{vor}' & \end{array}\right] \end{array}$$

The next example we present here, concerns the prefix *un-* (see Fig. 5) which we characterized at the moment very roughly by the $\neg$ operator, negating the propositional content of its complement

through the value of SCOPE. Again the subcategorization frame of the new word (e.g., the German *un+schön—not (very) nice*) is equal to the one of the complement, *un-* subcategorizes for and the syntactic category of the new compound remains the same, viz. adjective (A).

$$\text{(70)}\quad \textit{un-pref} = \textit{prefix} \wedge \textit{A-to-A} \wedge \textit{identical} \wedge \textit{1-place-operator} \wedge \textit{bind-A} \wedge \left[\begin{array}{l} \texttt{MORPH} \left[\begin{array}{l}\texttt{FORM "un-"}\\ \texttt{SUBCAT } \textit{un-A}\end{array}\right] \\ \texttt{SEM|OPERATOR} \neg \end{array}\right]$$

Note that *bind-A* (71) takes care that the MOD feature of the subcategorized adjective is coindexed with the one in the prefix *un-*. As a consequence of the Head Feature Principle *MHFP*, the mother also shares the value of the MOD feature.

$$\text{(71)}\quad \textit{bind-A} = \textit{affix} \wedge \left[\begin{array}{l} \texttt{MORPH|SUBCAT} \left[\begin{array}{l} A \\ \texttt{SYN|LOC|HEAD|MOD } \boxed{1}\end{array}\right] \\ \texttt{SYN|LOC|HEAD|MOD } \boxed{1} \end{array}\right]$$

Therefore a fully expanded feature structure for *un-* is of the following form:

$$\text{(72)}\quad \textit{un} = \left[\begin{array}{l} \textit{un-pref} \\ \texttt{MORPH} \left[\begin{array}{l} \texttt{POS left} \\ \texttt{FORM "un-"} \\ \texttt{ACAT Prefix} \\ \texttt{SUBCAT} \left[\begin{array}{l} \textit{un-A} \\ \texttt{SYN|LOC} \left[\begin{array}{l}\texttt{HEAD|MOD } \boxed{1} \\ \texttt{SUBCAT } \boxed{2}\end{array}\right]\end{array}\right]_{\boxed{3}} \end{array}\right] \\ \texttt{SYN|LOC} \left[\begin{array}{l}\texttt{HEAD} \left[\begin{array}{l}\texttt{MAJ A}\\ \texttt{MOD } \boxed{1}\end{array}\right] \\ \texttt{SUBCAT } \boxed{2}\end{array}\right] \\ \texttt{SEM} \left[\begin{array}{l}\texttt{OPERATOR} \neg \\ \texttt{SCOPE } \boxed{3}\end{array}\right] \end{array}\right]$$

The German diminution operators *-chen* (see Fig. 5) and *-lein* are also easy to characterize.[28]

$$\text{(73)}\quad \textit{chen-suff} = \textit{suffix} \wedge \textit{N-to-N} \wedge \textit{identical} \wedge \textit{1-place-operator} \wedge \textit{bind-N} \wedge \left[\begin{array}{l}\texttt{MORPH|FORM "-chen"} \\ \texttt{SEM|OPERATOR little}'\end{array}\right]$$

$$\text{(74)}\quad \textit{lein-suff} = \textit{suffix} \wedge \textit{N-to-N} \wedge \textit{identical} \wedge \textit{1-place-operator} \wedge \textit{bind-N} \wedge \left[\begin{array}{l}\texttt{MORPH|FORM "-lein"} \\ \texttt{SEM|OPERATOR little}'\end{array}\right]$$

We stop here in giving more detailed examples, but we would like to remark that many other German affixes can be straightforwardly represented in the way above (see Fig. 6), for instance

- German *-er* suffix—adjective comparison as derivation,
- German *ge-* prefix—passive as derivation,
- German *an-* and *be-* prefixes—category change IV$\rightsquigarrow$TV.

