



Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-93-02

User Manual of COKAM+

Gabriele Schmidt, Frank Peters, Gernod Laufkötter

February 1993

**Deutsches Forschungszentrum für Künstliche Intelligenz
GmbH**

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

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

Deutsches Forschungszentrum für Künstliche Intelligenz

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

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

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

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

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

Friedrich J. Wendl
Director

User Manual of ,COKAM+

Gabriele Schmidt, Frank Peters, Gernod Laufkötter

DFKI-D-93-02

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8902 C4).

© Deutsches Forschungszentrum für Künstliche Intelligenz 1993

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.

User Manual of COKAM+

Version 2.0, December 1992

Version 1.4.1 written by

Gernod Laufkötter

updated by

Gabriele Schmidt and Frank Peters


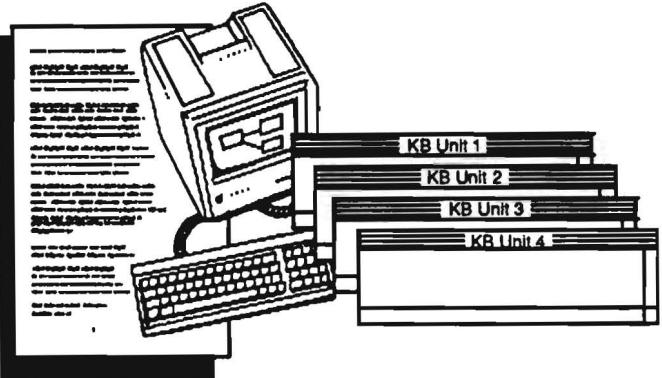
	<h1>COKAM+</h1>
<p>Deutsches Forschungsinstitut für Künstliche Intelligenz GmbH Erwin-Schrödinger-Straße P.O. Box 2080 D-6750 Kaiserslautern Tel.: (+49 631) 205-3211/13 Fax: (+49 631) 205-3210</p> <p>COKAM+ Implemented by</p> <p>Andreas Birk Holger Huf Otto Köhn Gernot Laufkötter</p> <p>Grapher V. 2.0 © G. Weloszek, University of Trier, May 1990 adapted to COKAM+ by O. Köhn, G. Laufkötter</p> <p>mail to: Gabriele Schmidt Tel.: (+49 631) 205-3462 e-mail: schmidt@dfki.uni-kl.de</p>	<p>Version 2.0, December 1992</p>  <p>Copyright Notice</p> <p>This software is distributed for non-profit and research purposes only. Non-profit redistribution of the current version or parts of the current version permitted if this copyright notice is included unchanged.</p> <p>There is no warranty of any kind for this prototype. It will be further improved as time permits.</p>

Table of Content

1. Installation.....	1
2. The Menu Bar of COKAM+	2
3. COKAM+	3
3.1. Marking Text Segments and Producing Knowledge Units	3
3.1.1. Source texts and text structures	3
3.1.2. Unit dialog window	3
3.1.3. The stack of units.....	6
3.2. Reset.....	6
3.3. Quit	6
4. Source Texts	7
4.1. Load a source text.....	7
4.2. Showing loaded source texts	7
4.3. Construction and application of the text structure.....	7
4.3.1. Construction of the text structure	7
4.3.2. Deleting nodes of the text structure.....	8
4.3.3. Navigating through the source text with the help of the text browser.....	8
4.4. Saving the text structure.....	8
5. Working with the Informal Knowledge Base	9
5.1. Loading the informal knowledge base.....	9
5.2. Looking for knowledge units and use of the unit stack	9
5.2.1. Search.....	10
5.2.2. Combining the selected search criteria	11
5.2.3. Building the unit stack	11
5.2.4. Quickly finding unit according to their title.....	11
5.3. Saving a knowledge base	11
6. The Model of Expertise and the Domain Model.....	12
6.1. Building a new model structure of the expertise / domain model	12
6.2. Saving a model structure of the expertise / domain model	12
6.3. Loading a model structure of the expertise / domain model.....	12
6.4. Associating the expertise / domain model with a knowledge base	13
6.5. Showing the expertise / domain model of a knowledge base.....	13
6.6. Associating keywords with the model of expertise	13

7. The Case Base	15
7.1. Initiating a new case base	15
7.2. Creating a new case in a case base.....	15
7.3. Showing a case.....	18
7.4. Deleting a case.....	18
8. Structuring the Knowledge Units.....	19
8.1. Relating to the model of expertise.....	19
8.2. Relating to cases	19
9. Templates.....	20
9.1. Construction of a template file.....	20
9.2. Loading a template file	22
10. Formalization of Knowledge Units (The Use of Templates).....	23
Index.....	i

1. Installation and Starting of COKAM+

For installing the program COKAM+ (and an example knowledge base) ca. 2 MB (+ 500 KB for the knowledge base) on the hard disk are necessary. For starting COKAM+ Allegro Common Lisp, Version 1.3.2 is required together with the Macintosh system software version 6.1.7. COKAM+ needs at least 2 MB RAM . A monitor of 19" allows to clearly arrange the multiple windows of an application of COKAM+.

Before working with COKAM+ the following requirements must be fulfilled:

- (i) The folder which contains the program of COKAM+ ("COKAM-x-y.image") *must* contain a folder called "COKAM".
- (ii) This folder "COKAM" must contain the following folders:
 - "data"
 - "knowledge-base"
 - "domain-models"
 - "expertise-models"

In the current version 2 ("COKAM-2.image") the user must only guarantee, that the folder "data" contains the file "Cokam.picture" and a file of the program RESEDIT or another program for editing resource files. This file is called "Cokam.resources" and contains graphical descriptions (pictures) of the cases which are applied in COKAM+. Furthermore, the text files from which knowledge will be acquired with COKAM+ must be ascii text files, e.g. WORD text files with the format "ASCII with line breaks", and must be transferred into the folder "data". These files must have the extension ".txt". The template files are stored in this folder, too (extension ".tpl").

All further folders or files which are stored in the folders mentioned above are constructed by the program itself. The user must **not** edit or change anyhow these files.

If the requirements mentioned above are fulfilled, COKAM+ can be started with a double click on the icon of the program.

