Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-95-06

FEGRAMED
An Interactive Graphics Editor for Feature Structures

Bernd Kiefer, Thomas Fettig

December 1995

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

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

Stuhlsatzenhausweg 3
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.

Dr. Dr. D. Ruland
Director

FEGRAMED
An Interactive Graphics Editor for Feature Structures

Bernd Kiefer, Thomas Fettig

DFKI-RR-95-06

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-9002 0 and FKZ ITWM-9403).

ISSN 0946-008X

FEGRAMED
An Interactive Graphics Editor for Feature Structures

Bernd Kiefer, Thomas Fettig

December 11, 1995

Contents

1 Terminology 5

2 How to use Fegramed 8

2.1 Viewing Feature Structures . . . 8

2.1.1 Scrolling . . . 8

2.1.2 Zooming . . . 8

2.1.3 Imploding . . . 9

2.1.4 Tag Expansion . . . 10

2.1.5 Sorting Edges . . . 11

2.1.6 Hiding Edges . . . 12

2.1.7 Obscured Edges . . . 13

2.1.8 Setting Depth and Other Preferences . . . 13

2.1.9 Searching for Atoms, Tags or Edges . . . 13

2.2 Editing . . . 15

2.2.1 Selection of Items . . . 15

2.2.2 Inserting Feature–Value (Edge–Vertex) Pairs . . . 16

2.2.3 Creating Complex Vertices . . . 16

2.2.4 Copy and Paste . . . 16

2.2.5 Deleting Vertices (CLEAR, REMOVE and CUT) . . . 17

2.2.6 Other Editing Features . . . 18

2.2.7 Building and Removing Coreferences . . . 18

3 Menus 20

3.1 The STRUCTURES Menu . . . 20

3.1.1 NEW . . . 20

3.1.2 OPEN, SAVE and SAVE AS . . . 20

3.1.3 CLOSE and CLOSE ALL . . . 21

3.1.4 SAVE & RETURN and RETURN . . . 21

3.1.5 PRINT and PAGE SETUP (Mac only) . . . 21

3.1.6 QUIT . . . 21

3.2 The EDIT Menu . . . 21

3.2.1 COPY and PASTE . . . 22

3.2.2 REMOVE, CLEAR and CUT . . . 22

3.2.3 BUILD TAG 22
3.2.4 REMOVE TAG 22
3.2.5 NEGATE 23
3.2.6 SET TEXT and SET SPECIAL... 23
3.2.7 EXPORT TO SCRAP (Mac only) 23
3.3 The FIND Menu 23
3.4 The VIEW Menu 24
3.4.1 IMPLODE 24
3.4.2 HIDE 24
3.4.3 ZOOM IN, ZOOM OUT and SHOW ROOT 24
3.4.4 HIDDEN FEATURES 24
3.4.5 FEATURE ORDER 25
3.4.6 DEPTH 25
3.4.7 SCROLLING (Mac only) 26
3.4.8 SET DEFAULTS (Mac only) 26
3.4.9 SAVE DEFAULTS (Mac only) 26
3.4.10 REFRESH and REFRESH ALL 26
3.4.11 SHOW OBSCURED 27
3.5 The FONTS menu (Mac only) 27

4 Building an Interface to Fegramed **28**
4.1 Feature Structure Files 28
4.2 Communication 29
4.3 Sort and Hide Files 31

A Motif Particulars **33**
A.1 Command Line Options 33
A.2 Resources 33
A.3 Menus and Keyboard Shortcuts 35

B Mac Particulars **36**
B.1 File Types Used by Fegramed 36
B.2 Menus and Keyboard Shortcuts 37

Introduction

This paper describes a tool for supporting grammar development in those linguistic frameworks which employ some constraint–based formalism, such as LFG (Lexical Functional Grammar), HPSG (Head-Driven Phrase Structure Grammar), FUG (Functional Unification Grammar) and CUG (Categorial Unification Grammar). These approaches have in common that all or at least a substantial part of the grammar (such as rules, lexical entries, node labels etc.) is represented as sets of attribute–value pairs.

In LISP or Prolog the structures can be internally represented as lists, but it is much more convenient and sometimes even indispensable to use graphical representations when developing grammars. During grammar processing, feature structures can become quite large (up to several thousand nodes), such that a customized view of the feature structure, which allows to selectively focus on relevant parts, becomes essential.

Fegramed provides a fully interactive editor for developing, maintaining and viewing feature structures. It is a tool that is built to cope with the complexity of feature structures in grammar development and use.

Chapter 1

Terminology

In the following, we will first clarify some technical terms we use in this manual and describe how they are related to the terms that are normally used in the linguistics literature.

Feature structures are directed graphs. Fegramed is a graph viewer with a specific graphical representation rather than a program that knows about the semantics of feature structures. We will therefore use graph terminology in this manual.

Vertices and Edges

What are normally called 'attributes' in linguistics are the (named) *edges* of a graph; the 'values' (both atomic and complex) are the *vertices*. There are different kinds of complex vertices (values) in Fegramed: *conjunctions*, *disjunctions*, *implications*, *lists* and *function applications*. There is *no* semantics

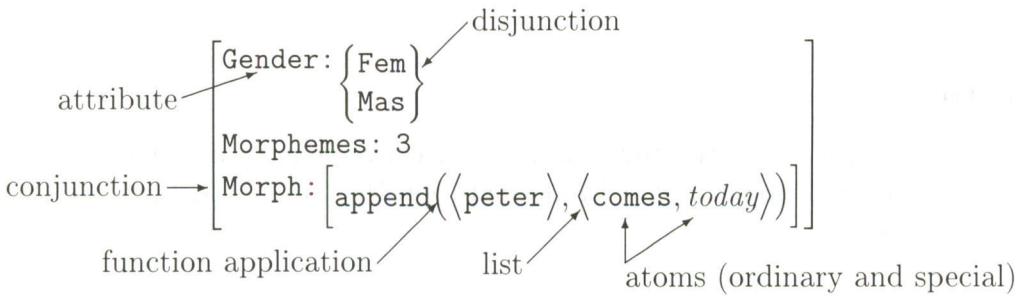

Figure 1.1: Basic notions

attached to these notions, they are simply vertices that are displayed in a particular manner and can have an outdegree (the number of edges leaving a vertex) that is greater than or equal to zero. All other vertices have an outdegree of zero, i.e. no edges can leave them. These vertices are also called *atoms* or *atomic vertices*. There are two types of atoms: ordinary and special atoms. A special atom will be displayed by Fegramed in a different font and

also has a special attribute in the external representation. Otherwise, there is no difference between ordinary and special atoms.

Look at the figure 1.1 to see how edges and complex and atomic vertices are displayed.

In our representation, as in many linguistic notations, edge names are separated from vertices by a ':'. Conjunctions are drawn with square brackets, disjunctions with curly braces. Atomic vertices are just shown by their names. To see what all those objects really look like on your computer and to have a small sample to play with, there should be a file `everything.fs` in your Fegramed distribution.

Coreferences

In feature structures two attributes can share the same (identical) information. This means that in the graph representation of the feature structure, several edges point to the same vertex. This vertex is only displayed once and a boxed number is attached to it. All other occurrences of this vertex are only represented by this boxed number, i.e. they are all marked by identical *tags*. In feature structure terms, these tags are called *coreferences*. An example of coreferences can be seen in figure 1.2.

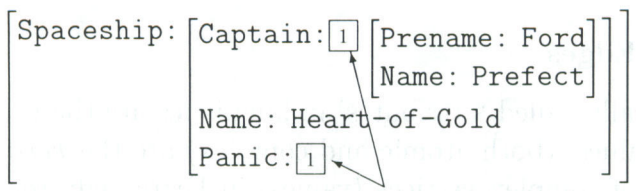

Figure 1.2: Coreferences are represented by numbers in boxes

The Hide Symbol

When drawing feature structures, it is often convenient to hide some part of the structure. In the graphical representation this is realized by a special *'hide' symbol.*

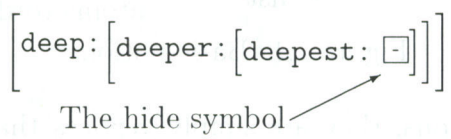

Figure 1.3: An example of a nested structure

This 'hide' symbol shows up in two different situations. If a restriction is put on the depth of display of the feature structure, any complex vertex that exceeds that depth is represented by the hide symbol. The user can also

create a hide symbol manually by imploding a complex vertex. This means that the vertex will be shown as a hide symbol until it is explicitly exploded. In contrast, hide symbols coming from depth restriction are dynamically generated and removed while zooming into or out of the feature structure.

Obscured Edges

In many formalisms, the members of disjunctions are not named. In these formalisms, the edges pointing to the different disjuncts do not matter.

