Word Manager: A System for the Definition, Access and Maintenance of Lexical Databases

Marc DOMENIG

Institut für Informatik
Universität Zürich-Irchel
CH-8057 Zürich
Switzerland

Abstract

This paper describes Word Manager, a system which is currently the object of a research project at the University of Zürich Computer Science Department. Word Manager supports the definition, access and maintenance of lexical databases. It comprises a formal language for the implemenation of morphological knowledge. This formal language is integrated in a graphics-oriented, high-level user interface and is *language independent.* The project is now in the prototyping phase where parts of the software are pretty far advanced (the user interface) and others are still rudimentary (the rule compiler/runtime system). The design of the system was strongly influenced by Koskenniemi's two-level model /Koskenniemi 1983/, its successors /Bear 1986/, /Black 1986/, /Borin 1986/, /Darymple 1987/, the ANSI-SPARC 3-Schema Concept /ANSI-X3-SPARC 1975/ and visual programming techniques /Bocker 1986/, /Myers 1986/. We will focus the discussion on one aspect: the *user interfacing* for the *construction* of the lexical data base.

1. Introduction

As I have argued elsewhere /Domenig 1986, 1987a, 1987b/, a dedicated system yields many advantages for the implementation, use and maintenance of lexical databases. The functionality of *general purpose database management systems* - e.g. relational ones - is too limited for lexical databases because they are not tuned to the task at hand; in particular, they do not provide for a formalism which is suited to describe *linguistic knowledge.* The reason why we would like to have such a formalism is that it allows us to take advantage of a computer's processing abilities, i.e. we may construct a lexical database which is not only a collection of purely 'static' information - a set of entries - but has 'dynamic' capabilities. For instance, the latter might be that it can analyse and generate inflected or composed word forms. "What would be the advantage of that?" one might ask. "It is no problem to *add on* these capabilities to a purely 'static' set of entries stored within a commercially available database management system by writing programs in the host language to this system!"

The answer is: there are a lot of advantages and I hope to clarify some of them in this paper. A dedicated system supports the construction, use and maintenance of lexical databases much more directly than a general purpose database management system in conjunction with a conventional programming language interface. Word Manager was designed as such a system, whereas Word Manager does not necessarily manage *all* the information stored in a lexical database. At this stage of the project, it manages only *morphological knowledge*, i.e. it would be quite feasible to use it as a front-end to a database managed by a general purpose system.

2. Overview of the user interfacing

Word Manager distinguishes two quite different interfaces for the construction and maintenance of lexical databases: one for the specification of what I term *conceptual knowledge* (linguist interface) and one for the specification of what I call *non-conceptual knowledge* (lexicographer interface). The former is the place where the kind of morphological knowledge is defined which can be typically found in grammars, the latter is a dialogue-oriented interface for the entering of the bulk of the data.

The relationship between the two interfaces is one of a strong dependency, i.e. the lexicographer interface depends very much on the specifications in the linguist interface. Much of the machine-lexicographer dialogue can be inferred automatically from these specifications. The formalism employed in the linguist interface was designed to be powerful enough to implement morphological knowledge of several natural languages on the one hand, yet dedicated enough to be easy to handle for linguists. Moreover, it provides the opportunity to experiment with different conceptual approaches within a certain framework. The following section will outline it.

3. The specification of morphological knowledge in the linguist interface

The linguist interface is conceived as a highly controlled environment which takes advantage of the latest hard- and software technology. This means that the user does not communicate with the computer on the level of its operating system except for when the application is started. On the level of the operating system, each morphological knowledge specification is represented by a so-called *document icon* (the two rightmost icons in Fig. 1 are document icons). By mousing such an icon, the user may start the application and load the specification stored in the document. Alternatively, he could start it by mousing the application icon (the leftmost icon in Fig. 1 is the application icon). Within the application environment, each document (morphological knowledge specification) is represented by a so-