2. The Menu Bar of COKAM+

After starting the program the menu bar of the Mac finder or of the present program will be substituted by the menu bar of COKAM+. It contains the following menus:



Figure 2.1: The menu bar of COKAM+

- "COKAM"
- "Sourcetexts"
- "Knowledge Base"
- "Domain Model"
- "Expertise Model"
- "Case Base"
- (• "*Graph*")

The menu "COKAM" (see chapter 3) offers the basic functions of the program. These functions allow to start the construction of an informal knowledge base by producing knowledge units from marked text segments. There are also functions to quit and to restart the program.

"Sourcetexts" summarizes all functions which allow to operate on the source texts and the structure of the texts (see chapter 4).

The menu "Knowledge Base" offers functions to load already constructed knowledge bases and to save new or modified knowledge bases (see chapter 5).

The menu "Domain Model" and "Expertise Model" permits to construct, to save and to load domain models and models of expertise (see chapter 6)

Finally the menu "Case Base" allows to operate with cases which are related to a certain knowledge base.

The menu "*Graph*" only appears if a window which presents its contents in a hierarchical manner is activated. The menu allows to show the hierarchy vertical or horizontal, to show different text fonts and sizes, to show different box sizes etc.

3. COKAM+

3.1. Marking Text Segments and Producing Knowledge Units

The construction of a new informal knowledge base begins by marking text segments in the source texts and by consequently producing knowledge units.

The menu function "Acquire Knowledge from Text .." in the menu bar "COKAM" initiate the knowledge acquisition from text. First a dialog window is shown by which a source text which is in the folder "data" can be selected. After selecting the text is shown on the screen. If the dialog window shows no text which can be selected then see chapter 4.5.

3.1.1. Source texts and text structures

The presented source text can be scrolled by clicking the mouse on the scroll bar on the right of the text. Corresponding to other Mac programs text segments can be marked with the mouse. Marked text segments are inverted. Above the source text window a second window called "text browser" is shown. In this window the hierarchical structure of the text is displayed as a tree. This structure allows to quickly navigate trough large source texts by clicking a node of the tree with the mouse. The construction of such structures for new source texts is explained in chapter 4.3.

3.1.2. Unit dialog window

On the right side of the window of the source text a dialog window is visible which allows the construction of knowledge units.

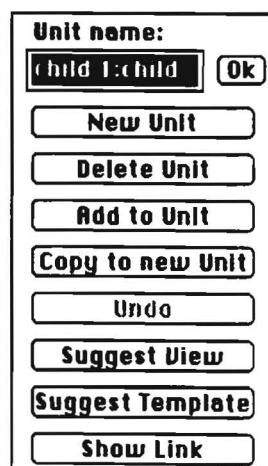


Figure 3.1: Unit dialog window

Until no unit is produced only the button "New Unit" is active for clicking with the mouse. If the button is clicked an empty knowledge unit which has the default title "Unit 1" is presented at the right side of the dialog window. This title can be changed in the editing field of the dialog window. Each new unit contains such a default title with a number at the beginning. The number is calculated by adding 1 to the number of existing units. The other buttons are related to the unit which is at the right side of the dialog window. This is the present unit. The following functions are related to it:

"Add to Unit"

The text which is already marked in the source text window will be copied to the present unit. For the user invisible a reference from the unit to the source text is stored so that later the text segment of the source text can be found (see button "Links"). The text segment which was marked in the source text before is now presented in bold letters. Thus, in future applications of the knowledge base it becomes obvious that this text segment was already used for the construction of a knowledge unit.

"Copy to Unit"

This function allows to decompose a unit into a subunit. A new unit will be produced. The text which is marked in the present unit will be copied into this new unit. If no text is selected, there will be an error message. Similar to the function "Add to Unit" the new unit has a reference to its source which can be found by clicking the button "Links" in the window of the function "Source".

"Delete Unit"

The present unit will be deleted from the knowledge base except that the unit is used in explanation structures. Then, a selection of all explanation structures in which the unit is used will be shown. The explanation can immediately be shown with the help of the selection window (see also "Links"). If a unit is deleted its children inherit its references to sources.

Notice!

In version 2 by using "Delete Unit" *each* present unit is deleted, i.e., a unit which is used in explanation structures is also deleted. If some units refer to the deleted unit in the explanation structures these units are deleted, too. There is no warning before!

"Undo"

This button is only efficient after choosing the action "Add to Unit". Then "Add to Unit" is completely undone.

"Suggest View"

By clicking the button "Suggest View" the window with the model of expertise will be shown if it is not already there. In the window of the model of expertise the views which were already selected are shown as inverted boxes. Additionally a dialog window which contains all views of the model of expertise appears. In this window the views which are suggested by the system are inverted. You can agree with the suggested views and click the button "ok" and the new views are shown in the model of expertise. Or you can correct the suggestion. Then, you can click on inverted views if you want to unselect them or on the other views if you want to add them. By clicking you can use the conventional Macintosh key combinations like "shift click" for marking several views following each other or like "apple click" for marking several views¹.

"Suggest Template"

If a knowledge unit has references to the model of expertise templates can be suggested (see chapter 9). The templates are shown in a dialog windows which allows to mark one template by clicking and to copy it to a child unit of the present unit with the button "Copy to Unit" (see above).

"Show Links"

By clicking on the button "Show Links" a pull down menu is opened, which shows the possible types of links which can be shown for the present knowledge unit:

"Source"

This link refers to the original text from which the text of the present knowledge unit was copied. If the text of the present unit refers to the source text which is already shown then the source text is scrolled so that at the top of the text the copied text segment is shown in bold letters. If the text of the present unit refers to another source text a dialog window allows you to show the referred source text or to keep the present source text.

If the source of the present unit is another unit, this one becomes the present unit of the stack.

¹Notice!

If you have defined your own key combinations for some functions in the control field of the system software (for example with "QuicKeys") then this functions can overrule the key functions defined in COKAM+. You should select some other key combinations in the system software.

"Children"

If a child unit has been produced from the present unit, this child unit becomes the present unit of the stack. .

"Expertise"

If there is a link from the present unit to the model of expertise, the model is shown and the corresponding categories of the model of expertise are inverted.

"Domain"

Similar to *"Expertise"* for the domain model.

For generating links to the model of expertise or the domain model see chapter 6.

