

**Installing and Using the Hybrid Knowledge Representation and  
Inference Environment  
BABYLON  
on the Apple Macintosh**

Jürgen Walther  
AI Research Division  
German National Research Center  
for Computer Science (GMD)  
P.O.Box 1316  
5205 Sankt Augustin 1  
Germany  
juergen@gmdzi.gmd.de

Andy Kohl  
AI Research Division  
German National Research Center  
for Computer Science (GMD)  
P.O.Box 1316  
5205 Sankt Augustin 1  
Germany  
kohl@gmdzi.gmd.de

This section focusses on the installation and use of BABYLON release 2.2 on Apple Macintosh. For more information on BABYLON see *The AI Workbench BABYLON*; Christaller, T., DiPrimio, F., and Voß, A. (eds), Academic Press 1992.

## 1. Hardware and Software Requirements

To run BABYLON on a Macintosh computer you should have at least 4 MB of RAM and 5 MB of free disk capacity. On the software side you need Macintosh Common Lisp 1.3.2 (CCL) or Macintosh Common Lisp 2.0 (MCL). The latter allows you to create a stand-alone application of BABYLON 2.2., called *BABYLON* if you load the file called *mac-make-mcl.lisp*; the Lisp version 1.3.2 allows you to create an image of BABYLON (thus still needing Lisp to run) if you load the file *mac-make-ccl.lisp*. All the sources that you need are in the folder BABYLON-2.2.

## 2. Installing BABYLON

Just copy the folder BABYLON-2.2 anywhere on your hard disk.

### 3. Starting BABYLON

In comparison to former versions the Macintosh Common Lisp release 1.2.1 has been considerably improved, as it now allows so-called image files to be stored. These files can be used to perform a selective memory dump of the Lisp system and to reload it later on. Loading such a selective memory dump is much faster than loading or evaluating the original Lisp forms. Thus, it is more convenient to store the whole BABYLON system as an image file. The creation of an image file is illustrated in Subsection 3.2.

#### 3.1 Starting BABYLON by Using an Image File

In order to start BABYLON open the **Babylon** folder and click on the icon named *Babylon* twice.

Please note that the Macintosh Common Lisp system is also stored on this file, consequently, it must not be loaded yet.

Having started BABYLON, you will find an extended Macintosh Common Lisp menu bar and a Lisp listener reporting the start. Later on the Lisp listener will serve as a dialog window during the consultation of expert systems.

**Note:** The Macintosh Common Lisp init data file should only be used to configure the Macintosh Common Lisp environment. Adjustments that are necessary to meet the BABYLON requirements should be carried out in the **babylon-init.lisp** file (cf. Section 6)

#### 3.2 Creating an Image File

If you want to create an image file (or an application) start your Lisp and then load either *mac-make-mcl.lisp*, if you have Macintosh Common Lisp 2.0, or *mac-make-ccl.lisp*, if you have Allegro Common Lisp 1.3.2. While loading the make file, there will be some y-or-n questions in the lisp listener

- Use development options for compiling files?  
Answer y, if you want these options

```
(defun development-options ()
  (setq *record-source-file* t
        *save-doc-strings* t
        *save-definitions* t
        *save-local-symbols* t
        *fasl-save-local-symbols* t))
```

otherwise you get the following options

```
(defun development-options ()
  (setq *record-source-file* t
        *save-doc-strings* nil
        *save-definitions* nil
        *save-local-symbols* nil
        *fasl-save-local-symbols* nil))
```

- Load graphic frame browser?

Its worth it!

- Is your AntiVirus software temporarily disabled?

Lisp will try to install a program on your hard disk. Thus, you have to either disable programs like Gatekeeper, or give MCL the right to install resources.

After the system has loaded a few basic BABYLON system files, you will see a dialog window (cf. Figure 1), which the programmer can use to specify an interpreter configuration. This configuration will be loaded and subsequently stored as an image file.

<b>Include</b>		<input checked="" type="checkbox"/> <b>free-text-mixin</b>
<b>Frame Interpreter</b>	<input type="radio"/> no	<input type="radio"/> basic-frame-mixin
<b>Rule Interpreter</b>	<input type="radio"/> no	<input type="radio"/> mini-frame-mixin
<b>Prolog Interpreter</b>	<input type="radio"/> no	<input checked="" type="radio"/> normal-frame-mixin
<b>Constraint Interpreter</b>	<input type="radio"/> no	<input type="radio"/> basic-rule-mixin
		<input type="radio"/> mini-rule-mixin
		<input checked="" type="radio"/> normal-rule-mixin
		<input type="radio"/> basic-prolog-mixin
		<input type="radio"/> mini-prolog-mixin
		<input checked="" type="radio"/> normal-prolog-mixin
		<input type="radio"/> basic-constraint-mixin
		<input type="radio"/> mini-constraint-mixin
		<input checked="" type="radio"/> normal-constraint-mixin
<b>Create Image:</b>	<input type="checkbox"/> compressed	<b>Save As ...</b> <b>Cancel</b>

Fig. 1: Creating an image file.

If you should terminate the configuration dialog by clicking on *Cancel*, no image file will be created at that point. Instead, you will find a special menu entry in the BABYLON main menu called *Configure Image*, which enables you to restart the configuration dialog. The menu entry *Load-KB* is still deactivated, with the result that no expert system can be loaded. But you can perform modifications of your Lisp environment, which you can store in your image data file. In order to store the new Lisp environment as an image file you will have to resume the configuration dialog.

#### 4. Menus

The menu bar of Macintosh Common Lisp is extended by BABYLON menus. At present, BABYLON provides five menus. A main menu, which offers general commands to operate expert systems (such as *Start*, *Kill*, ..) and four menus that provide interpreter-specific operations. One such menu is assigned to each of the four interpreters (*Frame*, *Rule*, *Prolog*, *Consat*). Which of the operations can be selected in these menus depends on the interpreter version. Those operations appearing in a grey tone cannot be selected. The operation *Explore Rule Terms* from the *rule* menu, for example, is only provided by the normal rule interpreter.

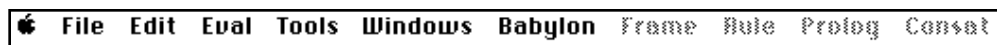


Fig. 2: The menu bar on the MAC II.

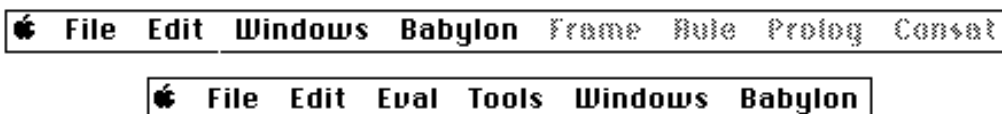


Fig. 3: Alternating menu bars on the MAC SE.

These menus and the five Macintosh Common Lisp menus combine to provide a menu bar consisting of ten entries. The screen of the Macintosh II accommodates the whole menu bar (cf. Figure 2), while the smaller screen of the Macintosh SE cannot contain all menus at the same time. For this reason, the Macintosh SE provides two menu bars (cf. Figure 3), which can be switched by using the Command-T key. This toggling operation is a menu entry in the BABYLON main menu. The first menu bar contains the *File*, *Edit* and *Windows* menu of Macintosh Common Lisp

and the five BABYLON menus, while the second one contains all Macintosh Common Lisp menus and the BABYLON main menu.

