

Freja
An object-oriented CASE Tool
Tutorial and Reference Manual

Mjølnær Informatics Report

MIA 93-31(2.2)

August 1996

Copyright © 1995-96 Mjølnær Informatics ApS.
All rights reserved.
No part of this document may be copied or distributed
without the prior written permission of Mjølnær Informatics

Contents

1	Introduction	1
2	The Notation.....	2
2.1	Pattern Diagrams.....	2
2.1.1	Pattern.....	2
2.1.2	Composition	2
2.1.3	Specialization	3
2.1.4	Association	3
2.2	Object Diagrams	4
2.2.1	Static Object	4
2.2.2	Dynamic Object.....	5
2.2.3	Procedure.....	5
2.2.4	Active Object.....	5
2.2.5	Dynamic Object Creation.....	5
2.2.6	Procedure Call	5
2.2.7	Composition	5
3	The Architecture.....	7
4	Tutorial	9
4.1	How to Get Started.....	9
4.1.1	Pattern Diagrams	9
4.1.2	Object Diagrams.....	9
4.2	Editing.....	9
4.2.1	Creating a New Diagram.....	9
4.2.2	Filling Out Templates.....	10
4.2.3	Text editing.....	11
4.2.4	Adding a New Pattern	13
4.2.5	Specifying Specialization	13
4.2.6	Adding Object References.....	14
4.2.7	Completing the Code in Sif	15
4.3	Reverse Engineering	21
4.3.1	Pattern Diagrams	21
4.3.2	Object Diagrams.....	23
5	Reference Manual.....	26
5.1	How to Get Started.....	26
5.1.1	Pattern Diagrams	26
5.1.2	Object Diagrams.....	26
5.2	Work Sheets	26
5.3	The Group Page	27
5.4	Object Pages.....	27
5.5	The Menu Bar	28
5.5.1	File Menu	28
5.5.2	Edit Menu	33
5.5.3	Templates Menu	35
5.5.4	Expand Menu	36
5.5.5	Relations Menu.....	38
5.5.6	View Menu	45
5.5.7	Create Menu	48
5.5.8	Makeup Menu.....	54
5.5.9	Align Menu.....	57
5.2.11	Page Menu.....	58

5.5.10 The Object Menu	59
6 Bibliography.....	63
Index.....	64

1 Introduction

Freja is an object-oriented CASE tool supporting system development with BETA as the implementation language. Together with Sif, the Mjølner Source Browser and Editor [MIA 91-11], Freja supports a smooth transition from design diagrams to implementation code and vice versa. The design notation is a graphical syntax for part of the BETA programming language. The basic idea is to use the same abstract language for design as well as implementation. A graphical syntax is used for design descriptions and the usual textual syntax is used for program code.

The CASE tool offers:

- **Diagram editing**
Design diagrams can be created and modified.
- **Automatic code generation**
Code skeletons are generated automatically from the design diagrams.
- **Reverse engineering**
Design diagrams can be automatically created from the program code.
- **Simultaneous editing of design descriptions and program code**
The design descriptions and the corresponding program code can be viewed and edited simultaneously, i.e. modifications in the design diagrams are reflected immediately in the program code and vice versa.
- **Structure editing**
The editing technique in the diagram editor as well as the program code editor is structure editing, which means that syntax errors are prevented. The user is offered templates of the legal constructs of the language.
- **Integrated system development environment**
The diagram editor Freja is integrated with the textual structure editor Sif which in turn is integrated with the rest of the Mjølner BETA System, e.g. the compiler and the debugger Valhalla. In this way the user is offered a system development environment that supports development of design diagrams, generation of program skeletons, filling out the code details, compiling and debugging. During the detailed implementation, testing and debugging process, the overall structure of the program may have changed, which makes the original design diagrams obsolete, but then new design diagrams can be generated automatically.

Freja is a Design/OA [Jensen 91] application written in BETA. Access to Design/OA is provided through a BETA library called DesignEnv [Knudsen 94], that interfaces to the Design/OA C library.

2 The Notation

As mentioned the graphical design notation is basically a graphical syntax for part of the BETA programming language. There are two kinds of diagrams: pattern diagrams and object diagrams.

Pattern diagrams illustrate the static structure of a BETA program, with symbols for pattern, composition and specialization. A symbol that is not directly found as a language construct in BETA is an association.

Object diagrams focuses on objects and the dynamic aspects of a BETA program. Object diagrams illustrate static and dynamic objects, composition, dynamic object creation and procedure calls.

Action sequences (i.e. the Do-Parts of objects) and enter/exit parts are not described graphically.

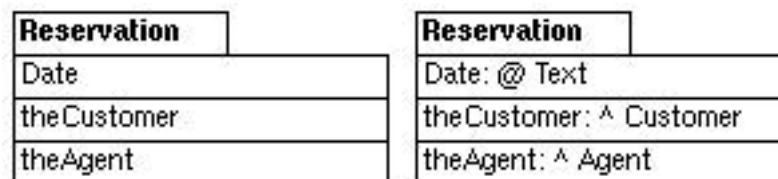
The concrete graphical syntax is not as important as the underlying concepts. Other existing design notations may be considered as alternatives in future versions of Freja.

2.1 Pattern Diagrams

Pattern diagrams are a mix of graphical symbols and usual textual BETA syntax.

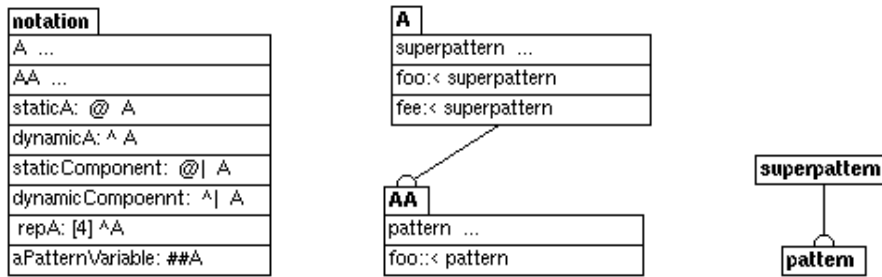
2.1.1 Pattern

A pattern is shown as a diagram with a title (the name of the pattern) and an optional list of rectangular boxes describing the attributes of the pattern. The two diagrams shown below are actually two different views on the same pattern (*Reservation*). The left diagram showing only the names of the attributes and the right diagram showing full type information of the attributes. In the following examples one or the other view will be used depending on the illustrative purposes of the example.



2.1.2 Composition

Attributes of a pattern are displayed in a list of rectangular boxes:

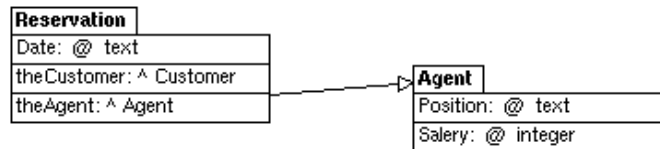


Whole-part Composition

Declaration of static references and static components are shown in the usual textual BETA syntax inside the boxes, except when the declarations contain object descriptors. In that case three dots (...) are shown instead, indicating that the details are suppressed. The object descriptor is shown in a separate pattern diagram.

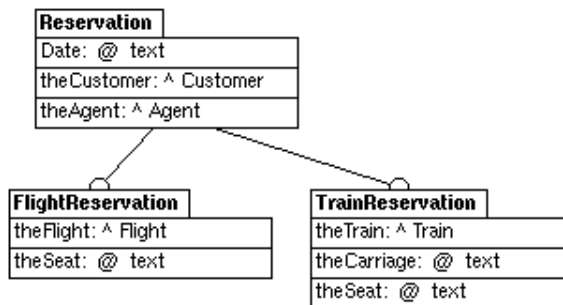
Reference Composition

Declaration of dynamic references and dynamic components can be shown in two ways. In the usual textual BETA syntax and/or using an arrow that points to the referenced pattern:



2.1.3 Specialization

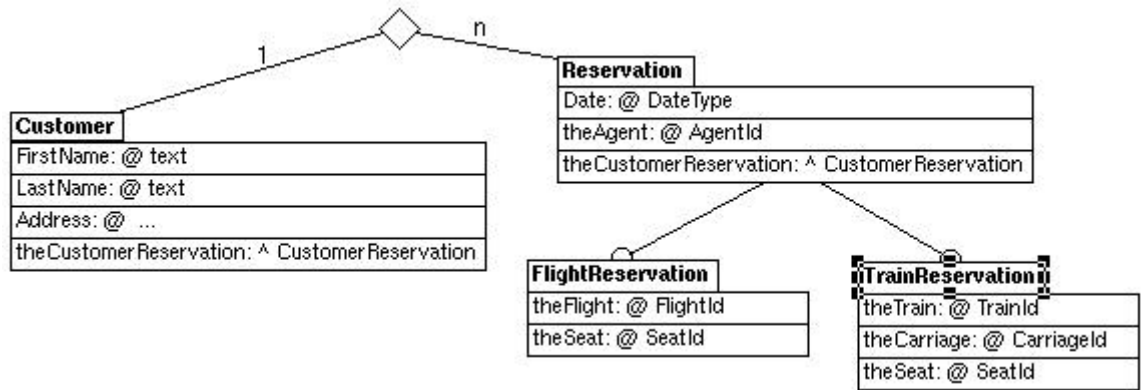
Specialization is illustrated as a tree of pattern diagrams. A connector is drawn between the subpattern and the superpattern. In the subpattern end the connector is attached to a half circle:



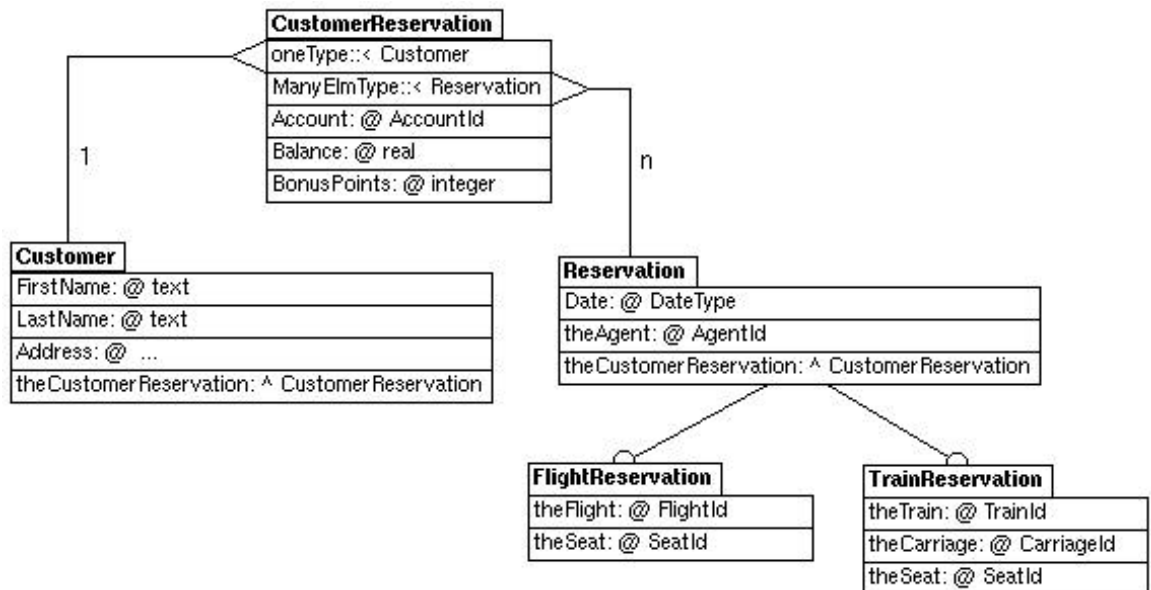
2.1.4 Association

An association defines a relation between objects of two patterns.

Three kinds of associations are possible. one to one, one to many and many to many:



An association can be shown in abstracted form as above or detailed as below:



As illustrated an association may have attributes like a pattern. In fact an association is a specialized pattern.

Except for associations the definition of the graphical syntax for pattern diagrams defines the mapping between design diagrams and BETA code.

2.2 Object Diagrams

2.2.1 Static Object

Rounded box in solid linestyle. If the object is pattern-defined the name of the pattern is shown following the “:” after the name of the object. If the object is singularly defined only the name of the object is shown.

MG:
Machinegroup

2.2.2 Dynamic Object

Rounded box in dashed linestyle. The name of the pattern that the object is an instance of is shown inside the box.

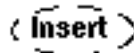


Notice:

A dynamic object is shown inside the object where the corresponding pattern is defined and not where the object is created (“the dynamic object is shown where it conceptually belongs”).

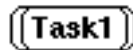
2.2.3 Procedure

Ellipse in a dashed linestyle. The name of the procedure is shown inside the ellipse. Detailed procedures are for practical reasons shown as rounded boxes in a dashed linestyle. Detailed procedures can be distinguished from detailed dynamic objects through the fact that the text “(PROC)” is appended to the name of the procedure when it is detailed.



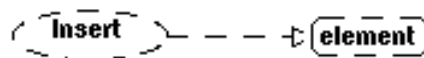
2.2.4 Active Object

Both static and dynamic objects can be active (“have an execution thread of their own”, components in BETA terms). These are shown with two parallel lines inside the object - one in each side.



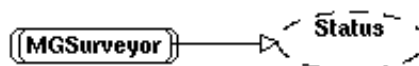
2.2.5 Dynamic Object Creation

Arrow in dashed linestyle. Going from the creating object to the created dynamic object.



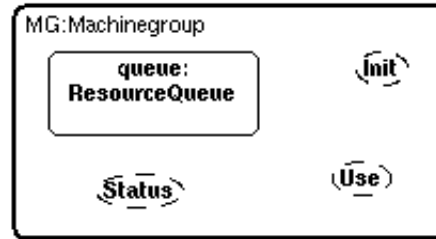
2.2.6 Procedure Call

Arrow in solid linestyle. Going from the calling object to the called procedure.

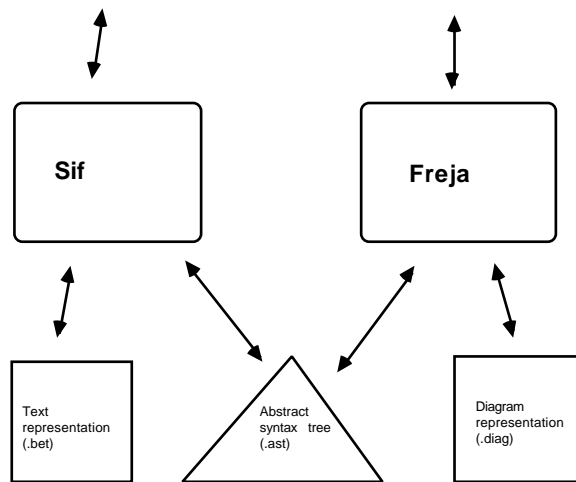
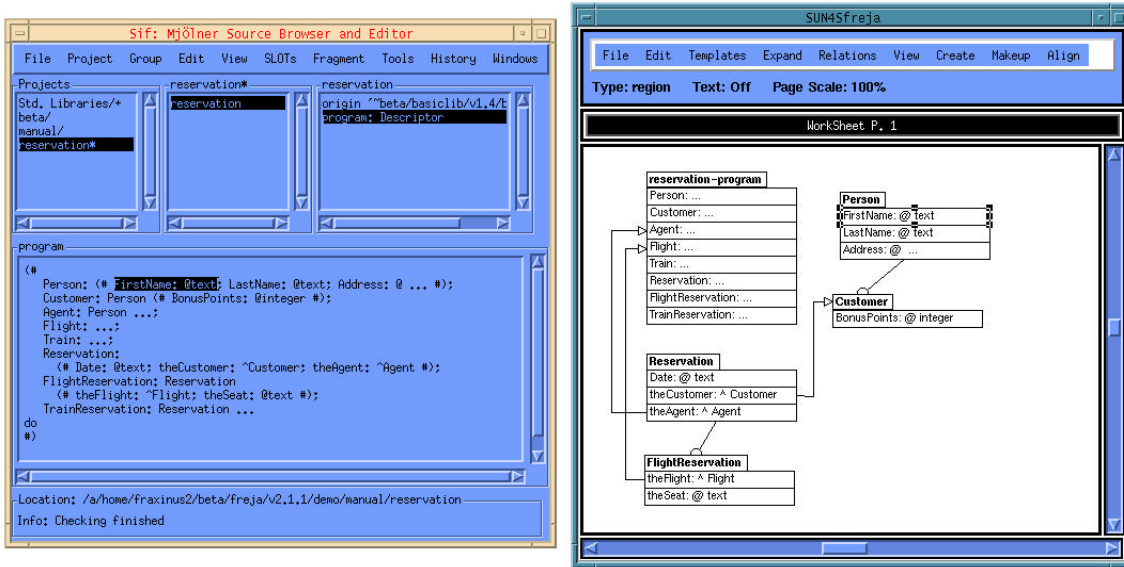


2.2.7 Composition

Currently only whole-part composition is displayed in the object diagrams. In the notation, part objects are simply shown nested inside their enclosing objects.



3 The Architecture



The figure illustrates how Freja and Sif work together. Freja and Sif are two applications that are tightly integrated in two ways.

Firstly, since the same abstract language is used for design as well as implementation, Freja and Sif can share one common representation of the program being developed. The structure editing technique gives a convenient representation, namely an abstract syntax tree (AST). The AST is presented (prettyprinted) textually in Sif and graphically in Freja.

Secondly, they communicate about changes in focus or modifications of the AST. Sif manipulates the AST through the textual representation and Freja manipulates the AST through the graphical representation. Whenever one of the editors modifies the AST, the other editor is notified and the other representation can be updated accordingly. Since the graphical syntax only reflects the overall structure of BETA programs, many modifications in the textual representation may not affect the graphical

representation. Conversely, almost every modification of the design diagram, except changing the layout of the diagrams, implies that the textual representation must be updated.

Both representations need not be visible at the same time. In the start of the development process the user might prefer only to see the diagrams, whereas the textual representation becomes more important later on. Since the diagrams and the program are prettyprints of the AST, they can always be reproduced. The textual prettyprinting is always reproducible. However if the user makes changes to the layout of the diagrams and want to keep the changes, the diagrams must be saved like the AST.

There are 3 important file types: .ast, .bet and .diag files.

The .ast and the .bet files are well-known to BETA programmers. The .ast file contains the AST and the .bet file is the textual version of the AST. Sif automatically produces the .bet file from the .ast file.

Layout of diagrams is only preserved through use of Freja

The .diag file contains a representation of the diagrams including layout information and references to the corresponding nodes in the AST. This means that if the user wants to keep the layout of the diagrams, the AST must only be modified through Freja (and/or Sif, when Sif has been started through Freja).

4 Tutorial

4.1 How to Get Started

4.1.1 Pattern Diagrams

Freja is started from an xterm in one of the following ways:

```
freja
freja foo.bet
freja foo.ast
freja foo.diag
freja foo
```

Using the first option only a menu bar will be presented to the user.

The second and the third option opens a BETA fragment (`foo`) and the fourth opens a saved diagram. The last option checks the timestamps of the files on disk and opens whichever is newest.

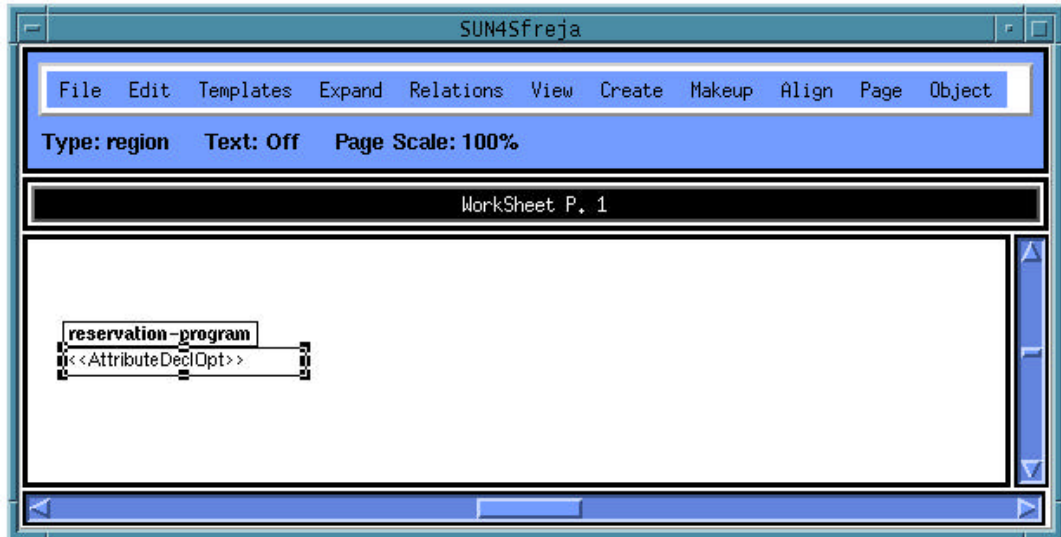
4.1.2 Object Diagrams

Object diagrams can be generated for any BETA program that has been checked. If the BETA program has not been checked or contains semantic errors object diagrams **cannot** be generated.