"Cancel"

The pull down menu is closed without carrying out any functions.

3.1.3. The stack of units

All units which are visible on the screen (or become visible by scrolling the stack) are called the unit stack. Within the unit stack a new unit is located on the right side of the unit dialog, i.e. the new unit is the present unit. Clicking on any unit causes a scrolling of the unit stack, so that the clicked unit is located on the right side of the unit dialog. Thus, the new unit becomes the present unit and all functions within the dialog window relate to it. The unit stack need not contain all units of an informal knowledge base. For loading an unit stack according to search criteria see chapter 5.3.

3.2. Reset

The menu "COKAM" contains the function "Reset". By selecting this function the program is set into the state it had after starting the program. All open windows and dialogues are closed and the knowledge base in the working space is deleted. If the knowledge base has unsaved modifications, e.g. new units, the system knows this. Before executing the reset it asks you whether to save the changes or to cancel the rest or to go on. The system can ask about unsaved changes in the knowledge base, the model of expertise and the domain model.

3.3. Quit

The selection of the function "Quit" in the "COKAM" menu finishes the program. If there are unsaved changes they can be saved or not or the function "Quit" can be canceled (similar to "Reset").

4. Source Texts

4.1. Load a source text

A new source text for producing or enlarging a knowledge base can be selected with the function "Load Sourcetext .." of the menu "Sourcetext". The source text is shown on the left side of the screen and a reference to the text is stored in the corresponding knowledge base. Therefore, a source text must be selected from the menu for a knowledge base only once. After that (even after starting the program again) the program finds the text in connection with the knowledge bases automatically. A prerequisite is that the knowledge base is stored within the folder "COKAM". This folder must contain the folder "data" which contains the source texts.

4.2. Showing loaded source texts

Loaded source text, i.e. source texts which already refer to the knowledge base, can be shown by clicking the function "Show Sourcetext .." of the menu "Sourcetext". For showing the source text and its text structure see 4.3.

4.3. Construction and application of the text structures

4.3.1. Construction of the text structure

If the source text is shown it is possible to construct a hierarchical structure of the text in the window above the source text which is called "Text Browser". If such a structure already exists for a source text the system always loads it together with the source text. The construction of such a structure can be done in the following manner:

The source text must be scrolled to a text segment which is to be used in the text structure, e.g. a heading of a chapter. This text segment must be marked with the mouse. Then, you can activate the text browser window by clicking on it. After that you must click on a node of the structure by pressing the apple key at the same time ("option-click"). The marked text segment becomes a new node under the node on which you have clicked. A node can stand for the level of the structure of the text.

It is also possible to add names for text segments in the structure which are not mentioned in the text. Then you must put the cursor on the desired part of the text without marking this part. Then you can activate the text browser window and again click on a node of the structure by pressing the apple key at the same time ("option-click"). Now the program asks you for a name of the new node.

4.3.2. Deleting nodes of the text structure

If you click on a particular node and press the keys "alt"- and "apple" at the same time (command-option-click) this node and all nodes which depend on this node are deleted from the structure..

4.3.3. Navigating through the source text with the help of the text browser

If the text browser window is activated you just have to click on a node of the structure and the source text is scrolled to the part of the text which is associated with the clicked node.

4.4. Saving the text structure

In the menu "Sourcetexts" there is a function "Save Textstructure". A new or changed text structure can be stored by selecting this function. The structure is then associated with the corresponding source text. The structure need not explicitly be loaded together with the source text.

5. Working with the Informal Knowledge Base

5.1. Loading the informal knowledge base

For working with an existing knowledge base the knowledge base must be loaded by selection the function "Load KB .." in the menu "Knowledge Base". The case base (see chapter 7) and models (see chapter 6) which are associated with the knowledge base and the references to the source texts are automatically loaded together with the knowledge base. When the loading function is finished the knowledge base is loaded in the working space, i.e. the unit stack is still empty after that. But now the window "search unit" (see below) and the "unit dialog" window (see above) are shown. Associated source text can be loaded by clicking on the function "Show Sourcetext .." in the menu "Sourcetexts". A unit stack can be selected and shown with the help of the "Search Units" window.

5.2. Looking for knowledge units and use of the unit stack

The function "Search Units .." or the loading of the informal knowledge base activates a dialog window which allows to ask queries about the knowledge base.

Search Units

Expertise Category:
 with subcategories

Domain Category:
 with subcategories

Substring:
 with parents

Title:
 combine criteria ..
 conjunctive disjunctive

add to unit stack
 replace unit stack

Search

Befestigungssystem v. Wendeschneidplatt ...
Befestigungssystem v. Wendeschneidplatt ...
Befestigungssystem v. Wendeschneidplatt ...
Eckenradius
Eckenwinkel
child 1:Einstellwinkel b. Schruppen
Ermittlung von Werkzeugen
Flächen
Freiwinkel 1
Freiwinkel 2
Freiwinkel 3

Figure 5.1: Dialog for searching units

The result of such a query is a set of units which build the loaded unit stack or which replace or enlarge the present unit stack. If there is no answer to a query which should answer particular criteria, the set of results is empty. The program answers every query and reports about the success or failure of the query. For asking queries the following functions can be used:

5.2.1. Search criteria

Searching according to a category of the model of expertise

In an editor field of the dialog box you can type in a category of the model of expertise. Only a substring which begins with the first letters of the category is necessary. The substring will be completed during the search process if the corresponding category exists in the model of expertise (also see chapter 6.)

The option "with Subcategories" can be selected or unselected by clicking with the mouse on the box on the right side of the editor field. This option determines whether only units of the category in the editor field are selected or whether also units of all subcategories of this category are selected (see chapter. 6).

Searching according to a category of the domain model

Another editor field allows to type in categories of the domain model. This field can be used similar to the editor field of the model of expertise.

Searching according to a substring in a unit

In the third field a substring can be typed in. According to this criteria all units are found which contain the substring somewhere in their text field.

Searching according to a title of a unit or a substring of a title

The fourth and last editor field allow to look for units with a particular title or substring in the title.

Searching units with or without older generations

The option "with parents" allows to look for all units without distinguishing whether the units are parent or child units. By default only the units of the youngest generation are searched.

The search process according to the criteria is activated by clicking the button "Search". If one or more criteria are not selected, i.e. their editor fields are empty, only that units are found which fit into the remaining criteria.