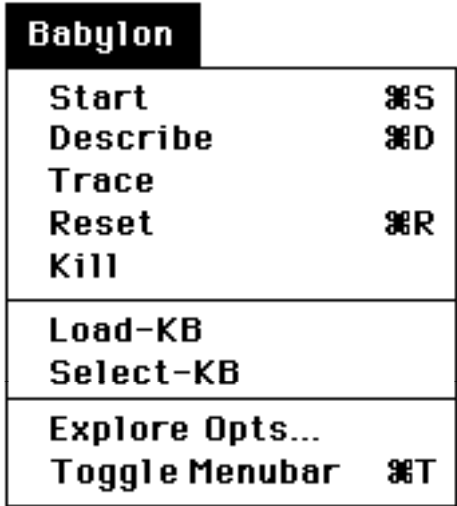
After starting BABYLON, and before loading an expert system, only the main menu of the five BABYLON menus is activated. The four interpreter-specific menus are deactivated, which is indicated by the grey color of the menu name. These menus are only activated, when an expert system with corresponding interpreters is loaded.

You should also note that the name of the BABYLON main menu, which is initially *Babylon*, will be changed when loading an expert system; it will then be that of the expert system.

The following subsections will explain the BABYLON menus and their operations.

### 4.1 The BABYLON Main Menu

The BABYLON main menu basically contains general commands to operate expert systems.



Babylon	
Start	⌘S
Describe	⌘D
Trace	
Reset	⌘R
Kill	
Load-KB	
Select-KB	
Explore Opts...	
Toggle Menubar	⌘T

Fig. 4: The BABYLON main menu.

*Start]*

serves to start the current expert system. This is a confirmation menu that informs the programmer about the setting of the trace mode. If this confirmation menu is positively acknowledged, the expert system begins to interpret its instruction part.

### *Describe*

yields some statistical information about the current expert system such as the number of frames, the number of instances and the number of rule sets, etc.

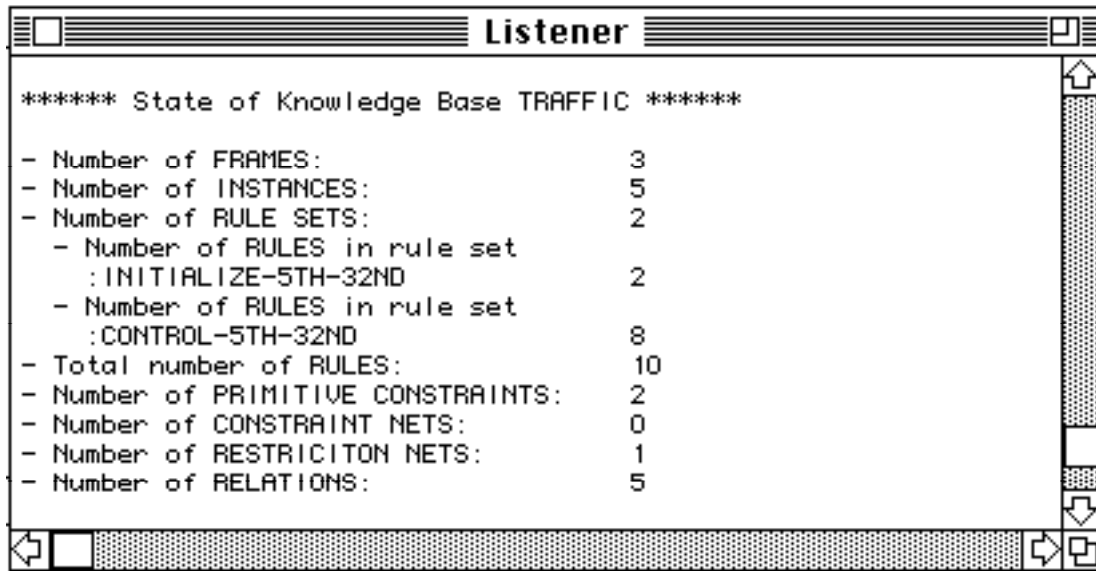


Fig. 5: The information provided by *DESCRIBE*.

### *Trace*

activates or deactivates the system trace. When the menu entry is marked with a hook, then the system trace has been activated.

### *Reset*

restores the initial state of the current expert system. This means that the modifications brought about during the dialog session with the expert system are reset. Such modifications can refer to changes in the slot values of objects, new entries in the free text data base etc. The resetting process is only started, when the user positively acknowledges a confirmation menu.

### *Kill*

presents to the user all the expert systems known to BABYLON. The expert system that has been selected will be deleted from the Lisp environment, as soon as the user confirms this command. If this was the current expert system, a different expert system (if available) will be selected to be the current one.

### *Load-KB*

presents to the user a menu containing files from the folder **babylon:samples:kbs**; these files store the external expert systems. When the user selects one expert system, it will be loaded, thus creating an instance of the referenced configuration and an editor window, which will display the source code of the expert system. Additionally, a trace window and an explanation window, which will at first be covered by the editor window, are created. At the same time the newly loaded expert system will be the current one, thus changing the name of the main menu, which will now contain the name of the new expert system.

### *Select-KB*

presents to the user a menu containing all loaded expert systems. When he/she selects an expert system, this one will be the current expert system, which will be reflected in the changed name of the main menu.

### *Explore Opts...*

yields a dialog box for the user to adjust a few parameters of the exploration options (cf. Figure 4).

<b>Frames:</b>	<input type="text"/>	<input checked="" type="checkbox"/> Sort	
<b>Supers:</b>		<input checked="" type="checkbox"/> Sort	<input type="checkbox"/> All
<b>Instances:</b>	<input type="text"/>	<input checked="" type="checkbox"/> Sort	<input type="checkbox"/> All
<b>Slots:</b>		<input checked="" type="checkbox"/> Sort	<input checked="" type="checkbox"/> All
<b>Rule Sets:</b>	<input type="text"/>	<input type="checkbox"/> Sort	
<b>Rules:</b>	<input type="text"/>	<input type="checkbox"/> Sort	
<input type="button" value="Do It"/>		<input type="button" value="Abort"/>	

Fig. 6: Specifying the exploration options.

The text fields of the dialog box can be used to enter patterns by using *jokers* or *wild cards*. The following jokers are admitted:

? stands for an arbitrary character

\* stands for a sequence of one or several characters

The pattern `b*o?`, for example, accepts the following character strings (using upper or lower case letters is irrelevant):

- *BOOK*
- *BARITONE*
- and of course *BABYLON*

The meaning of the various parameters will be explained in detail, when we discuss *Explore* of the *Frame* or *Rule* menu.



### *Toggle menu bar*

is only available and necessary on computers of the Macintosh SE type. This command is used to alternate between the two menu bars. For this purpose, the Command-T key can also be used.

## 4.2 The Frame Menu

When the configuration of the current expert system contains a frame interpreter, then the *Frame* menu will be activated. It provides operations for object and instance exploration as well as a facility to control the syntax check behavior.

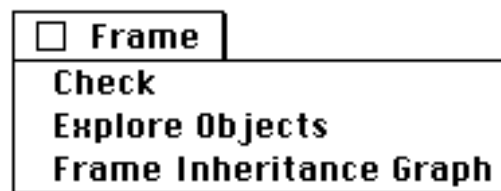


Fig. 7: The frame menu.

