Modechart Toolset User's Guide

Anne Rose, Manuel A. Pérez, Paul Clements

Center for Computer High Assurance Systems
U.S. Naval Research Laboratory
Washington, DC 20375-5320

Table of Contents

CHAPTER 1

Introduction to the Modechart Toolset (MT)

What is the purpose of this tool?

The Modechart Toolset (MT) is a set of tools designed to facilitate the specification, modeling, and analysis of real-time embedded systems using the Modechart language. MT supports the creation, modification, and on-line storage of Modechart specifications (Chapters 2-6). It also supports the analysis of Modechart specifications via a consistency and completeness checker (Chapter 7), a simulator (Chapter 8), and a verifier (Chapter 9).

Who are the intended users?

MT's intended users are systems engineers who want to model their systems using Modechart and to experiment with their models using MT's analysis tools.

What knowledge is assumed of users?

MT users should have:

- experience using a mouse and familiarity with common user interface components, such as buttons, pull-down menus, and scrollbars
- a working knowledge of the UNIX operating system, e.g., know about UNIX paths, directories, configuration files, etc.
- a working knowledge of the Modechart language and semantics
- elementary familiarity with X-windows and OpenWindows (see Appendix 1 and Appendix 2 for and introduction)

How can I find out more about Modechart and its capabilities?

To get a comprehensive definition of the Modechart language and semantics, see:

F. Jahanian, R.S. Lee, and A.K. Mok, "Semantics of Modechart in Real Time Logic," *Proceedings, 21st Hawaii International Conference on System Sciences*, (January 1988).

F. Jahanian and A.K. Mok, "Modechart: A Specification Language for Real-Time Systems," (to appear).

For insight into the engineering issues associated with the MT, see:

C. Heitmeyer, P.C. Clements, B. Labaw, and A.K. Mok, "Engineering CASE Tools to Support Formal Methods for Real-Time Software Development," *Proceedings, 5th Intl. Workshop on Computer-Aided Software Engineering*, (7-9 July 1992).

P. C. Clements, C. L. Heitmeyer, B. G. Labaw, A. T. Rose, "MT: A Toolset for Specifying and Analyzing Real-Time Systems," *Proceedings, 1993 Real-Time Systems Symposium,* (December 1993).

To understand more about the Modechart verifier, see:

F. Jahanian and A.K. Mok, "Safety Analysis of Timing Properties in Real-Time Systems," *IEEE Transactions on Software Engineering*, Vol. SE-12(9), pp. 890-904 (September 1986).

F. Jahanian, A. Mok, and D. Stuart, "Formal Specification of Real-Time Systems," Department of Computer Sciences, TR-88-25, University of Texas at Austin (June 1988).

F. Jahanian and D. Stuart, "A Method For Verifying Properties of Modechart Specifications," *Proceedings, 9th IEEE Real-Time Systems Symposium*, (1988).

D.A. Stuart, "Implementing a Verifier for Real-Time Systems," *Proceedings, 1990 Real-Time Systems Symposium*, pp. 62-71 (5-7 December 1990).

C. L. Heitmeyer and B. G. Labaw, “Requirements Specification of Hard-Real-Time Systems: Experience with a Language and a Verifier,” *Foundations of Real-Time Computing Formal Specifications and Methods*, A. van Tilborg and G. Koob, Eds., Kluwer Academic Publishers, Norwell, MA, pp. 291-313 (1991).

To understand more about the Modechart simulator and the semantic properties of Modechart that make the simulation useful, see:

D.A. Stuart and P.C. Clements, “Clairvoyance, Capricious Timing Faults, Causality, and Real-Time Specifications,” *Proceedings, 1991 Real-Time Systems Symposium*, (3-6 December 1991).

Conventions Used in This Document

- The steps required to perform an operation are numbered sequentially and are written in **bold** type.
- Problems that a user may encounter when performing a particular operation are documented following the steps in a subsection titled “What can go wrong?”.
- Each user operation either adds, clears, or makes no change to the undo stack. In the section describing a given operation, the text following the symbol ♦ indicates whether the operation is undoable. Performing an operation that is undoable adds the operation to the undo stack. Performing an operation that is not undoable clears the undo stack. If the section describing an operation does not contain the symbol ♦, performing the operation does not change the undo stack. The analysis tool operations have no effect on the undo stack.
- Within this document, the OpenWindow’s mouse button terminology will be used, i.e. mouse buttons will be referred to by function not by position.

Mouse Button	Function
Left	Select
Middle	Adjust
Right	Menu

  Appendix 1 (Using OpenWindows) describes what the different mouse button functions mean. When the documentation simply refers to clicking on or selecting an object without indicating which mouse button to use, assume the Select mouse button.

- A few operations described in this document have not been fully implemented yet. These operations are noted in the manual.