If the search button is pressed without selecting any criteria all unit of the knowledge base are found and shown in the unit stack.

5.2.2. Combining the selected search criteria

It is possible to combine the search criteria either conjunctively or disjunctively. By default the search is conjunctive, i.e. all criteria must be fulfilled at the same time so that a unit fits into the requirements. This can be changed by clicking on one of the two buttons under "combine search criteria ..".

5.2.3. Building the unit stack from the selected units

You can choose between adding the found units to an already existing unit stack or replacing an existing unit stack. This can be determined by selecting on of the two options "Add to Unit Stack" or "Replace Unit Stack".

5.2.4. Quickly finding units according to their title

On the bottom of the search dialog window the titles of the units which are shown in the present unit stack are listed in their alphabetical order. This order is only broken for child units. Child units can be found directly after their parent unit. The list of the titles can be scrolled with the help of the scroll bar on its right side. By clicking on the certain title of one of the units the unit stack is scrolled so that the unit with this title becomes the present unit, i.e. the unit is on the right side of the dialog window.

5.3. Saving a knowledge base

A knowledge base can be saved at any time so that provisional results can also be saved. By using the function "Save KB" a knowledge base can be saved with its previous name if the knowledge base was changed before. It is always possible to save a knowledge base with a new name by using the function "Save as ..". If you then type in the name of an existing knowledge base, a dialogue gives you the warning that you can overwrite the existing knowledge base if you want.

6. The Model of Expertise and the Domain Model

Exactly *one* model of expertise and *one* domain model is associated with *one* informal knowledge base. In the following the possible functions for working with the model of expertise are described. With the exception of using templates and keywords the same functions are offered to work with the domain model. Therefore the two models are explained together.

6.1. Building a new model structure of the expertise / domain model

In principle a model structure can be build like the text structure of a source text (see chapter 4.3). After selecting the function "New" in the menu "Expertise Model" (or "Domain Model") the system asks for a new name of the model and then opens a window. In this window the root of the structure has the name which you typed in as the new model name. (At the moment only hierarchical structures can be shown which could be too restrictive for showing a model of expertise.)

Adding a new category of the model of expertise (of the domain model)

An option-click (apple key + click) on a node or the root of the structure causes the program to asks for the name of the new category. The category is represents as a new node which contains the new name and is linked to the clicked node in the hierarchy.

Deleting a category of the model of expertise (of the domain model)

With command-option-click ("alt" + "apple" + click) a node of the structure and all nodes which depend on this node can be deleted from the model of expertise (from the domain model). The root can not be deleted.

6.2. Saving a model structure of the expertise / domain model

A model can be saved by selecting the functions "Save" or "Save as .." in the menu of the corresponding model. With "Save" changes of the model can be saved and with "Save as .." the model can be saved under a new name. If the program is restarted or you quit the program the systems reminds you if there are unsaved changes of the model and offers you the possibility to save it now.

6.3. Loading a model structure of the expertise / domain model

The menu function "Load .." in the menu of the corresponding model allows to load an already built structure. This can be necessary if the model was not associated with the knowledge base up to now (also see chapter 6.4). If there is already a model of expertise

(or a domain model) and its changes are not saved you can save it before loading the new model.

6.4. Associating the expertise / domain model with a knowledge base

After loading or building a model it is associated with the knowledge base which is loaded in the working space. *One* knowledge base can only be associated with *one* model of expertise and *one* domain model. But one model can be associated with several different knowledge bases. If a knowledge base is loaded and another model will be loaded or built, the association of the knowledge base to the former model is deleted. Before the link to the former model is deleted changes of this model can be saved, so that the model is not lost. The current association of a model with a knowledge base causes that the model is automatically loaded by loading the knowledge base! Thus if a model itself is loaded explicitly this causes a new association between this model and the knowledge base.

6.5. Showing the expertise / domain model of a knowledge base

The model of expertise (or the domain model) which is associated with a loaded knowledge base can be shown by selecting the function "Show" in the menu of the corresponding model.

6.6. Associating keywords with the model of expertise

Keywords can be associated with each view of the model of expertise. By clicking on a particular view and pressing the shift key at the same time ("shift-click") a dialog window appears which shows all keywords which are associated with this view in alphabetical order.

For managing the keywords the dialog window offers the following functions:

Adding a keyword

By clicking on the button "Add" a new window appears in which the name of a new keyword can be typed. After finishing the input the keyword is added into the list of the associated words.

Adding a marked word

If you have already marked a word in the text of the present knowledge unit with the help of the mouse then you can click on the button "Add marked" and the marked word will be added into the list of the associated words.

Deleting a keyword

In order to delete a keyword from the list you must mark the corresponding word and then click on the button "Delete".

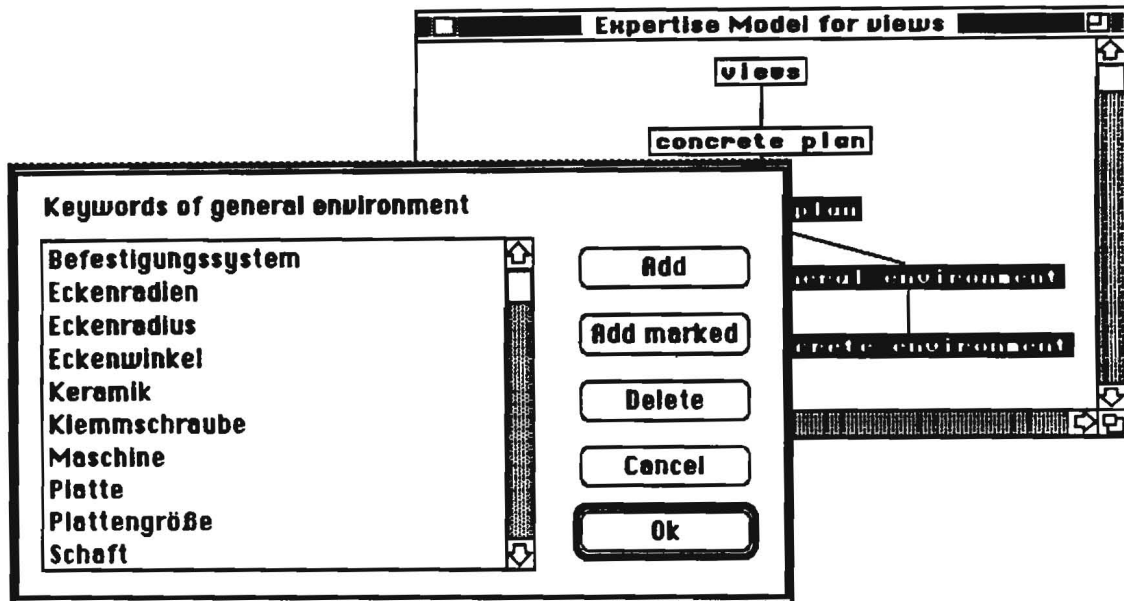


Figure 6.1: Dialog window which shows the keywords of the view "general environment" and the different functions

The changes of the keyword list can be undone by using the function "Cancel" or can be saved by using the function "Ok".