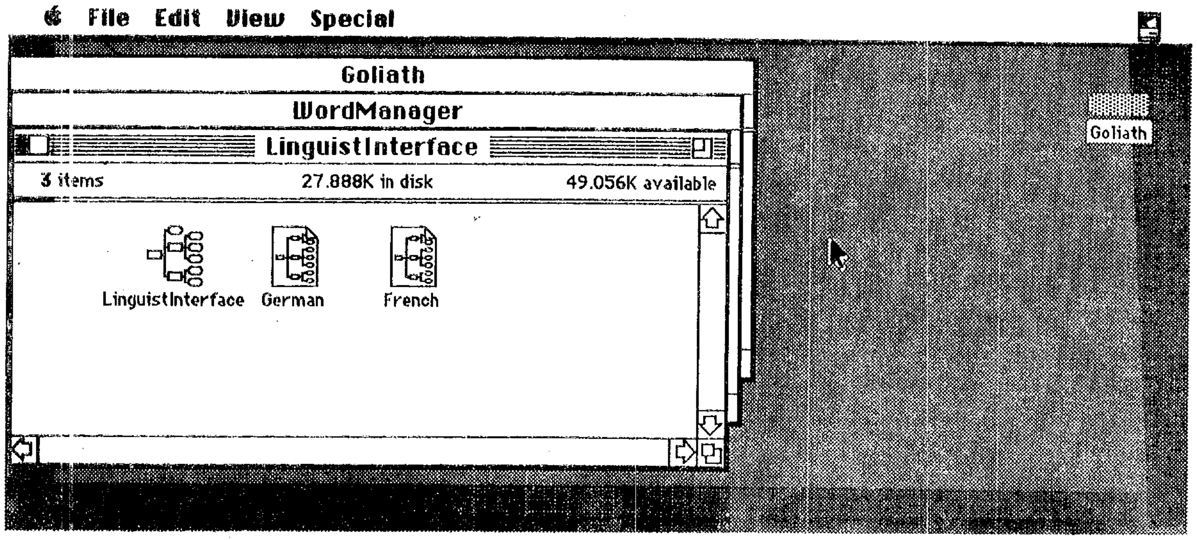
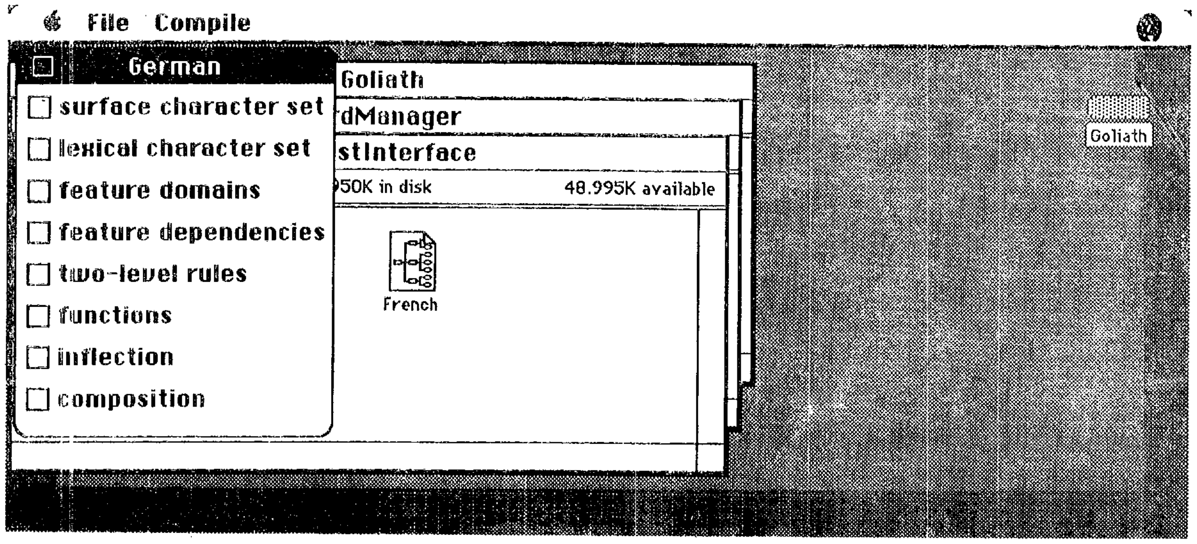