Acknowledgments

The authors acknowledge the hard work of the numerous people involved in the development of MT. Al Mok of the University of Texas invented the Modechart language and collaborated with the UT development team. Connie Heitmeyer, Bruce Labaw, Alan Bull, and Carolyn Gasarch of NRL contributed to the design and development of the facilities to create, edit and store Modechart specifications. Dennis Chen of UT built the initial version of the layout software. Simon Chow of UT modified the layout algorithm to deal with hierarchical structures. Bruce Labaw added several heuristics to the end product to make it perform better. Doug Stuart of UT built the verifier. Alex Ho of UT designed and implemented the graphical user interfaces for both the verifier and the simulator. Finally, we thank Connie Heitmeyer and Bruce Labaw for their many comments and reviews of this document.

CHAPTER 2

Starting MT

How do I invoke the tool?

1. **Start either X-windows or OpenWindows on your machine.**

   Appendix 1 and Appendix 2 give brief overviews of how to use OpenWindows and X-windows, respectively. For detailed information on either of these window environments, consult your local documentation.

2. **Set the required environmental variables.**

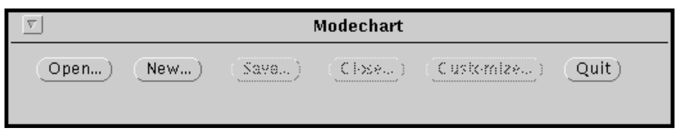
   To run MT, five variables must be set. You can use the “setenv” command to define these variables.

PRINTER	Postscript printer to use when printing Modechart specifications
SPAWNCMD	A program called “spawn”, which the tool uses to execute other stand-alone Modechart tools
VERIFYCMD	A program called “XMCVFY.sun4“, which the tool uses to verify Modechart specifications
SIMULATECMD	A program called “XMCSIM.sun4“, which the tool uses to simulate Modechart specifications
SIMOPTCMD	A program called “options_maker.sun4”, which the tool uses to create a default options file needed to run the simulator

If you do not know where these programs reside, contact a member of the MT team for information. It is a good idea to include these steps as part of the initialization sequence performed when you log into a UNIX system.

3. **If your machine supports X-windows or OpenWindows but it is not a Sun 4 or it does not host MT, do the following:**
   - **Start X-windows or OpenWindows on your machine.**
   - **Type "xhost +*xxx*" where "*xxx*" is the name of the machine hosting MT.**
   - **Log in to the MT host.**
   - **Once logged in, type "setenv DISPLAY *yyy*:0.0" where "*yyy*" is the name of the your machine.**
4. **Type "mcuitool" in any MT host window.**

   If the directory where "mcuitool" resides is not in your default path, you must type the entire pathname to invoke "mcuitool". You should add this path to your default path. If you have correctly started MT, a window, called the Modechart window, will appear.



This window contains the commands that could apply to more than one specification. In the current version of the tool, these commands include opening a specification, creating a specification and quitting the tool. These operations are described in detail later in this chapter.

Note: There are two command line options which allow you to create a new specification or load an existing specification automatically from the command line. After describing the process of performing these operations via the user interface, these shortcuts (using the command line options) will mentioned.

What files are associated with a specification?

The files associated with a specification created by MT include:

- a Modechart database file

This file contains information about the objects in the specification and how they are related to one another. It does not store any graphical information. The name of this file corresponds to the filename you provided when you opened or created this specification. The syntax of this file is described in Appendix 3.

- a graphics file

  This file captures the graphical information associated with your specification. The name of this file is the name of the database file appended with ".lay" (for layout).

- a simulation options file

  This file defines the environment attributes to be used during simulation. The default name of this file is the name of the database file appended with ".option". The purpose of this file is described in more detail in Chapter 8. Appendix 4 contains a sample options file.

- a simulation log file

  This file logs information associated with a particular simulation run. The default name of this file is the name of the database file appended with ".log". The information that can be contained in a log file is described in Chapter 8.

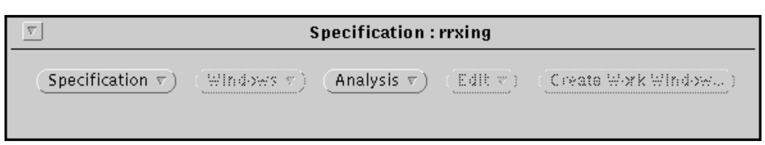
To load a specification, you are only required to have the database file. When a specification is loaded that does not have a graphics file, MT automatically lays out the specification. This layout process can take a noticeable amount of time (especially for larger specifications), so it is a good idea to save the specification after this process.

What windows are associated with each specification?