### *Check*

In order to support the development of expert systems the frame interpreter provides a choice of plausibility checks. For example, it checks whether the initial values of instance slots match the specification of their possible values. This menu entry allows the programmer to activate or deactivate the plausibility checks. Whether they are activated or not is reflected by the the hook before the menu entry.

As the plausibility checks require a lot of loading and resetting time, they are deactivated in the initial state and should only be activated during the development of an expert system.

### *Explore Objects*

provides a screen-oriented exploration option for the objects of the current expert system (cf. Figure 8). The screen will display six selection fields and one display field.

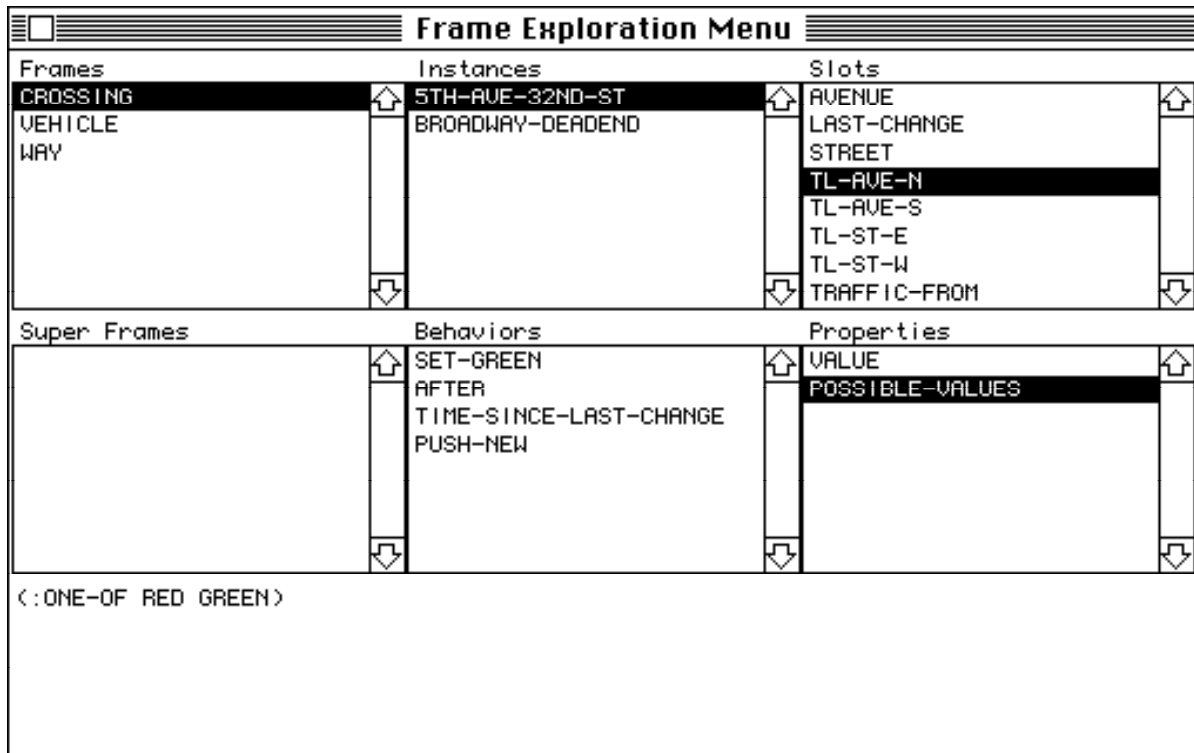


Fig. 8: The screen during the object exploration.

The selection field *Frames* lists all frame names that correspond to the reference pattern specified in the main menu (cf. *Explore Opts...*).

When the programmer selects a frame, the selection field *Super Frames* will display the corresponding superframes. If the option *All* is specified, not only the direct superframes will be displayed, but all frames preceding this frame in the inheritance hierarchy. The selection field *Instances* will display the instances of the selected frame. When choosing the option *All* the instances of the subframes will also be considered. The instance names must match the specified reference pattern.

The behaviors of the selected frame will be displayed in the selection field *Behaviors*.

When an instance is selected, its slots will be listed in the selection field *Slots*. The option *All* ensures that the slots inherited from the superframes are also considered.

The slot properties will be displayed in the selection field *Properties*.

The display field can present the parameter list and the documentation text of a behavior or the value of a slot property.

If specified, the frames, superframes, instances and slots are displayed in an alphabetical order.

Next to the titles of the selection fields *Frames*, *Instances* and *Behaviors* a button marked with > will appear. If the programmer clicks on one of these buttons, the knowledge base editor will be automatically activated. The editor will be automatically positioned to the construct that is selected in the corresponding selection field. If, for example, the programmer clicked on the button next to *Instances* in Figure 8, the system would position the editor to the line

(definstance 5th-ave-32nd-st of crossing .....

#### *Frame Inheritance Graph*

provides a tree oriented display of the inheritance structure of the objects of the current expert system (cf. Figure 9). Selecting this menu entry will pop up a selection dialog of all the frames of the current knowledge base and as a first entry \*ROOT\*. Selecting \*ROOT\* will show a window with **all** the known frames, selecting a frame name will show this frame as the root and all its subframes. Double clicking on a frame in this window will position to the frame definition in the editor buffer and option double clicking will pop up an exploration dialog for this frame (cf. Figure 10). If you want to change the root of the inheritance graph displayed, close the display window and select the *Frame Inheritance Graph* menu again.

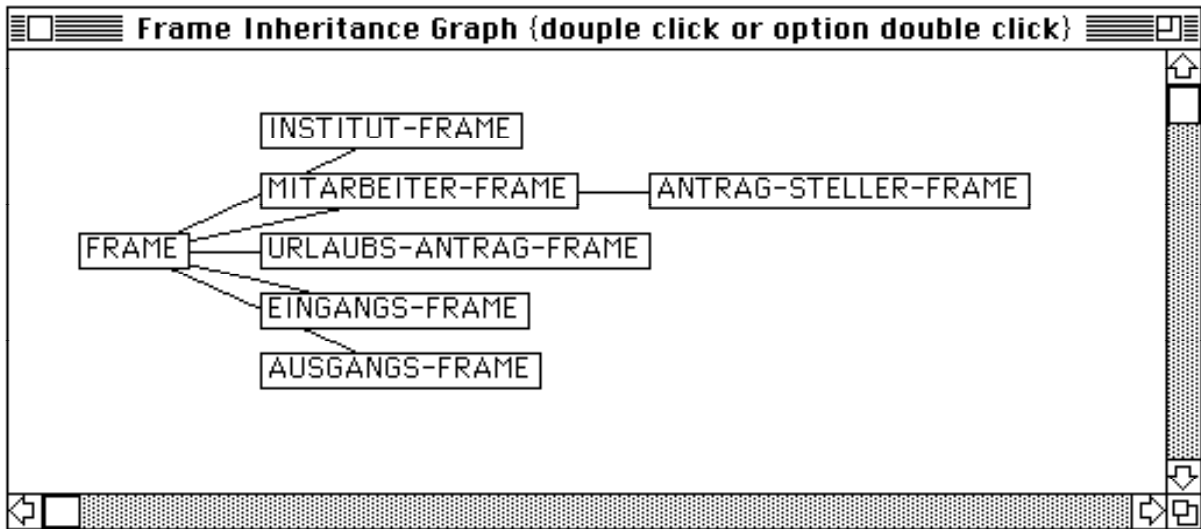


Fig. 9: The Frame Inheritance Graph Window.