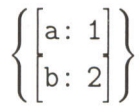

Figure 1.4: Disjunctions are normally not labeled

In Fegramed, all complex vertices have edge labels, but they can always be suppressed. Edges whose names are suppressed but whose values are visible are called *obscured edges*. The edge names can be made visible by checking the SHOW OBSCURED item in the VIEW menu. If you do this, you can see how obscured edges are specified:

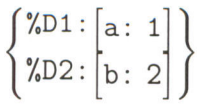

Figure 1.5: Obscured edges made visible

Obscured edges have names that start with the '%' character. If you have a formalism where the order of the disjuncts matters (as in distributed disjunction formalisms), you can use numbered edge names as in figure 1.5 to distinguish the disjuncts. But beware! The edges are sorted in lexicographic order, so you may have to have leading zeroes with the numbers to get the right order.

Obscured edges can be used in any complex vertex, not just in disjunctions. If you want to specify for example type information in a conjunction, you can do it with an obscured edge as shown in figure 1.6.

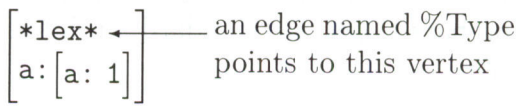

Figure 1.6: Representation of types using obscured edges

Chapter 2

How to use Fegramed

Some General Remarks

In this manual, we will repeatedly refer to certain actions like clicking the mouse or using keyboard shortcuts. As these actions differ on the different platforms, we will now briefly explain some of them.

Keyboard shortcuts (or accelerators as they are called in the X/Unix environment) exist for most of the actions described below. Because most of them differ in the implementations, only their existence is mentioned in the following text. Which shortcut to use for which action is explained in the appendices A.3 and B.2.

The mouse click actions differ because there is only one mouse button available on the Mac. Double clicking on the Mac is the same as middle–clicking the mouse under X/Unix. Single–clicking, on the other hand, is the same as left–clicking under X/Unix. The right mouse button under X/Unix has a special meaning that will be described later on.

2.1 Viewing Feature Structures

2.1.1 Scrolling

Scrolling is performed by clicking the mouse in the appropriate regions in the scroll bars. They are standard scroll bars, so there should be no problem using them.

2.1.2 Zooming

Feature structures are typically nested objects. If a feature structure is very complex and you want to examine all edges (with potentially many vertices attached to them), you would normally have to scroll through the whole

feature structure. But if you limit the depth to which the feature structure is displayed, any parts of the feature structure that are deeply nested are replaced by the hide symbols discussed earlier. This makes the feature structure smaller on the screen, but has the shortcoming that not all information is displayed at once.

To get more information about one specific vertex, you have to be able to select and redisplay certain vertices of the feature structure. This is achieved by zooming into the structure. If you double–click on some displayed vertex (double–click on the Mac, middle–click under X/Unix) other than the outermost vertex (which we call the *root node*), it will become the new root and the feature structure will be drawn up to the appropriate (deeper) level. See figure 2.1 for an example of a zooming operation.

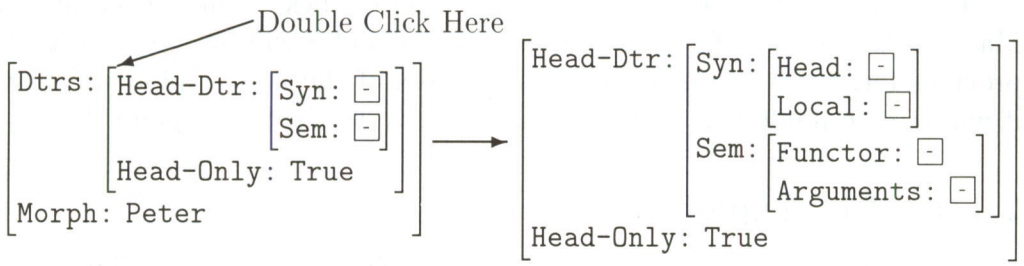

Figure 2.1: Zooming into the value of `Dtrs`

Another method to zoom in to a vertex is to select it with a single click (it will be displayed in reverse video) and select the ZOOM IN command from the EDIT menu.

To get back to the embedding structure, you can zoom out again. One method is to click outside of the whole structure.[1] This has the effect of zooming out one level. The same effect can be achieved by choosing the item ZOOM OUT in the EDIT menu or its keyboard shortcut.[2]

You can return immediately to the outermost vertex of the whole structure by choosing SHOW ROOT in the EDIT menu or using its keyboard shortcut.

2.1.3 Imploding

Another way to shrink large feature structures to obtain a better overview is to implode a (currently uninteresting) vertex. You can do this by marking the vertex (just single click on the vertex, it will be redisplayed in reverse

[1] In the X/Unix Version, this can be done by right–clicking the mouse anywhere in the display area of the window

[2] The shortcuts for the different versions of the feature editor are given in the appendices A.3 and B.2. Most of the menu options can be reached by a shortcut.

video) and selecting IMPLODE from the EDIT menu. The whole vertex will be replaced by a hide symbol.

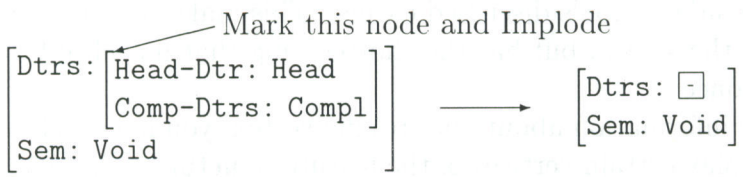

Figure 2.2: Imploding the vertex under `Dtrs`

Note that corefered vertices which were displayed under the imploded vertex will potentially be shown at another point. So you may have to implode several vertices to get the desired view.

To 'explode' the vertex again, simply double–click (double–click on the Mac; middle–click on X/Unix) the hide symbol and it will be expanded to its previous form. If you imploded a child vertex of the now 'exploded' vertex it will remain imploded. Only the clicked-on vertex will be expanded.

2.1.4 Tag Expansion

Two edges can share a single vertex. This is indicated by tags, as illustrated in figure 2.3. The little boxes with numbers are called tags (or coreferences).

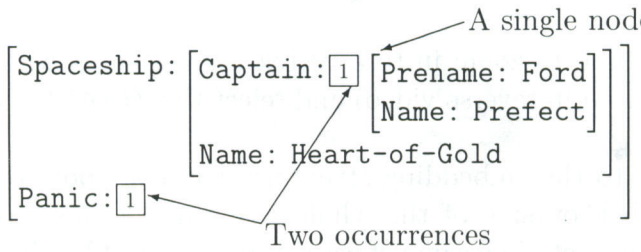

Figure 2.3: `Panic` and `Spaceship|Captain` share a single vertex

If two edges point to a single vertex, the number of the tag that appears behind them is the same. You can also see in this figure that complex vertices are shown only once, even if they are used in several places in the feature structure. All the occurrences will be marked with the same tag, but the whole vertex is only shown in one place. If you run into the situation that you want to look at a certain part of the feature structure but some of the vertices are only tags, you can double–click on a tag to make the expansion of the vertex appear at that edge.

It still holds that the vertex is only shown once; it has only disappeared from where it was shown before.

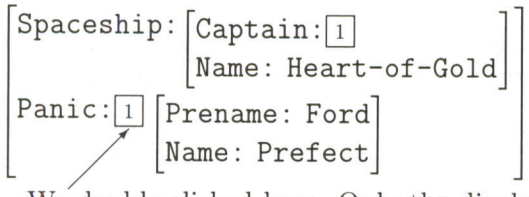

Figure 2.4: Expansion of the tag behind `Panic`

2.1.5 Sorting Edges

If you are working with big feature structures, you may be more interested in certain edges at every level of embedding. Ordering these edges in a specified way allows you to display the most important features at the top of the window while others, less important ones can be accessed by scrolling. The order in which edges are shown can be changed interactively. If you choose the item FEATURE ORDER from the VIEW menu you will be presented with a dialog box like the one in figure 2.5.

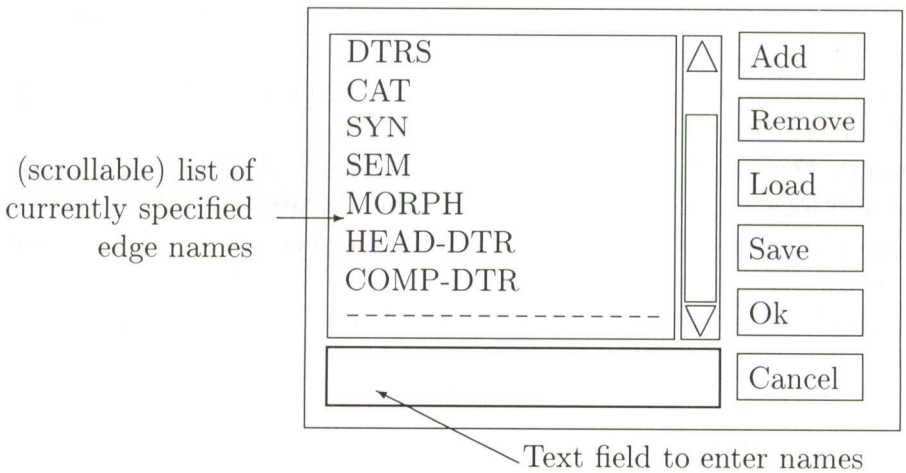

Figure 2.5: FEATURE ORDER dialog box

The list field shows you the edge labels that are specified so far. At the end of the list appears a dashed line that means 'end of list'. If nothing was specified yet, the dashed line will be the only entry in the list box. The edges in the feature structure are sorted in the order that is given by the list. The edge names in the feature structure have to match the entries in the list *exactly*, including case. All edges with edge names that are not mentioned in the list follow the specified ones in alphabetical order.