[28] It does make sense to propose a direct supertype (say *dim-suff*) of *chen-suff* and *lein-suff* which incorporates all features they share, so that the specifications of *chen-suff* and *lein-suff* reduce to *chen-suff* := *dim-suff* & [MORPH|FORM "-chen"] and *lein-suff* := *dim-suff* & [MORPH|FORM "-lein"].

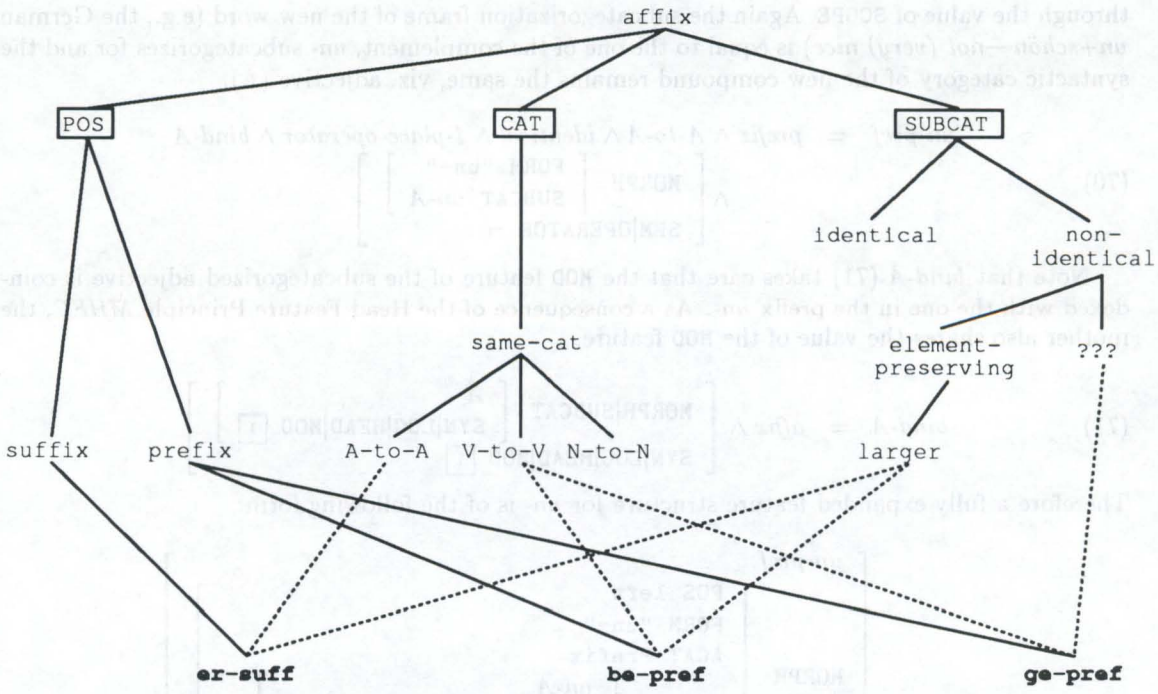


Figure 6: Structure of the cross-classified affix hierarchy with respect to the **SUBCAT** partition *non-identical*.

5 Summary and Conclusion

The paper presents a typed feature-based treatment to DERIVATION where the types are arranged in an inheritance hierarchy and unification is the primary information-building operation. This is accomplished by proposing more complex feature structures for words (and of course proposing feature structures for affixes) than the ones presented in HPSG and also by proclaiming certain morphological principles and a single rule schema fully in the spirit of HPSG, thus a step further towards a highly lexicalized theory which captures all aspects of linguistic knowledge. Therefore, this approach is an alternative to old-fashioned lexical rules as they are proposed, for instance, in HPSG I, Ch. 8. As a result of this approach, neither a parser nor a generator has to cope with the problem of how to use lexical rules during the processing of sentences—e.g., from a parser's point of view, it makes no difference to work on the word level or on the phrasal/sentential level, in other words, a control machinery doesn't have to switch between the application of traditional lexical rules (word level) and the application of principles and rule schemata (sentential level). But an approach like this one also leads to other benefits as we noted in the introductory chapter.