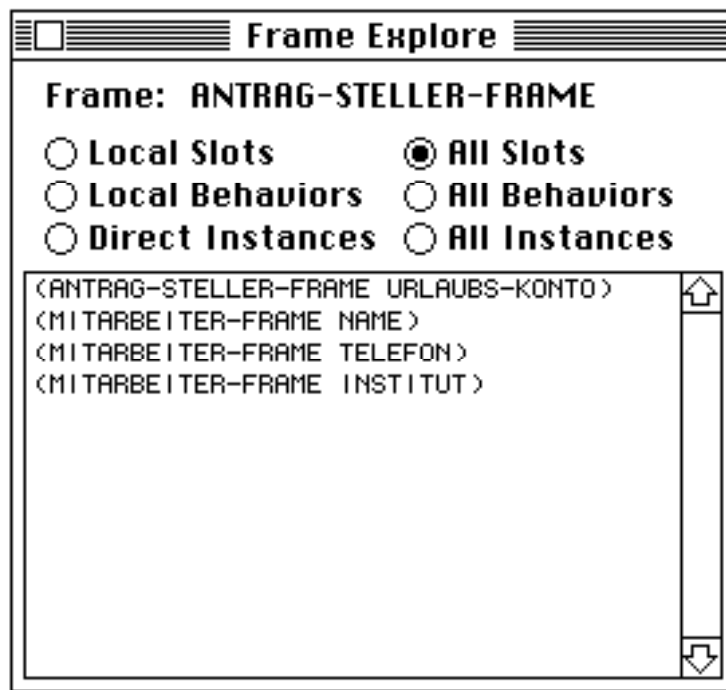


Fig. 10: The Frame Explore Dialog.

The Frame Exploration Dialog shows local or global features of the selected frame. Use the push buttons to select the features you are interested in. Selecting an entry will position to the

appropriate definition in the editor buffer. If you select a slot this will be the frame definition, containing the slot definition. Use option double click in the Frame Inheritance window to change the frame explored.

### 4.3 The Rule Menu

When an expert system with a rule interpreter as part of its configuration is current , the *Rule* menu from the menu bar can be selected. It contains operations to inspect the rule part (*Explore Rules*, *Explore Rule Terms*), to manipulate the trace behavior (*Trace*, *Set Rule Trace Options*) and to explain the consultation results (*Explore Facts*, *Hypotheses*, *Explain*).

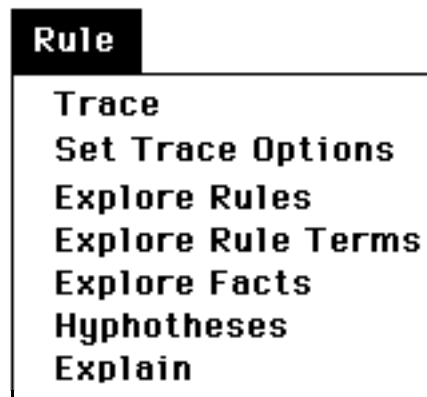


Fig. 9: The rule menu.

#### *Trace*

activates or deactivates the rule trace. If the menu entry is marked with a hook, the rule trace is deactivated. All operations that have been set by selecting the menu entry *Set Trace Options* will remain.

#### *Set Rule Trace Options*

when this menu entry is selected, a pop-up menu to regulate the rule trace behavior will appear on the screen. By using - *Exit* - it is possible to leave the menu.

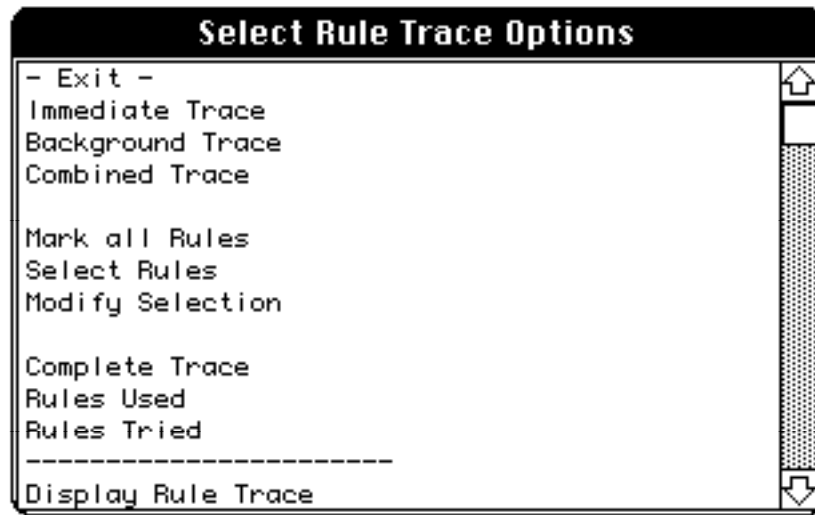


Fig. 10: Setting the rule trace behavior.

The trace can be directly, i.e. interactively, displayed in the trace window or stored in an internal data structure. In order to set this behavior the first three menu entries are used:

#### *Immediate Trace*

the trace will only appear in the trace window. This is done interactively, i.e. during a dialog session with an expert system.

#### *Background Trace*

the trace will only be stored in an internal data structure, which can be updated after the dialog by using the menu entries *Complete Trace*, *Rules Used*, *Rules Tried* and *Display Rule Trace* and displayed in the trace window.

#### *Combined Trace*

the trace can be displayed interactively as well as stored in internal data structures.

The rules that are to be included in the trace can be specified by the menu entries *Mark all Rules*, *Select Rules* and *Modify Selection*.

#### *Mark all Rules*

marks all rules of all rule sets to be included in the trace.

### Select Rules

this menu entry serves to mark the rules to be included in the trace. At the beginning, the programmer will find a menu containing all rule sets from which to choose one rule set. If only one rule set is known to the system, this menu will be omitted. Subsequently, the menu illustrated in Figure 11 will appear.

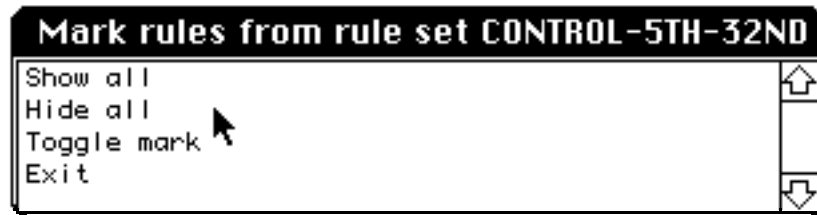


Fig. 11: The menu to mark rule sets.

- **Show all:** all rules of the rule set previously selected are marked.
- **Hide all:** the marks of all rules of the rule set previously selected are removed.
- **Toggle mark:** provides a menu that lists all rules of the rule set previously selected (cf. Figure 12).