There are three windows associated with a specification that remain active the entire time the specification is open. These windows are:

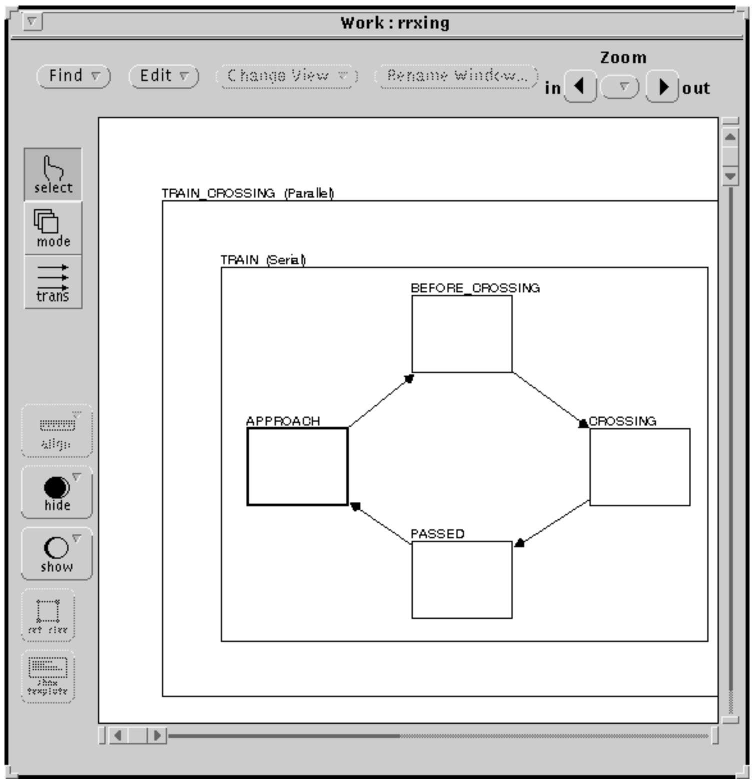
- Specification window

  This window allows you to execute commands that apply to a specification as a whole. These commands include saving, reverting to a saved specification, printing, closing and performing analysis. The operations contained in the Specification window are described in Chapter 3.



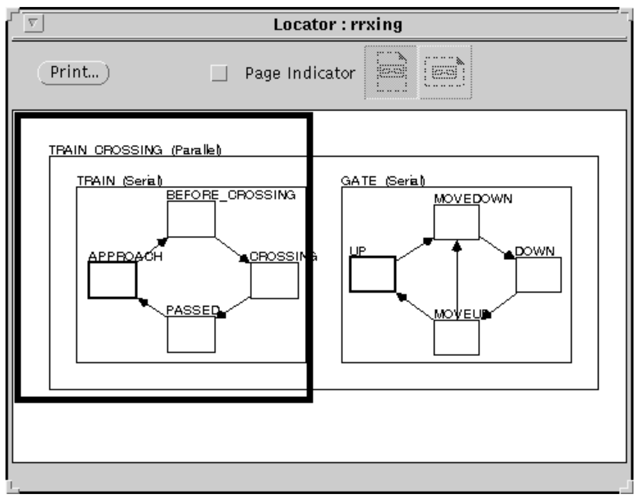
- Work window

  This window displays the graphical representation of a specification. It allows you to navigate through a specification and to make additions, deletions, and modifications to a specification. The operations contained in the work window are described in Chapters 4-6.



- Locator window

  The locator window displays the entire specification. In addition, it contains a work window indicator, a bold outline whose boundary corresponds to the section of a specification currently displayed by its work window. This indicator can be used for more rapid navigation around a specification than is possible using the navigation facilities available in the work window. You can directly manipulate the portion of a specification displayed in its work window by dragging and resizing the indicator. The navigation operations contained in the locator window are described in detail in Chapter 4. The locator window also contains controls to print a specification. See Chapter 3 for more detail on how to print.



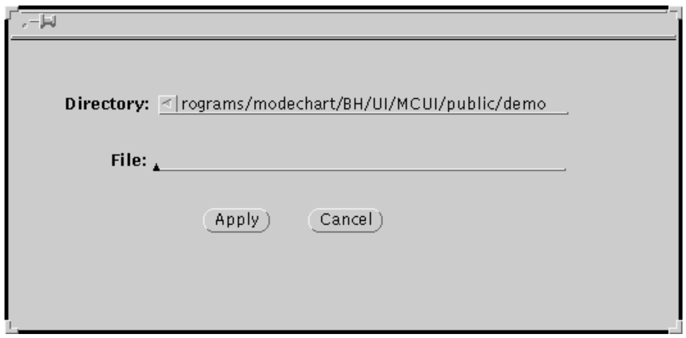
The analysis tool windows are active from the time the tool is invoked until the user issues the “quit” command. The windows associated with the consistency and completeness checker, the simulator and the verifier are described in Chapter 7-9.

How do I create a new specification?

1. **Click “New...” in the Modechart window.**

   A dialog box will appear asking for a directory and a filename. The default directory which is displayed in the dialog box is the last directory input by the user. Initially, it is the directory where MT was invoked.