Another topic we considered extensively, concerns the separation between morphotactics and morphophonemics. This was achieved by proposing a morphological and a sentential subcategorization and by directly recording the constituent order of the parts of a complex word. We argued that it is in fact possible to encode the whole morphotactics in the lexicon by means of the above given approach where the knowledge about morphotactics was made explicit instead of proposing a functional solution. The interface to morphophonemics is of course a functional one, viz., the function **realize-surface** in the Surface Realization Principle *MSRP*. This function, however, is set free completely from morphotactics.

One of the main results of this paper concerns the invention of a cross-classified hierarchy of affixes.[29] We have proposed a distributed representation of many known generalizations about affixes, where the linguistic knowledge is currently factored into five dimensions. A concrete affix

[29]Note that we suggest that *sign* is now exhaustively partitioned into *phrase*, *word*, and *affix* (see Fig. 1).

is then composed out of the different dimensions at once by means of unification/inheritance. The nice properties Pollard and Sag have given for the hierarchy of lexical types, can be obtained for free in classifying affixes in the same way.

There are other linguistic areas which we think can be captured by (small extensions of) the treatment we have presented here, viz, COMPOUNDS, IDIOMS, and the whole process of INFLECTION. These three topics will be under investigation in the near future.

References

[1] Hassan Aït-Kaci. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science*, 45:293–351, 1986.

[2] Hassan Aït-Kaci and Roger Nasr. Residuation: A paradigm for integrating logic and functional programming. Technical Report AI-359-86, MCC, Austin, TX, 1986.

[3] Mark Aronoff. *Word Formation in Generative Grammar.* MIT Press, Cambridge, MA, 1976.

[4] Steven Bird. Finite-state phonology in HPSG. In *Proceedings of the 14th International Conference on Computational Linguistics, COLING-92*, pages 74–80, 1992.

[5] Robert D. Borsley. Subjects and complements in HPSG. Technical Report CSLI-87-107, Center for the Study of Language and Information, Stanford University, 1987.

[6] Gosse Bouma. Feature structures and nonmonotonicity. *Computational Linguistics*, 18(2):183–203, 1992.

[7] Joan Bresnan, editor. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Mass., 1982.

[8] Jonathan Calder. Feature-value logic: Some limits on the role of defaults. In Mike Rosner, C.J. Rupp, and Rod Johnson, editors, *Proceedings of the Workshop on Constraint Propagation, Linguistic Description, and Computation*, pages 20–32. Instituto Dalle Molle IDSIA, Lugano, 1991.

[9] Luca Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76:138–164, 1988.

[10] Luca Cardelli and Peter Wegner. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys*, 17(4):471–522, 1985.

[11] Bob Carpenter. *The Logic of Typed Feature Structures.* Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1992.

[12] Bob Carpenter. Skeptical and credulous default unification with applications to templates and inheritance. In Ted Briscoe, Ann Copestake, and Valeria de Paiva, editors, *Default Inheritance Within Unification-Based Approaches to the Lexicon.* Cambridge University Press, 1992.

[13] Bob Carpenter and Carl Pollard. Inclusion, disjointness and choice: The logic of linguistic classification. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 9–16, 1991.

[14] Ann Copestake. Defaults in the LRL. In Ted Briscoe, Ann Copestake, and Valeria de Paiva, editors, *Default Inheritance Within Unification-Based Approaches to the Lexicon.* Cambridge University Press, 1992.

[15] Ann Copestake, Valeria de Paiva, and Antonio Sanfilippo. Functionality of the LKB. Draft version, unpublished, University of Cambridge, Computer Laboratory, March 1991.

[16] Walter Daelemans, Koenraad De Smedt, and Gerald Gazdar. Inheritance in natural language processing. *Computational Linguistics*, 18(2):205–218, 1992.

[17] Roger Evans and Gerald Gazdar. The DATR papers. Technical Report SCRP 139, School of Cognitive and Computing Sciences, University of Sussex, 1990.