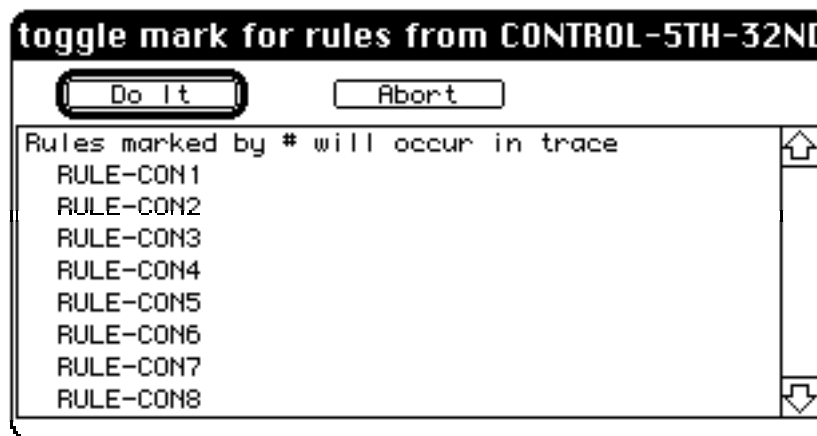


Fig. 12: Menu to mark rules.

The menu in Figure 12 can be used to select an arbitrary number of rules, whose marks are to be toggled. Rules not marked will be marked and those marked will lose theirs.

- **Exit:** to leave the menu.

**Note:** When selecting this menu entry all marks will **automatically** be removed.

#### *Modify Selection*

this menu entry can be used to modify the marks of rules. The dialog with the programmer will take place as described under *Select Rules*. When choosing this menu entry all the marks will **not** be deleted.

The next menu entries serve to filter and display the trace information internally stored.

#### *Complete Trace*

the trace information will be displayed unfiltered.

#### *Rules Used*

the trace will be filtered, with the result that the system will display only those rules that have been applied during the consultation.

#### *Rules Tried*

the trace will be filtered, with the result that the system will display only those rules that have been tested during the consultation.

#### *Display Rule Trace*

the trace will be displayed in the trace window. For this purpose, it will be filtered as previously chosen (cf. also *Complete Trace*, *Rules Used*, *Rules Tried*).

#### *Explore Rules*

provides the option of a screen-oriented exploration of rule parts (cf. Figure 13). The exploration window is divided into two selection fields and one display field.



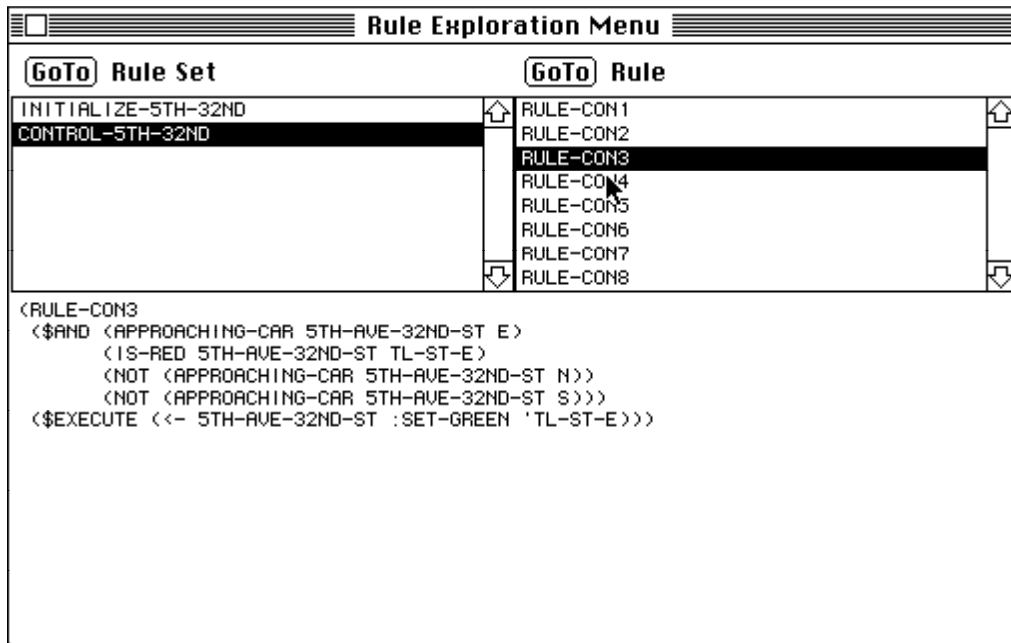


Fig. 13: The screen during the rule exploration.

The selection field *Rule Sets* will display the names of all rule sets. The selection field *Rules* will display all rule names of the rule set selected in *Rule Sets*. A rule once selected will be represented in the display field.

There is one button entitled *GoTo* for each of the two selection fields. When the programmer clicks on one of these buttons, the editor will be activated. It will be positioned at the definition of the construct that is selected in the corresponding selection field. When pressing the *GoTo* button in the *Rule* selection field (cf. Figure 13), the editor is positioned at the definition of the rule RULE-CON3.

### *Explore Rule Terms*

enables a screen-oriented exploration of the rule part about the search for terms and elements (cf. Figure 14). Terms in this context correspond to conditions in the condition part of a rule. The rule

```
(rule-2
  ($and
    (holiday-application number-days > (applicant holiday-entitlement)))
```

```
($conclude
  (holiday-application status = not-granted)
  (holiday-application :add-status
    "~%Days applied for > holiday-entitlement.")))
```

for example, contains the three terms

```
(holiday-application number-days > (applicant holiday-entitlement))
(holiday-application status = not-granted)
holiday-application :add-status
"~%Days applied for > holiday-entitlement.")))
```

and the elements

```
holiday-application
number-days
status
...
```

The exploration window has five selection fields, one display field and one option switch consisting of three alternatives.

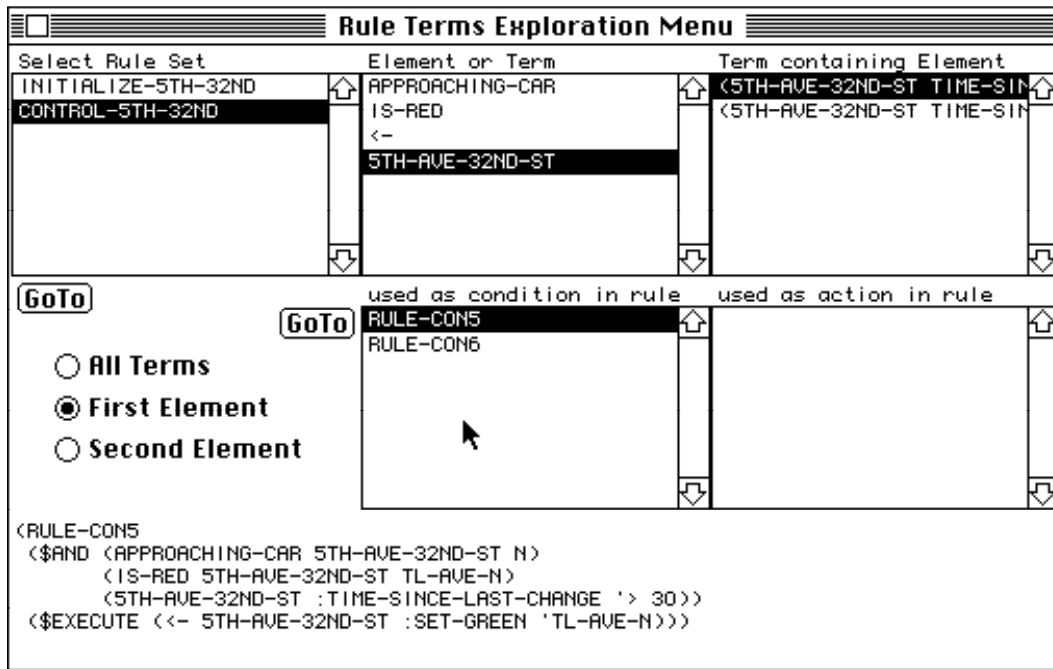


Fig. 14: The screen during the rule term exploration.