4.2 Editing

4.2.1 Creating a New Diagram

To create a new diagram use the command **New BETA program** or **New BETA library** in the **File Menu**. Below the command **New BETA program** has been used:



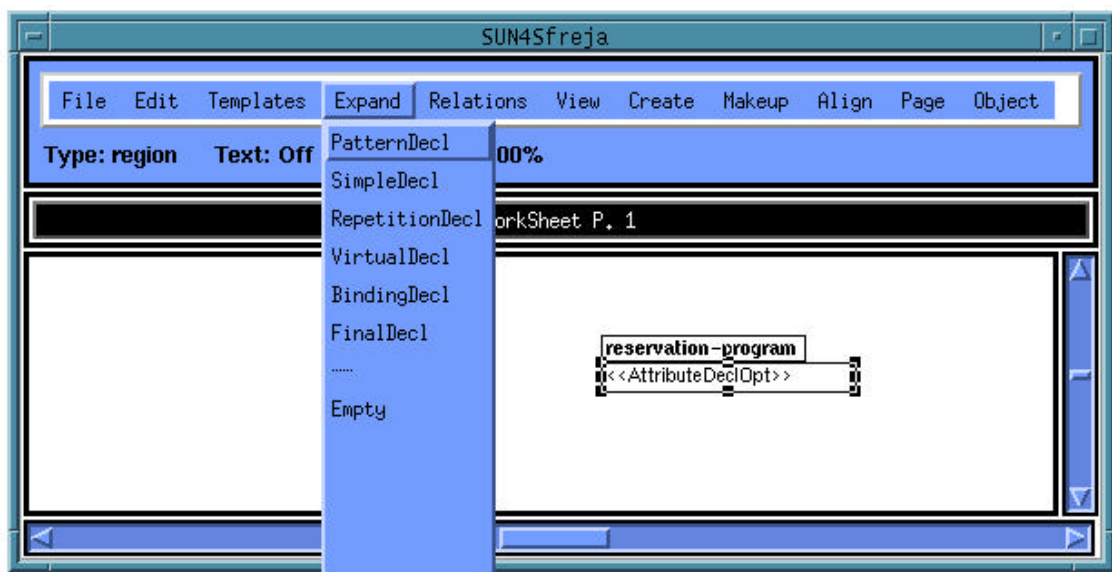
The title of the pattern diagram corresponds to the name and the category (program) of the new BETA fragment and the single attribute shown below the title corresponds to the single (unexpanded) attribute of the descriptor of the program:

```
ORIGIN '~beta/basiclib/v1.4/betaenv'
-- program: DescriptorForm --
<<PrefixOpt>> (# <<AttributeDeclOpt>> do <<ImpOpt>> #)
```

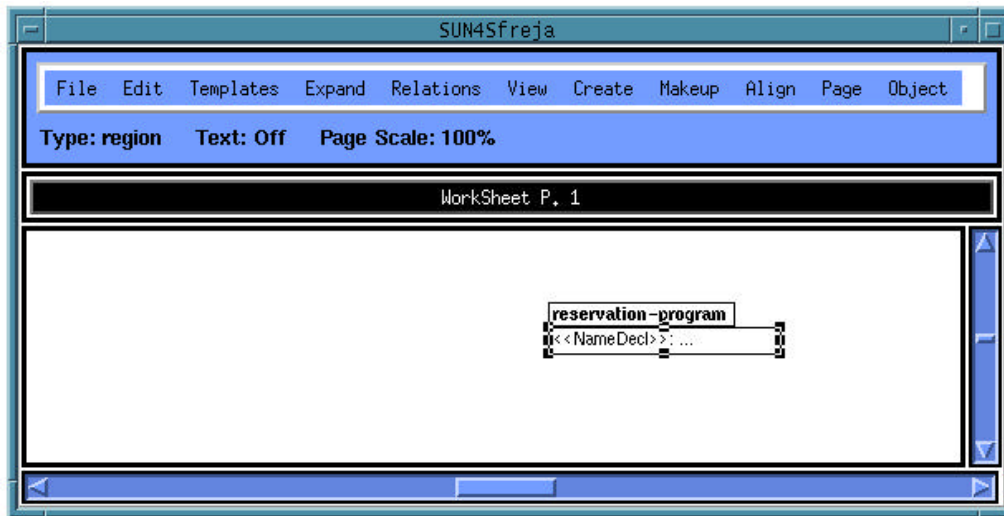
The diagram should be considered as a *graphical template* for a BETA program. In general a graphical template is a diagram containing *placeholders* (nonterminals) and *keywords* (terminals). The template constitutes an incomplete but syntactically correct design. Syntactically correct means that the diagram conforms to the rules of the graphical language. In this case the diagram is the graphical template is simply a title node and a node containing a placeholder for attribute declarations. The structure editing technique helps the user in creating diagrams that always conform to the rules of the design notation, i.e. the graphical syntax for BETA.

4.2.2 Filling Out Templates

When a placeholder is selected in a diagram the **Expand menu** contains the possible expansions according to the grammar of the design language, i.e. part of BETA:



If the `PatternDecl` attribute is selected the `AttributeDeclOpt` placeholder is substituted with a template for a pattern declaration. The graphical symbol representing the pattern declaration consists of a placeholder for a name declaration and three dots (...) indicating that the object descriptor is suppressed in the presentation.



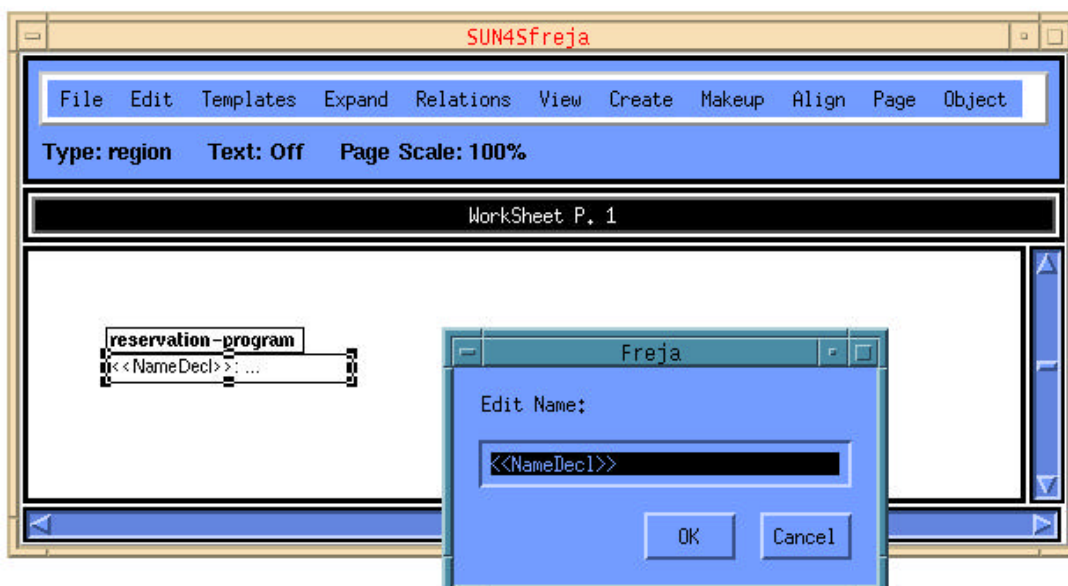
The corresponding BETA syntax for the pattern declaration template is:

```
<<NameDecl>>: <<PrefixOpt>>
  (# <<AttributeDeclOpt>>
  <<EnterPartOpt>>
  <<DoPartOpt>>
  <<ExitPartOpt>>
  #)
```

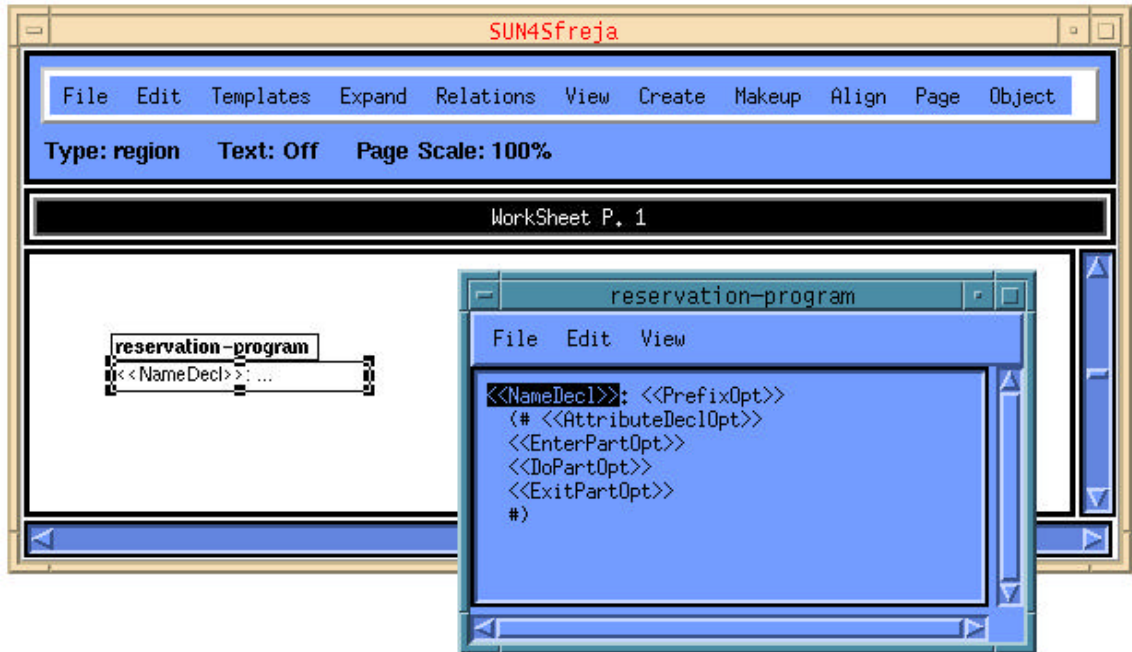
The detailed diagram representing the pattern declaration can be shown by double-clicking in the box.

4.2.3 Text editing

Now the name of the pattern can be typed by using the **Edit Name** command in the **Edit Menu**. This brings up a dialog in which the name of the pattern can be entered.



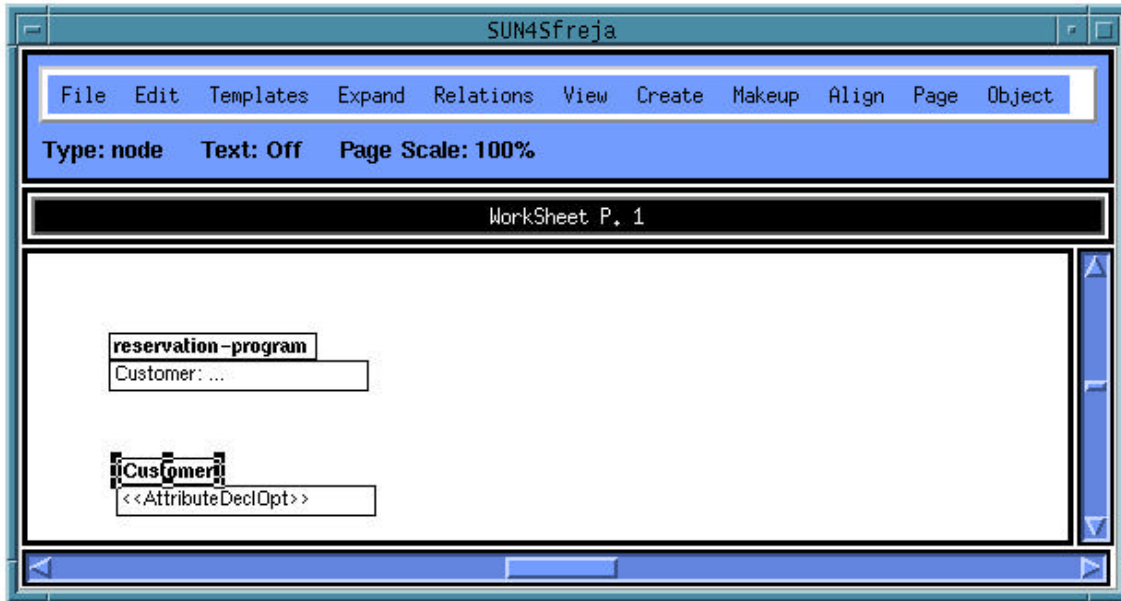
Alternatively choosing the **Open Subeditor** command in the **Edit Menu** will bring up a textual structure editor on the pattern to which the node corresponds. Here the name of the pattern can also be specified. It should although be noted that usually a subeditor would be started with the purpose of performing more complex textual editing than just entering the name of an attribute.



Being done editing in the subeditor one is free of choice to close the editor or to leave it open. If it is left open changes to the part of the code displayed in the subeditor will be reflected here and one is can resume editing it at any time wanted.



Below is the detailed diagram shown for the Customer pattern declaration:

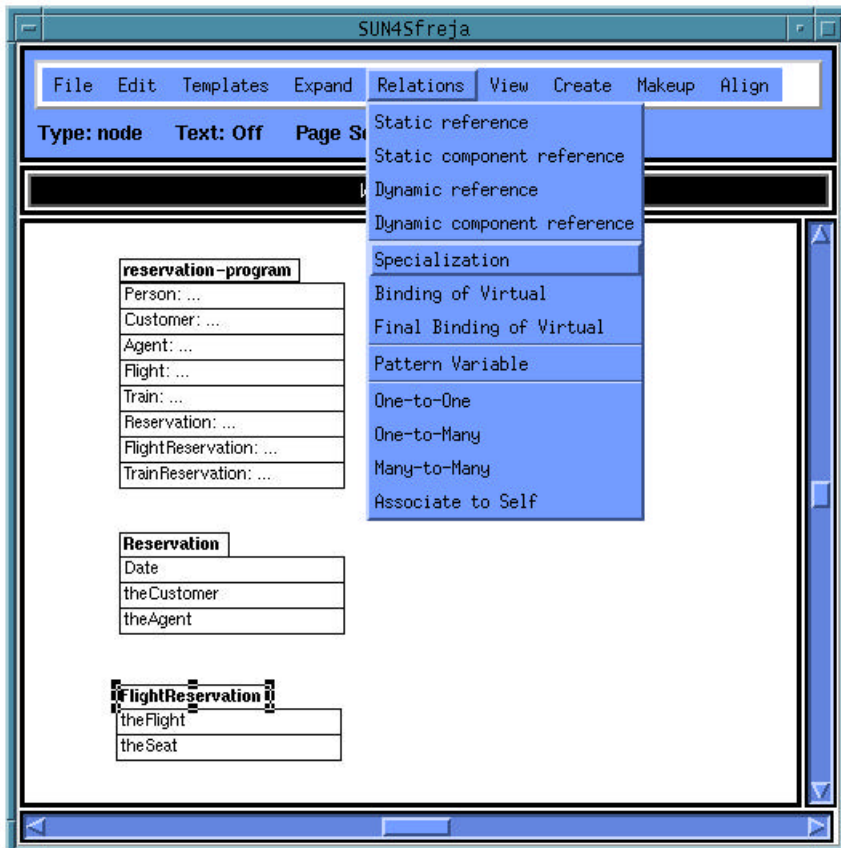


4.2.4 Adding a New Pattern

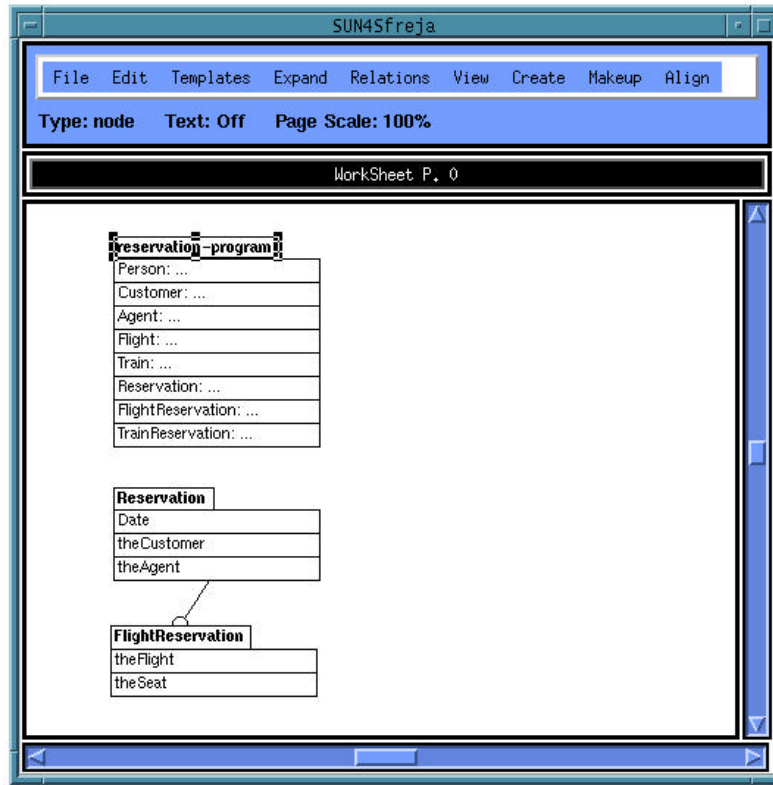
The **New Pattern** command in the **Template Menu** is a quick way to get a new detailed diagram of a pattern declaration. This command can be used as a short cut instead of the commands explained above.

4.2.5 Specifying Specialization

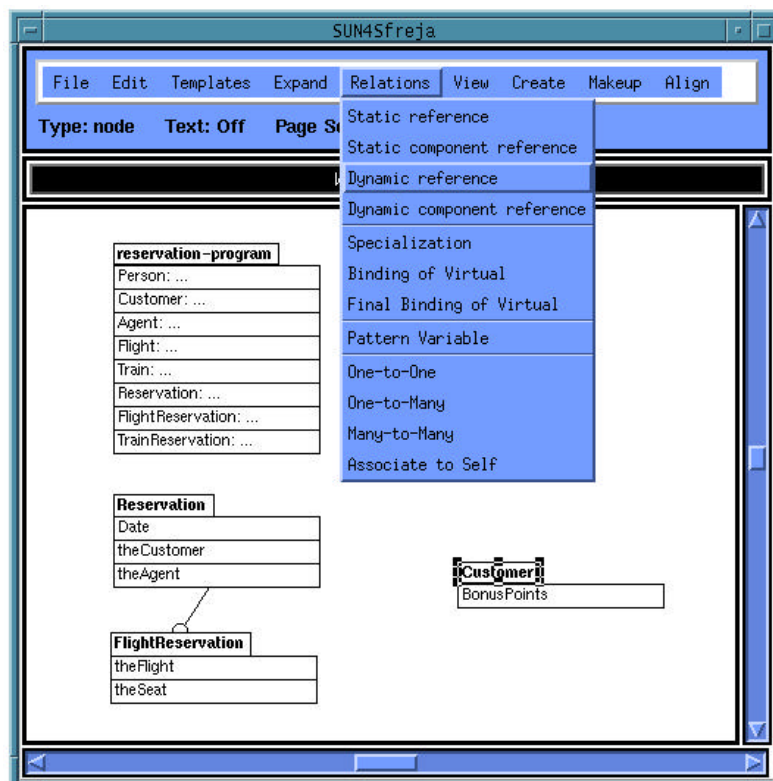
Consider the following situation where two pattern declarations of the `reservation` program has been detailed.