[18] Jens Erik Fenstad, Per-Kristian Halvorsen, Tore Langholm, and Johan van Benthem. *Situations, Language and Logic*. Reidel, Dordrecht, 1987.

[19] Daniel Flickinger. *Lexical Rules in the Hierarchical Lexicon*. PhD thesis, Stanford University, 1987.

[20] Daniel Flickinger, Carl Pollard, and Thomas Wasow. Structure-sharing in lexical representation. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 262–267, 1985.

[21] Gerald Gazdar. Linguistic applications of default inheritance mechanisms. In P. Whitelock, M.M. Wood, H.L. Somers, R. Johnson, and P. Bennett, editors, *Linguistic Theory and Computer Applications*, pages 37–67. Academic Press, London, 1987. Also available as Cognitive Science Research Paper CSRP 070, University of Sussex.

[22] Jack Hoeksema. *Categorial Morphology*. Garland, New York, 1985.

[23] Ronald M. Kaplan. Three seductions of computational psycholinguistics. In P. Whitelock, M.M. Wood, H.L. Sommers, R. Johnson, and P. Bennett, editors, *Linguistic Theory and Computer Applications*, pages 149–188. Academic Press, London, 1987.

[24] Lauri Karttunen. D-PATR: A development environment for unification-based grammars. Technical Report CSLI-86-61, Center for the Study of Language and Information, Stanford University, 1986.

[25] Andreas Kathol. Passives without lexical rules. In John Nerbonne, Klaus Netter, and Carl Pollard, editors, *HPSG and German*. Center for the Study of Language and Information, Stanford, 1993. CSLI Lecture Notes Series.

[26] Kimmo Koskenniemi. A general computational model for word-form recognition and production. In *Proceedings of the 10th International Conference on Computational Linguistics, COLING-84*, pages 178–181, 1984.

[27] Hans-Ulrich Krieger. Representing and processing finite automata within typed feature formalisms. Technical report, Deutsches Forschungsinstitut für Künstliche Intelligenz, Saarbrücken, Germany, 1993. Forthcoming.

[28] Hans-Ulrich Krieger. Single link overwriting—a conservative non-monotonic extension of unification-based inheritance networks. Technical report, Deutsches Forschungsinstitut für Künstliche Intelligenz, Saarbrücken, Germany, 1993. Forthcoming.

[29] Hans-Ulrich Krieger and John Nerbonne. Feature-based inheritance networks for computational lexicons. In Ted Briscoe, Ann Copestake, and Valeria de Paiva, editors, *Default Inheritance Within Unification-Based Approaches to the Lexicon*. Cambridge University Press, 1992. A version of this paper is also available as a DFKI Research Report. Also published in Proceedings of the ACQUILEX Workshop on Default Inheritance in the Lexicon, Technical Report No. 238, University of Cambridge, Computer Laboratory, October 1991.

[30] Hans-Ulrich Krieger, John Nerbonne, and Hannes Pirker. Feature-based allomorphy. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, 1993. A version of this paper is also available as a DFKI Research Report.

[31] John T. Maxwell III and Ronald M. Kaplan. The interface between phrasal and functional constraints. In Mike Rosner, C.J. Rupp, and Rod Johnson, editors, *Proceedings of the Workshop on Constraint Propagation, Linguistic Description, and Computation*, pages 105–120. Instituto Dalle Molle IDSIA, Lugano, 1991.

[32] Fernando C.N. Pereira and Stuart M. Shieber. *Prolog and Natural-Language Analysis.* CSLI Lecture Notes, Number 10. Center for the Study of Language and Information, Stanford, 1987.

[33] Carl Pollard. The syntax-semantics interface in a unification-based phrase structure grammar. In Stephan Busemann, Christa Hauenschild, and Carla Umbach, editors, *Views of the Syntax-Semantics Interface: Proceedings of the Workshop on "GPSG and Semantics", Technische Universität Berlin, 22-24.Feb 1989*, pages 167–184. KIT FAST, Technische Universität Berlin, 1989.