Fig. 1: The level of the operating system



☐ surface character set
☐ lexical character set
☐ feature domains
☐ feature dependencies
☐ two-level rules
☐ functions
☐ inflection
☐ composition

Fig. 2: The top level of the linguist interface application

called *tool-window* which contains eight labelled check-boxes (see Fig. 2). Each of these check-boxes represents a window, the name and purpose of which is indicated by the label:

The window *surface character set* provides for the definition of the character set out of which so-called *surface strings* are built. Surface strings are used for the surface representation of word forms. The window is graphics-oriented, i.e. most of the definitions are done with mouse- and menu commands (see Fig. 3).

The window *lexical character set* provides for the definition of the character set out of which so-called *lexical strings* are built. Lexical strings are used to define linguistically motivated abstractions of surface strings. The set is usually defined to include characters denoting morpheme boundaries and/or morphophonemes. The window is very similar to the surface character set window.

The window *feature domains* provides for the domain specifications of the *attribute-value pairs* which are used in the rule- and constituent specifications (see below). The window is a text-oriented editor. An example specification is shown in Fig. 4.

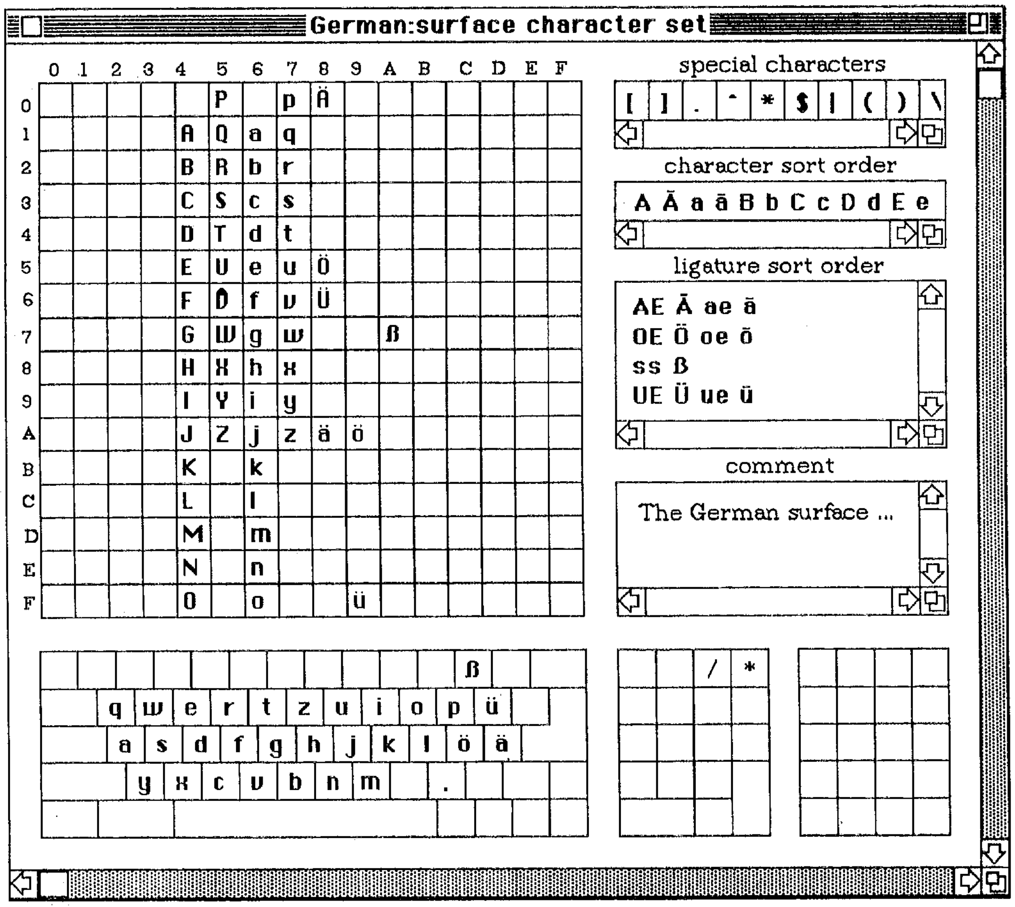
```
Cat      (N V A P Q)
Case     (NOM GEN DAT ACC)
Gender   (M F N)
Num      (SG PL)
```