The three options have the following meaning:

- **all terms:** All terms within the rule set can be selected by the programmer.
- **first elements:** All elements that appear in the first position of a rule term can be selected by the programmer. When one element has been selected, a second menu will appear. This menu contains the terms, where the element figures in the first position (can be used to search for instance names in rule terms).
- **second elements:** Like *first elements*, with the only difference being that the second element is considered in this case (can be used to search for slot names of instances in rule terms).

The selection field *Rule Sets* will display the names of all rule sets, *Element or Term* all terms (with the option *All Terms* being activated) or all elements that appear as the first element (with the option *First Element*) or as the second one (with the option *Second Element*) in the term of a rule of the selected rule set. In the last two cases, the respective terms will be displayed in the selection field *Term containing Element*.

When a term has been selected, the rules that contain this one in their condition part will be displayed in the selection field *used as condition in rule*, and those that contain it in their action part will be represented in the selection field *used as action in rule*. Selecting one rule from one of the last two selection fields leads to this one being shown in the display field.

The selection field *Select Rule Set* has a *GoTo* button. It is located below the field. When this one is clicked on, the editor with the knowledge base will appear. In this process, it is positioned to the definition of the selected rule set. The two selection fields *used as condition in rule* and *used as action in rule* have one common *GoTo* button, which is situated to the left of the field *used as condition in rule*. When clicking on this button the knowledge base editor will be activated and positioned to the definition of the selected rule.

#### Explore Facts

enables the screen-oriented exploration of the dynamic data base after the consultation of an expert system (cf. Figure 15). The exploration window consists of two selection fields and one display field.

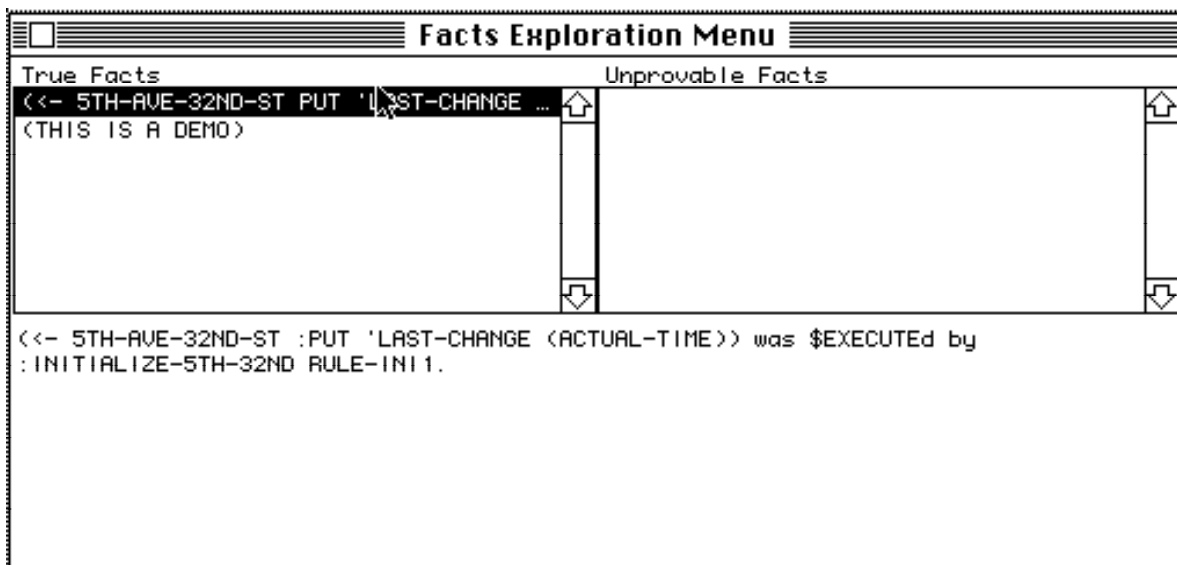


Fig. 15: The screen during the fact exploration.

The selection field *True Facts* shows the facts that have been evaluated to true during the last consultation, while the selection field *Unprovable Facts* displays the facts that cannot be proved. For the facts evaluated to true the display field can show the action that has led to the verification. An explanation as to why the listed hypotheses are not provable cannot be provided for the time being.

### *Hypotheses*

displays all hypotheses that have been verified during the last consultation in the dialog window.

### *Explain*

exposes the explanation window displaying all the facts that have been evaluated to true during the last dialog session. Subsequently, the programmer will find a menu providing the following options:

### *No*

terminates the menu without performing an action.

### *How?*

presents a menu consisting of the facts evaluated to true. When selecting a fact the explanation window will display the reason as to why this fact has been evaluated to true.

### *How all?*

presents a menu consisting of all the facts that have been verified or falsified. When selecting a fact the explanation window will display the reason behind its evaluation.

### *Print Rule*

Like *Show Rule*, but the rule is presented in a pop-up menu, with the result that its terms can be selected. When a term is selected from the condition part, the system will display a menu with rules that contain this term in its action part, provided it finds some. Terms from the action part will undergo the reverse process.

#### 4.4 The Prolog Menu

When the current expert system contains a Prolog interpreter in its configuration, then the menu *Prolog* can be selected from the menu bar. It contains operations that can be used to manipulate the trace behavior (*Trace*, *Set Trace Options*), to inspect axiom sets (*Explore Axset*), to load and add or remove axiom sets (*Load Axset*, *Select Axset*) and to verify a Prolog hypothesis (*Prove*, *Next*).

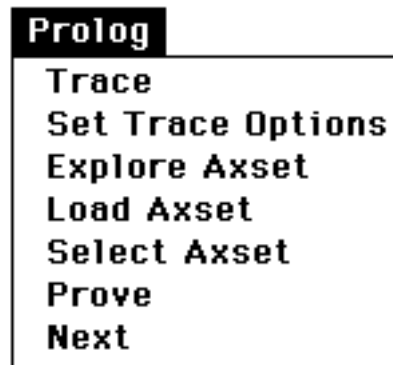


Fig. 16: The Prolog menu.

#### *Trace*

activates or deactivates the Prolog trace. When the menu entry is marked with a hook, then the Prolog trace is deactivated. All operations that have been set by selecting the menu entry *Set Trace Options* will remain.

#### *Set Trace Options*

produces another menu that can be left by using *Exit* (cf. Figure 17).