2. **Type in the directory and the filename of the specification you wish to create.**



3. **Click "Apply".**

   The dialog box will disappear, and a specification window, a work window and a locator window will appear. If you decide you do not want to create a new specification, click "Cancel".

Shortcut: You can create a new specification from the command line by typing "mcuitool -new <filename>". This shortcut assumes the specification path to be the current working directory.

What can go wrong?

- The current version of MT only allows one specification to be open at a time. Close all open specifications before invoking this command again.
- The specification you want to create already exists.
- A graphics file associated with the specification you are trying to create already exists.

How do I open an existing specification?

1. **Click "Open" in the Modechart window.**

   A dialog box will appear requesting a directory and file name.

2. **Type in the directory and filename of a Modechart specification.**

   This file might have been generated as the result of a "Save" operation from a previous MT session, or it may have been created independently of MT. For information about the legal format of a Modechart specification file, see Appendix 3.

3. **Click "Apply".**

   The dialog box will disappear, and a specification window, a work window, and a locator window will appear. If you decide you do not want to open the specification, click "Cancel".

Shortcut: You can open a specification from the command line by typing “mcuitool <filename>”.

What can go wrong?

- The current version of MT only allows one specification to be open at a time. Close all open specification before trying this command again.
- The file named by the given directory path and filename doesn’t exist.
- The file permissions do not allow you to read the file.
- The name of a graphics file corresponds to the specification being opened but contents of the graphics file do not correspond to the contents of the specification. One solution to this problem is to remove the graphics file (or at least change its name) and then load the specification. This will force MT to generate a new graphics file for the specification.

How do I quit the tool?

1. **Click on the “Quit” button in the Modechart window.**

   If a specification is open and needs to be saved, you will be asked whether you wish to save the specification before quitting, to quit the tool without saving the specification, or to cancel the quit.

CHAPTER 3

Managing a Modechart Specification

How do I save a specification?

1. **Choose "Save" from the Specification menu in the specification window.**

   It is prudent to execute a "Save" operation from time to time to guard against unexpected system failures. The "Save" operation causes the old contents of the file to be lost. When you save a specification, a Modechart database file and a graphics file will be written in the directory you provided when you opened/created this specification:

   Shortcut: Click "Specification" in the specification window. "Save" is the default menu item for the Specification menu so "Save" is automatically chosen.

What can go wrong?

- You do not have write permission in the directory you specified.
- One or more of the files for this specification already exists and you do not have permission to write over them.

How do I save the specification under a different name?

1. **Choose "Save as..." from the Specification menu in the specification window.**
2. **Enter the new file name.**

The same dialog box which appears when you create or open a specification will pop up.

3. **Click "Apply".**

   The current work will be saved in the new file, and the old file will not be altered. If you entered a filename that already exists, you will be given the choice of either cancelling the save or writing over the existing file (assuming you have write permission).

What can go wrong?

- You entered a directory which does not exist.
- You do not have write permission in the directory you specified.
- One or more of the files corresponding to the new specification name already exist and you do not have permission to write over them.

How do I close a specification?

1. **Choose "Close" from the Specification menu in the specification window.**

   If you have not made any changes (syntactic or semantic) to the specification since it was last saved, the specification will be closed without further prompting. If you have made changes, you will be given the option of either saving the specification before closing, closing the specification without saving the changes, or cancelling the close operation.

How do I revert to working on the last saved version of my specification?

1. **Choose "Revert to Save" from the Specification menu in the specification window.**

   If your specification differs from the last saved version, you will be asked to either confirm or cancel this operation.

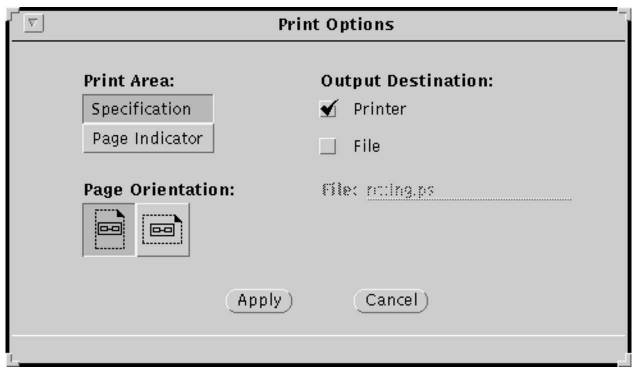
How do I print a specification?