If the list of specified names is to long to fit into the list field, you may have to use the scroll bars to inspect it fully or to select a certain item.

To add a new edge name, first type it into the text field. Then mark the list item in front of which the new edge name should be placed, and press the

ADD button. The new edge name will appear in the list immediately before the marked name. To add names at the *end* of the list, you have to mark the dashed line. If you have selected an edge or an atom before you invoked the FEATURE ORDER command, the name of that object will already be entered into the text field. Thus, you can easily add new items to the list while working with the structures.

To remove an item from the list, select it and press the REMOVE button. The dashed line cannot be removed because it is not a real entry but it is necessary to make it possible to add entries to the end of the list.

Pressing the OK button takes you back to the currently active window accepting all changes that you made to the list. The new list reflects the order that will be used to sort the current and all subsequently opened windows. Pressing the CANCEL button will discard all changes you made to the list and leave the windows unchanged. If you want to re-order an already opened window, bring it to the front and select REFRESH from the VIEW menu. REFRESH ALL reorders all currently open windows.

The LOAD and SAVE buttons allow you to save these lists to disk for use in another Fegramed session or as a default order. Thus, you do not have to specify the same lists again and again.

Pressing one of these buttons gives you a file selection dialog to specify the file to load or to save to, respectively.[3]

On startup, Fegramed loads the default sort order from a file. On the Mac, name and location of this file belong to the saveable defaults (see section 3.4.8) and is preset to the name 'FeditSort' and Fegramed's application directory. In the X/Unix version, either the file is specified by the contents of the `Fegramed.sort` resource (cf. appendix A, section A.2) or it tries to load '~/.fedit_sort'.

2.1.6 Hiding Edges

To hide an edge means that edges with this name will be hidden throughout the feature structure. As a consequence, the vertices these edges are pointing to will not show up either, except in the case that another edge with a different name is *not* hidden and points to the same vertex. (The coreference tags will remain even if only one visible edge points to that vertex. This way you know that it is shared at different places that may be invisible.)

To specify an edge name as hidden, you can use either of two methods. Select an edge with the appropriate name and issue the HIDE command from the EDIT menu. Alternatively, use a dialog window by selecting HIDDEN FEATURES in the VIEW menu. The dialog window is identical to the

[3]In the current X/Unix version, the file cannot be selected. The list is always in the file "~/.fedit_sort" unless you specify another file using the `Fegramed.sort` resource.

FEATURE ORDER dialog, but the order of the features in the list does not matter.

To hide all edges with a certain name, type the name into the text field and press the ADD button (the HIDE command is just an abbreviation for this).

To reveal a hidden edge, you have to use the dialog window: select the name in the list and press the REMOVE button

The LOAD and SAVE buttons work the same way as in the feature order dialog, just the default files are different: On the Mac, it is the file with name 'FeditHide' in Fegramed's application directory, in the X/Unix version, it is '~/.fedit_hide' or the file specified by the `Fegramed.hide` resource (cf. appendix A, section A.2).

2.1.7 Obscured Edges

Obscured edges were introduced in Chapter 1 (see page 7). They are normally not visible and are therefore used in situations where only values (vertices) are expected, for example in disjunctions. You can toggle a switch that affects the visibility of the obscured edges by choosing SHOW OBSCURED in the VIEW menu.

2.1.8 Setting Depth and Other Preferences

The depth up to which feature structures are shown can be limited (see Chapter 1). This limit can be set interactively by choosing the menu item DEPTH from the VIEW menu. You will get a dialog box in which you can specify the number of levels that should be shown.

In the Mac version of the feature editor, there is a similar dialog to set the scroll speed for the stepping scroll. You can reach it via the SCROLLING menu item. You can enter the number of pixels that will be scrolled for each press of one of the scroll bar arrows.

Most of the settings that affect the appearance of feature structures only affect the currently active and subsequently opened windows, but not the other open windows. To see the effects of a sort or hide list change in every window, you can use the REFRESH ALL menu entry in the VIEW window. To obtain the effect in the active window only, use REFRESH.

2.1.9 Searching for Atoms, Tags or Edges

The FIND function can be used (as in most text editors) to locate a certain edge or atom in the structure. If you choose FIND from the FIND menu, the dialog box shown in figure 2.6 appears.

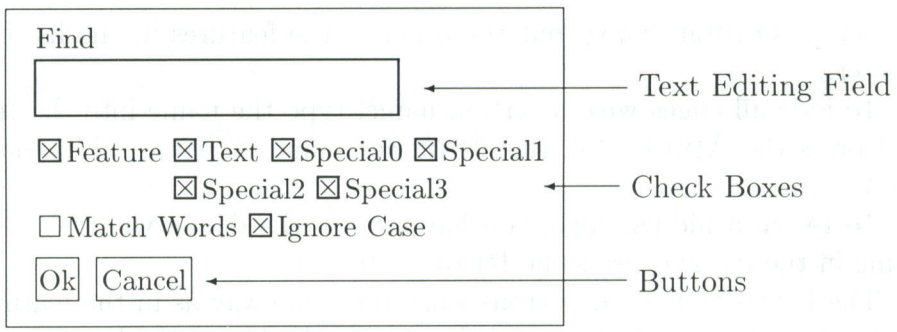

Figure 2.6: FIND dialog box

In this dialog, the check boxes FEATURE, TEXT and SPECIAL0 ... SPECIAL3 specify which items are examined during the search (edge labels, ordinary or special atoms). The FIND function examines the structure recursively, comparing the string given in the FIND text field with the names of the specified items. If you have selected an edge or an atom before opening the FIND dialog, the name of this edge or atom will be entered into the text field.

The case of letters is ignored if the IGNORE CASE box is checked. If WHOLE WORD is checked, the comparison of the find string and the names only succeeds if they have the same length. You can determine the place from where the search is started as follows:

- If a vertex is marked, the search starts at that vertex.
- If an edge is marked, the search starts at the vertex the edge points to.
- If nothing is marked, search starts at the current root node.

If the find action is successful, the item that is found will be marked by highlighting and scrolled into sight if necessary. If nothing appropriate is found, nothing happens.

There is another FIND function that works on the coreferences in the structure: FIND TAG. It lets you type in the tag number you want to look for. There are no options to choose, but the actual search procedure is the same as for FIND in all other respects.

The FIND AGAIN function resumes the last find action, be it a FIND or a FIND TAG.

2.2 Editing

Fegramed is not only suitable to view structures but also allows you to create and edit them.

2.2.1 Selection of Items

First of all, if you want to edit something in a structure, you have to select it. This is done by single clicking the object you want to edit. Items that are selected are shown white on black.

If you have selected an edge or an atom, you can change it simply by typing. There are certain special facilities for string editing:

1. If the whole string is selected and you start to type, the string will be completely replaced by what you type in.
2. If the whole string is selected, you can delete it completely by tapping Backspace.
3. You can move the cursor inside the string by:
   - clicking where it should go; or
   - using the left and right arrow keys (← and →) (as long as you do not leave the string).
4. Backspace deletes a character to the left (except when the whole string is selected as in case 2), Del deletes a character to the right.
5. Return has no meaning in string editing.

If an item is selected, you can select other items by using the arrow keys. But beware: if you select an atom or an edge and press the right arrow key, the cursor will appear. If you want to avoid this, you have to press the command key[4] while tapping the arrow keys. Just play a bit with the cursor keys to see how they work.

So far we have seen how the information contained in a feature structure can be selected and how text can be changed. We now show how vertices and edges can be added and deleted.

[4]With the Macintosh version press the ⌘ key, with the Motif version use Meta

2.2.2 Inserting Feature–Value (Edge–Vertex) Pairs

If you have selected a complex vertex, you can insert a new feature–value pair (a vertex and an edge pointing to it) by pressing Return.

In disjunctions, lists and function lists the edge name is "%?" by default, so you will not see it if you have not checked SHOW OBSCURED. See also figure 1.4 on page 7 and section 3.4.11 on page 27.

2.2.3 Creating Complex Vertices

To create a complex vertex, you have to select an *atomic* vertex first and then choose the appropriate command item from the COMPLEX menu or use its keyboard shortcut. The atom will then be replaced by the chosen complex vertex.

2.2.4 Copy and Paste

You can copy any selected vertex or edge to the internal scrap using the COPY command either from the EDIT menu or by its key shortcut.

The possible targets to PASTE the copied object to depend on the type of the object: edges can only be pasted into complex vertices, while vertices can be pasted to any vertex, complex or atomic. When a vertex is pasted to a complex vertex, the effect of the operation is as if a CLEAR operation had been done before the PASTE (described later in the section 2.2.5).