Fig. 4: Example definition in window *feature domains*

The window *feature dependencies* provides for the definition of dependencies between features. An example specification is shown in Fig. 5.

```
(Cat N)    demands    Gender
```

Fig. 5: Example definition in window *feature dependencies*


German:surface character set

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0						P		p	Ä							
1					A	Q	a	q								
2					B	R	b	r								
3					C	S	c	s								
4					D	T	d	t								
5					E	U	e	u	Ö							
6					F	V	f	v	Ü							
7					G	W	g	w			ß					
8					H	X	h	x								
9					I	Y	i	y								
A					J	Z	j	z	ä	ö						
B					K		k									
C					L		l									
D					M		m									
E					N		n									
F					O		o			ü						

special characters

[ ] . ^ * $ | ( ) \

character sort order

A Ä a ä B b C c D d E e

ligature sort order

AE Ä ae ä
OE Ö oe ö
ss ß
UE Ü ue ü

comment

The German surface ...

Fig. 3: The window *surface character set*

The window *two-level rules* provides for the definition of morphophonemic rules which realize the mapping function between the surface- and lexical strings. The rules specified here are similar to those in DKIMMO/TWOL /Darymple 1987/. The window is a text-oriented editor. An example specification is shown in Fig. 6 (the two rules handle noun genitive [e]s: the first one replaces "+" by "e" as in Strausses, Reflexes, Reizes, the second one duplicates "s" as in Verhältnisses, Verhängnisses, Erschwernisses).

The window *functions* provides for the definition of rules for the kind of string-manipulations which should not be realized with two-level rules (because their power would be excessive or they would imply the introduction of linguistically unmotivated morphophonemes). The window is a text-oriented editor. An example specification is shown in Fig. 7 (ReCap recapitalizes prefixed nouns).

```
ReCap
    "(.*)A(.*)/\1a\2" value
    "(.*)B(.*)/\1b\2" value
    ...
    "(.*)Z(.*)/\1z\2" value

     "^a(.*)/A\1" value
     "^b(.*)/B\1" value
     ...
     "^z(.*)/Z\1" value
```

Fig. 7: Example definition in window *functions*

```
".*[sxz]" (ICat N-ROOT)   ""          (ICat N-ENDING)   "+s/es" (Case GEN)

".*"      (ICat N-ROOT)   "nis/niss"  (ICat N-ENDING)   "+s/es" (Case GEN)
```