[34] Carl Pollard and Ivan Sag. *Information-Based Syntax and Semantics. Vol. I: Fundamentals.* CSLI Lecture Notes, Number 13. Center for the Study of Language and Information, Stanford, 1987.

[35] Susanne Riehemann. Word formation in lexical type hierarchies. a case study of *bar*-adjectives in german. Master's thesis, Eberhard-Karls-Universität Tübingen, Seminar für Sprachwissenschaft, 1993.

[36] Graeme D. Ritchie, Stephen G. Pulman, Alan W. Black, and Graham J. Russell. A computational framework for lexical description. *Computational Linguistics*, 13(3-4):290–307, 1987.

[37] Graham Russell, John Carroll, and Susan Warwick-Armstrong. Multiple default inheritance in a unification-based lexicon. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 215–221, 1991.

[38] Stuart Shieber, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and Mabry Tyson. The formalism and implementation of PATR-II. In Barbara J. Grosz and Mark E. Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge*, pages 39–79. AI Center, SRI International, Menlo Park, Cal., 1983.

[39] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes, Number 4. Center for the Study of Language and Information, Stanford, 1986.

[40] Gert Smolka. A feature logic with subsorts. LILOG Report 33, WT LILOG–IBM Germany, Stuttgart, Mai 1988.

[41] Gert Smolka. Feature constraint logic for unification grammars. IWBS Report 93, IWBS–IBM Germany, Stuttgart, November 1989.

[42] Gert Smolka. Residuation and guarded rules for constraint-logic programming. Research Report RR-91-13, DFKI, Saarbrücken, 1991.

[43] Gert Smolka and Hassan Aït-Kaci. Inheritance hierarchies: Semantics and unification. *Journal of Symbolic Computation*, 7:343–370, 1989.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-67608 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse 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 can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-92-34
Philipp Hanschke: Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35
Manfred Meyer:
Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment
18 pages

RR-92-36
Franz Baader, Philipp Hanschke:
Extensions of Concept Languages for a Mechanical Engineering Application
15 pages

RR-92-37
Philipp Hanschke: Specifying Role Interaction in Concept Languages
26 pages

RR-92-38
Philipp Hanschke, Manfred Meyer:
An Alternative to H-Subsumption Based on Terminological Reasoning
9 pages

RR-92-40
Philipp Hanschke, Knut Hinkelmann: Combining Terminological and Rule-based Reasoning for Abstraction Processes
17 pages

RR-92-41
Andreas Lux: A Multi-Agent Approach towards Group Scheduling
32 pages

RR-92-42
John Nerbonne:
A Feature-Based Syntax/Semantics Interface
19 pages

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-93-18
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-93-26
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-93-33
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

DFKI Technical Memos

TM-91-13
Knut Hinkelmann: Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

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

DFKI Documents

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.):
UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse 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

DFKI Documents

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht
1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Software-komponenten für natürlichsprachliche Systeme
139 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
264 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatischen Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deklarations-basierte Unterstützung des Dokumentenanalyse zur Fehlererkennung 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: Parsing 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 Buchholz: Klassik bei Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26
Arno Tollmann: Realisierung eines Werkzeugs zur Hilfe des Constraint-Systems CONTAX
26 Seiten

D-92-27
Hans-Ulrich Krieger, Klaus Rüdenauer, Thomas Lukasiak: 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 Buschke, Thom Frühwirth: Terminological Reasoning with Constraint Handling Rules
12 pages

D-93-02
Gabriele Schmidt, Frank Peters, Gernot Lamprecht: 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
101 Seiten

D-93-05
Stephan Kudoske, Ingrid Gnoth, Jochen Heinsohn, Rainer Hoch, Werner 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 erkennungsgesteuerter Kontrollrahmen zur partiellen Textanalyse
55 Seiten

D-93-08
Hans-Ulrich Krieger, Ulrich Schäfer: TDL ExtraBlight User's Guide
35 pages

Derivation Without Lexical Rules

Hans-Ulrich Krieger

RR-93-27
Research Report