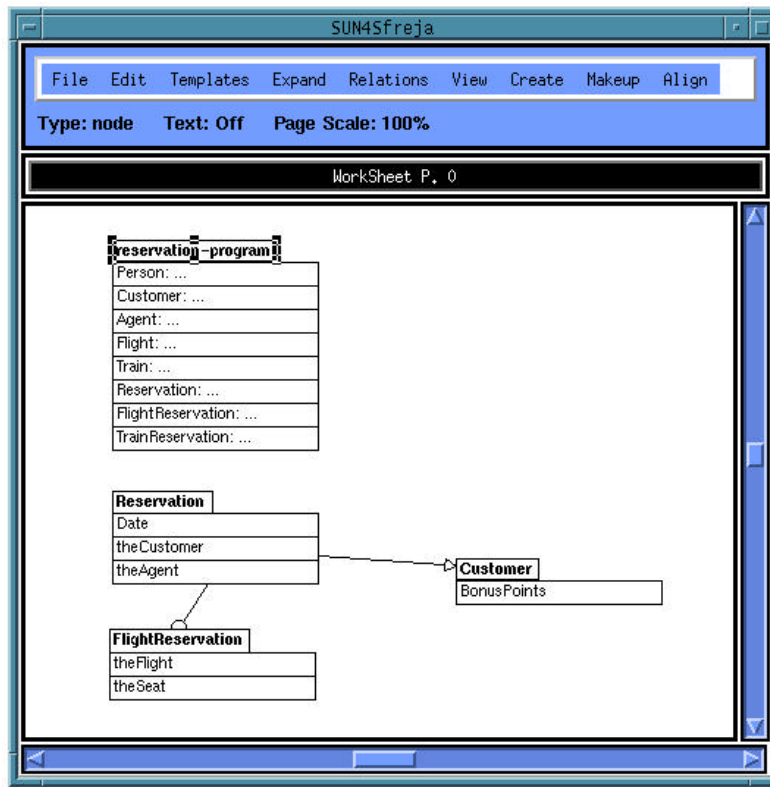
sTo specify that `FlightReservation` is a subpattern of `Reservation` the **Specialization** command of the **Relations Menu** is used. This command makes it possible to drag an inheritance connector from the subpattern title to any node in the super pattern:



4.2.6 Adding Object References

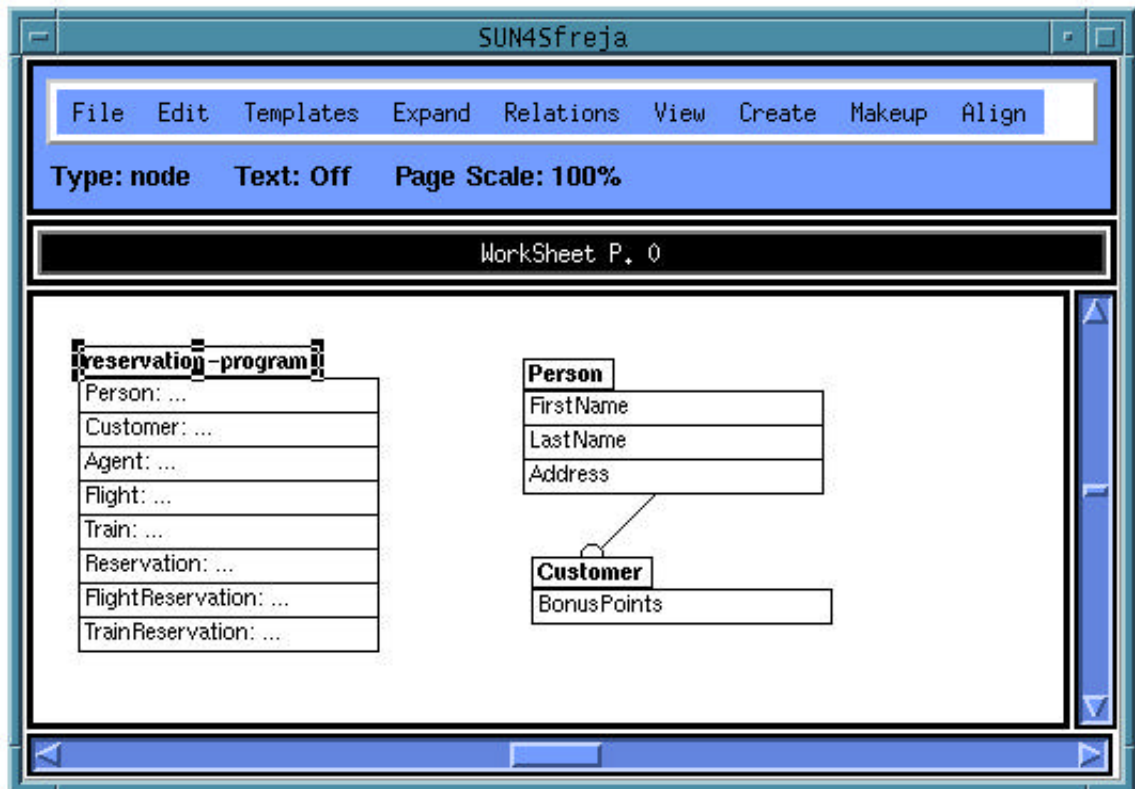


To specify that `theCustomer` is an object reference to a `Customer` pattern, the **Dynamic Reference** command of the **Relations Menu** is used. This command makes it possible to drag a dynamic reference arrow from a static or dynamic reference declaration to a pattern declaration diagram or node:



4.2.7 Completing the Code in Sif

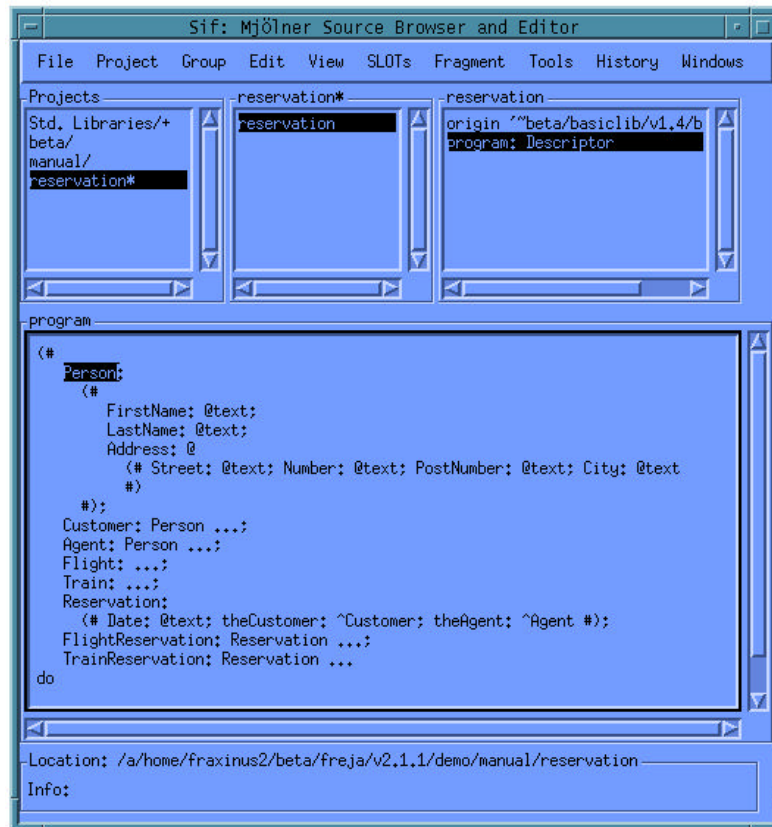
Consider the following design diagram which could be considered as complete at the design level:



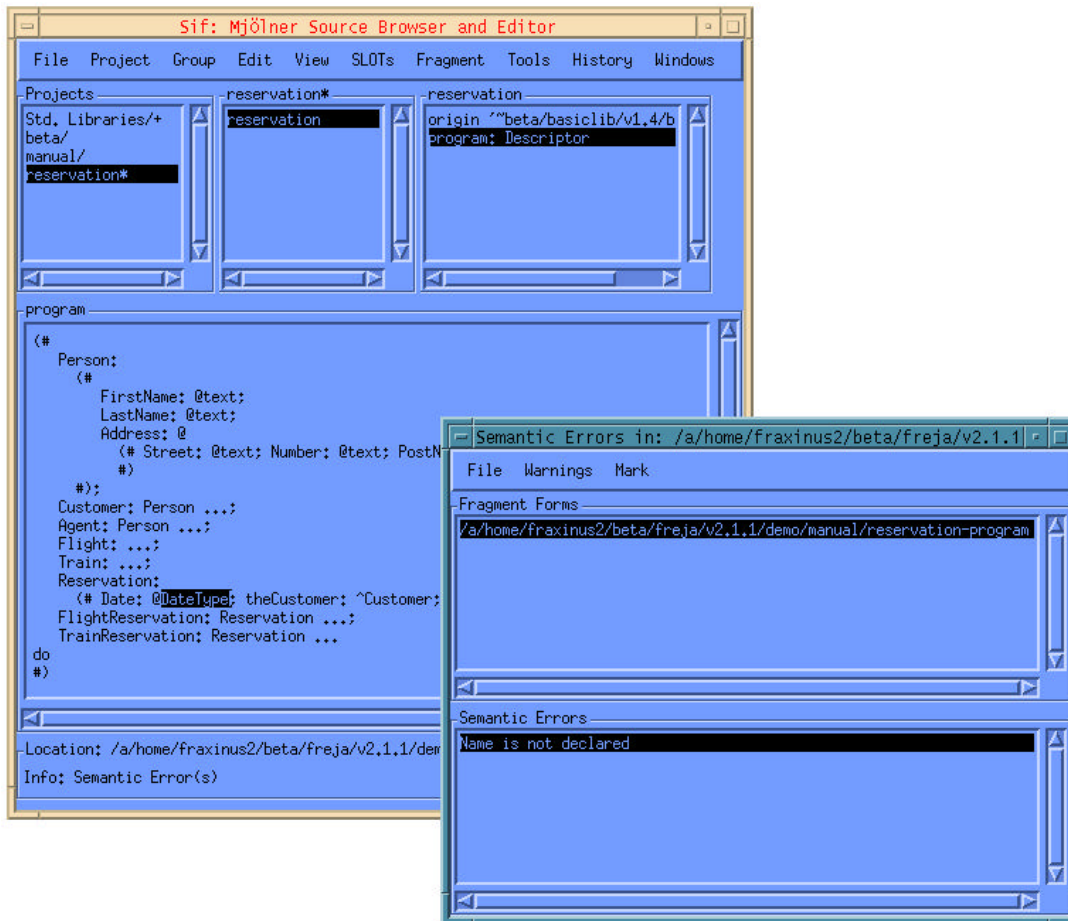
To fill out the code details, Sif, the textual representation editor must be activated, if not already visible:



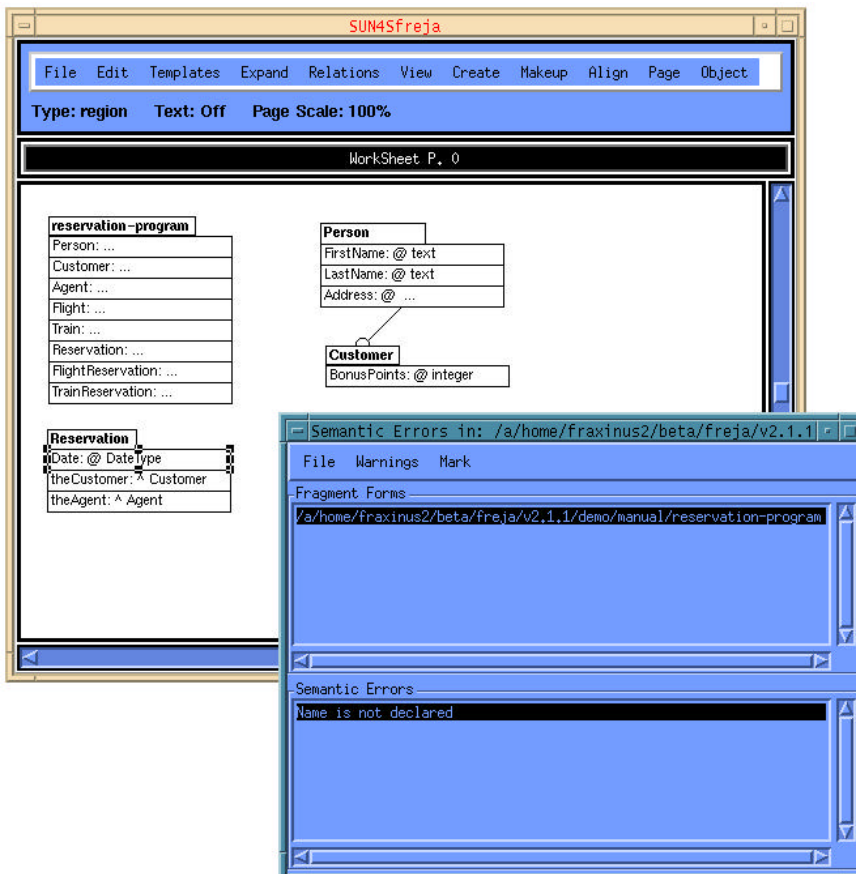
The textual representation of the design is, however, not complete. It still contains unexpanded placeholders. To get rid of these the **Remove Optionals** command is activated in the **Edit Menu** of either Freja or Sif:



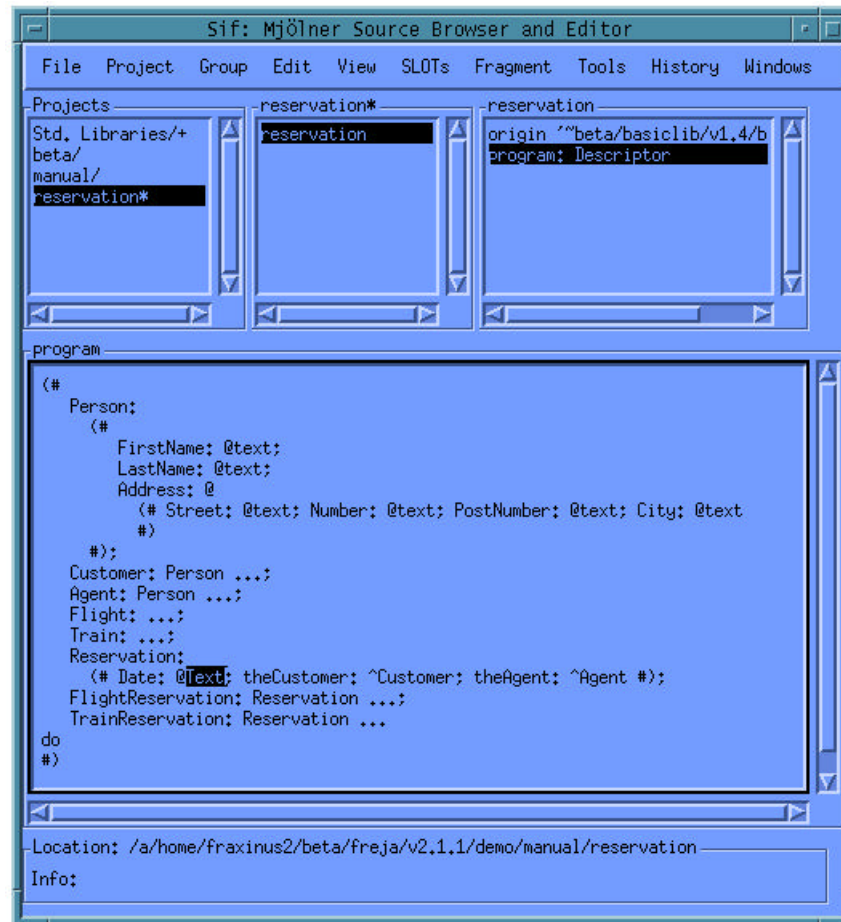
In this case all placeholders disappear, but this is not always the case. E.g. a name of a pattern declaration must always be filled out. After filling out more code in Sif, e.g. the action sequences of the objects (the Do-parts) (not shown below), the compiler is activated in the **Tools Menu** of Sif (or alternatively, if one only wishes to check the program for semantic errors, the checker can be activated using the **Check** entry of Freja's **Edit Menu**):



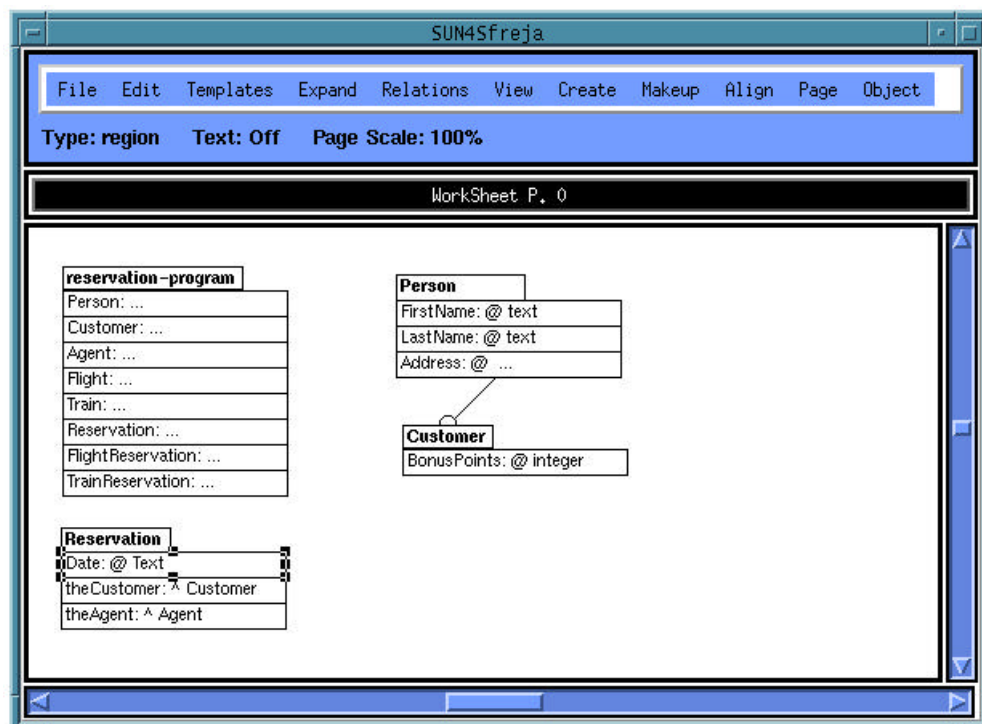
The compiler reports a semantic error, and the corresponding node is also selected in Freja:



The `DateType` is not declared. Instead of declaring it, the predefined `Text` pattern is entered in Sif:



This change is immediately reflected in Freja:



After a number of corrections the program is semantically correct and can be executed and tested. In this phase Freja will typically not be activated. When the program

has been tested and considered fulfilling the requirements, the design diagrams may have become obsolete, and need to be updated, i.e. reverse engineering is necessary. This is done by activating Freja on the BETA program.