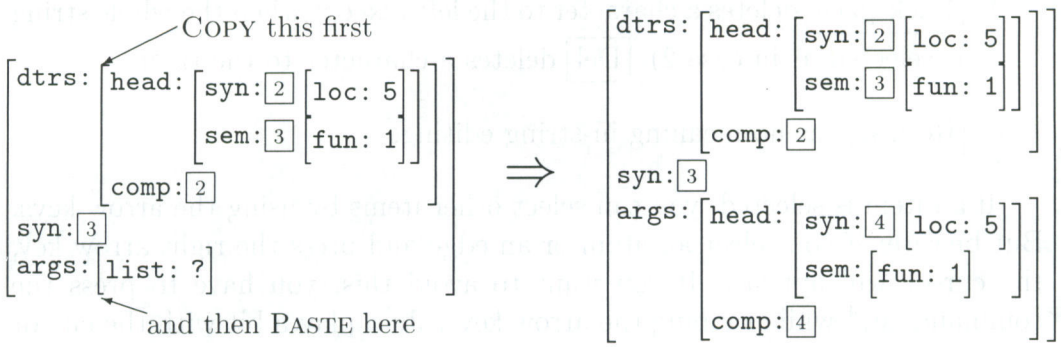

Figure 2.7: COPY and PASTE of a complex vertex

If a complex vertex has been copied, all *internal* coreferences are preserved, while the *external* ones, i.e. edges pointing into the copied subgraph from the outside, will disappear (cf. figure 2.7; there is no coreference between `syn` and `args|head|sem`). A method to move a complex vertex preserving all coreferences will be described in section 2.2.7.

You can also copy and paste between feature structures displayed in different windows.

2.2.5 Deleting Vertices (CLEAR, REMOVE and CUT)

There are three different deletion operations, CLEAR, REMOVE and CUT.

CLEAR works only on complex vertices and has the effect of destroying the internal structure of the selected vertex while all edges pointing to that vertex stay intact. The selected vertex is replaced by an atomic vertex with the name "?".

Note that vertices remain in the graph if they are reachable not only through the cleared vertex but also through other vertices. Look at figure 2.8 for an example of a CLEAR operation.

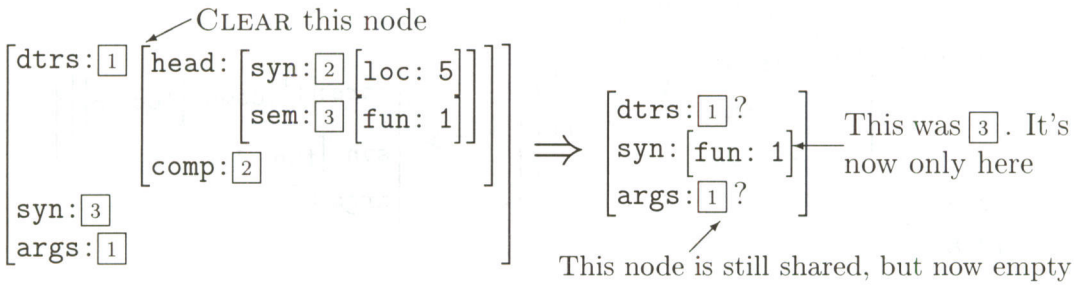

Figure 2.8: CLEAR applied to the value of `dtrs`

A REMOVE action on a vertex deletes the vertex itself and all edges pointing directly to it. All vertices and edges that lose connection to the outermost vertex are also deleted. All vertices that are pointed to by edges from outside will remain in the graph. Figure 2.9 shows an example of the REMOVE operation.

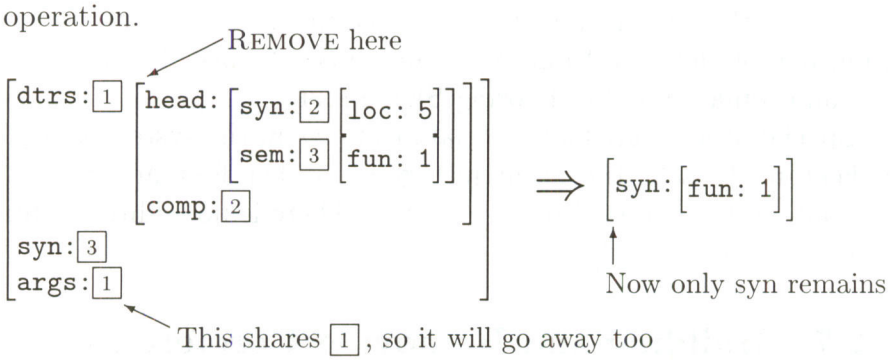

Figure 2.9: REMOVE applied to the value of `dtrs`

REMOVE also works on edges. In this case, the edge and the vertex to which it points are deleted. All vertices pointed to from outside remain in the graph. See also figure 2.10 and figure 2.11 as examples of REMOVE operations on edges.

CUT works on all vertices as well as on edges. The selected object is first copied to the internal cut buffer as by a COPY operation. A vertex is then deleted with CLEAR, an edge with REMOVE.

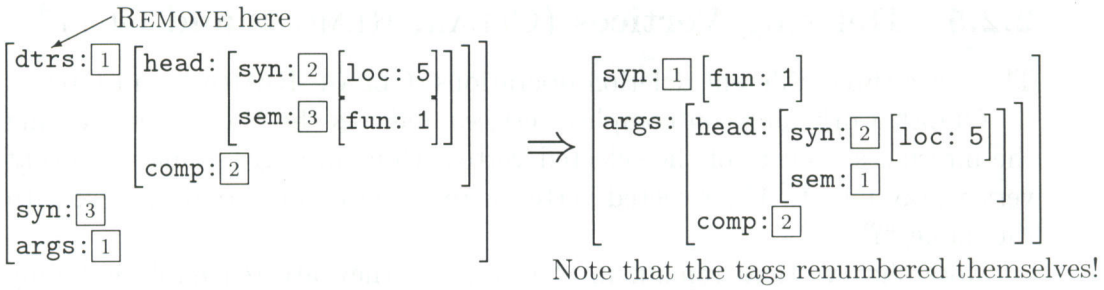

Figure 2.10: REMOVE applied to the edge dtrs

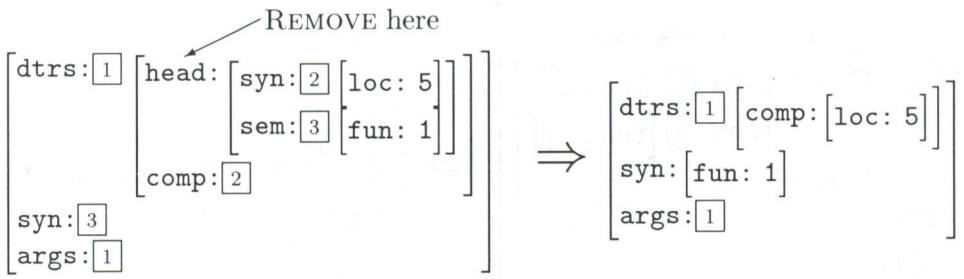

Figure 2.11: REMOVE applied to the edge head

2.2.6 Other Editing Features

A vertex can be negated by choosing the NEGATE command from the EDIT menu. Similarly, you can change a selected atom into a *special* atom (for Fegramed, that just means it is displayed in the appropriate special font) by using one of the SET SPECIAL commands. A special atom can be turned into an ordinary one by choosing SET TEXT.

Special atoms may have special meanings in the system *using* Fegramed, in Fegramed itself, the atom just gets another font and it gets a special attribute in the external representation. There is no additional functionality that applies to special atoms.

2.2.7 Building and Removing Coreferences

How do you make two or more edges point to the same vertex, creating a coreference? Let's assume you have an edge which you want to point to an already existing vertex. Select the edge and select BUILD TAG from the EDIT menu. (On the Macintosh a hand cursor will appear to indicate "Build Tag" mode.) Now select the vertex you want the edge to point to. Matching coreference boxes will appear at the selected vertex (if there was not one already) and on the edge that was selected in the first place.

If you are in the "Build Tag" mode and change your mind, you can leave it by either reselecting BUILD TAG from the menu (on the Mac, the item

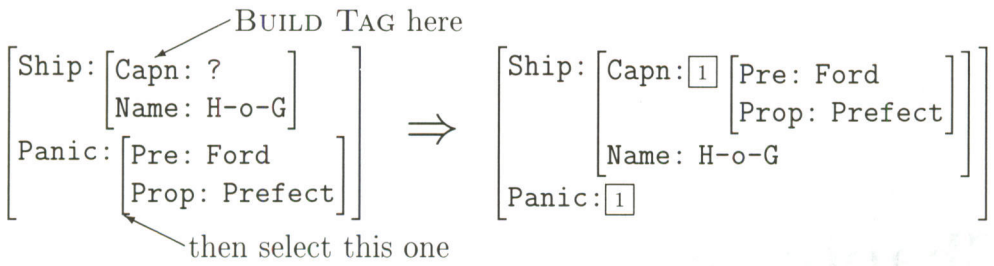

Figure 2.12: Select `Capn`, BUILD TAG and then the vertex under `Panic`

changed its name into CANCEL TAGGING) or clicking outside the structure (in the X/Unix version, right click anywhere in the window).