Fig. 6: Example definition in window *two-level rules*

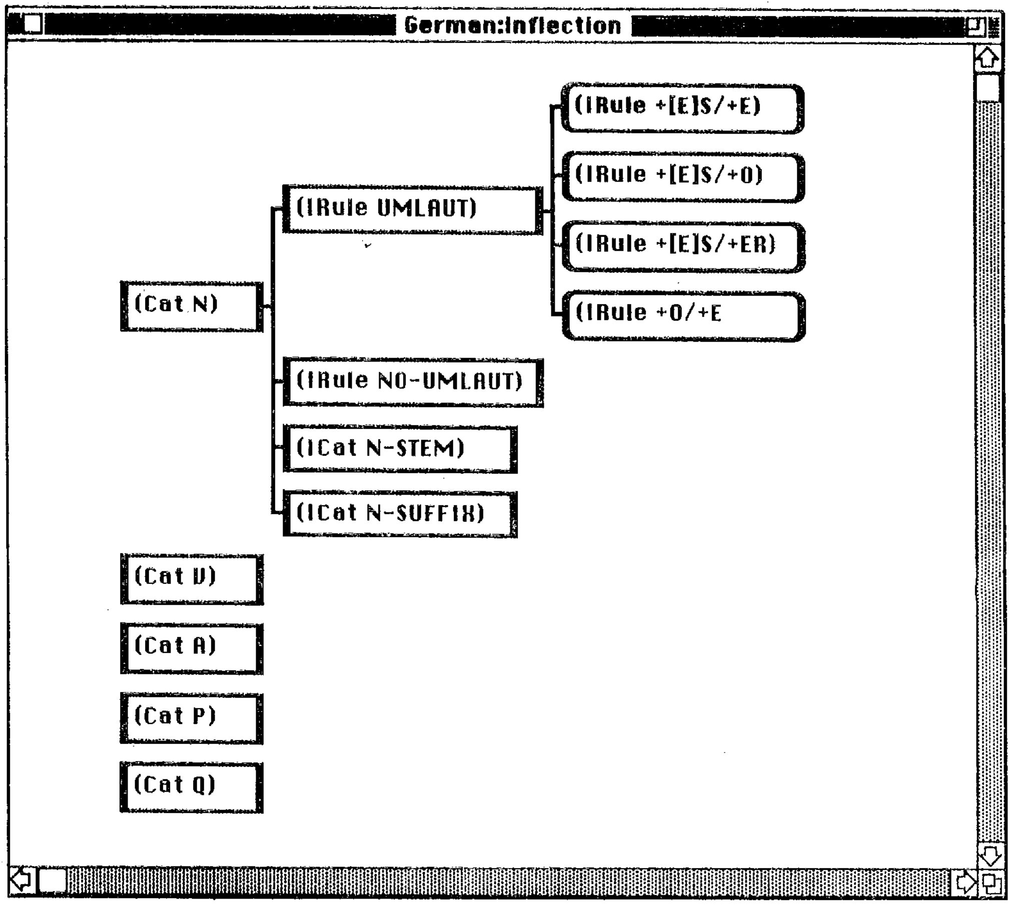
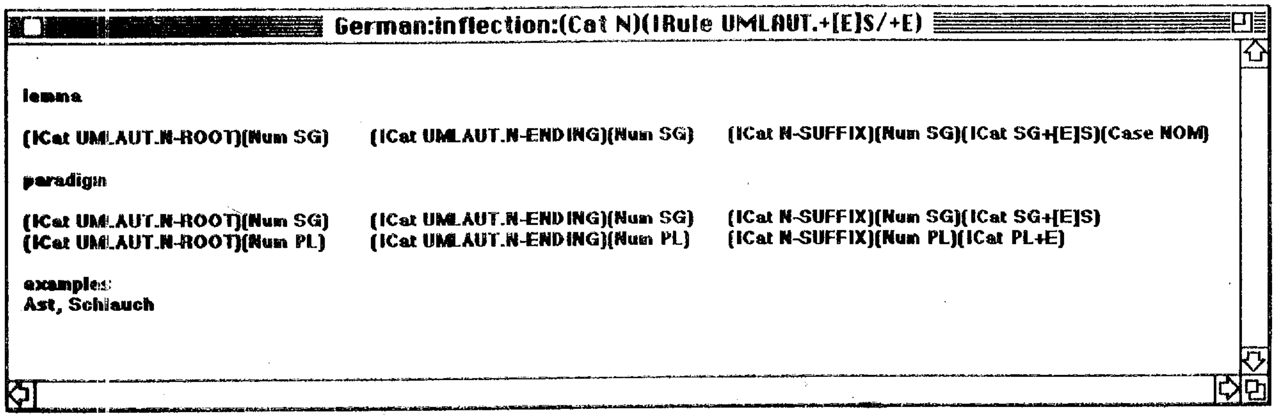