4.3 Reverse Engineering

4.3.1 Pattern Diagrams

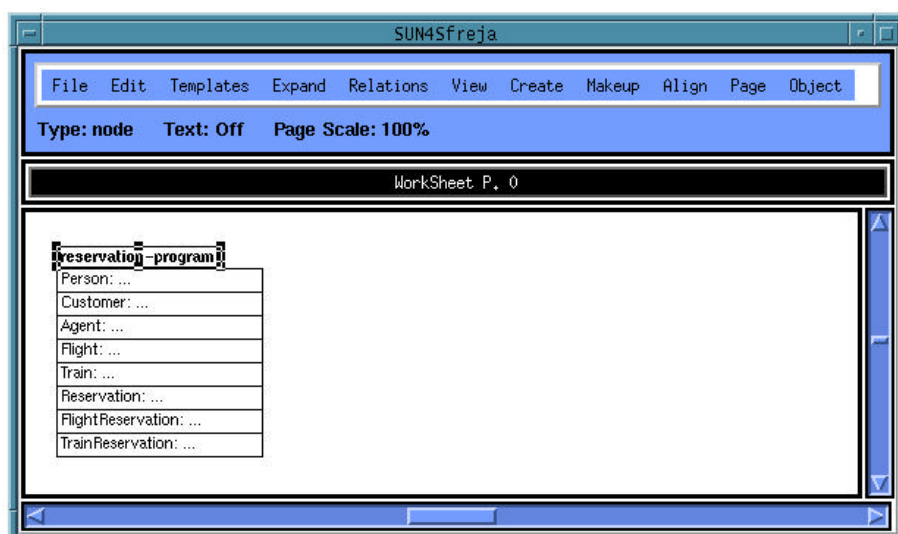
To extract the design diagrams from the BETA code, the program is opened in Freja. Consider the following program shown in Sif:

```

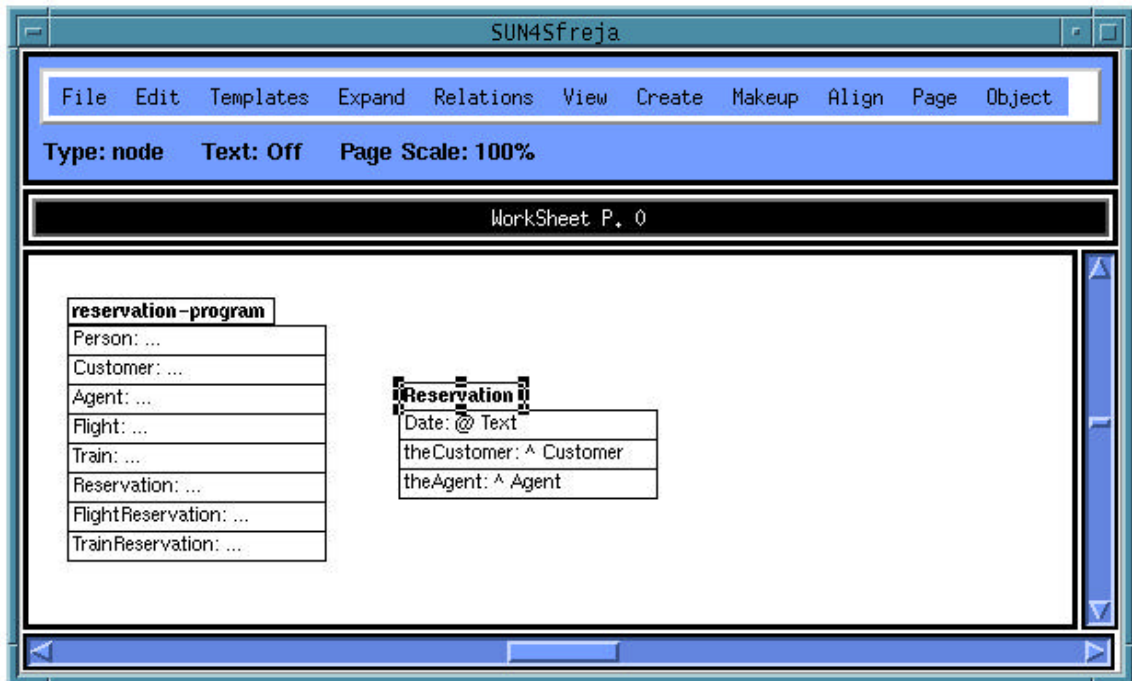
Sif: Mjölner Source Browser and Editor
File Project Group Edit View SLOTS Fragment Tools History Windows
Projects reservation* reservation
Std. Libraries/+ beta/ manual/ reservation*
reservation* reservation
origin "~/beta/basiclib/v1.4/b
program; Descriptor
program
(<#
  Person: (<# FirstName: @text; LastName: @text; Address: @ ... #);
  Customer: Person (<# BonusPoints: @integer #);
  Agent: Person ...;
  Flight: ...;
  Train: ...;
  Reservation:
    (<# Date: @Text; theCustomer: ^Customer; theAgent: ^Agent #);
  FlightReservation: Reservation (<# theFlight: ^Flight; theSeat: @text #);
  TrainReservation: Reservation ...
do
#)
Location: /a/home/fraxinus2/beta/freja/v2.1.1/demo/manual/reservation
Info:

```

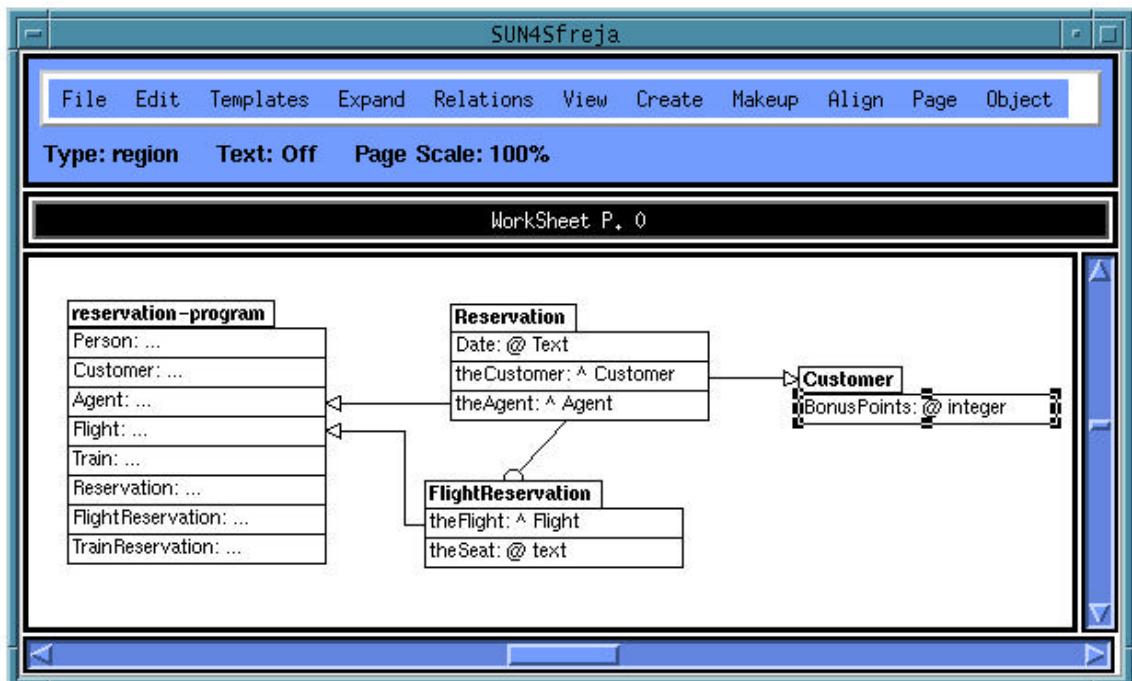
When opening this program in Freja the following diagram initially appears:



This is an abstract design view of the program. More detailed design diagrams can be generated by detailing selected parts of the diagrams. If a diagram node contains three dots (...) it can be detailed. Below the pattern `Reservation` is detailed by double-clicking on the `Reservation` node:

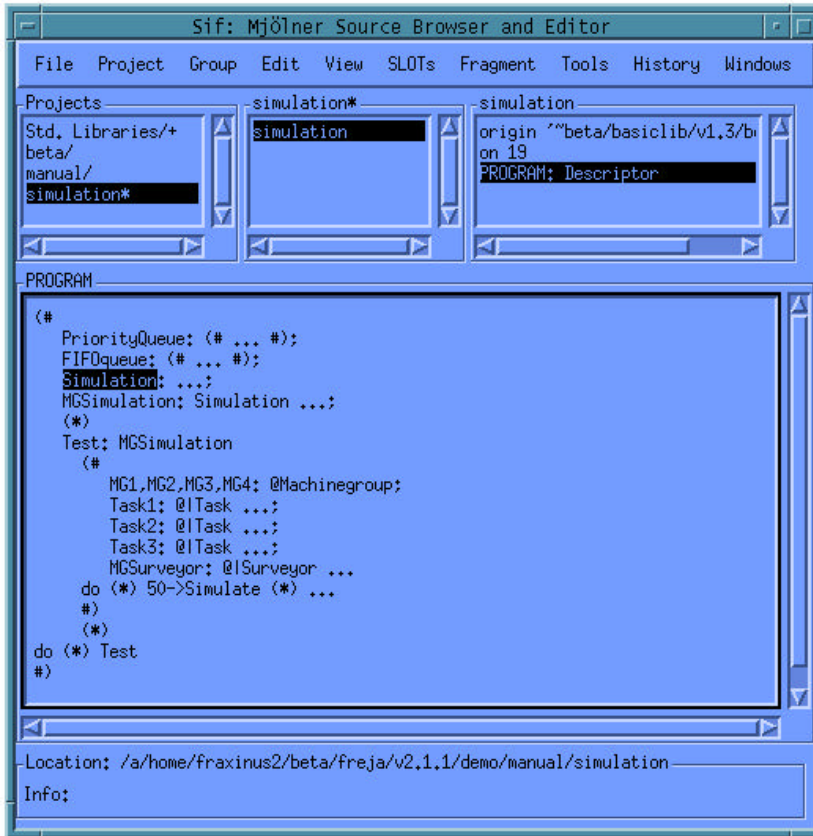


If a detailed diagram is related to other visible diagrams, possible relations like specialization or dynamic reference is automatically shown (if the program has been checked for semantic errors):

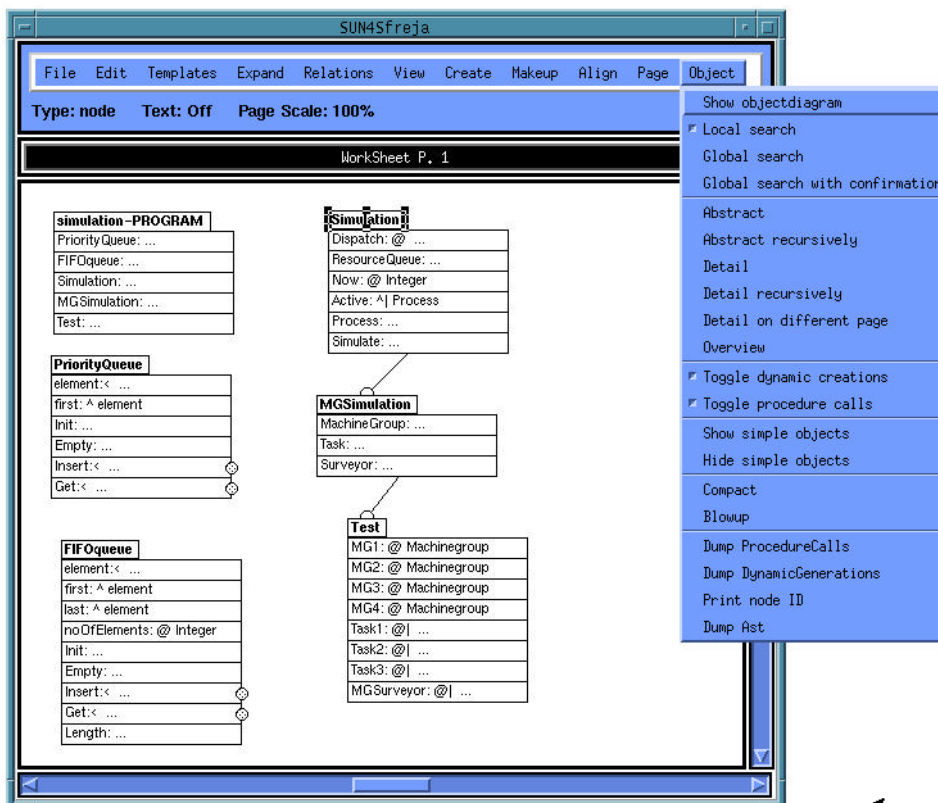


4.3.2 Object Diagrams

Object diagrams are generated in the same dynamic way as when generating the pattern diagrams. The reason for not detailing all the diagrams automatically is that, a fully detailed diagram often will evolve into something to big and to difficult to overview. Consider the following BETA program shown in Sif.



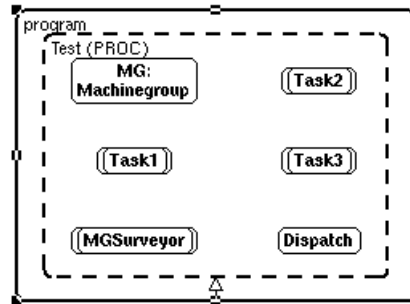
The corresponding pattern diagram is shown in Freja:



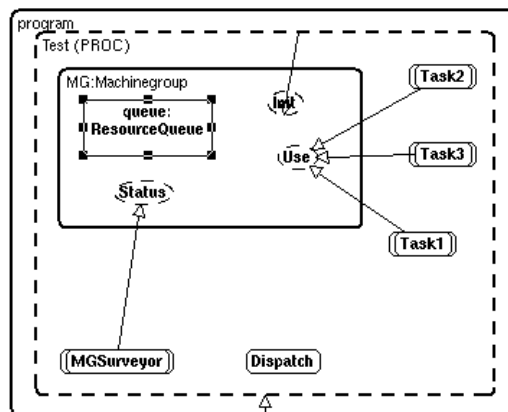
To get an object diagram for this program the **Show Object Diagram** command is used in the **Object Menu**, and a new window, a so-called object page is opened, containing:



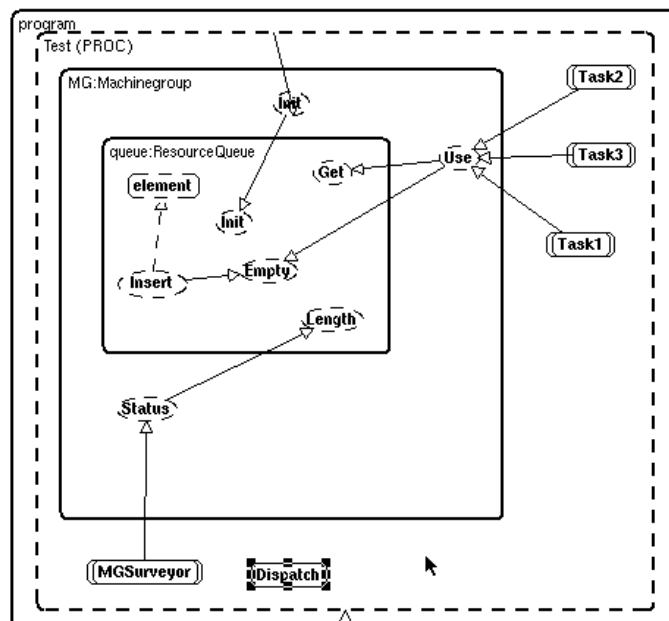
Which tells that the do part of the object program calls a procedure `test`. Double-clicking on `test` shows the local objects of the procedure:



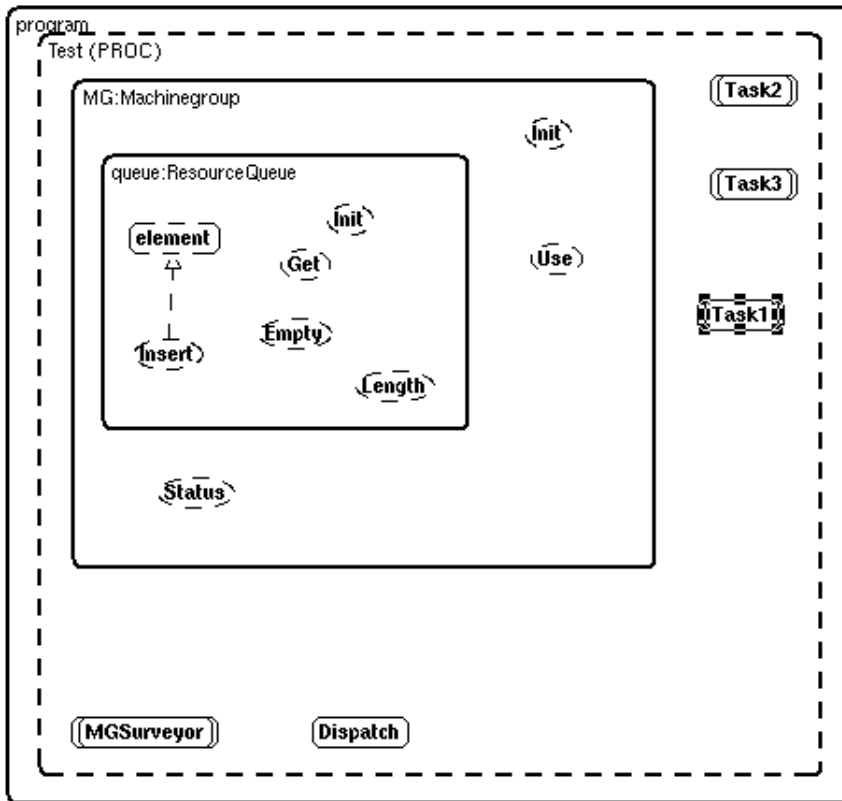
It contains 4 active objects and 2 static objects. Detailing `MG` gives:



Now we can, among other things, see that `MGSurveyor` calls the local procedure `status` of `MG`. Detailing `queue` gives an even more detailed picture:



We can see that `Insert` creates the dynamic object `element`, and that `MGSurveyor` calls `Status` which in turn calls the `Length` procedure pattern of `queue`. If the number of arrows get to many, they can be filtered away. Below the procedure call arrows are hidden:



5 Reference Manual

5.1 How to Get Started

5.1.1 Pattern Diagrams

Freja is started from an xterm in one of the following ways:

```
freja
freja foo.bet
freja foo.ast
freja foo.diag
freja foo
```

Using the first option only a menu bar will be presented to the user. How to continue from here: see the **File Menu** section.

The second and the third option opens a BETA fragment (foo) and the fourth opens a saved diagram. All three of these situations are described in further detail under **Open...** in the **File Menu** section. The last option checks the timestamps of the files on disk and opens whichever is newest.

5.1.2 Object Diagrams

Object diagrams can be generated for any BETA program that has been checked. If the BETA program has not been checked or contains semantic errors object diagrams **cannot** be generated.

To activate the object diagrams you first have to activate the pattern diagram part of Freja. Having done this one or more work sheets containing pattern diagrams will be open. To generate an object diagram you select the title-node of a pattern diagram shown on a work sheet and choose **Show Object Diagram** from the **Object** menu. The result of this will be a new page ("**Object Page**") with a box whose title corresponds to the chosen pattern diagram. The only exception here is that you cannot generate an object diagram for a pattern diagram that corresponds to a BETA library. In that case you will have to detail one of the entries of this pattern diagram and then go forth as described above.

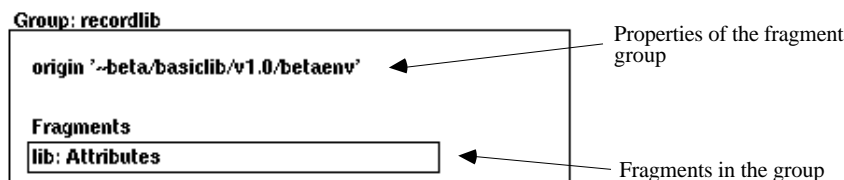
5.2 Work Sheets

The page title of a work sheet is on the form "**WorkSheet p. n**", where n is the page number. A work sheet is a page containing pattern diagrams. The pattern diagrams on a single work sheet is a reflection of the patterns contained in one or more BETA fragments. An example of a work sheet displaying the pattern diagram of a BETA program called foo.bet :



5.3 The Group Page

If one or more work sheets with pattern diagrams are open choosing **Show Group Window** from the Pages menu a page called “**Group Window**” will be opened. This page will contain one or more *group diagrams*.



Group Diagram

The number of group diagrams in the Group Window corresponds to the number of currently open BETA fragment groups.

The Group Diagram displays the properties and the fragments defined in the group. The properties and the fragments can be detailed, i.e. shown in detail, by selecting a node and then select **Detail** in the **View** menu. The detailing can also be performed by double-clicking on the nodes. By e.g. double-clicking on the `lib: Attributes` node, the declarations in the fragment will be presented as a pattern diagram in on a work sheet (if it is already detailed, the detailed version will become current focus). Doubleclicking `Origin`, `Include` or `Body` fields in a group diagram will open that group and bring up a group diagram for it.