The print operation is based on the concept of what you see is what you get. When you print a specification, the hard copy of the specification will reflect the characteristics of the specification that

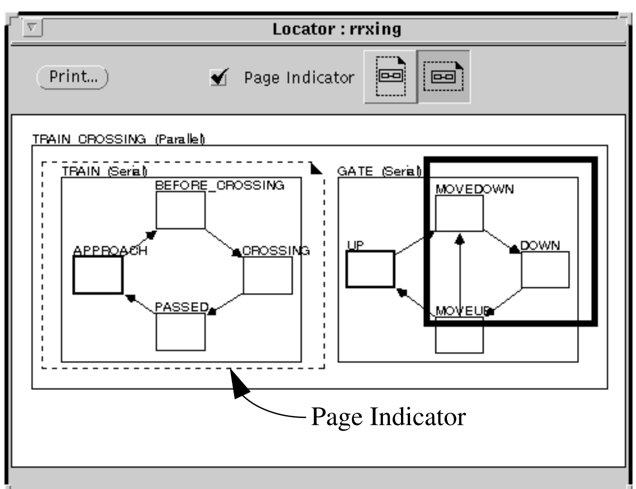
MT displays on your screen. In Chapter 6, you will learn how you can show and hide certain display characteristics. To print a specification:

1. **Click the “Print...” button in the locator window or choose “Print...” from the Specification menu in the specification window.**

   A dialog box will appear asking you to specify the print area, page orientation and output destination.



2. **Enter the information in the print dialog box.**
   - You can print either the entire specification or the contents of the page indicator. To toggle the page indicator on and off, click on the “Page Indicator” box in the locator window.



When the page indicator is shown, you can control its orientation with the horizontal and vertical page buttons to the right of the check box. The page indicator can be manipulated in the same manner as the work window indicator. To move the page indicator, click and hold the Select mouse button inside of the page indicator, and drag it to its new location. To resize the page indicator, click and hold the Select mouse button on the page indicator border, and stretch it to its new size.

- You can indicate whether you want the whole specification printed on the page vertically or horizontally. If you choose to print the area shown by the page indicator, the page orientation will be automatically assigned to that of the page indicator.
- You can send the output directly to the printer and/or save it to a file. If you choose to send the specification directly to the printer, it will be sent to the printer defined by the PRINTER variable. If you choose to send the output to a file, you must specify a filename. If the filename you enter already exists, you will be given the option of overwriting the file or cancelling the print operation.

3. **Click “Apply”.**

   The dialog box will disappear and your specification will be printed and/or saved to a file. Click “Cancel” if you wish to abort the print operation.

What can go wrong?

- You indicated that you wanted to save the output to a file but you forgot to specify a file.
- Your PRINTER variable is not defined.
- Your PRINTER variable defines a printer that is not PostScript compatible.

CHAPTER 4

Navigating through a Modechart Specification

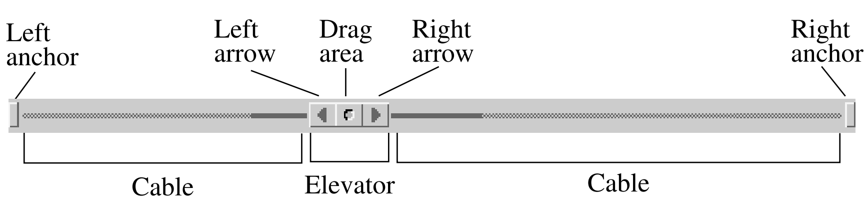
What is navigation?

To display a desired portion of a specification in the work window, the user must often “navigate” through the specification. Good navigation mechanisms are essential in MT, since without such mechanisms a user cannot effectively manage a large specification without getting lost. MT provides three basic navigation mechanisms, namely, scrolling, zooming and searching.

How do I scroll through a specification?

Scrolling is the vertical and/or horizontal movement of the specification in the work window. Scrolling can be accomplished in either of two ways:

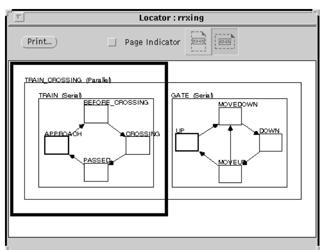
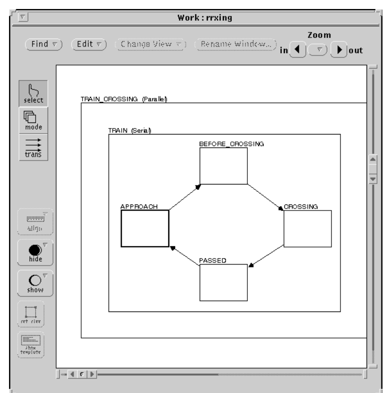
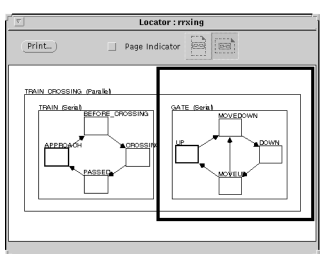
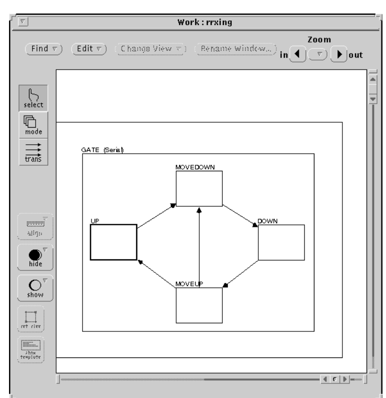
1. Use the scrollbars in the work window.