If you want to remove a coreference (you think that an edge should not point to some shared vertex anymore) you can select it and either delete it by using the REMOVE command or use REMOVE TAG, which will make the edge point to a newly created vertex named "?". With REMOVE TAG, you can then proceed to create a new structure under the old edge. Notice that the vertex the edge pointed to will not be changed by this operation.

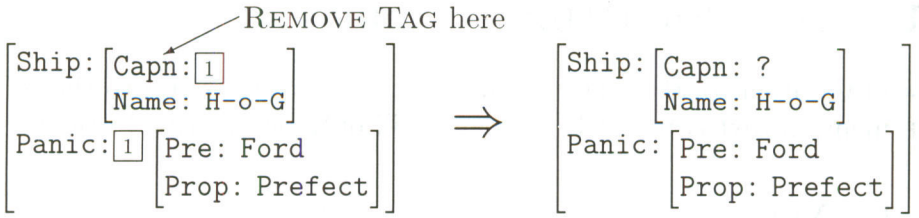

Figure 2.13: Removing a tag.

A hint for advanced users: BUILD TAG and REMOVE TAG can be used to *move* a structure from one point to another. Suppose you want to move a structure from an edge `a` to an edge `b`. To do this, create tags between them (using BUILD TAG) and then immediately remove the tag behind edge `a`. The effect will be that the tags are deleted and the structure is pasted in at edge `b`.

This differs from doing a CUT behind `a` and a subsequent PASTE behind `b`. CUT and PASTE will not preserve coreferences that point into the substructure behind `a` from the outside. Using the BUILD/REMOVE TAG mechanism preserves all (internal and external) coreferences.

Chapter 3

Menus

Since most of the functionality of Fegramed was described in the previous chapter, we say little about it here and refer to the appropriate sections in the previous chapter. This is just a short reference guide to every menu item.

3.1 The STRUCTURES Menu

Most of the items of the STRUCTURES menu are very similar to those of the FILE menu in text editors. These parts will only be described briefly.

3.1.1 NEW

This command allows you to create a completely new feature structure. Choosing NEW gives you a new window with only one (atomic) vertex in it, which has the name "?". You can now start editing your feature structure.

3.1.2 OPEN, SAVE and SAVE AS

These items load a feature structure from or save it to a file. If you select OPEN, a file dialog box pops up that lets you choose the file to load.

SAVE will save the feature structure of the current window (in X/Unix the window where you selected the SAVE item) into the file it was loaded from. If you select SAVE in a window that was created by the NEW command and does not know a filename, Fegramed will react differently in the different implementations. On the Mac, it will behave as if you had selected SAVE AS (see below). Under X/Unix, it will just display an alert box that tells you to use SAVE AS to save this window.

SAVE AS is used either if you want to save a feature structure in a file other than the original, e.g. after having edited the structure, or if you created

a completely new feature structure (with the NEW command) and want to specify the file it should be written to. If you select this item, a file selection dialog box pops up and lets you choose the directory and type in a name for your file.

3.1.3 CLOSE and CLOSE ALL

CLOSE attempts to close the current (topmost) window. If the window contents were changed, it will pop up a dialog box to ask you if you want to save the changes to disk, discard the changes or cancel the whole CLOSE operation. If you choose to save or discard the changes, the window will be closed; otherwise it will be kept open.

CLOSE ALL attempts to close all windows by invoking CLOSE on every open window of Fegramed. If you choose to cancel the operation for one of your changed windows, the whole CLOSE ALL operation will be interrupted.

3.1.4 SAVE & RETURN and RETURN

The commands SAVE & RETURN and RETURN return to the host application if Fegramed runs in client mode. Both tell the host application which of these commands was selected. SAVE & RETURN returns the name of the file that belongs to the window where the structure was saved, whereas RETURN simply tells the host to take control again.

For a detailed description of this mechanism see section 4.2, page 29.

3.1.5 PRINT and PAGE SETUP (Mac only)

Use PRINT to send the contents of the current window or the current selection to the printer. PAGE SETUP will allow you to modify printer settings before printing. These are common commands on the Mac.

3.1.6 QUIT

QUIT attempts to quit the Fegramed application. It issues a CLOSE ALL operation to close all windows first. If this operation is not canceled, it quits Fegramed.

3.2 The EDIT Menu

Note that each time you single-click some part of your feature structure it will be displayed in reverse video to indicate that it has been selected.

3.2.1 COPY and PASTE

You can easily copy parts of your feature structure from one point to another: Mark an object (an edge or a vertex) by clicking on it with the mouse. Then select COPY to copy it into the internal cut buffer. To paste it back in, select the target structure and select PASTE. If you copied an edge, the target has to be a complex vertex, if it was a vertex, it can be any vertex. For a detailed description, see section 2.2.4.

You can also copy and paste between feature structures displayed in different windows.

3.2.2 REMOVE, CLEAR and CUT

REMOVE and CLEAR are slightly different deletion operations. REMOVE works on edges as well as on vertices, CLEAR only on vertices. CUT first copies the selected object to the internal scrap, like COPY, and then deletes the object, a vertex with CLEAR, an edge with REMOVE. For a detailed description with examples, see section 2.2.5.

3.2.3 BUILD TAG

This command is used to create coreferences, i.e. more than one edge pointing to the same vertex. This is achieved by selecting an edge and specifying an existing vertex the edge should point to.

First, select the edge, then choose BUILD TAG and select the vertex you want the edge to point to. The new coreference is shown by identical tag boxes behind the previously selected edge and the selected vertex.

To cancel a BUILD TAG operation, reselect BUILD TAG or click anywhere outside the structure (in the X/Unix version, right-click anywhere in the window).

For a detailed description, see section 2.2.7.

3.2.4 REMOVE TAG

The item called REMOVE TAG gives you the possibility to discard tags, i.e., to make an edge that points to a shared vertex point to a fresh (atomic) vertex. To remove a tag, select the corresponding *edge* (not the tag itself!) and choose REMOVE TAG. The vertex the edge pointed to will not be affected, the edge will just point to a new atom named "?". This command will only work on edges pointing to a vertex that is shared by more than one edge (see also section 2.2.7).[1]

[1] A hint for advanced users: BUILD TAG and REMOVE TAG can be used to *move* a structure from one point to another. Suppose you want to move a structure from an

3.2.5 NEGATE

NEGATE toggles the negation of a (complex or atomic) vertex. Negated vertices are displayed with the negation symbol '¬' in front of them.

3.2.6 SET TEXT and SET SPECIAL...

Any ordinary atom in the structure can be turned into a special atom by using one of the SET SPECIAL... commands. There are currently four different special attributes, each of which has its own font and is marked differently in the external representation. Likewise, any special atom can be turned into an ordinary one using SET TEXT.

3.2.7 EXPORT TO SCRAP (Mac only)

This feature is only available in the Mac version of Fegramed. The menu item EXPORT TO SCRAP allows you to copy structures to the clipboard and use them later in other programs, e.g. in a text processor.[2] To do this, select a structure with the mouse and choose EXPORT TO SCRAP. To use it in the target application, simply use this application's PASTE command.

3.3 The FIND Menu

The menu items in the FIND menu let you search for edges, ordinary and special atoms in large feature structures that you cannot easily overview. There are three different commands, FIND, FIND TAG and FIND AGAIN, which will be described in this section.

After selecting FIND you will be given a dialog box in which you can enter the following information (see also figure 2.6 on page 14):

- what to find
- what to search: edges, ordinary or special atoms – or all of these
- whether to search for matching words
- whether to make the search sensitive to case

Fegramed will display the first matching item of the feature structure in reverse video to indicate success. If the search fails, the display is not changed.

---

edge a to an edge b. To do so, create tags between them (using BUILD TAG) and then immediately remove the tag behind edge a. The effect will be that the tags are deleted and the edge b will point to that vertex.

[2]Note, that the displayed structure is transferred to the clipboard, not its internal representation.

FIND TAG works similar to FIND, except that it searches for a tag number that you specify in the dialog.

Fegramed only searches for the first occurrence of your search key. You can use FIND AGAIN to continue the search. Fegramed will then go on to the next occurrence of a matching item and display it.

See also section 2.1.9 for a detailed description.

3.4 The VIEW Menu

3.4.1 IMPLODE

The IMPLODE item is useful to hide temporarily unwanted details of your feature structures. It reduces a (possibly very large) vertex to a special symbol ⊡ and so shrinks the whole structure. Double–clicking this symbol will bring back the original view.

3.4.2 HIDE

HIDE is another method to make your feature structure more transparent. After selecting an edge and the HIDE item all edges with this name (and the vertices they point to) are no longer displayed. If you want to bring them back to the screen, use the HIDDEN FEATURES command, described in section 2.1.6 on page 12.