5.4 Object Pages

The page title of an object page is on the form “**Object Page p. n**”, where *n* is the page number. An object page is a page containing an object diagram. The object diagram on an object page is a reflection of the dynamic aspects of a BETA program. Object diagrams illustrate static and dynamic objects, composition, dynamic object creation and procedure calls.

The box that is initially shown on the object page is an entity whose contents corresponds to the objects contained in the object descriptor of the pattern that was chosen to be subject to object diagram generation (in the case where the chosen pattern diagram represents a BETA program-descriptor the object diagram will simply represent the objects contained in the program).

5.5 The Menu Bar



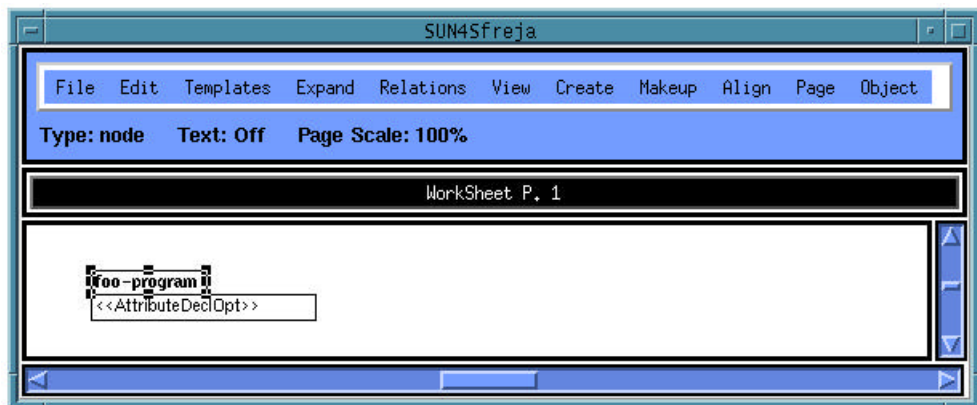
The majority of the commands in the menus operate on the *active page*. A page is made active either by clicking in the contents of the page or by clicking in the field containing the title of the page. This field is shown in reverse video when the page is active.

Many of the commands of the menus also operate on the current selection of the active page. The current selection is called *current focus*.

5.5.1 File Menu

New BETA program

Selecting this command will make a dialog popup. Entering a file name without extension and clicking **ok** will result in the creation of a minimal BETA program, which is given the entered file name. The immediate result of this will be a pattern diagram showing up on the active work sheet:



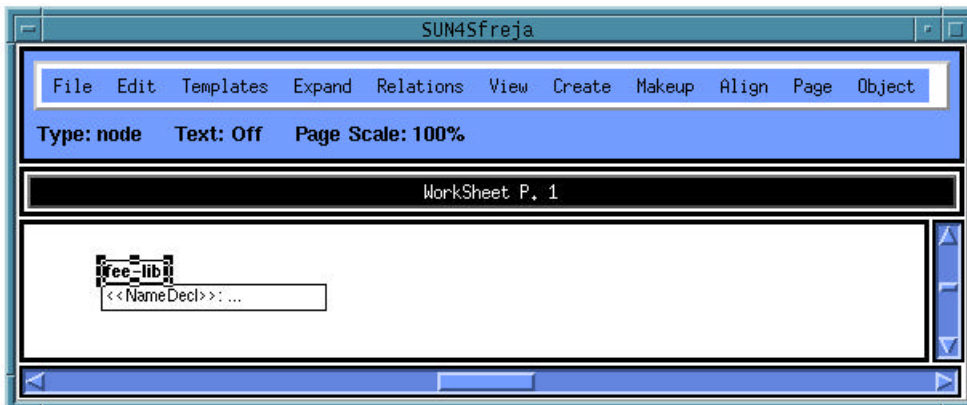
The title of the pattern diagram is the name of the new fragment group (the file name: `foo`) followed by a dash (-) and the name (`program`) of the new BETA fragment form. The single attribute shown below the title corresponds to the single (unexpanded) attribute of the descriptor of the program:

```
ORIGIN '~beta/basiclib/v1.4/betaenv'
-- program: DescriptorForm --
<<PrefixOpt>> (# <<AttributeDeclOpt>> do <<ImpOpt>> #)
```

Notice:

- Pattern diagrams displaying the descriptor of a program, like the one mentioned above, are called *descriptor diagrams*.

New BETA library



Makes a dialog popup. Entering a file name without extension and clicking **ok** will result in the creation of a minimal BETA library, which is given the entered file name. The result of this will be a pattern diagram, like the one shown above, showing up on the active work sheet. The title of the pattern diagram is the name of the new BETA fragment group (the file name: `fee`) followed by a dash (-) and the name of the new BETA fragment form (`lib`). The single attribute shown below the title corresponds to the single pattern attribute of the library:

```

ORIGIN '~beta/basiclib/v1.4/betaenv'
-- lib: Attributes --
<<NameDecl>>: <<PrefixOpt>>
(# <<AttributeDeclOpt>>
  <<EnterPartOpt>>
  <<DoPartOpt>>
  <<ExitPartOpt>>
#)

```

Notice:

- Pattern diagrams displaying the attributes of a library, like the one mentioned above, are called *attributes diagrams*.

A general note:

The common name for descriptor and attributes diagrams is *fragment diagrams*.

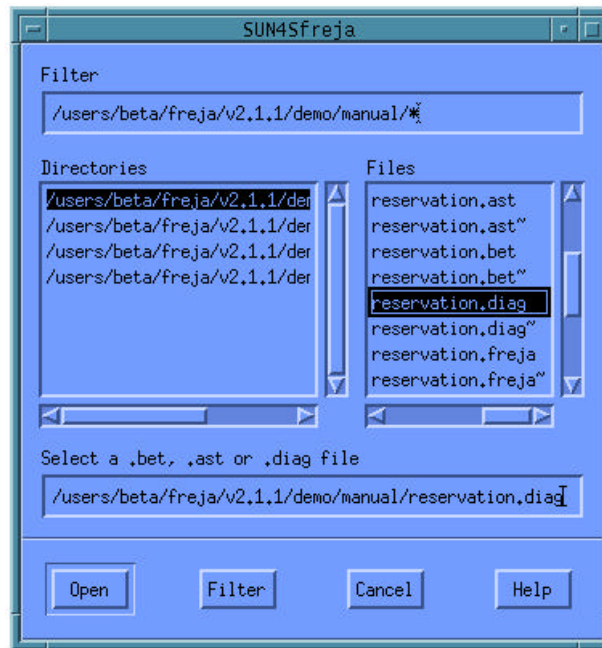
New Graphics Page...

If other pages are already open invoking this command will create a new page entitled “**Graphics Page p. n**”. If no other pages are open at the time where the command is invoked two pages will be created - a graphics page and a group page.

Graphics pages are as the name implies meant for graphics only. That is all objects appearing on these pages are interpreted as having no other semantics than the purely graphical one. Basically the commands that apply on these pages are the purely graphical ones that can be found in the **Create**, **Makeup** and **Align** menus. Also **Cut**, **Copy**, **Paste** and **Clear** from the **Edit** menu applies to single objects, regions and groups of objects on these pages.

As a special case it should be mentioned that editing pattern and object diagrams as graphics only (i.e. editing the diagrams arbitrarily without this having any consequences on the underlying code) can be done simply by copying such a diagram from a work sheet or an object page and afterwards pasting it onto a graphics page (for an alternative way of editing these diagrams as pure graphics see **Load Graphics...**).

Open...



Selecting **Open...** makes the open dialog popup. From here a .bet, .ast file or .diag file can be selected and opened.

Selecting a .bet or .ast file will, depending on the category of the selected fragment, result in either a descriptor or attributes diagram on the active worksheet. This pattern diagram reflects the attributes of the opened fragment. One exception from this should be noted: When a file containing more than one fragmentform is opened the first thing to show up will be the group page with a diagram displaying the fragmentforms of the selected file (see section: 'The Group Page', below). Now, by doubleclicking one of these fragmentforms its attributes can be shown on the active worksheet as described above.

Selecting a .diag file will open pages containing pattern diagrams corresponding to the situation in Freja when a save was last performed (see **Save** and **Save As...** below). This also has the effect that all the BETA fragments related to the pattern diagrams shown on these pages are being opened. That is, if the pattern diagrams on the pages of the opened .diag file are edited and saved hereafter, the related fragments will be edited and saved correspondingly.

Notice:

- A BETA fragment is said to be *related* to a pattern diagram if the pattern diagram reflects attributes declared in this fragment.
- If the fragments related to a saved .diag file are being edited and saved outside the scope of Freja the .diag file becomes inconsistent with these and therefore the .diag file can no longer be opened. However the only data lost this way will be the layout of the diagrams on the pages, as new diagrams, reflecting the changes, can be generated through opening the .bet or .ast files anew (here-through performing reverse engineering).
- Only one .diag file can be open at the time, but the different pattern diagrams on the work sheets can be related to several different BETA fragments.

Close

Closes all open pages in Freja and the fragments related to the pattern diagrams shown on these pages.

If any editing of the diagrams has been done since the last save the user is asked if he wishes to save these changes. Answering yes to this question results in a save to the currently open .diag file.

Save

Performs a save of all shown diagrams on all open pages to the currently open .diag file.

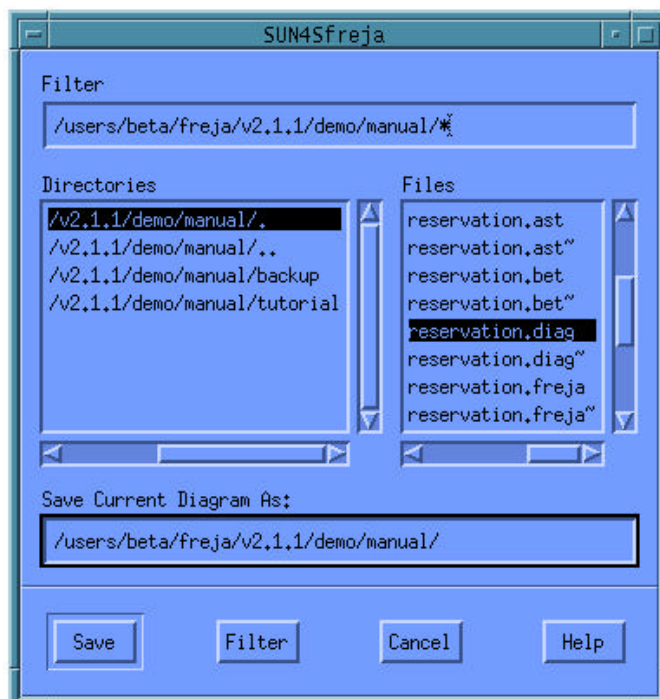
This means that the layout of the diagrams on the pages are saved, and that the fragments related to the diagrams are saved to their .bet and .ast files. The .diag file can then later be opened for further browsing and editing and eventually be saved again.

Notice:

- The default name for the diagram is the name of the BETA fragment that was opened or created first during the session. To change this name use **Save As** (see below).

Save As...

Using this command invokes the dialog:



Entering a name in the text field of the dialog with extension .diag and choosing Save will result in a save of all shown diagrams on all open pages to this file (as described above).

Revert

Reverts all edits done during the current Freja session. I.e. the diagram will appear as it did when it was initially opened.

Page Setup

Use **Page Setup** before printing. To get a satisfactory print the recommended settings are:

- **Paper: A4 letter.**
- **Output form: Scale to fit.**
- **Omit page borders.**

Print...

Initiating the print command will bring up a dialog asking the user to indicate which pages are to be printed. If all pages are to be printed simply select the **All** field of the dialog.

Having selected the **ok** button of the dialog the following string will appear in the xterm from which Freja was originally started: "Enter name of PostScript

file (without .ps): “. Simply do so and both a PostScript and an Encapsulated PostScript version of the print will be available in the directory where Freja was started.

To this purpose the Freja script sets the environment variable "DesignPrintCommand" to "frejaprinter" if it does not already have a value. Alternatively the "DesignPrintCommand" can be used to specify that Freja is to print directly on a given printer. This can e.g be done like this:

```
>setenv DesignPrintCommand "lpr -Pr312"
```

Where `lpr -Pr312` is the command for printing on a printer called `r312`.

Preferences...

Brings up a dialog for manipulating general settings of the program.

Load Text...

Is used for loading text from a file on disc into a new box node in the diagram.

Save Text...

Saves the text contained in the node that is current focus onto a specified file on disc.

Load Graphics...

Makes a dialog popup from which you can choose a file on disc that contains pages that has been saved as graphics form within Freja at an earlier time. These pages can now be edited as graphics pages and saved again as pure graphics (see e.g. **New Graphics Page...**).

Also if in an earlier session pattern and/or object diagrams where created and saved together with the code, these diagrams can at a later time be loaded into Freja as graphics only (i.e. without the code). This is done by inoking this menu item and selecting the .diag file (the graphics) that was saved earlier together with the .ast file (the code).

Notice:

- Having loaded a .diag file into Freja as graphics only it is generally not recommended to save this graphics under the *same* name again - i.e. as a .diag file. This will cause the .diag file to become *inconsistent* with the code saved in the .ast file. That is reverse engineering has to be performed before code and diagram can be edited simultaneously again.

Save Page As Graphics...

Using this command you can, using the dialog that pops up, save the current page on disc as graphics only.

About

Will bring up a dialog displaying general information (version no. etc.) about the running Freja.

Help

Brings up a version of this document.

Show Sif / Hide Sif

This menu item initially contains the text **Show Sif** and by invoking it the Mjølner BETA Source Browser and Editor, Sif, will appear on the screen. In the case where a pattern diagram is currently being edited in Freja, Sif will appear with focus on the corresponding piece of code. The code can now be edited in Sif as well as through Freja and the changes will constantly be reflected in the other tool.

After invoking **Show Sif** the menu item will contain the text **Hide Sif** which can now be invoked to make Sif disappear from screen again.

Quit

Performs a **Close** (see above) and quits the application.

5.5.2 Edit Menu

Undo

Work sheets:

Choosing **Undo** from the **Edit Menu** will undo the latest editing operation on a work sheet in Freja. If this operation itself was **Undo** the state before the **Undo** will be restored.

The operations which can be undone are **Cut**, **Copy**, **Paste**, **Insert Before**, **Insert After**, **Paste Before**, **Paste After**, **Expand**.

Cut

In the general case **Cut** will remove current focus and put it on the graphical clipboard as a graphics only object. Also groups and regions of objects can be cut, but not groups that contain both graphics only objects and parts of pattern diagrams. See **Paste** for further explanation of when the graphics only clipboard is used and when code is also being pasted.

Work sheets:

Initiating **Cut** removes the contents of current focus on the active work sheet. The deleted part of the pattern diagram is moved onto a clipboard and can later be pasted into any other pattern diagram. Only titles and attributes can be cut from a pattern diagram (not any of the relations). Performing a **Cut** with a title of a pattern diagram as current focus results in the removal of the entire pattern diagram and the removal of the attribute of which this diagram is a detailed version. Notice that this type of cut will also cut the corresponding code.

Object Pages:

Only cutting of graphics only objects, regions and groups is currently implemented on object pages.

Copy

In the general case **Copy** will put current focus on the graphical clipboard as a graphics only object. Also groups and regions of objects can be cut. See **Paste** for further explanation of when the graphics only clipboard is used and when code is also being pasted.

Work sheets:

Copies the contents of current focus on the active work sheet onto the clipboard and can later be pasted into any other pattern diagram. Only titles and attributes can be copied from a pattern diagram (not any of the relations).

Paste Before

Work sheets:

When current focus is an attribute of a pattern diagram this command performs an **Insert Before** followed by a **Paste**.

Paste

Work sheets:

If current focus is part of a pattern diagram - i.e. a title or an attribute - and if the clipboard contains part of a pattern diagram, then **Paste** can only be performed when the contents of current focus and the pattern diagram part on

the clipboard are of exchangeable syntactic categories. If this is the case and a **Paste** is performed the effect is that the contents of current focus is replaced by the pattern diagram part on the clipboard and the code is updated correspondingly.

In this context when cutting or copying with a title of a pattern diagram as current focus the syntactic category is the same as the syntactic category of the attribute of which the pattern diagram is a detailed version.

If the clipboard contains a graphics only object this object is simply pasted as such onto the work sheet.

Object pages:

On object pages only paste of pure graphics can be performed.

Graphics Pages:

If the clipboard contains a graphics only object this is pasted as such onto the graphics page.

If the clipboard contains part of a pattern diagram the purely graphical part of this will be pasted onto the graphics page.

Paste After

When current focus is an attribute of a pattern diagram this command performs an **Insert After** followed by a **Paste**.

Clear

Removes the contents of current focus on the active work sheet (nothing is moved to the clipboard).

Edit Name

If current focus is an attribute of a pattern diagram or the title of a diagram detailed from such an attribute, this command will invoke a dialog in which the name of the attribute can be entered. The entered name must be a lexem (i.e. some legal name in BETA). Therefore before accepting it, the name is parsed and if illegal the dialog will ask the to change it.

Open Subeditor

If current focus is an attribute of a pattern diagram or the title of a diagram detailed from such an attribute, this command will invoke a so-called *subeditor* on the code that corresponds to current focus. The functionality of a subeditor is mainly identical to the functionality of the codeviewer/-editor of the browser window of Sif. Subeditors are therefore typically used when textual structure editing on some part of the code is called for.

Insert Before

If the contents of current focus is an attribute of a pattern diagram this command will insert a new unexpanded attribute before current focus.

Insert After

If the contents of current focus is an attribute of a pattern diagram this command will insert a new unexpanded attribute after current focus.

Remove Optionals

If current focus is part of a pattern diagram this command removes all nonterminals representing optional productions from the contents of current focus.

Show Optionals

Inserts all nonterminals representing optional productions in current focus, if it is part of a pattern diagram.

Check

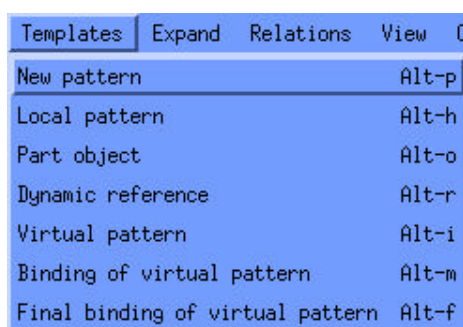
Invokes the semantic checker on the code related to the pattern diagram which is current focus.

If any semantic errors are detected in the code it is reported in a “Semantic Error” dialog. Selecting an error in this dialog will make the attribute in Freja, referring to the code containing the error, become current focus.

If no errors are detected during checking all drawn relations, that have somehow become invalid using e.g text editing, are removed and the ones, that were not displayed before the check, are now drawn.

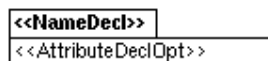
5.5.3 Templates Menu

In general the templates menu contains entries that are composed of two or more ordinary commands and are provided to make some of the more frequently used series of commands easier to perform.



New Pattern

This command basically creates a new pattern and makes a corresponding pattern diagram show up on the active work sheet. A newly created pattern like this consists only of unexpanded parts and is ready to be filled in with a name and further attributes:



In further detail the command is a combination of:

- an **Insert After** which creates a new unexpanded attribute,
- an **Expand** of this new attribute into a pattern declaration,
- and finally a **Detail** of the pattern declaration.

(for explanations of **Expand** and **Detail** see below)

The initial **Insert After**, which of course decides where the new pattern attribute is declared, is performed on the bottom most attribute of what is called the *current fragment diagram*.

If the current focus is a title or an attribute of a pattern diagram P the current fragment diagram is the fragment diagram that is related to the same fragment as the diagram P.

Local Pattern

Like the **New Pattern** command this command creates a new pattern and makes a corresponding pattern diagram show up on the active work sheet. Also in the same way this command is a combination of **Insert After**, **Expand** and **Detail**. Only difference is that **Local Pattern** will insert the new pattern attribute in the diagram where current focus is set. That is, the pattern diagram which includes the title or attribute which is current focus. This diagram is called the *current diagram*.

Part Object

Part Object is a combination of **Insert After** and **Expand** and will insert a new static item in the current diagram.

Dynamic Reference

Dynamic Reference is a combination of **Insert After** and **Expand** and will insert a new dynamic item in the current diagram.