- ❑ To scroll a screen full at a time, click the cable on the appropriate side of the elevator.
- ❑ To scroll very precisely (small movements), click the appropriate arrow.
- ❑ To scroll to one extreme or the other, click the appropriate anchor.
- ❑ To move to an arbitrary location, click and hold the Select mouse button in the elevator drag area and release it at the desired location.

2. Move the indicator in the locator window.

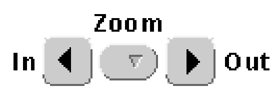
Click and hold the Select mouse button inside of the indicator, and drag the indicator to the desired location. When you release the mouse button, the work window will scroll to the area shown inside of the indicator.

Locator Window	**Corresponding Work Window**
 Initially	
 After Scrolling	

How can I change the zoom level of a specification?

Zooming is the apparent magnification or demagnification of the specification displayed in the work window, so that as much or as little of the specification as desired is displayed. There are two ways to change the zoom level of a specification:

1. Use the zoom controls in the work window.



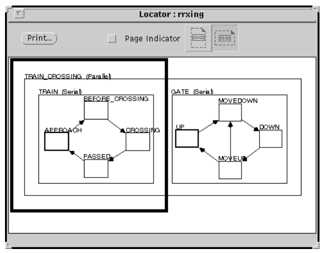
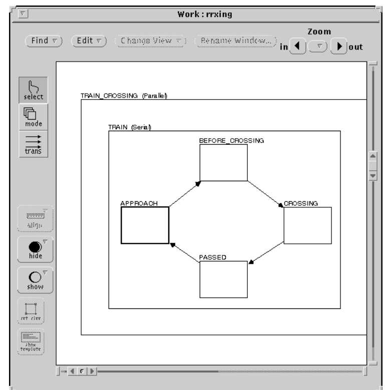
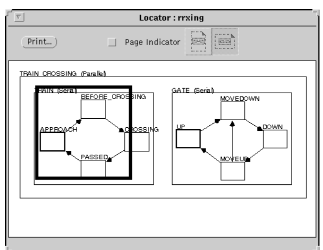
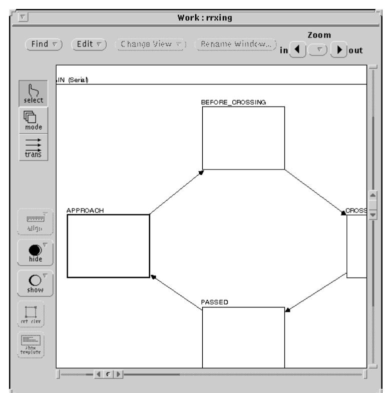
❑ To magnify the specification, click the “In” button.

❑ To demagnify the specification, click the “Out” button.

❑ To magnify or demagnify the specification by powers (2, 4, 8 or 16), choose the appropriate command from the zoom menu button.

2. Resize the indicator in the locator window.

Click and hold the Select mouse button on the indicator’s border, and stretch the indicator to the desired size. When you release the mouse button, the zoom level of the work window will change to correspond to the portion of the specification shown inside of the indicator.

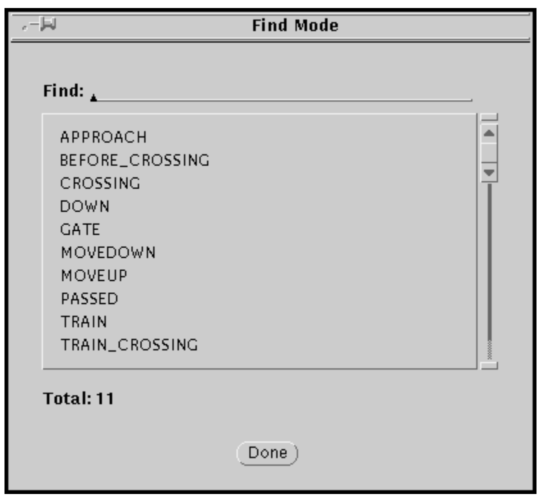
Locator Window	Corresponding Work Window
 Initially	
 After Zooming	

How can I find specific Modechart objects?

1. From the Find menu in the work window, choose the type of find you wish to perform.

To Find:	Do this:
Modes	Choose “Find Mode ...” from the Find menu.
Transitions	Choose “Find Transition ...” from the Find menu.

A dialog box will appear containing a list of all the names of the objects defined in your current specification which correspond to the type of find you are performing. This list will be dynamically updated as you continue to work on your specification, i.e. if you delete a mode, it will be deleted from the list.