3.4.3 ZOOM IN, ZOOM OUT and SHOW ROOT

You can zoom in to a structure by double–clicking it (middle–clicking it in X/Unix) or select it and choose the ZOOM IN command. To zoom out again, either click anywhere outside the feature structure (in the X/Unix version, you can also zoom out by Right–clicking the mouse anywhere in the window) or choose ZOOM OUT. The difference between ZOOM OUT and SHOW ROOT is the following: whereas ZOOM OUT zooms out to the next enclosing structure (that means one level higher), SHOW ROOT zooms out to the root of the whole structure (i.e. to the *absolutely* highest level).

3.4.4 HIDDEN FEATURES

This command gives you a dialog to interactively change the list of hidden features. To hide a feature means that all edges with a certain name will not be shown on the display (see also HIDE above).

To add a new item to the list, type its name into the text field (all edges bearing exactly that name will not be displayed) and press the ADD button.[3] To remove an item (the appropriate edges will show up again), select it in the list box and press the REMOVE button.

The LOAD and SAVE buttons allow you to store the current list to disk for use in later Fegramed sessions or as default list.

To return to editing, press OK to accept all changes you made to the list or press CANCEL to discard them. For a detailed description of the dialog, see sections 2.1.6 and 2.1.5.

3.4.5 FEATURE ORDER

Choosing this menu item opens a dialog that lets you specify the order in which edges are displayed by Fegramed. The edges having names that match an entry in the list exactly (including case) will be sorted to the top in the specified order at any level of the feature structure. All other edges will follow in alphabetical order the ones mentioned in the list.

To add a new item, enter its name into the text field, select the item in front of which you want it to appear and press ADD. If you want to add it at the end of the list, select the dashed line. To remove an item, simply select it in the list box and press REMOVE.

The LOAD and SAVE buttons allow you to store the list to disk for use in later Fegramed sessions or as default order.

To return to editing, press OK to accept all changes you made to the list (the current and all subsequently opened windows will be sorted according to the new order) or press CANCEL to discard them. For a detailed description of the dialog, see section 2.1.5.

3.4.6 DEPTH

In the DEPTH dialog that is invoked by selecting DEPTH from the VIEW menu, you can specify the maximum depth of structure that will be displayed, i.e. you can tell Fegramed when to display the hide symbol (⊡) instead of more deeply embedded structure. The default value for this option is 10, so Fegramed will not use the symbol unless you have a structure that has more than 10 nested complex vertices (conjunctions, disjunctions, ...) on the screen.

[3] In fact this has the same effect as HIDE in the EDIT menu. You can see HIDE as an abbreviation for this operation.

3.4.7 SCROLLING (Mac only)

This item allows you to set the scrolling speed for the editor's windows. Just enter a value to change it. Small values cause slow scrolling, big values (very) fast scrolling. This feature is available only on the Mac.

3.4.8 SET DEFAULTS (Mac only)

To make the settings for the current window permanent for the rest of your work session with the editor, use SET DEFAULTS. The following values are fixed by this command:

- size and position of the window
- all changeable fonts (References, Text, and Special0 ... Special3)
- the depth limit for feature structure display
- the scroll speed
- name and location of the default sort and hide files

All subsequently opened windows will take these settings. Note that they will be "forgotten" when leaving Fegramed (unless you save them). This feature is available only on the Mac.

3.4.9 SAVE DEFAULTS (Mac only)

When choosing SAVE DEFAULTS, Fegramed will save the values that you fixed with SET DEFAULTS as startup preferences to a file called "FeditPrefs" in the Fegramed application directory. Next time you start Fegramed, the values listed in the previous section will be set as you fixed them.

You can copy the Preferences file to another directory and start Fegramed by double–clicking that file. Thus it is possible to have local vs. global preferences or different preferences for different systems. This feature is available only on the Mac.

3.4.10 REFRESH and REFRESH ALL

REFRESH redraws the topmost window using the actual settings. It was created because windows that are already open when you change for example the feature order are not refreshed automatically; you can also use it to refresh the window contents in case something weird happened to the display.

REFRESH ALL performs REFRESH on all currently open windows.

3.4.11 SHOW OBSCURED

If you generate a disjunction, it looks as if it had no edges leaving from it, just a list of vertices. However, this is not the whole truth. Actually, Fegramed generates an edge with name "%?" that is hidden as long as you do not switch on SHOW OBSCURED. Every edge that has a name that starts with a '%' character is an obscured edge and will not be visible if SHOW OBSCURED is switched off. On the Mac, this item will be checked in the menu when the feature is switched on.

3.5 The FONTS menu (Mac only)

This menu allows you to select the fonts for ordinary atoms, special atoms and coreference numbers interactively. Each menu point invokes a font selection dialog that is identical for all fonts. The font settings of the current window appear in this dialog as preset values in the controls.

In the dialog, there is a pop up menu for the available fonts, a pop up menu for the most common size values and a text edit field to enter rarely used size values. If the numbers in the size pop up menu appear outlined, the selected font is directly available in this size. Otherwise, the display font is computed by shrinking or expanding an available size. Furthermore, the check boxes in the dialog determine the font attributes that are applied.

Pressing the OK button will set the chosen window font to the values that were selected in the dialog and the current window will be redrawn. CANCEL closes the dialog too, but leaves the window font and the window itself unchanged.

Chapter 4

Building an Interface to Fegramed

This chapter is only relevant for people who want to use Fegramed as a tool inside another system that controls it. In the following, we describe what the interface looks like. "Ordinary" users of Fegramed do not have to worry about these things, they are mostly for developers.

4.1 Feature Structure Files

Exchange of feature structures with Fegramed takes place via feature structure files. These are plain text files containing the description of a feature structure. To interface to Fegramed, you must convert both ways between this format and your internal representation. The syntax of a feature structure file is described now. First the tokens:

Name	see text
Tag	#[0–9]+
TagIs	#[0–9]+ =
Attribute	#[A–Z]
AttributeIs	#[A–Z]=

Names can contain any character. If the first character of a name is one of '[', ']', '{', '}', '|', '(', ')', '#' or '\', it must be preceded by a '\' (which is removed when the token is read). If a name contains spaces, each blank must also be escaped with a '\'. The end of a name is marked by a non-escaped blank, so the last character of 'one-of) ' is ')'.

Only the attributes #S and #N are used at the moment, where #S= n specifies a special atom of type n (where n has to be in $0 \ldots 3$) and #N specifies that the vertex is negated. #S is not appropriate for complex vertices; it will be ignored at complex vertices as well as all other invalid attributes given.

The input syntax for the feature editor in BNF notation is as follows:

node	→	complexnode	
	→	atomicnode	
	→	Tag	; a coreference
	→	TagIs node	; the definition
			; of a coreference
	→	'(' path ')'	; a coreference specified
			; by a feature path
complexnode	→	ComplexName attributes edgelist	
ComplexName	→	'{'	; Disjunction
	→	'|'	; Implication
	→	'['	; Conjunction
	→	']'	; Function application
	→	'}'	; List
atomicnode	→	AtomName attributes	
attributes	→	Attribute attributes	; Only #N
	→	AttributeIs Name attributes	; Only #S=0...3 (atoms)
	→	ε	
edgelist	→	'(' EdgeName node ')' edgelist	
	→	ε	
path	→	EdgeName path	
	→	ε	
AtomName	→	Name	
EdgeName	→	Name	

4.2 Communication

Fegramed has a communication interface which allows it to work in a kind of 'client' mode. In this mode, it can also receive commands from a host application via a communication channel. Possible commands include opening a window containing a certain feature structure file, closing one or all windows and quitting completely.

The implementation of the communication channel differs between the Macintosh and the X/Unix versions of Fegramed.

On the Mac, the communication channel is implemented using communication files and Apple Events. To talk to Fegramed, you have to write your commands to a file, give it the appropriate file type (`'FBLK'`) and file creator (`'FEDI'`) and send Fegramed an 'Open Document' (`'odoc'`) Apple Event for that file. Fegramed will determine that it is a communication file by looking at its type (in contrast to the communication files, the type of an ordinary feature structure file has to be `'TEXT'`) and then read and execute

the commands it contains.

The communication channel of X/Unix Fegramed can only be activated by the command line option "-poll". The feature editor then checks its standard input regularly for incoming command sequences. Thus if you want to interface Fegramed with your application, you must execute Fegramed as a subcommand of the application with the "-poll" option enabled and write the 'remote' commands to its standard input.

The following remote commands are supported by both implementations. Note that each command must be followed by a newline:

- `feature=`*pathname*
  The feature editor loads the file *pathname* and opens a new window containing the feature structure. If the file cannot be found, an error dialog box will be displayed.

  The two subsequent commands affect only an immediately following `feature` command.

- `selected=`*edge-name**
- `path=`*edge-name**
  The `selected` command highlights the vertex at the end of the specified feature path while the `path` command zooms into the feature path.

  If a path can not be resolved completely, a warning will be displayed and the longest prefix path that could be matched is used.

  The `path` and `selected` commands have no effect except when given immediately before a `feature` command is transmitted. They affect only the view in the window that is opened by *this* `feature` command. If other commands are given in between, the `path` and `selected` commands are ignored.

- `close=`*pathname*
  This command closes the window that displays the file given by the pathname. If more than one window displays this file, only one of them is closed. The command is ignored if no such window exists. This command has the same effect as closing the window interactively. If the window contents have been changed, a dialog window will open and ask whether to SAVE or DISCARD CHANGES, or to CANCEL the `close` operation. If CANCEL is chosen, the window will not be closed despite the external command.