COKAM+ uses the keywords to suggest particular views of the model of expertise for a particular knowledge unit (see 3.1.2 "Unit dialog window"). For this reason the keywords should closely related to the corresponding view of the model of expertise. So, if COKAM+ finds a keyword in the text of the knowledge unit, it will suggest the corresponding view.

7. The Case Base

A case base can be associated with an informal knowledge base. For such a case base several cases can be defined. A case should consist of a problem description and the solution of the problem. It should be domain-specific and prototypical in the domain. It is assumed that such cases are very often represented graphically. In order to present the cases in their usual form they are scanned in, stored in a resource file of the folder "data" (see chapter 1). From this resource file the pictures of cases, e.g. a picture of the problem description and a picture of the solution can be loaded and used to construct a case. Several of such cases are collected in one case base. There are the following functions in the menu "Case Base":

7.1. Initiating a new case base

With the help of the function "New CB.." a new case base can be constructed. After selecting this function you are asked for the name of the new case base in a dialog window. After finishing this input a new case base with the new name is constructed and associated with the knowledge base. Therefore a knowledge base must be loaded in order to use this function correctly. If a case base which is related to the knowledge base already exists a warning informs you that this case base is deleted by constructing the new case base. Then, you can decide whether to go on or to cancel the process.

7.2. Creating a new case in a case base

For creating a new case the function "Create Case" in the menu "Case Base" must be selected. A dialog window shows the names of the pictures of the problem descriptions or a case² which are stored in the "Cokam.resources" file. If you select one of the names in the dialog window a new window will be opened in which the corresponding picture will be shown. Such a newly selected problem description will be automatically saved in the present case base. The complete case can now be constructed by adding the case solution (case solution steps) to the problem description³:

²Often we sloppily call a problem description a case!

³In our domain a case consists of a problem description and several solution steps. Both are represented graphically. For the construction of a complete case we first select the problem description, which can be selected as already described. This problem description shows the problem and the desired goal and refers to solution steps. These solution steps (we call them operators) can now be added as pictures in their correct sequence. In other words we show the complete case by structuring several pictures in a flat hierarchy (also see figure 7.1.)

Adding a solution step (an operator) to a problem description

The option-click on the picture of the problem description opens the dialog window again. Now a solution step (operator) can be marked by clicking on its name. If you press the "OK" button the operator is selected, graphically connected to the problem description by drawing a line, and stored together with the complete case. If you press the "Cancel" button the window disappears and nothing else happens.

Deleting a solution step (an operator)

Deleting an operator can be done by command-option-clicking on the picture of the operator. The selected solution step is deleted in the structure of the case.

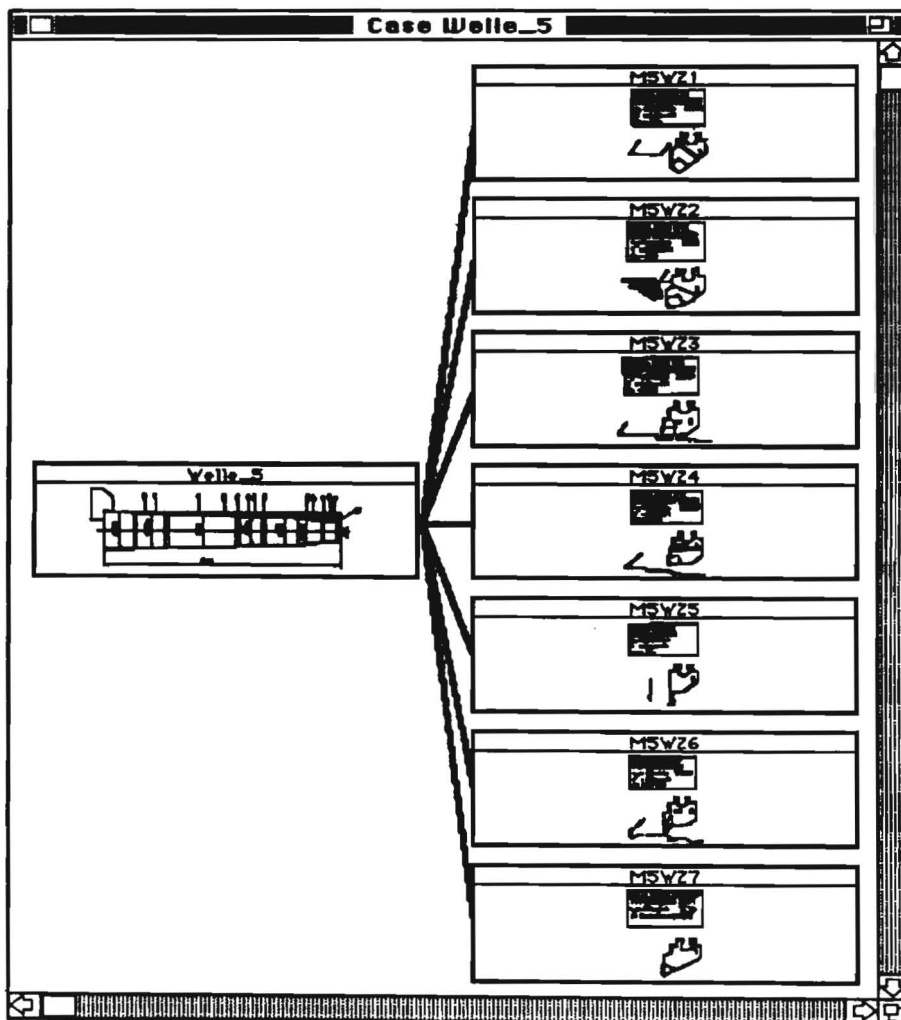


Figure 7.1 One case of a case base in our domain.
This case consists of the problem description (called "Welle_5") and seven solution steps.