Fig. 8: The window *inflection*

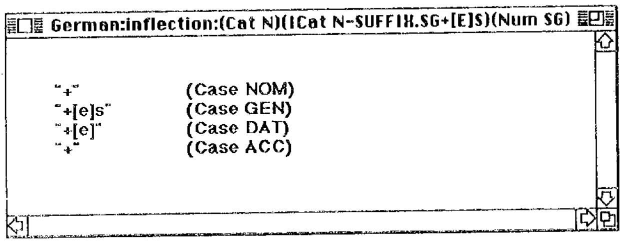
The window *inflection* provides for the definition of word classes with their inflectional rules and paradigms. This window is a graphical *tree editor* which allows the interactive construction of an n-ary tree. This tree is used to structure the rules and constituents which define the word classes. The structuring criteria are *features* (attribute value pairs) and the structure has the following semantics: the rules specified in a subtree operate on the constituents specified within the same subtree. Fig. 8 shows a subtree which contains rules and constituents for German noun inflection (only the top branch (IRule UMLAUT) is expanded down to the terminal nodes). The terminal nodes of the tree contain either rules or constituents. By mousing them, the user may open text-oriented editor windows. An example of a rule is shown in Fig. 9: it consists of matching constraints (realized by feature sets) on the constituents and specifies a set of *lemma forms* and a set of *word forms*. In the example, the set of lemma forms - specified below the keyword 'lemma' - is a single word form (nominative singular;



```
German:inflection:(Cat N)(IRule UMLAUT.+[E]S/+E)

lemma

(ICat UMLAUT.N-ROOT)(Num SG)    (ICat UMLAUT.N-ENDING)(Num SG)    (ICat N-SUFFIX)(Num SG)(ICat SG+[E]S)(Case NOM)

paradigm

(ICat UMLAUT.N-ROOT)(Num SG)    (ICat UMLAUT.N-ENDING)(Num SG)    (ICat N-SUFFIX)(Num SG)(ICat SG+[E]S)
(ICat UMLAUT.N-ROOT)(Num PL)    (ICat UMLAUT.N-ENDING)(Num PL)    (ICat N-SUFFIX)(Num PL)(ICat PL+E)

examples:
Ast, Schlauch
```

Fig. 9: Example inflectional rule window

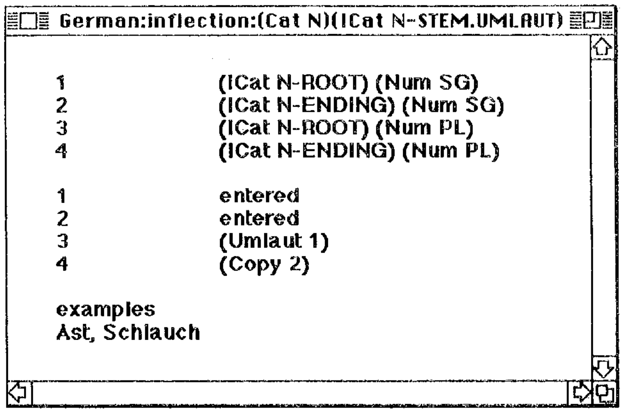
the pattern of feature sets identifies exactly one form which is put together by the concatenation of three constituents). The set of word forms - specified below the keyword 'paradigm' - consists of eight elements (the case paradigm; the two patterns of feature sets identify exactly eight forms, each of which is put together by the concatenation of three constituents). The constituent windows specify either so-called *hard-coded constituents* or *constituent types*. The former are feature sets which are associated with 'hard-coded' lexical strings (see Fig. 10); they are typically used to specify inflectional



```
German:inflection:(Cat N)(ICat N-SUFFIX.SG+[E]S)(Num SG)

"+"        (Case NOM)
"+[e]s"    (Case GEN)
"+[e]"     (Case DAT)
"+"        (Case ACC)
```

Fig. 10: Example window with hard-coded constituents

affixes. The latter are feature sets where the associated strings are represented by *place holders*, i.e. the strings are not specified yet but will be entered later, either via the lexicographer interface or by the firing of compositional rules (see Fig. 11).