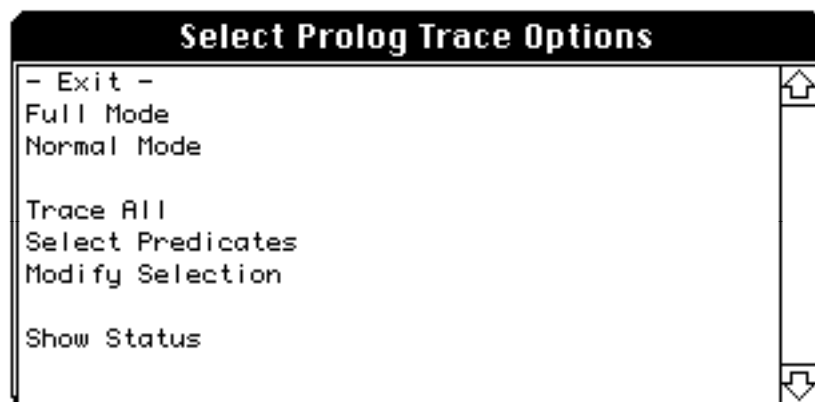


Fig. 17: The menu with Prolog trace options.

The operations that are provided by the subsequent menu are as follows:

*Full Mode*

modifies the trace to provide detailed information.

*Normal Mode*

modifies the trace to provide brief information.

*Trace All*

includes all predicates in the trace.

*Select Predicates*

This menu entry serves to mark predicates to be traced. At first, the programmer finds a menu with all axiom sets from which one can be chosen. If only one axiom set is known to the system, this choose menu will be omitted. Then the menu illustrated in Figure 18 will appear.



Fig. 18: The menu to mark axiom sets.

- **Trace all:** All predicates of the axiom set previously selected are marked.
- **Trace none:** The marks of all predicates of the axiom set previously selected are removed.
- **Toggle trace:** presents a menu that lists all predicates of the axiom set previously selected (cf. Figure 19).

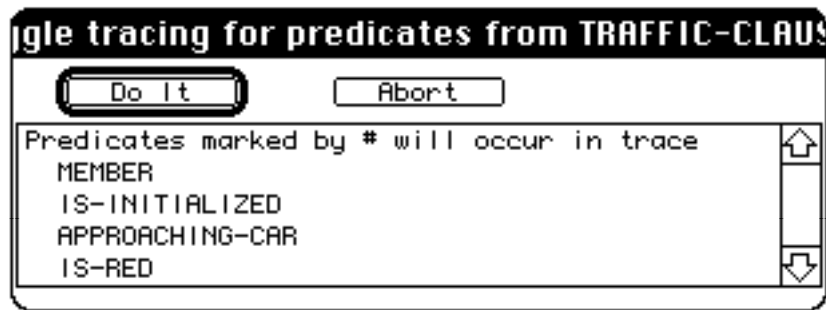


Fig. 19: The menu to mark predicates.

This can be used to select predicates: those not marked will be marked, and those marked will lose their marks.

- **Exit:** to leave the menu.

**Note:** When choosing this menu entry all marks made so far will **automatically** be removed.

#### *Modify Selection*

This menu entry can be used to modify the marks of predicates. The dialog with the programmer will take place as described for Select Predicates. When choosing this menu entry the marks made so far will **not** be deleted.

#### *Show Status*

describes the current trace behavior.

#### *Explore Aset*

provides a screen-oriented exploration option for the Prolog part of the current expert system (cf. Figure 20). The screen will display two selection fields and one display field.



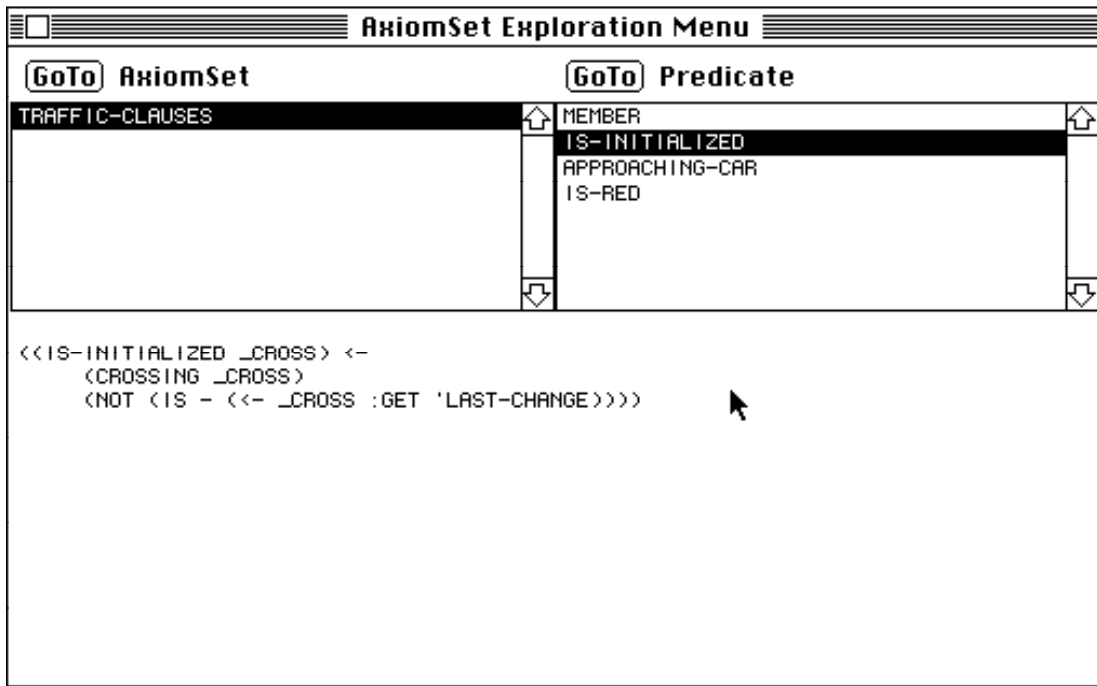


Fig. 20: The screen during the Prolog exploration.

The selection field *AxiomSet* will display all axiom set names. When selecting an axiom set the names of the predicates contained in it will be shown in the selection window *Predicate*.

When the programmer selects a predicate, its definition will be shown in the display field.

Both selection fields have one *GoTo* button each. When the programmer clicks on the button in the selection field *AxiomSet*, then the knowledge base editor will appear. This one is positioned to the definition of the selected axiom set. The same applies to the button in the selection field *Predicate*.

#### *Load Axset*

presents a data choose menu with the data files of the folder **babylon:samples:axsets**. The files contain predicate definitions. The definitions from the selected file are loaded. In order to make them known to the Prolog interpreter, they will have to be activated by using the Select operation (cf. the next menu entry).

### *Select Axiom*

it is possible to select one or several axiom sets that are to be introduced to the Prolog interpreter from a given menu. But note that afterwards all axiom sets that have not been selected are unknown to the interpreter, i.e. they are deactivated, even if they were activated before.

### *Prove*

enables the programmer to enter a Prolog hypothesis to be verified into the dialog window, and yields the first proof that has been established. To compute further solutions, please, use the *Next* operation (cf. the following menu entry).

For example, load and select (*Load, Select*) the axiom set *set-ax* and verify the hypothesis:

```
((member _x (1 2 3 4))(member _x (3 4 5 6)))
```

But note the Lisp-oriented notation.

### *Next*

yields the next proof of the hypothesis that is being verified.