Virtual Pattern

Virtual Pattern is a combination of **Insert After** and **Expand** and will in the current diagram insert a new virtual pattern definition on the form:

```
<<NameDecl>>:< <<PrefixOpt>>
  (# <<AttributeDeclOpt>>
    <<EnterPartOpt>>
    <<DoPartOpt>>
    <<ExitPartOpt>>
  #)
```

Binding of Virtual Pattern

Binding of Virtual Pattern is a combination of **Insert After** and **Expand** and will in the current diagram insert a new virtual pattern definition on the form:

```
<<NameDecl>>::< <<PrefixOpt>>
  (# <<AttributeDeclOpt>>
    <<EnterPartOpt>>
    <<DoPartOpt>>
    <<ExitPartOpt>>
  #)
```

Final Binding of Virtual Pattern

Final Binding of Virtual Pattern is a combination of **Insert After** and **Expand** and will in the current diagram insert a new virtual pattern definition on the form:

```
<<NameDecl>>:: <<PrefixOpt>>
  (# <<AttributeDeclOpt>>
    <<EnterPartOpt>>
    <<DoPartOpt>>
    <<ExitPartOpt>>
  #)
```

5.5.4 Expand Menu

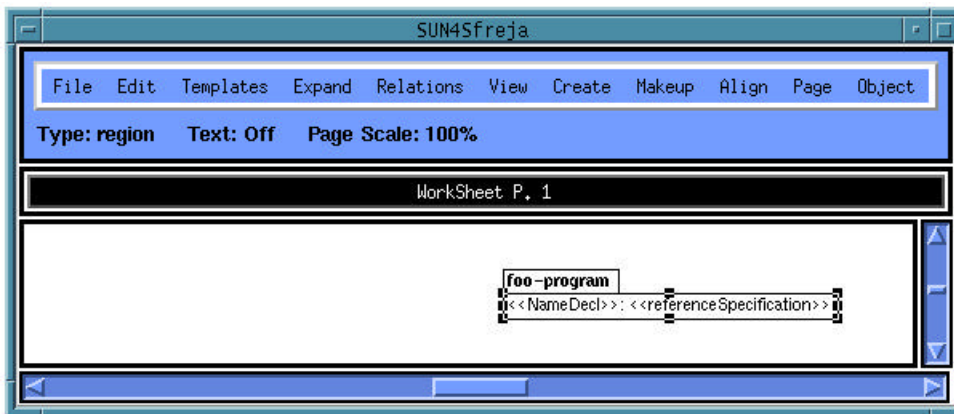
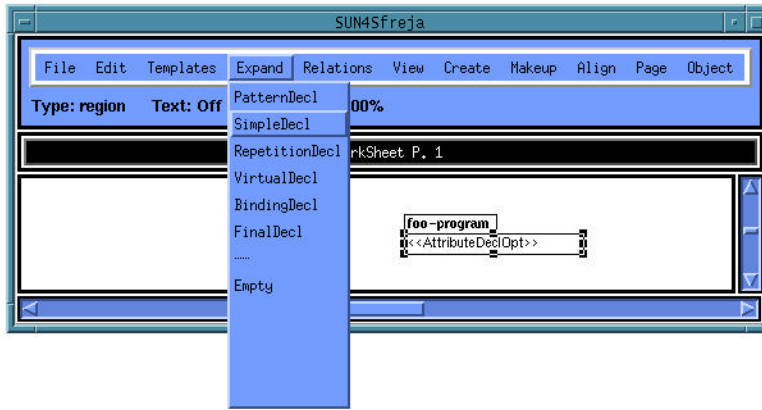
The content of the **Expand** menu is dependent on the current focus.

If current focus is anything else than an attribute of a pattern diagram it will be empty.

If current focus is an attribute of a pattern diagram the menu will contain an entry for each language construct that can replace a possible nonterminal in the attribute.

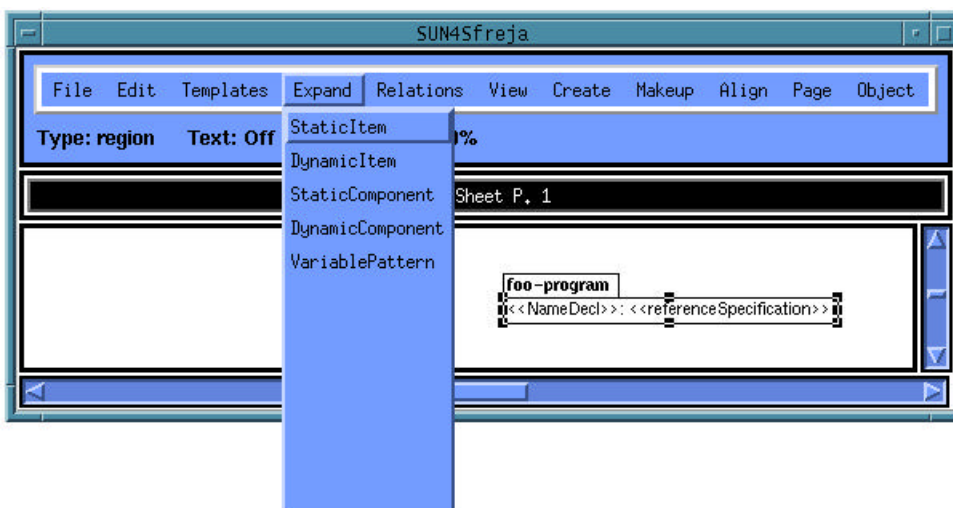
If the attribute that is current focus does not contain a nonterminal the **Expand** menu will be empty.

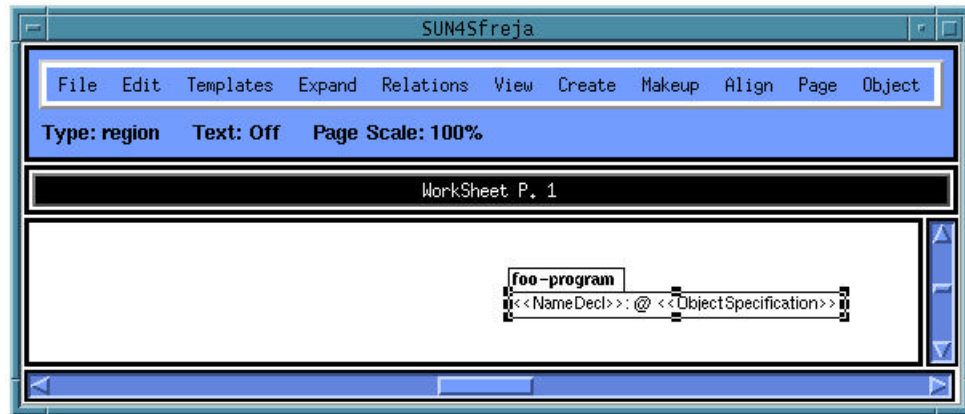
Having selected an attribute that does contain a nonterminal choosing one of the entries in the **expand** menu will result in the nonterminal being replaced with the chosen construct (see below). If the nonterminal in current focus is an optional or a list element the menu will also contain an **Empty** entry. Choosing this entry will remove the nonterminal.



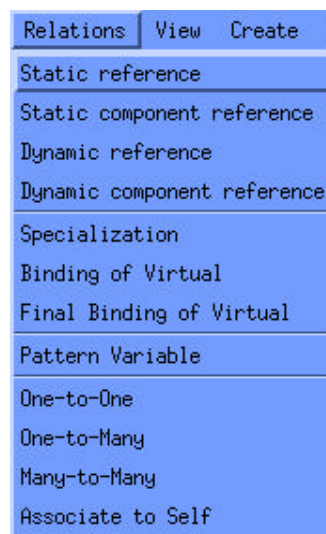
In some cases the attribute that is current focus will apparently contain more than one nonterminal as in the example shown below (<<NameDecl>> and <<ReferenceSpecification>>).

However in these cases the situation will always be that only one of the nonterminals can be replaced by a number of language constructs and all other present nonterminals will be so called *lexem nonterminals* that can not be further expanded. Therefore the Expand menu will at any time contain the language constructs which are possible replacements for the (single) replaceable nonterminal in the attribute (see below). In the example above <<NameDecl>> is a lexem nonterminal and the replaceable nonterminal determining the contents of the menu is <<ReferenceSpecification>>. To replace a lexem nonterminal like <<NameDecl>> use the **Edit Name** command of the **Edit** menu. Other lexem nonterminals can e.g. be replaced using a subeditor (see **Open Subeditor**) on the attribute containing the lexem.





5.5.5 Relations Menu



The commands of the **Relations** menu all relate one node to another. For this reason they are all basically used in the same manner:

- Choose the wanted relation,
- click down the mouse on the source of the relation and
- drag a connector to the destination of the relation.

Static Reference

In this case the source of the relation must either be a node containing an unexpanded attribute declaration (an `<<AttributeDeclOpt>>`) or some kind of simple declaration - like f. ex.:

```
<<NameDecl>>: <<ReferenceSpecification>>,
aReference: ^aPattern OR
anInstance: @<<ObjectSpecification>>
```

The destination must either be the title of a pattern diagram or an attribute containing a pattern declaration, virtual declaration, binding declaration or final binding declaration.

The result of creating a relation as such will be that the source attribute contains a static reference to the pattern identified by the destination.

If in the three examples mentioned above the destination had been a node containing a pattern with name DEST the contents of the source would become:

```

<<NameDecl>>: @DEST
aReference: @DEST
anInstance: @DEST

```

Notice:

- In the second of the above examples the creation of the **Static Reference** relation is used to change the source from being a dynamic reference to being a static reference.
- The dragged connector between source and destination will disappear when the relation has been created.

Static Component Reference

Behaves Exactly like the **Static Reference** relation except that using this entry will create static components in the source instead of static items.

Dynamic Reference

Using this command the constraints on source and destination are exactly the same as the constraints for two first mentioned relations.

The result of creating a **Dynamic Reference** relation will be that the source attribute contains a dynamic reference to the pattern identified by the destination.

Using the same sources and destinations as mentioned in the three examples of **Static Reference** the contents of the source would become:

```

<<NameDecl>>: ^DEST
aReference: ^DEST
anInstance: ^DEST

```

In this case the dragged connector will remain between the source and destination with an arrow pointing to the destination.

Notice:

- In the last of the above examples the creation of the **Dynamic Reference** relation is used to change the source from being a static reference to being a dynamic reference.

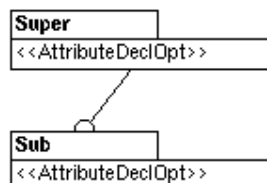
Dynamic Component Reference

Behaves Exactly like the **Dynamic Reference** relation except that using this entry will create dynamic components in the source instead of dynamic items.

Specialization

Creating this relation the source and destination must be detailed versions of either pattern declarations, virtual declarations, binding declarations or final binding declarations.

The result of creating a **Specialization** relation will be that the descriptor corresponding to the source diagram gets the destination pattern inserted as its prefix.



The corresponding BETA code before the operation:

```

Super: (# <<AttributeDeclOpt>> #);
Sub: <<PrefixOpt>> (# <<AttributeDeclOpt>> #)

```

and after:

```

Super: (# <<AttributeDeclOpt>> #);
Sub: Super (# <<AttributeDeclOpt>> #)

```

Binding of Virtual

For this relation the source must either be an attribute containing an unexpanded attribute declaration or an attribute containing a pattern, virtual or binding declaration. The destination must be an attribute containing a binding or virtual declaration.

Having created a relation like this the source will have become a binding of the virtual of the destination.

Notice:

- A possible descriptor of the source will not be lost through this operation.

Final Binding of Virtual

Behaves exactly like the **Binding of Virtual** relation except that the source in this case becomes a final binding of the virtual of the destination.

Variable Pattern

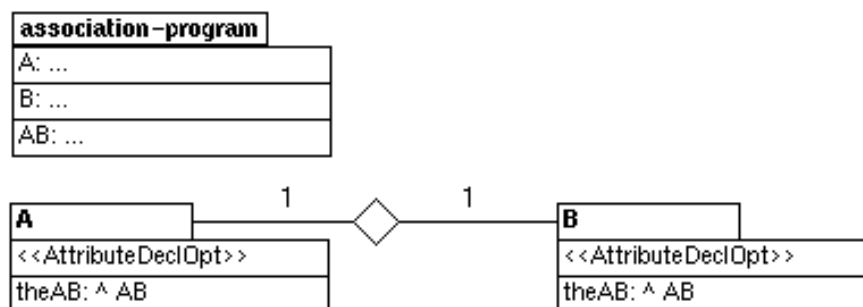
For the **Variable Pattern** relation the source must be an attribute containing either an unexpanded attribute declaration or some kind of simple declaration. The destination must either be the title of a diagram that is a detailed version of an attribute containing a pattern declaration or it must be an attribute containing a pattern declaration.

The result will be that the source becomes a declaration of a pattern variable with the destination pattern as qualification.

One-to-One Association

Creating this relation the source and destination must be detailed versions of either pattern declarations, virtual declarations, binding declarations or final binding declarations.

The result of creating a **One-to-One Association** relation will be that a subpattern of the pattern `OneToOneAssociation`¹ is created. In this pattern the virtual `leftType` will be bound to the source pattern and the virtual `rightType` will be bound to the destination pattern. Furthermore, both the source and the destination will get a new reference attribute qualified with the above mentioned subpattern of `OneToOneAssociation`.



The corresponding BETA code before the operation:

¹ `OneToOneAssociation` is declared in the library `associations.bet`, situated in the directory of the current version of Freja.

```
(#
  A: (# <<AttributeDeclOpt>> #);
  B: (# <<AttributeDeclOpt>> #)
#)
and after:
```

```
(#
  A: (#
    <<AttributeDeclOpt>>;
    theAB: ^AB
  #);
  B: (#
    <<AttributeDeclOpt>>;
    theAB: ^AB
  #);
  AB: OneToOneAssociation
    (#
      leftType::< A;
      rightType::< B
    #)
#)
```

The generated code can be used for generally associating an instance of one pattern to an instance of another pattern.

The superpattern `OneToOneAssociation` defines the references `left` and `right` qualified with virtuals `leftType` and `rightType`.

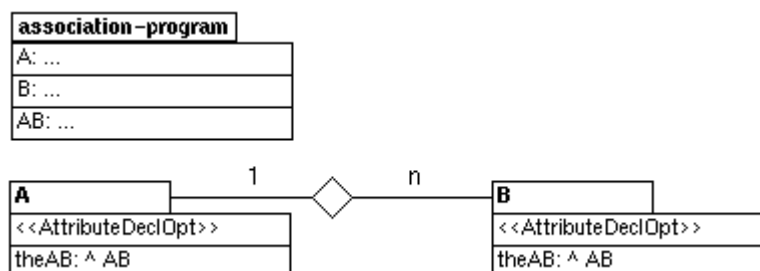
To associate two instances of patterns `A` and `B` in the example one would normally:

- Create an instance of pattern `AB`.
- Set reference `left` of the instance of `AB` to refer to the instance of `A`.
- Set reference `right` of the instance of `AB` to refer to the instance of `B`.
- Set references `theAB` of both the instance of `A` and the instance of `B` to refer to the instance of `AB`.

One-to-Many Association

Creating this relation the source and destination must be detailed versions of either pattern declarations, virtual declarations, binding declarations or final binding declarations.

The result of creating a **One-to-Many Association** relation will be that a subpattern of the pattern `OneToManyAssociation`² is created. In this pattern the virtual `oneType` will be bound to the source pattern and the virtual `manyElmType` will be bound to the destination pattern. Furthermore, both the source and the destination will get a new reference attribute qualified with the above mentioned subpattern of `OneToManyAssociation`.



² `OneToManyAssociation` is declared in the library `associations.bet`, situated in the directory of the current version of Freja.

The corresponding BETA code before the operation:

```
(#
  A: (# <<AttributeDeclOpt>> #);
  B: (# <<AttributeDeclOpt>> #)
#)
```

and after:

```
(#
  A: (#
    <<AttributeDeclOpt>>;
    theAB:^AB
  #);
  B: (#
    <<AttributeDeclOpt>>;
    theAB:^AB
  #);
  AB: OneToManyAssociation
    (#
      oneType::< A;
      ManyElmType::< B
    #)
#)
```

The generated code can be used for generally associating an instance of one pattern to one or more instances of another pattern.

The superpattern `OneToManyAssociation` is itself a subpattern of the container type `list` which implements linked lists. `OneToManyAssociation` defines a reference `one` qualified with the virtual `oneType` and defines the element type of the `list` to a pattern containing a reference `manyElm` qualified with virtual `manyElmType`.

To associate an instance of pattern `A` to a number of instances of `B` in the example one would normally:

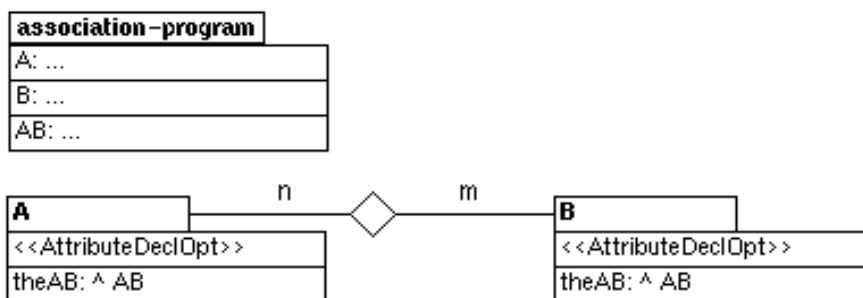
- Create an instance of pattern `AB`.
- Set reference `one` of the instance of `AB` to refer to the instance of `A`.
- Insert the instances of `B` in the instance `AB` using the `insert` pattern defined in `OneToManyAssociation` (remember that the instance of `AB` is inherently a list with the above mentioned element type)
- Set references `theAB` of both the instance of `A` and the instances of `B` to refer to the instance of `AB`.

Many-to-Many Association

Creating this relation the source and destination must be detailed versions of either pattern declarations, virtual declarations, binding declarations or final binding declarations.

The result of creating a **Many-To-Many Association** relation will be that a subpattern of the pattern `ManyToManyAssociation`³ is created. In this pattern the virtual `leftType` will be bound to the source pattern and the virtual `rightType` will be bound to the destination pattern. Furthermore, both the source and the destination will get a new reference attribute qualified with the above mentioned subpattern of `ManyToManyAssociation`.

³ `ManyToManyAssociation` is declared in the library `associations.bet`, situated in the directory of the current version of Freja.



The corresponding BETA code before the operation:

```

( #
  A: ( # <<AttributeDeclOpt>> # );
  B: ( # <<AttributeDeclOpt>> # )
# )

```

and after:

```

( #
  A: ( #
    <<AttributeDeclOpt>>;
    theAB: ^AB
  # );
  B: ( #
    <<AttributeDeclOpt>>;
    theAB: ^AB
  # );
  AB: ManyToManyAssociation
    ( #
      leftType::< A;
      rightType::< B
    # )
# )

```

The generated code can be used for generally associating one or more instances of one pattern to one or more instance of another pattern.

The superpattern `ManyToManyAssociation` is itself a subpattern of the container type `list` which implements linked lists. `ManyToManyAssociation` defines the element type of the `list` to a pattern containing a references `left` and `right` qualified with viruals `leftType` and `rightType`.

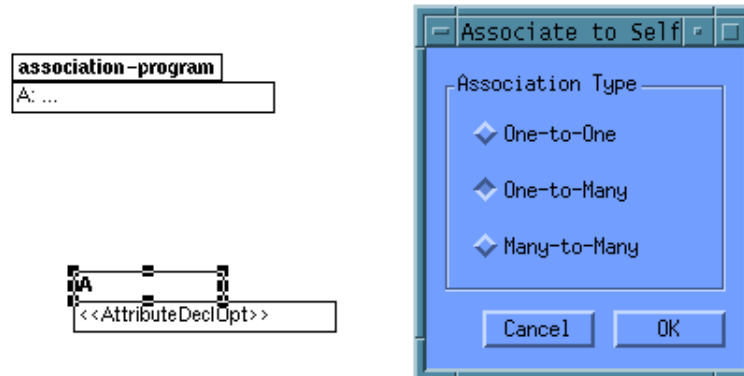
To associate a number of instances of `A` to a number of instances of `B` in the example one would normally:

- Create an instance of pattern `AB`.
- Insert the instances of `A` in the instance `AB` using the `insert` pattern defined in `ManyToManyAssociation` (remember that the intance of `AB` is inherently a list with the above mentioned element type)

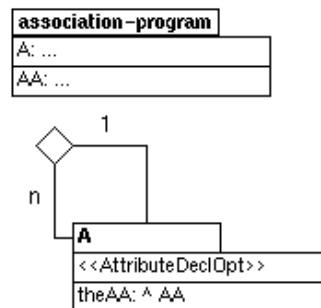
- Insert the instances of **B** in the instance **AB** using the `insert` pattern defined in `OneToManyAssociation` (remember that the instance of **AB** is inherently a list with the above mentioned element type)
- Set references `theAB` of both the instances of **A** and the instances of **B** to refer to the instance of **AB**.