2. To display an object in the work window, either click on its name in the scrolling list or type it and hit return.

White space is ignored when you type the name of an object. For transitions, use the following format:

source mode name -> *destination mode name*

Whenever possible, the object you clicked on will be centered in the work window. For modes, the top left corner of the mode will be centered. For transitions, the point where the transition is attached to its source mode will be centered. The object being displayed will not be centered if it requires changing the specification size, i.e. moving the work window outside the current scrollbar boundaries.

3. **When you are finished searching for the objects displayed in the scrolling list, click "Done".**

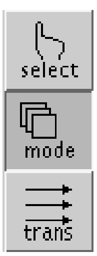
   The "Find" dialog box will disappear.

What can go wrong?

- You entered the name of an object that does not exist in the current specification.

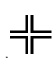
CHAPTER 5

Creating Modechart Objects



How do I create a mode?

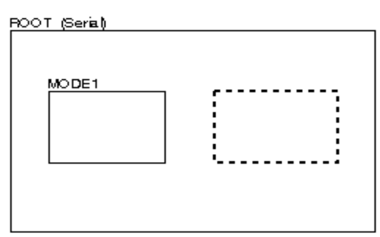
1. **Click the "Mode" tool button in the work window.**

   Your cursor will change to crosshairs, , when you start using the Mode tool.

2. **Inside of the work window canvas, click and hold the Select mouse button while moving the mouse diagonally.**

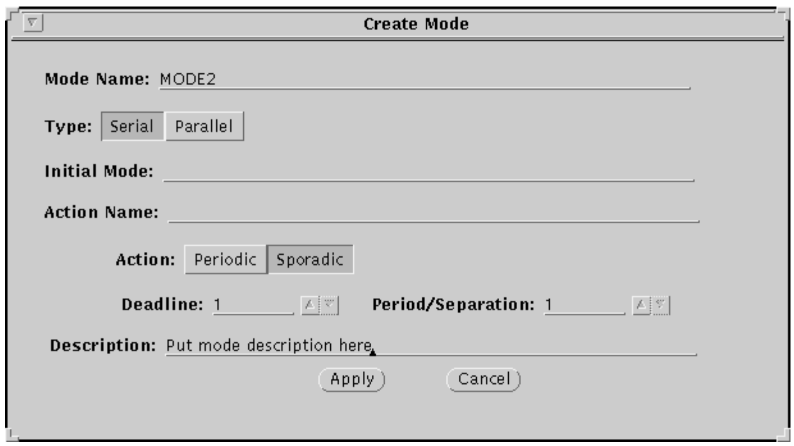
   As you move the mouse, an outline of the mode will be drawn. Upon release of the mouse button, the mode will be drawn as a bold dashed box and a mode template window will appear.

   The mode which immediately contains the box you have drawn will become the parent of this mode and any existing modes contained by the box will become its children. If there is no mode which contains the new mode, it will become a root mode. For example, the parent of the mode being created below is ROOT.



The bold dashed box indicates that you have defined the mode graphics but you have not yet provided the required textual mode information (name, type, action, etc.). The mode template window is the means by which you will provide this information.

3. **Enter the information in the mode template window**.



When creating a mode, you must specify the "Mode name" and "Type" fields in the mode template window. All of the other mode information is optional.

4. **Click "Apply".**

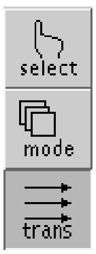
The mode template window will disappear. You have now added a mode to your specification. Assuming it is not an initial mode, it will appear as a plain box with the mode name written on top of it. If it is an initial mode, it will appear as a bold box with the mode name written on top of it. If you change your mind and decide you don't want to add this mode to your specification, click the "Cancel" button.

What can go wrong?

- You positioned the new mode so it overlapped an existing mode.
- The parent of the mode you tried to create is atomic. Remember that even though you cannot explicitly make the type of a mode atomic within the MT you can if you manually create the database file. See Appendix 3 for more details.
- You gave the mode the name of an already existing mode.
- You gave the mode an invalid name. The rules for valid mode names are listed in Appendix 3.
- You gave the mode an invalid initial mode name. The same rules which apply to mode names apply to initial mode names.

- You assigned an action deadline which was greater than the period/separation.

◆ Is creating a mode undoable? **Yes**



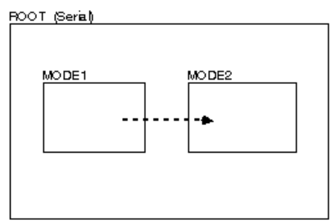
How do I create a transition?

1. **Click the "Trans" tool button in the work window.**

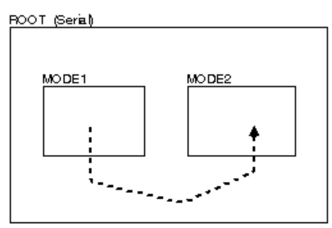
   Your cursor will change to a horizontal arrow, ⟹ , when you start using the Trans tool.