- `closeall`
  Tries to close all currently open windows, like CLOSE ALL from the STRUCTURES menu. If the contents of a window were changed, a dialog

is initiated as with `close` (see above). If CANCEL is chosen, the whole `closeall` operation is aborted.

- `quit`
  Tries to close all open windows in the same way as `closeall` and quits Fegramed if it succeeded in closing all windows.

The communication channel is bidirectional, i.e. it is not only used to transmit commands to Fegramed but also to get a "return value" back. Fegramed tells you if RETURN or SAVE & RETURN from the STRUCTURES menu were selected. Here is what it returns:

- `return`
  RETURN was selected, that means that Fegramed returns to its host application without a feature structure as result.

- `feature=`*pathname*
  SAVE & RETURN was selected; in this case, Fegramed returns the name of the feature structure file that was displayed in the current window when SAVE & RETURN was selected. The host program can then load this feature structure as the result of calling Fegramed.

X/Unix Fegramed writes this information to its standard output if the command line option "-poll" was enabled.

Mac Fegramed will determine the sender process as well as the communication file by looking at the *last* Apple Event used for communication. It will overwrite the contents of the file with the return commands and make the sender process the current process. Note that only the last Apple Event to open a communication file is relevant, others that were received between the last return action of the user and this event will be lost.

To receive Fegramed's response, the host application could remember the file change date and time of the communication file when switching to Fegramed and check if it changed when it handles a 'Resume' operating system event (which means it is the front process again). If the communication file changed, the host application should read it and take the appropriate actions.

4.3 Sort and Hide Files

These files contain the lists of features mentioned in the FEATURE ORDER respectively HIDDEN FEATURES dialogs. They are plain text files where each feature contained in the list appears on a separate line. You can also edit these files with a text editor and load them into Fegramed.

In the X/Unix version, these files are always named "~/.fedit_sort" and "~/.fedit_hide" unless other names are specified in the `Fegramed.sort` and `Fegramed.hide` resources, respectively (cf. appendix A, section A.2).

In the Mac version, the names of these files can be chosen by the user. When you select the SAVE button in the FEATURE ORDER or HIDDEN FEATURES dialog, a file selection dialog will appear that lets you specify the directory and the name of the file to save the current list into. These pathnames are among the options that are remembered using the SET DEFAULTS and SAVE DEFAULTS commands.

Appendix A

Motif Particulars

A.1 Command Line Options

–file or –fs	feature structure file to be loaded
–refFont	font used for coreference numbers
–textFont	font used for ordinary atoms and edges
–spec0Font	font used for special atoms of type 0
–spec1Font	font used for special atoms of type 1
–spec2Font	font used for special atoms of type 2
–spec3Font	font used for special atoms of type 3
–featureDepth	displayed depth
–sort	name of the sort file
–hide	name of the hide file
–poll	use stdin/stdout for communication

The –geometry option affects the Fegramed_Menu window. If you want to specify a geometry for the Fegramed work window, you have to use a command line option in the following style:

–xrm "Fegramed.geometry: 300x200+10+5"

A.2 Resources

Apart from the existing Xt-resources, you can specify fonts, the displayed feature depth and the names of the sort and hide files via application resources. The following example should suffice as documentation. Further information about the resource mechanism can be found in the X manual page or your local X Toolkit Manual. An explicit command line option overrides the resource specification.

```
! Geometry of the menu window: size should not be given:
Fegramed_Menu.geometry: +1+80

!The default geometry for the work windows can be given by:
Fegramed.geometry: 300x100-310+9

!fonts for the work windows (any available font may be used):
Fegramed.refFont:        8x13
Fegramed.textFont:       7x14
Fegramed.spec0Font:      10x20
Fegramed.spec1Font:      -adobe-helvetica-bold-r-*-*-10-*-*-*-*-*-*-*
Fegramed.spec2Font:      -adobe-helvetica-medium-o-*-*-10-*-*-*-*-*-*-*
Fegramed.spec3Font:      -*-courier-bold-r-*-*-16-*-*-*-*-*-*-*

!The displayed depth of the feature structures
Fegramed.featureDepth:  40

!The names of the sort and hide files
Fegramed.sort:          "$HOME/fegramed/sortfile"
Fegramed.hide:          "$HOME/fegramed/hidefile"
```

For the specification of resources for other widgets, a part of the widget hierarchy is given below. Be careful when setting these resources, as not every resource is set to a safe value when the widgets are created.

Menu bar of Fegramed work window:	`Fegramed*.top_box.*`
Fegramed work window:	`Fegramed.*.edit_window`
Vertical scrollbar:	`Fegramed.*.scroll1`
Horizontal scrollbar:	`Fegramed.*.scroll2`

Resources one might want to change could be the font or the foreground and background colour. These values will not affect the functionality of the Feature Editor. Here is an example that changes the colours of Fegramed. Simply try it to see the results.

```
! change the background colours of the menu window,
! the menu region and the scroll bars. The edit region has to have
! a white background.
Fegramed_Menu*background: green
Fegramed*background: green
Fegramed*borderColor: red
Fegramed*BottomShadowColor: pink
Fegramed*HighlightColor: pink
Fegramed*edit_window*background: white
```

A.3 Menus and Keyboard Shortcuts

Structures	
New	◇ N
Open	◇ O
Save	◇ S
Save As	
Close	◇ W
Close All	
Save & Return	
Return	
Quit	◇ Q

Edit	
Cut	◇ X
Copy	◇ C
Paste	◇ V
Remove	◇ R
Clear	◇ D
Build Tag	◇ T
Remove Tag	
Negate	◇ –
Set Text	◇ 0
Set Special0	◇ 1
Set Special1	◇ 2
Set Special2	◇ 3
Set Special3	◇ 4

Complex	
Conjunction	◇ [
Disjunction	◇ {
Implication	◇ >
List	◇ <
Function List	◇ +

Find	
Find	◇ F
Find Tag	◇ #
Find Again	◇ A

View	
Implode	◇ I
Hide	◇ H
Zoom in	
Zoom out	◇ Z
Show root	◇ Y
Hidden Features	
Feature Order	
Depth	
Reorder	
Reorder All	
Show Obscured	

Appendix B

Mac Particulars

B.1 File Types Used by Fegramed

Feature Structure Files have type `'TEXT'` and creator `'FEDI'` (optional). They contain feature structure descriptions in the format described in section 4.1.

Communication Files have type `'FBLK'` and creator `'FEDI'`. They contain remote commands to talk to Fegramed as well as Fegrameds responses. If Fegramed gets an 'Open Document' (`'odoc'`) Apple Event, the file type tells it to interpret the file contents as commands rather than as feature structure. The details are described in section 4.2.

Preferences Files have type `'PREF'` and creator `'FEDI'`. They contain the default settings for the following values:

- size and position of a new window
- all changeable fonts (References, Text, and Special0 ... Special3)
- the depth limit for feature structure display
- the scroll speed
- name and location of the default sort and hide files

Sort and Hide Files have type `'TEXT'` and creator `'FEDI'` (optional). They contain lists of feature names separated by newlines. They are read and written at startup time (provided they are specified as default files, see above) and when using the LOAD and SAVE buttons in the HIDDEN FEATURES and FEATURE ORDER dialog.

B.2 Menus and Keyboard Shortcuts

Structures	
New	⌘ N
Open	⌘ O
Save	⌘ S
Save As	
Close	⌘ W
Close All	
Save & Return	
Return	
Page Setup	
Print	⌘ P
Quit	⌘ Q

Edit	
Cut	⌘ X
Copy	⌘ C
Paste	⌘ V
Remove	⌘ R
Clear	⌘ D
Build Tag	⌘ T
Remove Tag	
Negate	⌘ –
Set Text	⌘ 0
Set Special0	⌘ 1
Set Special1	⌘ 2
Set Special2	⌘ 3
Set Special3	⌘ 4
Export to Scrap	⌘ E

Complex	
Conjunction	⌘ [
Disjunction	⌘ {
Implication	⌘ >
List	⌘ <
Function List	⌘ +

Find	
Find	⌘ F
Find Tag	⌘ #
Find Again	⌘ A

View	
Implode	⌘ I
Hide	⌘ H
Zoom in	
Zoom out	⌘ Z
Show root	⌘ Y
Hidden Features	
Feature Order	
Depth	
Scrolling	
Set Defaults	
Save Defaults	
Reorder	
Reorder All	
Show Obscured	

Fonts
References
Text
Special0
Special1
Special2
Special3

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
PF 2080
67608 Kaiserslautern
FRG

Telefon (0631) 205-3506
Telefax (0631) 205-3210

e-mail
dfkibib@dfki.uni-kl.de
WWW
http://www.dfki.uni-
sb.de/dfkibib

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder (so sie als per ftp erhaeltlich angemerkt sind) per anonymous ftp von ftp.dfki.uni-kl.de (131.246.241.100) im Verzeichnis pub/Publications bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or (if they are marked as obtainable by ftp) by anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) in the directory pub/Publications.
The reports are distributed free of charge except where otherwise noted.