Associate to Self

Creating this relation current focus must be a detailed version of either a pattern declaration, a virtual declaration, a binding declaration or a final binding declaration. A dialog like the one shown below will popup upon choosing this command.



From here one of the three types of associations can be chosen. Depending on the chosen type the effect of this will be exactly the same as mentioned above in **One-to-One Association**, **One-to-Many Association** or **Many-to-Many Association**, except that instead of associating one pattern to another, the pattern that is current focus is associated to itself.



The corresponding BETA code before the operation:

```
(#
  A: (# <<AttributeDeclOpt>> #);
#)
and after:
```

```
(#
  A: (#
    <<AttributeDeclOpt>>;
    theAA: ^AA
  #);
  AA: OneToManyAssociation
  (#
    oneType::< A;
    ManyElmType::< A
  #)
#)
```


5.5.6 View Menu

Abstract

Work sheets:

If current focus is a title of a pattern diagram **Abstract** will remove this pattern diagram and all diagrams detailed from it.

If current focus is an attribute that has been detailed the command will remove the detailed version of the attribute and all diagrams detailed from that.

Object Pages:

Abstracting an object (going from the detailed to the abstracted form of an object) is done by selecting a detailed object and choosing **Abstract** or by doubleclicking the detailed object (double-click inside the detailed object without hitting any of the contained objects).

Abstract Recursively

Work sheets:

If current focus is a title of a pattern diagram **Abstract Recursively** will remove all diagrams detailed from it (but *not* the diagram to which the title belongs).

If current focus is an attribute that has been detailed the command will remove the detailed version of the attribute and all diagrams detailed from that.

Object Pages:

Abstract does not apply to detailed objects that themselves contain detailed objects. To abstract such objects you will have to choose **Abstract Recursively** instead of **Abstract** (doubleclicking the object to abstract recursively is not possible).

Overview

Work sheets:

If current focus is a title or an attribute of a pattern diagram this command will remove all pattern diagrams not on the "detail path" to the diagram which is current focus.

Object Pages:

Having detailed a lot of objects in the object diagram you might want to focus on one specifically interesting object and so to speak "undo" all the other detailing that has been done. This is the purpose of the **Overview** command. To use it you select an object (of any kind, abstracted or detailed) and choose **Overview**. As a consequence of this all detailed objects in the surrounding of the selected object will be abstracted (except of course the objects containing the selected object itself).

Detail

Work sheets:

If current focus is an attribute of a pattern diagram and this attribute contains an object descriptor, like e.g. a pattern declaration does, **Detail** will display the attribute on detailed form; that is, as a pattern diagram displaying the attributes of this descriptor.

If current focus is a title of a pattern diagram **Detail** will perform a **Detail** on all attributes that can be detailed in the pattern diagram.

Notice:

- Doubleclicking a detailable attribute of a pattern diagram will perform a **Detail** on the attribute. If the attribute is already detailed doubleclicking it will make the title of the detailed version the new current focus.

Object Pages:

To display the contents of the first *abstracted object* select it and then choose **Detail** from the menu or you simply double-click the abstracted object. This will result in having the abstracted object displayed on *detailed* form; that is the objects that are directly contained in the detailed object (in the BETA program) will be displayed inside the box of the detailed object. These objects can now be detailed exactly the same way as the first one etc. - getting a diagram that illustrates the static and dynamic object structure and the creation and call relations between the objects of the initially chosen pattern.

Notice:

- When an object has been detailed the program automatically does a reprettyprint of the surrounding objects. The intention of this reprettyprint is to preserve the ordering of the objects on screen. In normal cases this works as intended, however, some abstracted objects may grow very large when being detailed with the effect that some of the surrounding objects are moved far apart. A help in preventing this to happen can be being aware of the fact that detailed objects only expand to the right and downwards (never to the left or upwards), so only objects residing in these locations may become subject to relocation

Detail Recursively**Work sheets:**

If current focus is an attribute of a pattern diagram and this attribute contains an object descriptor, like e.g. a pattern declaration does, **Detail Recursively** will display the attribute on detailed form; that is, as a pattern diagram displaying the attributes of this descriptor. Then it will perform this command recursively on the detailable attributes of the diagram that has now been detailed.

If current focus is a title of a pattern diagram **Detail Recursively** will perform a **Detail Recursively** on all attributes that can be detailed in the pattern diagram.

Object Pages:

If you select an abstracted object and choose the entry **Detail Recursively** the operation **Detail** will recursively be performed on the contained objects of the abstracted object.

Detail on Different Page**Work sheets:**

Not yet implemented.

Object Pages:

It is also possible to show the details of an object on another page than the one containing the abstracted form of the object. To do this select the abstracted object you wish to detail and choose **Detail on Different Page**. As a result of this the detailed form of the chosen abstracted object will be displayed on a new page. The abstracted object will still exist in its original surroundings but now displayed with a bold linestyle to indicate that the corresponding detailed object can be found on another page. In fact, doubleclicking an abstracted object like this will activate the page containing the detailed form of it.

Search

Work sheets:

By means of the **Search** command it is possible to search for a substring of a lexem, i.e. a name definition, a name application or a string.

Notice that this command does not search text auxiliary to the pattern diagrams, e.g text in user created labels.

Replace...

Work sheets:

Extends the search dialog with a replace text field so that the found text can be replaced with a text specified in this text field.

Layout Sub-patterns

Work sheets:

If current focus is the title of a pattern diagram and if there are currently displayed sub-patterns of this pattern, **Layout Sub-patterns** will rearrange the positions of these sub-pattern diagrams in a manner that makes them appear "nicely" in a tree structure below the selected diagram.

Edged Specialization Connectors

Work sheets:

If current focus is the title of a pattern diagram and if there are currently displayed sub-patterns of this pattern, **Edged Specialization Connectors** will make specialization connectors only follow horizontal and vertical lines.

Text Attributes

Makes a dialog popup through which the text attributes of current focus can be changed.

Fit To Text

If current focus is a graphical object auxiliary to the pattern diagrams, the command will make the object fit to the text contained in the object. This command does not work for attributes and titles of pattern diagrams.

Layout Preferences...

Makes a dialog popup where preferences regarding the layout of the pattern diagrams can be set.

The following explains in further detail the consequences of enabling or disabling the different checkboxes of the dialog.

Show:

Attributes

If enabled the attributes of the pattern diagrams are visible. If disabled the attributes of the pattern diagrams are invisible (only the titles of the diagrams will be visible).

References

If enabled the reference arrows are visible. If disabled the reference arrows are invisible.

Specializations

If enabled the specialization connectors are visible. If disabled the specialization connectors are invisible.

Associations

If enabled the association connectors (and diamonds) are visible. If disabled the association connectors (and diamonds) are invisible.

Show attributes with:

The four following entries concerns the appearance of the information inside the attributes of the pattern diagrams - that is, the prettyprint of the attributes. Only one of the following can be checked in at the time.

Name And Type

In this mode both the name and the type of the declaration inside an attribute is shown, as for example in:

```
anInstance: @aPattern
```

Name

This is the default prettyprint mode for attributes. This prettyprint mode only displays the name (the `NameDecl` part) of the declaration, making the above example appear as:

```
anInstance
```

Type

This prettyprint mode only displays the type of the declaration, making the above example appear as:

```
: @aPattern
```

Name And Kind

The **Name and Kind** prettyprint mode displays the name and the kind (e.g. static item symbolised by a “@”) of the declaration and makes the example appear as:

```
anInstance: @
```

5.5.7 Create Menu

The **Create** menu commands draw non-formal graphic objects (nodes and connectors).

Connector

Draws connectors between two nodes.

The cursor, a light arrow \rightarrow , indicates that the system is in connector creation mode.

Drawing Single-Segment Connectors

Click inside one node (the source node) and drag the connector tool inside another node (the destination node). Freja always draws connectors from border to border, along a line from the center of the source node to the center of the destination node.

Delete a connector by double-clicking outside a node during a drag. Connector length and placement are determined by the nodes they connect.

Drawing Multi-Segment Connectors

To create a segmented connector, follow the above process, with one difference: release the mouse at the location of the first bend point, outside the destination node. Subsequent clicks outside the destination node create other bend points. Click in the destination node to complete the connector.

Pressing the mouse on a bend point and pressing the ALT or META key snaps the two segments of the bend point into a right angle.

Delete a bend point by placing the pointer tool on the bend point and pressing the mouse button along with SPACEBAR.

Add a bend point by placing the pointer tool on a midpoint selection handle and dragging it to a location.

Effect of Moving Source/Destination Nodes

Freja automatically redraws connectors when their source and destination nodes are moved or resized.

Endpoint Handle Use

Reattach a connector by selecting the endpoint handle and dragging it to another object. After the connector has been created, use the endpoint regions to change the position of the source or destination endpoint.


Terminating Connector Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Box

Creates nodes with a box shape

The cursor, a box , indicates that the system is in box creation mode.

Changing Size and Position

Mouse clicks create default size boxes. Holding down the mouse and dragging the cursor also creates a box, the size of which is determined by the extent of the drag. Pressing SHIFT while dragging the cursor moves the box.

Creating Multiple Boxes

While in box creation mode, each mouse click creates a new box.


Terminating Box Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Rounded Box

Creates nodes with a rounded box shape.

The cursor, a rounded box , indicates that the system is in rounded box creation mode.

Changing Size and Position

Mouse clicks create default size rounded boxes. Holding down the mouse and dragging the cursor also creates a rounded box, the size of which is determined by the extent of the drag. Pressing SHIFT while dragging the cursor moves the rounded box.

Creating Multiple Rounded Boxes

While in rounded box creation mode, each mouse click creates a new rounded box.


Terminating Rounded Box Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Ellipse

Creates nodes with an ellipse shape.

The cursor, an ellipse , indicates that the system is in ellipse creation mode.

Changing Size and Position

Mouse clicks create default size ellipses. Holding down the mouse and dragging the cursor also creates an ellipse, the size of which is determined by the extent of the drag. Pressing SHIFT while dragging the cursor moves the ellipse.

Creating Multiple Ellipses

While in ellipse creation mode, each mouse click creates a new ellipse.


Terminating Ellipse Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Polygon

Creates an node, shaped as a polygon.

The cursor, a polygon , indicates that the system is in polygon creation mode.

Changing Size and Position

Mouse clicks create default size polygons. Holding down the mouse and dragging the cursor also creates a polygon, the size of which is determined by the extent of the drag. Pressing SHIFT while dragging the cursor moves the polygon.

Creating Multiple Polygons

While in polygon creation mode, each mouse click creates a new polygon.

Drawing Polygons

The command sets the first vertex or bend point. Subsequent vertices are created by clicking the mouse. To complete the polygon, double-click on the last vertex. A polygon must have at least three vertices or it will be removed when double-clicked.

Pressing the mouse on a bend point and pressing the ALT or META key snaps the two segments of the bend point into a right angle.

Delete a bend point by placing the pointer tool on the bend point and pressing the mouse button along with SPACEBAR.

Add a bend point by placing the pointer tool on a midpoint selection handle and dragging it to a location.


Terminating Polygon Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Regular Polygon

Creates nodes with a regular polygon shape.

The cursor, a regular polygon  , indicates that the system is in regular polygon creation mode.

Changing Size and Position

Mouse clicks create default size regular polygons. Holding down the mouse and dragging the cursor also creates a regular polygon, the size of which is determined by the extent of the drag. Pressing SHIFT while dragging the cursor moves the regular polygon.

Creating Multiple Rounded Boxes

While in regular polygon creation mode, each mouse click creates a new regular polygon.


Terminating Regular Polygon Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Wedge

Creates nodes with a wedge shape.

The cursor,  , indicates that the system is in wedge creation mode.

Changing Size and Position

Mouse clicks create default size wedges. Holding down the mouse and dragging the cursor also creates a wedge, the size of which is determined by the extent of the drag. Pressing SHIFT while dragging the cursor moves the wedge.

Creating Multiple Wedges

While in wedge creation mode, each mouse click creates a new wedge.


Terminating Wedge Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Line

Creates multi-point lines classified as nodes.

The cursor, a cross  , indicates that the system is in line creation mode.


Terminating Line Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Label

Creates nodes shaped as labels.

The cursor, a label  , indicates that the system is in label creation mode.

Label creates a special borderless node to contain text. Text is always turned on when the label tool is active. A click of the label tool places an insertion point on the page.

Dimensions

A label's dimensions are determined by the text typed into it. Freja does not word-wrap text typed in a label. Press the RETURN key to break lines.

Restrictions

A label's dimensions may not be changed by dragging or by any other graphic operation. For example, a label that is a group member does not change with the group, and a label that is a region does not change with its parent.

Labels cannot have fill added nor can the layering logic be altered.

Moving Labels

Move a label around the diagram with the graphic tool. The label's boundary is indicated by selection handles.

Terminating Label Creation Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Make Region

Turns nodes into regions or selects a new parent for regions. The command subordinates the current node to another object on the page.

Creating a Region

1. The status bar prompts: Select a Node, Region, or Connector to contain the new Region.
2. Place the pointer within the region's parent.
3. When the object's borders flash, click the mouse and the action is complete.
4. Selection handles reappear on the boundary of the new region and the status bar displays: Type: Region.

Effect on Parents

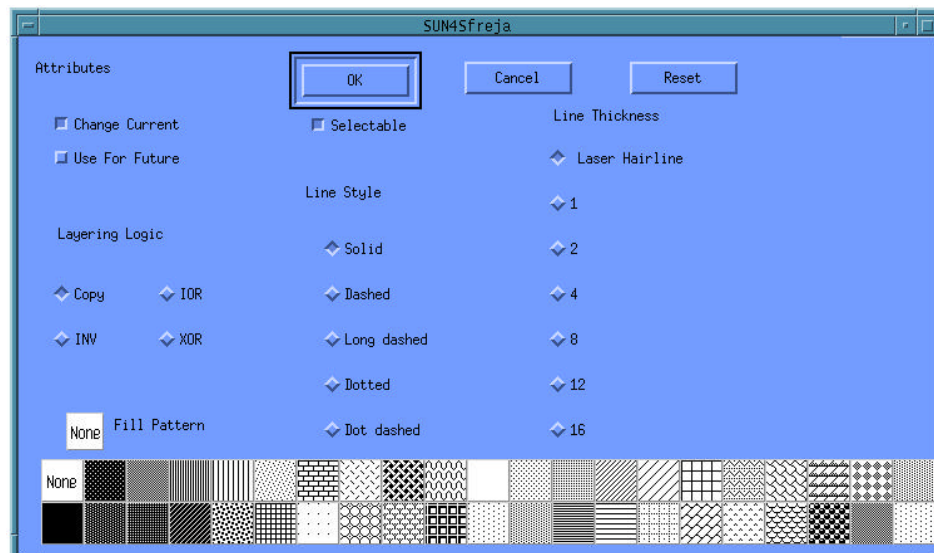
Some operations performed on a parent are also performed on its regions (with the exception of a label), including moving and resizing.

Operations performed on a region do not affect its parent.

Unmake Region

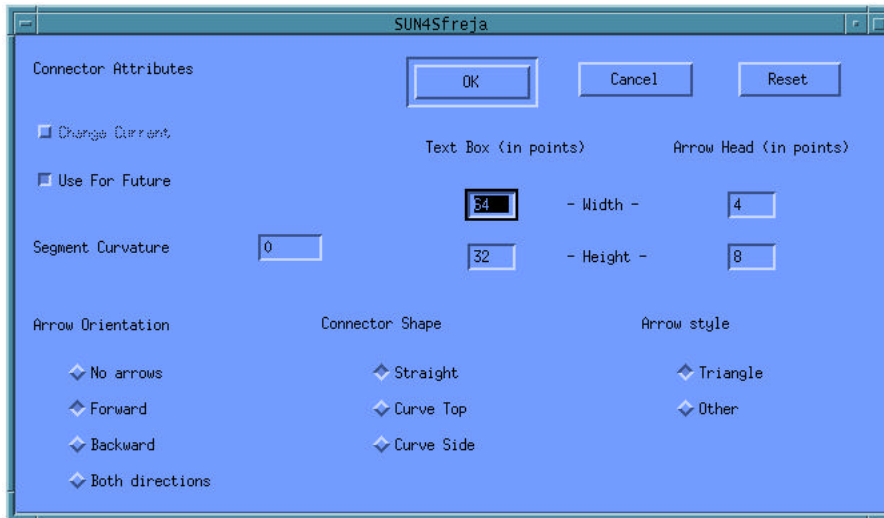
Turns regions into auxiliary nodes.

Set Attributes...



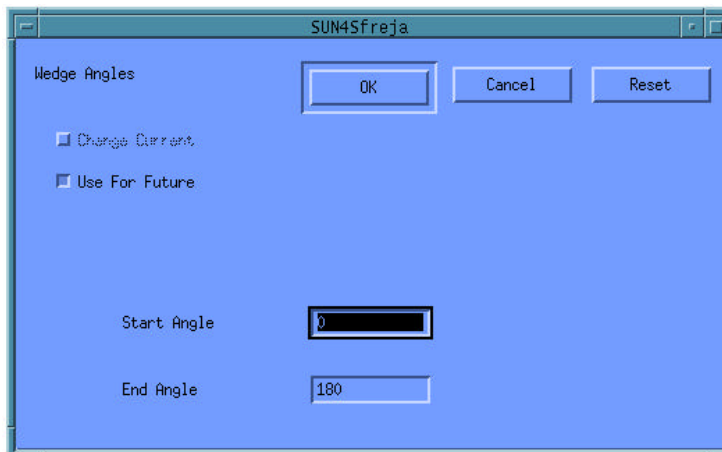
This command brings up a dialog through which general attributes of the graphical objects can be set

Set Conn Attributes...



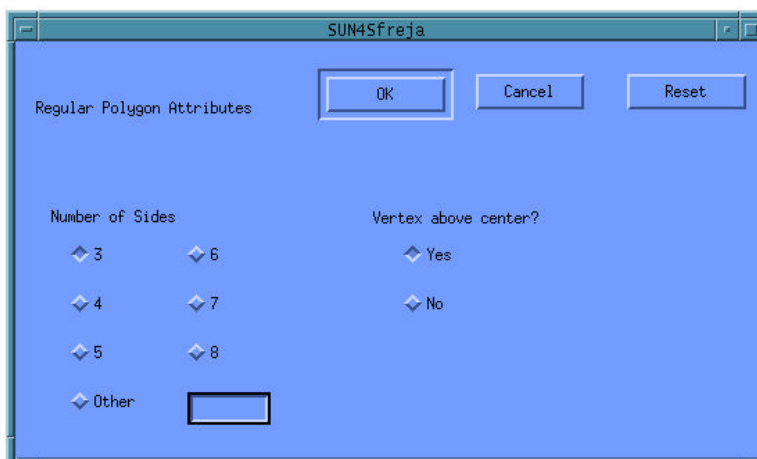
This command brings up a dialog through which attributes of connectors can be set

Set Wedge Attributes...



This command brings up a dialog through which attributes of wedges can be set

Set Polygon Attributes...



This command brings up a dialog through which attributes of Polygons can be set

5.5.8 Makeup Menu



The **Makeup** menu commands manipulate and alter graphic objects.

Select

Selects objects by temporarily hiding other objects.

When objects are layered, closely spaced, or contained within other objects, they may be difficult to select with the graphic tool.

Selecting Visible Objects

Moving the pointer tool across an object causes its borders flash. Clicking on a flashing object makes it become the current object.