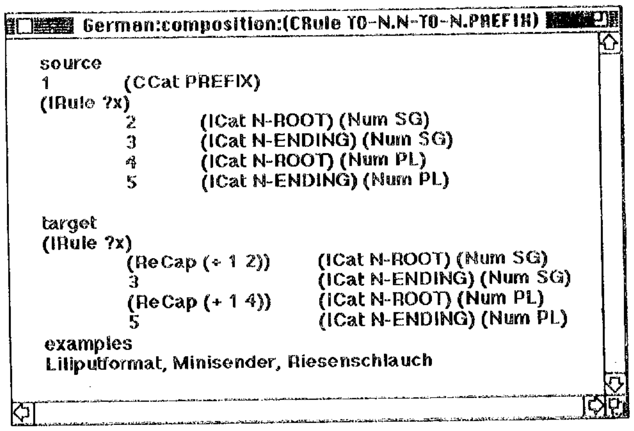


```
German:inflection:(Cat N)(ICat N-STEM.UMLAUT)

1          (ICat N-ROOT) (Num SG)
2          (ICat N-ENDING) (Num SG)
3          (ICat N-ROOT) (Num PL)
4          (ICat N-ENDING) (Num PL)

1          entered
2          entered
3          (Umlaut 1)
4          (Copy 2)

examples
Ast, Schlauch
```

Fig. 11: Example window with constituent types

They are typically used to specify word roots. From what has been said so far, we may infer how an entry into the database is made and what it will generate: the specification of an entry requires the identification of an inflectional rule and the specification of the lexical strings which are represented as place holders in the constituents matched by the rule. Usually, this means that one or two strings have to be entered. From this, the system may generate the entire inflectional paradigm of the word. Notice that the user of the linguist interface defines with his specification what a *word* is (viz. a set of lemma forms and a set of word forms). Moreover, Word Manager imposes the convention that only entire words - and no isolated word forms - may be entered into the database.

The window *composition* provides for the definition of compositional rules and constituents (affixes). This window is a graphical tree editor similar to the window *inflection* where the rules and constituents are structured by features which qualify them. The rules in the terminal nodes (see Fig. 12) define new *potential* word entries by specifying how constituents of existing entries are combined with



```
German:composition:(CRule TO-N.N-TO-N.PREFIX)

source
1        (CCat PREFIX)
(IRule ?x)
         2        (ICat N-ROOT) (Num SG)
         3        (ICat N-ENDING) (Num SG)
         4        (ICat N-ROOT) (Num PL)
         5        (ICat N-ENDING) (Num PL)

target
(IRule ?x)
         (ReCap (+ 1 2))     (ICat N-ROOT) (Num SG)
         3                   (ICat N-ENDING) (Num SG)
         (ReCap (+ 1 4))     (ICat N-ROOT) (Num PL)
         5                   (ICat N-ENDING) (Num PL)
examples
Liliputformat, Minisender, Riesenschlauch
```

Fig. 12: Example compositional rule window

each other and with constituents defined in the window *composition* (derivational affixes). These rules are usually not applied generatively but analytically, because a generative application is likely to *overgenerate* (theoretically, the user may specify an arbitrary number of features which restrict excessive generation, but I believe that this is unpractical in most cases, because it implies that the lexicographer has to specify a host of features for each entry). The purpose of the rules is that all derived and compound words may be entered into the database via the triggering of such rules. This has the advantage that the system (automatically) keeps track of the derivational history and therefore the morphological structuring of each entry.

4. The lexicographer interface

Given a compiled specification of the conceptual morphological knowledge defined within the linguist interface, Word Manager may generate a dialogue which guides the lexicographer towards the identification of the inflectional/compositional rules that must be triggered in order to add an entry to the database. In the case of non-composed words, for example, Word Manager may simply navigate in the tree which structures the inflectional rules (specified in the window *inflection*), posing questions according to the structuring criteria.