The knowledge units can be associated with the solution steps (operators) of a case. Corresponding to the solution steps the knowledge units are conditions. They can be distinguished in *preconditions*, *consequences* and *abstraction and refinement rules*. After

clicking on a solution step (or a problem description) a dialog window offers the possibilities to show the picture of the solution step (or the problem description) or to show the structure of the conditions of the solution step.

The picture of a solution step (or a problem description)

If you click on the button "Picture" the picture of the corresponding solution step (or problem description) appears in the upper left corner of the screen. The size of the window is variable.

Preconditions, consequences and abstraction and refinement rules of solution steps

If you select the function "Conditions" in the dialog window, then two windows appear on the screen. The window at the top shows a solution step, its preconditions (knowledge units which directly refer to the solution step) and abstraction and refinement rules of the preconditions (knowledge units which refer to other knowledge units). The other window shows the same solution step, its consequences (knowledge units to which the solution step directly refers) and abstraction and refinement rules of the consequences (knowledge units which refer to other knowledge units).

By option-clicking on the solution step or on a card of a knowledge unit the present knowledge unit of the unit stack is linked to the card on which you have clicked. Depending on the window in which you have clicked on the solution step the linked knowledge unit is interpreted as a precondition or as a consequence.

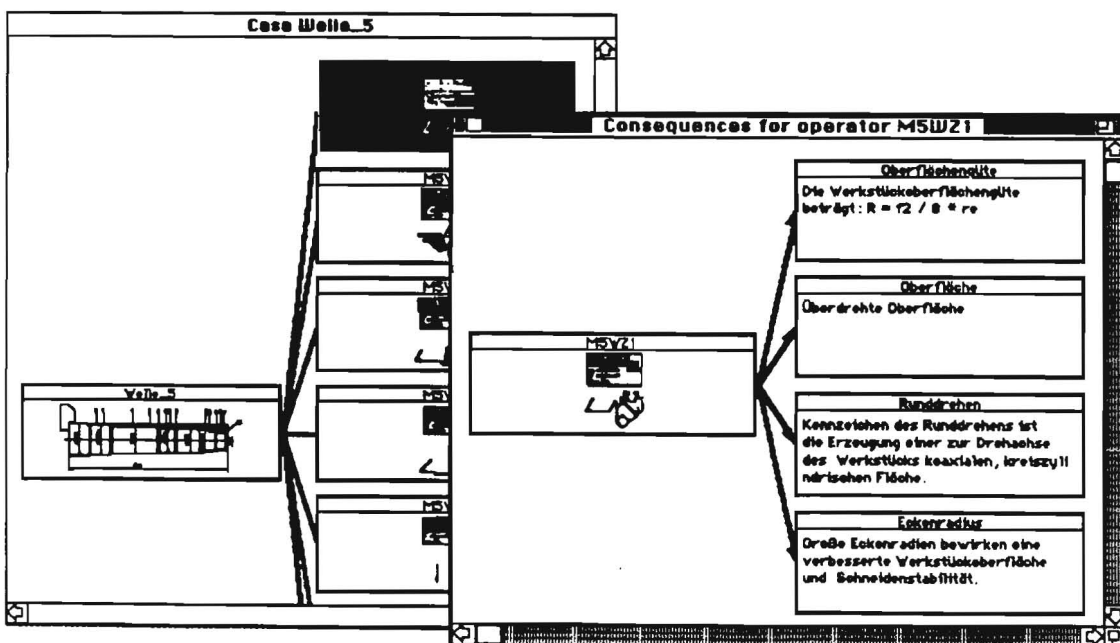


Figure 7.2.: The case, one of its operators and consequences of the operator

7.3. Showing a case

After clicking on the button "Select old Case" a dialog window shows you the names of all cases which are stored in the present case base. After selecting a case by clicking on its name the case is presented in a window. Now, the case can be edited like described in chapter 7.2.

7.4. Deleting a case

In order to delete a case you must select the function "Delete Case" in the menu "Case Base". Again, a dialog window with the names of all cases of the present case base appears. You can either cancel the function or mark the to be deleted case with the mouse. This case will be eliminated from the list of cases of the present case base.

8. Structuring the Knowledge Units

COKAM+ offers two different possibilities to structure knowledge units. The two types of structuring are briefly summarized in the chapter.

8.1. Relating to the model of expertise

Knowledge units can be associated with the views of the model of expertise. If a knowledge unit is assumed to be relevant for the problem solving process it must be associated with at least one view. On the one side the association can be done by clicking on the corresponding view. On the other side the function "Suggest View" of the unit dialog (see chapter 3.1.2) can help by finding the correct relations. Based on the views with which a knowledge unit is associated templates can be suggested (see chapter 9).

8.2. Relating to cases

To each solution step of a case a knowledge unit can be linked (see chapter 7.2.). Depending on the type of the link the knowledge unit is a precondition, a consequence or an abstraction and refinement rule. The type of such a link can influence the kind of templates which are suggested (see chapter 9).

9. Templates

In order to support the formalization of knowledge units COKAM+ offers the possibility of managing templates. To a given knowledge unit the program suggests a set of templates. From this set you have to select that template of which you assume that it fits the knowledge unit best. That template can be added to the knowledge unit by the function "Add to unit". Or it can be copied to a new child unit of the corresponding knowledge unit with the function "Copy to new unit". The set of possible templates depends on the selected views of the model of expertise and can be influenced by kind of the structuring of the the knowledge units according to the cases.

9.1. Construction of a template file

A template file can be constructed with each text editor which can save the file in ascii format. The template file has to be stored in the folder "data" and must have the extension "tpl". In the file there are two lists: .