DFKI Research Reports

1995

RR-95-11
Anne Kilger, Wolgang Finkler
Incremental Generation for Real-Time Applications
47 pages

RR-95-09
M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf
A Refined Architecture for Terminological Systems:
Terminology = Schema + Views
71 pages

RR-95-07
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt
The Complexity of Concept Languages
57 pages

RR-95-04
M. Buchheit, H.-J. Bürckert, B. Hollunder, A. Laux, W. Nutt,
M. Wójcik
Task Acquisition with a Description Logic Reasoner
17 pages

RR-95-03
Stephan Baumann, Michael Malburg, Hans-Guenther Hein, Rainer Hoch,
Thomas Kieninger, Norbert Kuhn
Document Analysis at DFKI
Part 2: Information Extraction
40 pages

RR-95-02
Majdi Ben Hadj Ali, Frank Fein, Frank Hoenes, Thorsten Jaeger,
Achim Weigel
Document Analysis at DFKI
Part 1: Image Analysis and Text Recognition
69 pages

1994

RR-94-39
Hans-Ulrich Krieger
Typed Feature Formalisms as a Common Basis for Linguistic Specification.
21 pages

RR-94-38
Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne,
Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger,
Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman.
DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling.
13 pages

RR-94-37
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 1: Overview.
54 pages

RR-94-36
Manfred Meyer
Issues in Concurrent Knowledge Engineering. Knowledge Base and Knowledge Share Evolution.
17 pages

RR-94-35
Rolf Backofen
A Complete Axiomatization of a Theory with Feature and Arity Constraints
49 pages

RR-94-34
Stephan Busemann, Stephan Oepen, Elizabeth A. Hinkelman,
Günter Neumann, Hans Uszkoreit
COSMA – Multi-Participant NL Interaction for Appointment Scheduling
80 pages

RR-94-33
Franz Baader, Armin Laux
Terminological Logics with Modal Operators
29 pages

RR-94-31
Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-23
Gert Smolka
The Definition of Kernel Oz
53 pages

RR-94-20
Christian Schulte, Gert Smolka, Jörg Würtz
Encapsulated Search and Constraint Programming in Oz
21 pages

RR-94-18
Rolf Backofen, Ralf Treinen
How to Win a Game with Features
18 pages

RR-94-17
Georg Struth
Philosophical Logics—A Survey and a Bibliography
58 pages

RR-94-16
Gert Smolka
A Foundation for Higher-order Concurrent Constraint Programming
26 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr
Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-13
Jana Koehler
Planning from Second Principles—A Logic-based Approach
49 pages

RR-94-12
Hubert Comon, Ralf Treinen
Ordering Constraints on Trees
34 pages

RR-94-11
Knut Hinkelmann
A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-10
Knut Hinkelmann, Helge Hintze
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-08
Otto Kühn, Björn Höfling
Conserving Corporate Knowledge for Crankshaft Design
17 pages

RR-94-07
Harold Boley
Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-06
Dietmar Dengler
An Adaptive Deductive Planning System
17 pages

RR-94-05
Franz Schmalhofer, J. Stuart Aitken, Lyle E. Bourne jr.
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-03
Gert Smolka
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-02
Elisabeth André, Thomas Rist
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-01
Elisabeth André, Thomas Rist
Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

1993

RR-93-48
Franz Baader, Martin Buchheit, Bernhard Hollunder
Cardinality Restrictions on Concepts
20 pages

RR-93-46
Philipp Hanschke
A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

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

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt
Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-43
M. Bauer, G. Paul
Logic-based Plan Recognition for Intelligent Help Systems
15 pages

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

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

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

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

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

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.)
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

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

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

RR-93-32
David R. Traum, Elizabeth A. Hinkelman
Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman
Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman
Corporate Agents
14 pages

RR-93-29
Armin Laux
Representing Belief in Multi-Agent Worlds via Terminological Logics
35 pages

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

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

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

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

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

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

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

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

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

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

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz
Object-Oriented Concurrent Constraint Programming in Oz
17 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-93-14
Joachim Niehren, Andreas Podelski, Ralf Treinen
Equational and Membership Constraints for Infinite Trees
33 pages

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

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

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

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

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

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

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

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

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

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

RR-93-03
Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi
An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 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-93-01
Bernhard Hollunder
An Alternative Proof Method for Possibilistic Logic and its Application to Terminological Logics
25 pages

DFKI Technical Memos

1995

TM-95-02
Michael Sintek
FLIP: Functional-plus-Logic Programming on an Integrated Platform
106 pages

TM-95-01
Martin Buchheit, Rüdiger Klein, Werner Nutt
Constructive Problem Solving: A Model Construction Approach towards Configuration
34 pages

1994

TM-94-04
Cornelia Fischer
PAntUDE – An Anti-Unification Algorithm for Expressing Refined Generalizations
22 pages

TM-94-03
Victoria Hall
Uncertainty-Valued Horn Clauses
31 pages

TM-94-02
Rainer Bleisinger, Berthold Kröll
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

TM-94-01
Rainer Bleisinger, Klaus-Peter Gores
Text Skimming as a Part in Paper Document Understanding
14 pages

1993

TM-93-05
Michael Sintek
Indexing PROLOG Procedures into DAGs by Heuristic Classification
64 pages

TM-93-04
Hans-Günther Hein
Propagation Techniques in WAM-based Architectures — The FIDO-III Approach
105 pages

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

TM-93-02
Pierre Sablayrolles, Achim Schupeta
Conflict Resolving Negotiation for COoperative Schedule Management Agents (COSMA)
21 pages

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

DFKI Documents

1995

D-95-12
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'95 Workshop:
KRDB-95 - Reasoning about Structured Objects:
Knowledge Representation Meets Databases
61 pages

D-95-09
Antonio Krüger
PROXIMA: Ein System zur Generierung graphischer Abstraktionen
120 Seiten

D-95-07
Ottmar Lutzy
Morphic - Plus
Ein morphologisches Analyseprogramm für die deutsche Flexionsmorphologie und Komposita-Analyse
74 pages

D-95-06
Markus Steffens, Ansgar Bernardi
Integriertes Produktmodell für Behälter aus Faserverbundwerkstoffen
48 Seiten

D-95-05
Georg Schneider
Eine Werkbank zur Erzeugung von 3D-Illustrationen
157 Seiten

D-95-03
Christoph Endres, Lars Klein, Markus Meyer
Implementierung und Erweiterung der Sprache $\mathcal{ALCP}$
110 Seiten

D-95-02
Andreas Butz
BETTY
Ein System zur Planung und Generierung informativer Animationssequenzen
95 Seiten

D-95-01
Susanne Biundo, Wolfgang Tank (Hrsg.)
Beiträge zum Workshop „Planen und Konfigurieren", Februar 1995
169 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

1994

D-94-15
Stephan Oepen
German Nominal Syntax in HPSG
— On Syntactic Categories and Syntagmatic Relations —
80 pages

D-94-14
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 2: User Guide.
72 pages

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.)
Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'94 Workshop: KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 pages

Note: This document is no longer available in printed form.

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.)
Working Notes of the 1994 International Workshop on Description Logics
118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-94-09
Technical Staff
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-08
Harald Feibel
IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-07
Claudia Wenzel, Rainer Hoch
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

D-94-06
Ulrich Buhrmann
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 Seiten

D-94-04
Franz Schmalhofer, Ludger van Elst
Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 Seiten

D-94-03
Franz Schmalhofer
Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung
54 Seiten

Note: This document is no longer available in printed form.

D-94-02
Markus Steffens
Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-01
Josua Boon (Ed.)
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

1993

D-93-27
Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman,
Hans Uszkoreit (Eds.)
Report of the EAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken
110 pages

D-93-26
Frank Peters
Unterstützung des Experten bei der Formalisierung von Textwissen INFOCOM - Eine interaktive Formalisierungskomponente
58 Seiten

D-93-25
Hans-Jürgen Bürckert, Werner Nutt (Eds.)
Modeling Epistemic Propositions
118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-93-24
Brigitte Krenn, Martin Volk
DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen
66 Seiten

D-93-22
Andreas Abecker
Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

Note: This document is no longer available in printed form.

D-93-21
Dennis Drollinger
Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-20
Bernhard Herbig
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt
Design & KI
74 Seiten

D-93-15
Robert Laux
Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-14
Manfred Meyer (Ed.)
Constraint Processing – Proceedings of the International Workshop at CSAM'93, St.Petersburg, July 20-21, 1993
264 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

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

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

Note: This document is no longer available in printed form.

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

D-93-09
Hans-Ulrich Krieger, Ulrich Schäfer
TDLExtraLight User's Guide
35 pages

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

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

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 for a nominal charge of 25 DM (or 15 US-$).

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-93-04
Technical Staff
DFKI Wissenschaftlich-Technischer Jahresbericht
1992
194 Seiten

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

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

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

FEGRAMED
An Interactive Graphics Editor for Feature Structures

Bernd Kiefer, Thomas Fettig

RR-95-06
Research Report