In the case of composed words, Word Manager may apply the compositional rules in analytical mode, provided that the 'initial' information consists of a word string. Such an analytical application of the rules is usually not very overgenerating - in contrast to a generative application-, i.e. the system will be able to present a reasonably limited number of selection choices.

5. Conclusion

The advantages of a *dedicated* system like Word Manager for the management of lexical databases are manifold. In this paper, we have restricted the discussion to the advantages yielded during the construction of the database. These are by no means the only ones: the dedication also implies that the *overhead* of non-dedicated systems (e.g. general purpose DBMS in conjunction with conventional programming languages), i.e. the features which are superfluous for lexical databases, is avoided. On the other hand, Word Manager provides features which a general purpose system will never have, viz. the special formalism to implement morphological knowledge. This is not only beneficial from the point of view of the interfacing to the database but also from the point of view of the software design: in the dedicated system, the morphological knowledge is a part of the *conceptual database schema* (in the terminology of database theory) and thus belongs to the *kernel* of the system, as it were. When a general purpose database management system in conjunction with a conventional programming language is used to implement the same kind of knowledge, it has to be implemented within the *external schemata* to the database and thus repeatedly for each of them. The so-called *code factoring* is therefore much better in a dedicated system: the knowledge is more centralized and implemented with a minimum of redundancy.

References

/ANSI-X3-SPARC 1975/ ANSI/X3/SPARC Study Group on Data Base Management Systems: "Interim Report 75-02-08." *FDT (Bull. of the ACM SIGMOD) 7*, 1975.

/Bear 1986/ Bear J.: "A Morphological Recognizer with Syntactic and Phonological Rules." in: *Proceedings of the 11th International Conference on Computational Linguistics*, Bonn, August 25-29, 1986.

/Black 1986/ Black A.W., et al.: "A Dictionary and Morphological Analyser for English." in: *Proceedings of the 11th International Conference on Computational Linguistics*, Bonn, August 25-29, 1986.

/Bocker 1986/ Bocker H.D., et al.: "The Enhancement of Understanding Through Visual Representations." in: *Human Factors in Computing Systems*, CHI'86 Conference Proceedings (Special Issue of the SIGCHI Bulletin), Boston, April 13-17, 1986.

/Borin 1986/ Borin L.: "What is a Lexical Representation?" in: *Papers for the Fifth Scandinavian Conference of Computational Linguistics*, Helsinki, December 11-12, 1985. University of Helsinki, Department of General Linguistics, Publications No. 15, 1986.

/Darymple 1987/ Dalrymple M., et al.: *DKIMMO/TWOL: A Development Environment for Morphological Analysis*, in Kaplan R., Karttunen L.: "Computational Morphology." Course Script LI283, 1987 Linguistic Institute, Stanford University, June 29-August 7, 1987.

/Domenig 1986/ Domenig M., Shann P.: "Towards a Dedicated Database Management System for Dictionaries." in: *Proceedings of the 11th International Conference on Computational Linguistics*, Bonn, August 25-29, 1986.

/Domenig 1987a/ Domenig M.: *Entwurf eines dedizierten Datenbanksystems für Lexika. Problemanalyse und Software-Entwurf anhand eines Projektes für maschinelle Sprachübersetzung*. Niemeyer Verlag, Tübingen, Reihe 'Sprache und Information' Bd. 17, 1987.

/Domenig 1987b/ Domenig M.: "On the Formalisation of Dictionaries." in: *Sprache und Datenverarbeitung*, 1/1987.

/Koskenniemi 1983/ Koskenniemi K.: *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Dissertation at the University of Helsinki, Department of General Linguistics, Publications No. 11, 1983.

/Myers 1986/ Myers B.A.: "Visual Programming, Programming by Example, and Program Visualization, a Taxonomy." in: *Human Factors In Computing Systems*, CHI'86 Conference Proceedings (Special Issue of the SIGCHI Bulletin), Boston, April 13-17, 1986.