## 4.4 The Consat Menu

Supposing an expert system with a constraint interpreter as part of its configuration is the current one, the menu *Consat* can be selected from the menu bar. It contains operations to manipulate the trace behavior (*Trace*), to define and display constraints (*Define Constraint, Explore Constraint*) and to satisfy constraints locally and globally (*Satisfy Locally, Satisfy Globally*).



Fig. 21: The Consat menu.

### *Trace*

presents a new menu that can be exited by using *do nothing*. It contains operations to select constraints that are to be included in the trace.

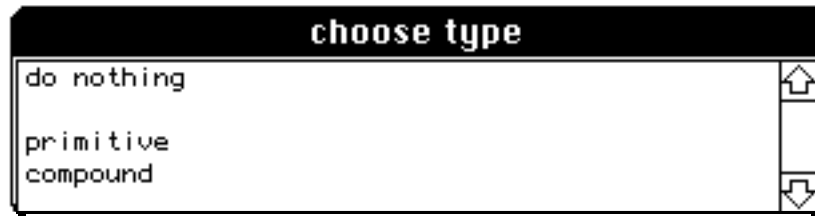


Fig. 22: Menu with Consat trace options.

### *primitive*

It is possible to select one or several constraints that are to be traced from the menu of all primitive constraints.

### *compound*

It is possible to select one or several constraints that are to be traced from the menu of all compound constraints.

### *Explore Constraints*

enables a screen-oriented exploration of the constraint part (cf. Figure 23). The exploration window is divided into a selection field, two option switches with two alternatives each and one display field.

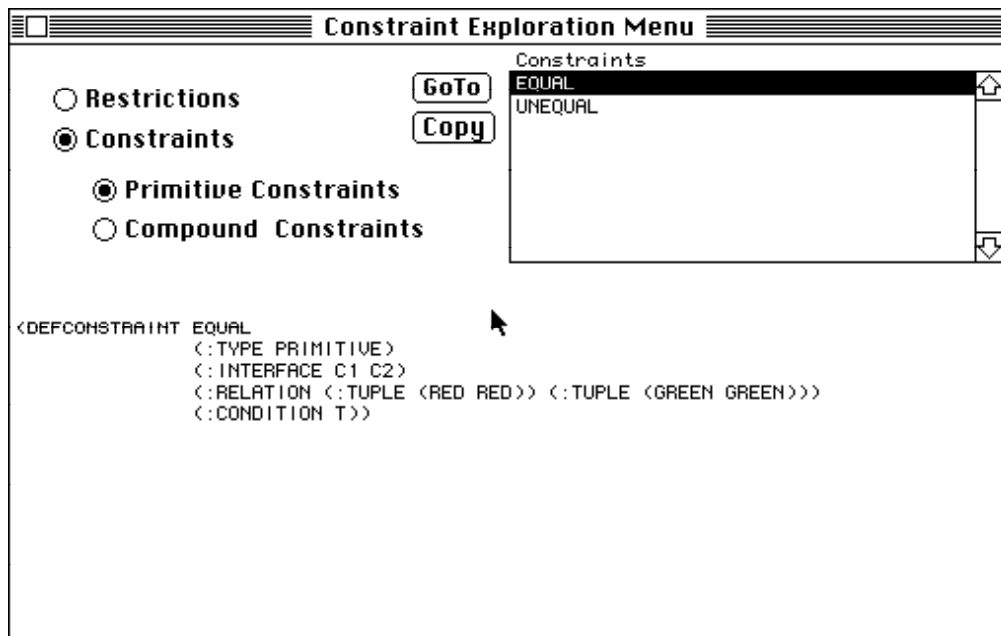


Fig. 23: The screen during the constraint exploration.

The selection field *Constraints* can list names of constraints or restrictions. This depends on the first option switch. When this option switch is set to *Constraints*, the second option switch can be used to choose between primitive and compound constraints. When a restriction or a constraint is selected, its representation will be shown in the display field.

In addition, there is a *GoTo* and a *Copy* button. Clicking on the first one activates the knowledge base editor and positions it to the definition of the selected construct.

The second button can be used to copy the text shown in the display field into the kill-buffer of the knowledge base editor. Then, the text can be transferred and copied into the knowledge base by means of the editor (emacs) (which is usually achieved by pressing the Control-Y key). This is necessary, when new constraints are created by using the operation *Define Constraint* (cf. the next menu entry). These constraints are only known in the internal, but not in the external representation of the knowledge base, which is stored in the corresponding file.

### *Define Constraint*

enables the interactive definition of constraints. The definition process is supported by menus and input requests.

### *Satisfy Locally*

When the programmer has selected a constraint (primitive or compound), he/she will be successively asked to enter its initial values. Afterwards the system will try to find a locally consistent solution.

### *Satisfy Globally*

When the programmer has selected a constraint (primitive or compound) and specified the maximum number of global solutions that are to be computed, he/she will successively be asked to enter its initial values. Afterwards, the system will try to find globally consistent solutions.

## **5. The Interface Mixin**

The user-interface described here is usually based on the normal-interface-mixin, which is also available on other computers. In the configuration of an expert system it must be specified by the option `:interface`. For example:

```
(def-kb-configuration crossingc
  (:procs normal-frame-mixin
          normal-rule-mixin
          normal-constraint-mixin
          lisp-mixin
          normal-prolog-mixin
          free-text-mixin)
  (:interface normal-interface-mixin))
```

The normal-interface-mixin mainly provides menus and windows.

### **5.1 Menus**

There are two types of menus:

- menus to select **one** node, which is activated by the selection.

- menus to select **several** nodes, which are individually selected and then activated by clicking on the confirmation field *Do It*. When selecting entries the Shift and Command keys play an important role:

**Shift:** all entries from the first one to the last one are selected.

**Command:** the last entry is toggled. This means that the entry is selected, if it has not been selected yet, otherwise the selection will be annuled.

A lot of menus also provide additional information in the form of menu entries that are not to be selected. To this end, other computers provide such entries with a *:no-select* property. The Macintosh pop-up menus do not have such a property. Thus, these entries can be selected, but yield nil.

## 5.2 The Window

The **normal-interface-mixin** creates three windows for any expert system:

- a trace window as an instance of the editor window (*\*fred-window\**), which displays the system, the rule, the Prolog as well as the constraint trace.
- an explanation window as an instance of the editor window (*\*fred-window\**), which displays all explanations.
- and an editor window, which can edit the knowledge base of the expert system.

The dialog window of all expert systems is mapped to the Top Lisp Listener.

## 6. Additional Information

You can adjust BABYLON in different ways to meet your requirements. You can develop new interpreters or user-interfaces or modify the supplied source code. Apart from this rather time-consuming method there are also a few global variables, which can be modified in the file **babylon:babylon-init.lisp**.

A few crucial variables are the keyboard functions for special actions. For example, after an input request you can ask for a context explanation during a BABYLON consultation by pressing the **Help** key (bound to the ?-key), and for information about the possible answers by pressing the **Control-Help** key (bound to the Escape-key, but note that this one is internally described as Clear!).

In addition, you can vary the maximum length of menu entries (initially set to 50) and define the maximum number of menu entries (initially set to 20).

```
(setf *help-key*          #\?  
      *c-help-key*       #\escape  
      *end-key*          #\return  
      *item-width*       50  
      *max-menu-entries* 20)
```

In order to change other important system parameters refer to the documentation in the book or the sources of the implementation.