Head of definition

In the first list which is called head of definition the individual views of the model of expertise are referred to symbols. Each of these relations is written in parentheses so that the syntax looks like this:

```
( ("View A of Expertise"      Expertise_Symbol_A)
.....
("View Z of Expertise"      Expertise_Symbol_Z) )
```

Body of definition

In the second list which is called body of definition a combination of the condition symbols and the expertise symbols which are defined for the views of the model in the first list is associated with a specific template. The syntax of the second list looks like:

```
(Condition_Symbol      (Selection List of Expertise_Symbols)
                          (Form of Templates))
```

Condition symbols can be :

precondition, consequence, ar-rule (for abstraction and refinement rule; also see chapter 7.2) and Nil (empty condition symbol).

Contrary to the condition symbols the expertise symbols can be defined according to the view of the model of expertise (see first list). In the selection list they are now combined to a logical term. All atomic expertise symbols in the list are seen as a conjunction. If there is a list of some expertise symbols within the list the symbols of this embedded list are seen as disjunction.

Example:

```

("concrete plan"      cplan)           Head of definition
("general plan"      splan)
("general product"   aprod)
("concrete product"  cprod)
("general environment" aenv)
("concrete environment" cenv))

((consequence (aprod (splan cplan) (aenv cenv))
  ("operator" (replace operator 1))
  ("product" (replace aprod *))           Template 1
  ("plan" (replace (splan cplan) *))
  ("environment" (replace (aenv cenv) *))
  "-->"
  ("product" (replace (aprod cprod))) )

(consequence ((splan cplan) (aenv cenv))
  ("operator" (replace operator 1))
  ("plan" (replace (splan cplan) *))     Template 2
  ("environment" (replace (aenv cenv) *))
  "-->"
  ("environment" (replace (aenv cenv) 1))
  ...
  ...
  ...

(precondition ((aprod cprod) (splan cplan))
  ("operator" (replace operator 1))
  ("product" (replace (aprod cprod)      Template n
  "-->"
  ("plan" (replace (splan cplan) 1))))
)

```


Now, assume the program must suggest a set of templates for a knowledge unit. This knowledge unit is related to a solution step as a consequence (condition symbol: "consequence"). Further it is related to the views "general plan" (splan), "general product" (aproduct) and "concrete environment" (cenv). Then, corresponding to the example only template 1 fits since it requires the condition symbol "consequence" and the expertise symbol "aproduct" *and* ("splan *or* "cplan") *and* ("aenv *or* cenv").

Template 2 would not be suggested since the symbol "aproduct" is not in its selection list of expertise symbols. Template n does not fit at all because it requires the condition symbol "precondition".

9.2. Loading a template file

In order to load a template file, the function "Load Template" in the menu "Expertise Model" must be selected. A dialog window shows the names of all template files which have the extension ".tpl" and are stored in the folder "data". You can click on one of these names and the corresponding template file is loaded in the working space. By clicking on the function "Suggest Template" in the unit dialog window it is checked whether the relations of the present knowledge unit fit with the selection lists of these templates (see chapter 10).

10. Formalization of Knowledge Units (The Use of Templates)

With the help of the function "Suggest Template" of the unit dialog window the program suggests a set of templates. This suggestion depends on the relation of the present knowledge unit to the views of the model of expertise and its relations to the case solutions. If there is no case solution activated the template is selected based on the views. The suggested templates are shown in a dialog window.

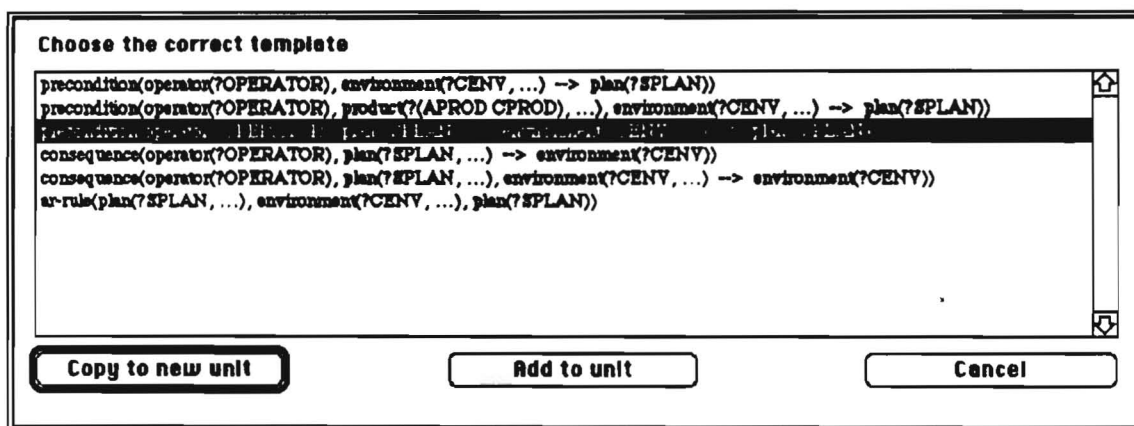


Figure 10.1.: Dialog window for selecting templates

After marking the template which fits the present knowledge unit best this template can be copied to the knowledge unit by the function "Add to unit". Or it can be copied to a new child unit of the corresponding knowledge unit with the function "Copy to new unit". After that you can edit the template and fill in the concrete data which are mentioned in an informal form in the parent knowledge unit.

Index

- abstraction and refinement rule**
 6; 17; 19; 20
 "Acquire Knowledge from Text .." 3
 "Add to Unit Stack" 11
 "Add to Unit" 4
Allegro Common Lisp 1
body of definition 20
 "Cancel" 6
 "Case Base" 2; 15
case base 15
case solution 15
case 15; 18; 19
category 10
child unit 11
 "Children" 6
 "COKAM-2.image" 1
 "Cokam.resources" 1; 15
 "COKAM" 2; 3
condition symbol 20
condition 16; 17
consequence 16; 17; 19; 20
 "Copy to Unit" 4
current version 1
 "Delete Unit" 4
 "Domain Model" 2
domain model 12
 "Domain" 6
 "Expertise Model" 2
expertise symbol 20
 "Expertise" 6
folder 1
formalization 20
 "Graph" 2
graphical description 1
hard disk 1
head of definition 20
informal knowledge base ... 3; 12
Installation 1
keyword 13
 "Knowledge Base" 2
knowledge base 1; 11
knowledge unit 3; 19; 20
 "Load KB .." 9
menu bar 2
model of expertise 5; 12; 19
monitor 1
 "New Unit" 4
older generations 10
operator 16
picture 1; 15; 17
precondition 16; 17; 19; 20
present unit 4; 11
problem description 15
 "Quit" 6
RAM 1
 "Replace Unit Stack" 11

"Reset"	6
"Save Textstructure"	8
"Search Units .."	9
Search	10
"Show Links"	5
solution step	15
source text	3; 7
"Source"	5
"Sourcetexts"	2; 8
Starting	1
structuring	19
substring	10
"Suggest Template"	5; 22; 23
"Suggest View"	5; 19
syntax	20
template file	20; 22
template	5; 19; 20
"text browser"	3
text file;	1
text segment	3
text structure	3; 7
title	4; 10
"Undo"	4
Unit dialog window	3; 22
unit stack	6; 11
view	5; 19
"with parents"	10



Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.

Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far can be ordered from the above address.

The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-92-11

Susane Biundo, Dietmar Dengler, Jana Koehler:
Deductive Planning and Plan Reuse in a
Command Language Environment
13 pages

RR-92-13

Markus A. Thies, Frank Berger:
Planbasierte graphische Hilfe in
objektorientierten Benutzungsoberflächen
13 Seiten

RR-92-14

Intelligent User Support in Graphical User
Interfaces:

1. InCome: A System to Navigate through
Interactions and Plans
Thomas Fehrle, Markus A. Thies
2. Plan-Based Graphical Help in Object-
Oriented User Interfaces
Markus A. Thies, Frank Berger

22 pages

RR-92-15

Winfried Graf: Constraint-Based Graphical
Layout of Multimodal Presentations
23 pages

RR-92-16

*Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel,
Hans-Jürgen Profitlich:* An Empirical Analysis of
Terminological Representation Systems
38 pages

RR-92-17

Hassan Ait-Kaci, Andreas Podelski, Gert Smolka:
A Feature-based Constraint System for Logic
Programming with Entailment
23 pages

RR-92-18

John Nerbonne: Constraint-Based Semantics
21 pages

RR-92-19

*Ralf Legleitner, Ansgar Bernardi, Christoph
Klauck:* PIM: Planning In Manufacturing using
Skeletal Plans and Features
17 pages

RR-92-20

John Nerbonne: Representing Grammar, Meaning
and Knowledge
18 pages

RR-92-21

Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The
Geometrical Approach
25 pages

RR-92-22

Jörg Würtz: Unifying Cycles
24 pages

RR-92-23

Gert Smolka, Ralf Treinen:
Records for Logic Programming
38 pages

RR-92-24

Gabriele Schmidt: Knowledge Acquisition from
Text in a Complex Domain
20 pages

RR-92-25

*Franz Schmalhofer, Ralf Bergmann, Otto Kühn,
Gabriele Schmidt:* Using integrated knowledge
acquisition to prepare sophisticated expert plans
for their re-use in novel situations
12 pages

RR-92-26

Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaischian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27

Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29

Zhaohui Wu, Ansgar Bernardi, Christoph Klauck: Skeletal Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

RR-92-30

Rolf Backofen, Gert Smolka
A Complete and Recursive Feature Theory
32 pages

RR-92-31

Wolfgang Wahlster
Automatic Design of Multimodal Presentations
17 pages

RR-92-33

Franz Baader: Unification Theory
22 pages

RR-92-34

Philipp Hanschke: Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35

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

RR-92-36

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

RR-92-37

Philipp Hanschke: Specifying Role Interaction in Concept Languages
26 pages

RR-92-38

Philipp Hanschke, Manfred Meyer:
An Alternative to Θ -Subsumption Based on Terminological Reasoning
9 pages

RR-92-40

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

RR-92-41

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

RR-92-42

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

RR-92-43

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

RR-92-44

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

RR-92-45

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

RR-92-46

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

RR-92-47

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

RR-92-48

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

RR-92-49

Christoph Klauck, Ralf Legleitner, Ansgar Bernardi: Heuristic Classification for Automated CAPP
15 pages

RR-92-50

Stephan Busemann:
Generierung natürlicher Sprache
61 Seiten

RR-92-51

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

RR-92-52

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

RR-92-54

Harold Boley: A Direkt Semantic Characterization of RELFUN
30 pages

RR-92-55

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

RR-92-56

Armin Laux: Integrating a Modal Logic of Knowledge into Terminological Logics
34 pages

RR-92-58

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

RR-92-59

Karl Schlechta and David Makinson: On Principles and Problems of Defeasible Inheritance
14 pages

RR-92-60

Karl Schlechta: Defaults, Preorder Semantics and Circumscription
18 pages

RR-93-02

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

RR-93-03

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

RR-93-05

Franz Baader, Klaus Schulz: Combination Techniques and Decision Problems for Disunification
29 pages

RR-93-08

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

DFKI Technical Memos**TM-91-12**

Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13

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

TM-91-14

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

TM-91-15

Stefan Busemann: Prototypical Concept Formation An Alternative Approach to Knowledge Representation
28 pages

TM-92-01

Lijuan Zhang: Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02

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

TM-92-03

Mona Singh: A Cognitive Analysis of Event Structure
21 pages

TM-92-04

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

TM-92-05

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

TM-92-06

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

TM-92-08

Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

DFKI Documents**D-92-07**

Susanne Biundo, Franz Schmalhofer (Eds.):
Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08

Jochen Heinsohn, Bernhard Hollunder (Eds.):
DFKI Workshop on Taxonomic Reasoning
Proceedings
56 pages

D-92-09

Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10

Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-11

Kerstin Becker: Möglichkeiten der Wissensmodellierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12

Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13

Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-14

Johannes Schwagereit: Integration von Graph-Grammatiken und Taxonomien zur Repräsentation von Features in CIM
98 Seiten

D-92-15

DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16

Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17

Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
254 pages

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

D-92-18

Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19

Stefan Dütrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21

Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-23

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

D-92-24

Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25

Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26

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

D-92-27

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

D-92-28

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

D-93-02

Gabriele Schmidt, Frank Peters, Gernod Laufkötter: User Manual of COKAM+
23 pages

User Manual of COKAM+

Gabriele Schmidt, Frank Peters, Gernod Laufkötter

D-93-02
Document