Selecting Hidden Objects


While the obscuring object is flashing, press SPACEBAR to make it invisible. Continue to hide objects until the desired object is visible. Select the object by clicking the mouse. All the hidden objects are restored.

Clicking anywhere in the active window restores all the temporarily hidden objects.

Adjust

Resizes the current node without moving its center.

Resizing in Two Dimensions

The hand pointer  appears at the lower right corner of the current object. Drag the hand to resize the object in either dimension. Click the mouse to complete the operation.

Resizing in One Dimension

To resize one dimension while retaining the object's other dimension and its center, hold down the CTRL key.

Drag horizontally, with the vertical dimension locked.

Drag vertically and the horizontal dimension is locked.

Click the mouse to complete the operation.

Resizing Regions and Labels

When a label or a key region is selected, the command has no effect unless the object is part of a group. In such a case, the command does not affect the object itself but only its descendants, which get a new size and position.

When a selected group of regions contains ancestors/descendants of each other, some regions may change more than once.

Change Shape

Changes the size and shape of objects.

Change Shape changes the shape and size of the current node or group to match that of any other node on the page.

Shape Changing

The status bar prompts the selection of a node or region as a model for the shape and size change.

Placing the pointer within the selected object causes its borders flash.

Clicking on the flashing object completes the change.

Region Changing

Note that when a selected group of regions contains ancestors/descendants of each other, some regions may change more than once.

Drag

Repositions objects.

The drag tool  indicates that the system is in drag mode.

Use this tool to move the current object or group without placing the pointer inside a selected object. This is especially useful for moving small objects and objects inside other objects.

Dragging Text

If text is turned on, text can be moved as a block within its object. To begin a drag text, the drag tool must be inside the graphic object containing the text.

Terminating Drag Mode

The mode is terminated by:

- Pressing ESC.
- Selecting another command.

Displace

Repositions objects by moving the current node or group the same distance (horizontally and vertically) as the last drag or alignment of an object or group.

Note: The command cannot be applied to groups of regions.

Move To...

Moves the current node or group to another page in the diagram.

Moving Nodes

The hierarchy page window appears and the status bar prompts: Click on page to receive object.

Move Node returns to the current page after the target page has been selected.

Effect on Regions and Connectors

Regions and connectors are moved, along with the corresponding nodes, in the usual way:

- A region is moved if the corresponding node is moved.
- A connector is moved if both the source and destination nodes are moved. If only one of these nodes is moved, the connector is deleted.

Hide Substructure

Hides descendants of selected objects, making them non-visible and non-selectable.

Nodes and Connectors

Hides all the descendants of the selected objects.

Regions

Hides the selected regions, together with all their descendants.

Show Substructure

Shows descendants of selected objects, making the descendants visible and selectable.

Next Object

Selects the object one layer above the current object.

This process is repeated with each selection of **Next Object** until the top layer of the stack of objects is reached. **Previous Object** (**Makeup** menu) reverses the process.

Previous Object

Selects the previous object (or text pointer).

Previous Object selects the object one layer below the current object. This process is repeated with each selection of **Previous Object** until the bottom layer of the stack is reached. **Next Object** (**Makeup** menu) reverses the process.

Bring Forward

Changes the object order of objects on a page.

Reordering Object Layering

1. Select the object to bring forward.
2. Choose **Bring Forward**.
3. Place the pointer on the object to put one layer below the selected object. The object chosen to underlie the selected object flashes.
4. Click to bring the selected object forward .

To travel through the layering order sequentially, use **Next Object** (**Makeup** menu) and **Previous Object** (**Makeup** menu).

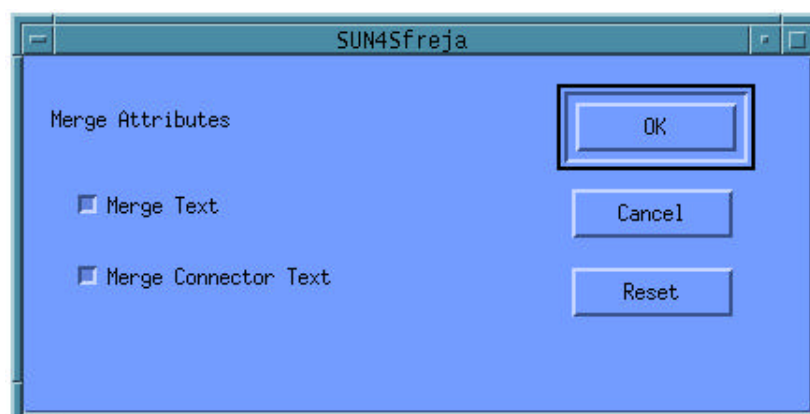
Restrictions

The command is not yet implemented for groups of regions.

Merge

Merges nodes into a single node by combining a node or group into a single target node.

Specify options for the merge with **Set Merge Attributes**.

Set Merge Attributes

Options for **Merge** can be set using this dialog.

Edit Polygon

Makes the vertices of the current polygon selectable.

The vertices of the polygon can then be edited in exactly the same way as bend points of connectors.

5.5.9 Align Menu



The **Align** menu commands reposition the currently selected node or group by aligning it to one or two reference objects.

Aligning is a two-step process: first select the command and then the reference object(s). A click of the mouse on the reference object accomplishes the alignment.

Horizontal

Performs a horizontal alignment.

Horizontal aligns the center of the current object along a horizontal axis drawn through the center of the reference object.

Vertical

Performs a vertical alignment.

Vertical aligns the center of the current object along a vertical axis drawn through the center of the reference object.

Left to Left

Performs a left-to-left alignment.

Left to Right

Performs a left-to-right alignment.

Right to Left

Performs a right-to-left alignment.

Right to Right

Performs a right-to-right alignment.

Top to Top

Performs a top-to-top alignment.

Top to Bottom

Performs a top-to-bottom alignment.

Bottom to Top

Performs a bottom-to-top alignment.

Bottom to Bottom

Performs a bottom-to-bottom alignment.

Center

Performs a center alignment.

Center places the center of the current object over the center of the reference object.

Between

Places the center of the current object at the midpoint of an imaginary line drawn between the centers of the two reference objects.

Projection

Places the current object at one end of an imaginary line, whose first endpoint is the center of the first reference object, and whose midpoint is the center of the second reference point.

5.2.11 Page Menu

Refresh

Refreshes the current page.

Refresh redraws the diagram on the current page, restoring the on-screen resolution of objects and text distorted by operations such as **Blowup** and **Reduce**.

Reduce

Reduces the page scale for the selected set of pages.

Each time **Reduce** is used, the active page is reduced to one half its present size to the minimum of 25 percent of normal. **Reduce** reverses the effect of **Blowup**, and like **Blowup**, may affect screen resolution.

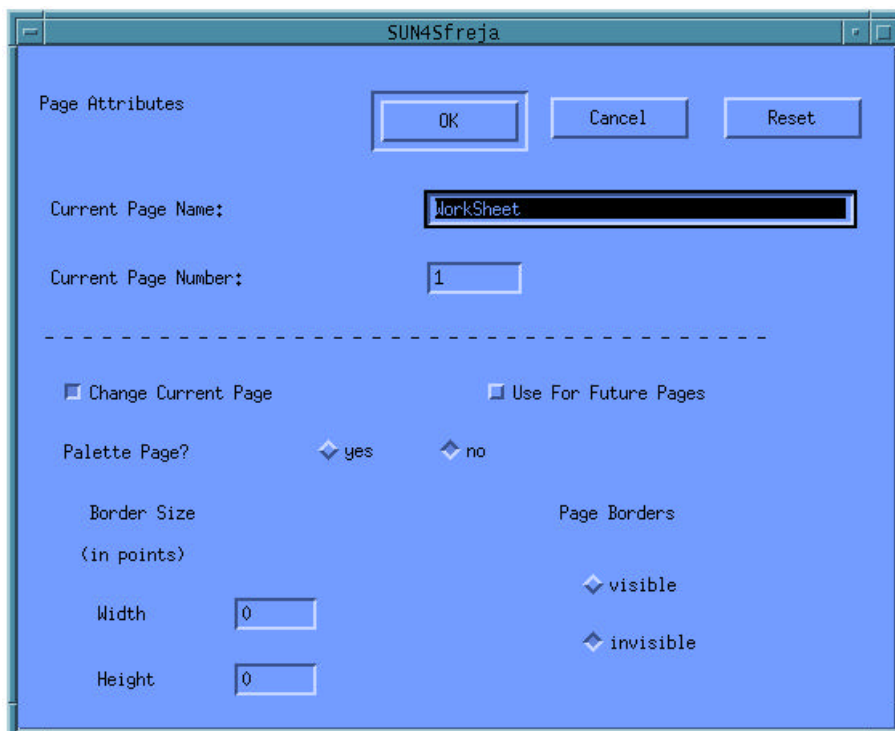
Blowup

Enlarges the page scale for a selected set of pages.

Each time **Blowup** is used, it enlarges the active page to twice its present size up to the maximum of 400 percent of normal. The page scale is indicated in the status bar.

Some degrading of the screen image may occur. The actual diagram and the printed diagram are not degraded by **Blowup** or **Reduce**. Use the **Refresh (Page menu)** command to restore on-screen resolution after using **Blowup**.

Attributes...



The attributes of the current page can be set using this dialog.

Group Window

Makes the page entitled “**Group Window**” the current page.

If current object on the **Work Sheet** is a title or an attribute of a pattern diagram, the current object that is selected in the **Group Window** is the fragment that this title or attribute belongs to.

Work sheet

Makes the page entitled “**Work Sheet**” the current page.

Hide page

Hides the current page (i.e. makes it invisible).

Show all pages

If a number of pages has been hidden this command makes them all reappear on screen.

5.5.10 The Object Menu**Show Object Diagram**

This menu entry is used when initially generating an object diagram. For further details see the Getting Started section.

The Search Modes

Having chosen **Show Object Diagram**, as explained above, the text “**Scanning AST...**” will appear in the status field of the menu bar (the field that initially contains the text “**Type: node**”). This means that an analysis is being performed on the target program to detect dynamic object generations and procedure calls, so that these can be displayed in the resulting object diagram. Those BETA-fragments that are in some way used by the target program (included files, body files etc.) also have to be analysed to give a complete picture of the dynamics of the program. Some programs use a lot of different frag-

ments that themselves use a number of fragments etc. Analysing all these fragments can sometimes be a time-consuming task. Furthermore it may not be necessary to do this to get an object diagram that illustrates the essentials of the program. For this reason you may choose one of three different search modes before choosing **Show Object Diagram**. The three search modes are:

- **Local Search:** The default search mode. Analysis is only carried out on fragments located in the same directory as the target program or in subdirectories hereof.
- **Global Search:** All fragments used by the target program are analysed.
- **Global Search with Confirmation:** Like **Global Search** except that for every included fragment you are asked to confirm if the fragment should be analysed.

Notice:

- Excluding a fragment from analysis also means excluding all fragments used by this fragment.

During analysis the fragment that is currently analysed is printed out in the xterm from which the program was started.

Filters

Some programs may have a very complex object creation and/or procedure call structure. The illustration of this in the object diagram may result in a very large number of arrows. This again may hinder the user in getting a clear picture of the dynamic structures of the program. To help out in these situations two filters are provided.

Toggle Dynamic Creations

To remove all dynamic creation arrows from an object diagram you choose

Toggle Dynamic Creations. To have them displayed again at sometime afterwards you choose **Toggle Dynamic Creations** once more. Whether or not a check mark is present before this entry in the menu visualises whether or not dynamic creations are currently shown in the object diagram. Default is that dynamic creations are displayed in the diagram.

Toggle procedure calls

Same as for **Toggle Dynamic Creations**.

Notice:

- The filters on dynamic creations and procedure calls work on page level. That is, if you choose **Detail on Different Page** for some object then both dynamic creations and procedure calls will as a default be shown in the diagram on the new page. Afterwards checking out one or both of these will only have an effect on the page currently in focus.

Showing and Hiding Simple Objects

Instances of the basic patterns Integer, Char and Boolean are in the context of object diagrams considered to be *simple objects*. Simple objects are as a default not displayed in the object diagram. It is however possible to have them displayed as follows: Select the detailed object for which you wish to see the simple objects that are part of it. Then choose **Show Simple Objects**. The result of this is that all simple objects that are part of the chosen detailed object are displayed at the bottom of this. The simple objects of a detailed object can in a similar fashion be hidden again by selecting the detailed object and choosing **Hide Simple Objects**.

Compact and blowup

The objects in the diagram can freely be rearranged (moved) by the user except that objects that are part of another object cannot be moved outside the frame of the surrounding object. While rearranging the objects in some detailed object

one might wish for more space to do so; that is to enlarge the rectangle that makes up the detailed object. To do this you select the detailed object and choose **Blowup**. The effect is that the detailed object is enlarged by 10% and the surrounding objects are reprettyprinted accordingly (of course this procedure may be repeated until a satisfactory size is obtained).

If you on the contrary wish to have an object displayed on an as compact form as possible you select a detailed object and choose **Compact**. Thereby the rectangle of the chosen detailed object and those detailed objects containing it will be made as small as possible.

Notice:

- You should not resize objects using the standard “mouse dragging” method as the effect of this will be that all contained objects will also be resized and no reprettyprint will be performed.
- If you move a detailed object all objects that are part of it are also moved. This effect can sometimes be a help in detecting which objects are part of a detailed object.

6 Bibliography

- [Jensen 91] K. Jensen, S. Christensen, P. Huber, M. Holla: *Design/OA: A Reference Manual*, Meta Software Corporation, 125 Cambridge Park Drive, MA 02140, USA, 1991.
- [Knudsen 94] J. L. Knudsen, M. Löfgren, O. L. Madsen, B. Magnusson (eds.): *Object-Oriented Environments – The Mjølner Approach*, Prentice Hall, 1994, ISBN 0-13-009291-6.
- [Madsen 93] O. L. Madsen, B. Møller-Pedersen, K. Nygaard: *Object-Oriented Programming in the BETA Programming Language*, Addison-Wesley, 1993, ISBN 0-201-62430-3
- [MIA 90-1] Mjølner Informatics: *The Mjølner BETA System: – Overview*, Mjølner Informatics Report MIA 90-1.
- [MIA 90-2] Mjølner Informatics: *The Mjølner BETA System: BETA Compiler Reference Manual* Mjølner Informatics Report MIA 90-2.
- [MIA 90-11] Mjølner Informatics: *Sif – A Hyper Structure Editor, Tutorial and Reference Manual* Mjølner Informatics Report MIA 90-11.
- [MIA 92-12] Mjølner Informatics: *The Mjølner BETA System – The BETA Source-level Debugger – Users's Guide*, Mjølner Informatics Report MIA 92-12

Index

- About 32
- Abstract 45
- Abstract Recursively 45
- abstract syntax tree 7
- Active Object 5
- active page. 28
- Adjust 54
- Align Menu 57
- Architecture 7
- Associate to Self 44
- Association 3
- Associations 47
- Attributes 47
- attributes diagrams. 29
- Attributes... 59
- Between 58
- Bibliography 63
- Binding of Virtual 40
- Binding of Virtual Pattern 36
- Blowup 58, 61
- Bottom to Bottom 58
- Bottom to Top 58
- Box 49
- Bring Forward 56
- Center 58
- Change Shape 55
- Check 18, 35
- Clear 34
- Close 30
- common representation 7
- Compact and blowup 60
- Composition 2, 5
- Connector 48
- Copy 33
- Create Menu 48
- Creating a New Diagram 9
- current diagram. 35
- current focus. 28
- Cut 33
- descriptor diagrams. 28
- Detail 45
- Detail on Different Page 46
- Detail Recursively 46
- Displace 55
- Drag 55
- Dragging Text 55
- Dynamic Component Reference 39
- Dynamic Object 5
- Dynamic Object Creation 5
- Dynamic Reference 15, 36, 39
- Edged Specialization Connectors 47
- Edit Menu 33
- Edit Name 12, 34, 37
- Edit Polygon 57
- Editing 9
- Ellipse 50
- Endpoint Handle Use 49
- ew BETA library 9
- Expand Menu 36
- File Menu 28
- file types 8
- Filters 60
- Final Binding of Virtual 40
- Final Binding of Virtual Pattern 36
- Fit To Text 47
- Global Search 60
- Global Search with Confirmation 60
- graphical template 10
- Group Page 27
- Group Window 59
- Help 32
- Hidden Objects 54
- Hide page 59
- Hide Substructure 55
- Horizontal 57
- inconsistent 32
- Insert After 34
- Insert Before 34
- keywords 10
- Label 51
- Layout Preferences... 47
- Layout Sub-patterns 47
- Left to Left 57
- Left to Right 57
- Line 51
- Load Graphics... 32
- Load Text... 32
- Local Pattern 35
- Local Search 60
- Make Region 52
- Makeup Menu 54
- Many-to-Many Association 42
- Menu Bar 28
- Merge 56
- Move To... 55
- Moving Labels 52
- Moving Nodes 55
- Moving Source/Destination Nodes 49
- Multi-Segment Connectors 49
- Multiple Boxes 49
- Multiple Ellipses 50
- Multiple Polygons 50
- Multiple Rounded Boxes 50, 51
- Multiple Wedges 51
- Name 48
- Name And Kind 48
- Name And Type 48
- New BETA library 29
- New BETA program 9, 28
- New Graphics Page... 29

- New Pattern 13, 35
- Next Object 56
- nonterminals 10
- Notation 2
- Object Diagrams 4, 9, 23, 26
- Object Layering 56
- Object Menu 59
- Object Pages 27
- Object References 15
- One-to-Many Association 41
- One-to-One .i.Association 40
- Open Subeditor 12, 34
- Open Subeditor) 37
- Open... 30
- Overview 45
- Page Menu 58
- Page Setup 31
- Parents 52
- Part Object 36
- Paste 33
- Paste After 34
- Paste Before 33
- Pattern 2
- Pattern Diagrams 2, 9, 21, 26
- placeholders 10
- Polygon 50
- Preferences... 32
- Previous Object 56
- Print... 31
- Procedure 5
- Procedure Call 5
- Projection 58
- Quit 33
- ragment diagrams. 29
- Reduce 58
- Reference Composition 3
- Reference Manual 26
- References 47
- Refresh 58
- Region Changing 55
- Regular Polygon 51
- Relations Menu 38
- Remove Optionals 17, 34
- Replace... 47
- Resizing in One Dimension 54
- Resizing in Two Dimensions 54
- Resizing Regions and Labels 54
- Reverse Engineering 21
- Revert 31
- Right to Left 57
- Right to Right 57
- Rounded Box 49
- Save 31
- Save As... 31
- Save Page As Graphics... 32
- Save Text... 32
- Search 47
- Search Modes 59
- Select 54
- Set Attributes... 52
- Set Conn Attributes,, 53
- Set Merge Attributes 56
- Set Polygon Attributes... 53
- Set Wedge Attributes... 53
- Shape Changing 55
- Show 47
- Show all pages 59
- Show attributes with 48
- Show Object Diagram 24, 26, 59
- Show Optionals 34
- Show Sif / Hide Sif 32
- Show Substructure 56
- Sif 7, 16
- Single-Segment Connectors 49
- Specialization 3, 13, 39
- Specializations 47
- Static Component Reference 39
- Static Object 4
- Static Reference 38
- structure editing 7, 10
- Templates 10
- Templates Menu 35
- Terminating Box Creation Mode 49
- Terminating Connector Creation Mode 49
- Terminating Drag Mode 55
- Terminating Ellipse Creation Mode 50
- Terminating Label Creation Mode 52
- Terminating Line Creation Mode 51
- Terminating Polygon Creation Mode 51
- Terminating Regular Polygon Creation Mode 51
- Terminating Rounded Box Creation Mode 50
- Terminating Wedge Creation Mode 51
- Text Attributes 47
- Text editing 12
- Toggle Dynamic Creations 60
- Toggle procedure calls 60
- Top to Bottom 57
- Top to Top 57
- Tutorial 9
- Type 48
- Undo 33
- Unmake Region 52
- Variable Pattern 40
- Vertical 57
- View Menu 45
- Virtual Pattern 36
- Visible Objects 54
- Wedge 51
- Whole-part Composition 3
- Work sheet 59
- Work Sheets 26