2. **Draw the transition graphics.**

   Two types of transitions can be created, straight transitions and segmented transitions. To create a straight transition, click the Select mouse button inside the transition's source mode and hold it while moving the mouse until it is inside the transition's destination mode.

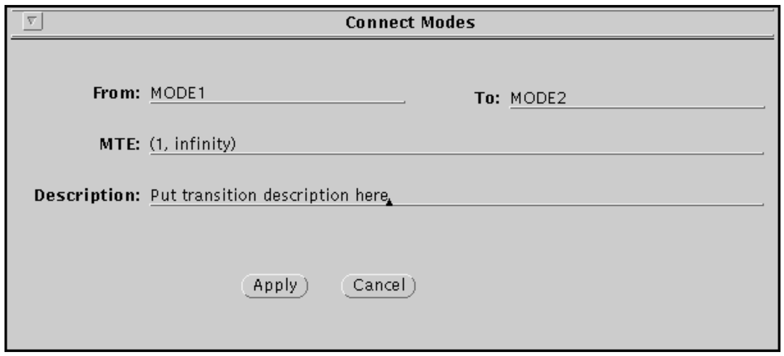


To create a segmented transition, first click the Select mouse button inside of the source mode. Subsequent clicks with the Select mouse button mark the segment endpoints. You can end the transition by double clicking (using the Select mouse button) inside of the destination mode.



When you are done drawing the transition, it will be drawn as a bold dashed arrow and a transition template window will appear. The bold dashed arrow indicates that you have defined the transition graphics but you have not provided the required textual transition information (mode transition expression) yet. The transition template window is the means by which you will provide this information.

3. **Enter the information in the transition template window**.



When creating a transition, you must specify the “MTE” (mode transition expression) field in the transition template window. The “Description” field is optional. The “From” and “To” fields are not editable; they indicate which modes you connected the transition from and to, respectively. The only way to change these fields is to reconnect the transition, after it has been added to your specification. Reconnecting transitions is described in Chapter 6.

4. **Click “Apply”.**

The transition template window will disappear. You have now added a transition to your specification. It will appear as a plain arrow. If you change your mind and decide you don’t want to add this transition to your specification, click the “Cancel” button.

What can go wrong?

- You did not attach the transition to two existing modes. If you tried to start drawing the transition outside of all modes then you will not get any feedback while you are moving the mouse.
- The transition already exists.
- The lowest common ancestor of the source and destination modes is not serial.
- You gave an invalid mode transition expression.

♦ Is creating a transition undoable? **Ye**s

How do I define an action?

(not implemented)

How do I define an external event?

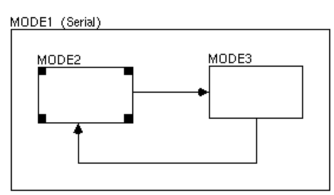
(not implemented)

CHAPTER 6

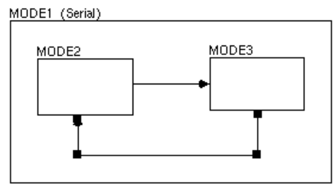
Editing a Modechart Specification

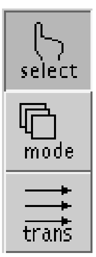
What is a selection set and why do I need one?

The current selection set is the set of modes and transitions that are currently selected. The current selection set is important because most of the edit operations are performed on the selected items. A selection set is useful when you want to apply an operation to several objects at once. Selection handles, which are respresented by small black squares, indicate that a mode or transition is selected. MODE2 is an example of a selected mode.



The transition from MODE3 to MODE2 is an example of a selected transition.





How do I select modes and transitions?

1. **Click the "Select" tool button in the work window.**

   Your cursor will change to a pointing finger,  , when you start using the Select tool. When you leave the Select tool (click on the "Mode" tool or "Trans" tool button), the current selection set will be unselected.

2. **Click the Select mouse button on the mode or transition that you wish to select.**

   You will know that the item has been selected when selection handles appear on it. When you select an item in this manner, all previous selections are unselected.

3. **To extend the current selection set, click the Adjust button on the item you wish to add.**

   Clicking on an already selected item will cause that item to be unselected.

In order to make selecting a group of items less tedious, several short-cuts are provided.

- You can draw a selection box by pressing the Adjust mouse button and dragging diagonally until the selection box contains all of the modes and/or transitions you wish to select.

  When you release the mouse button, all of the items contained entirely within the selection box will be selected. If an item contained within the selection box is already selected, it will be unselected.
- You can select a mode and all of the modes and transitions contained within that mode by holding down the Shift key and then clicking the Select mouse button.
- You can select all of the modes and transitions in your specification by choosing the "Select All" command in the Edit menu in the work window.

The following table is provided as a quick how-to reference for performing select operations.

To:	Do this:
Select a mode or transition	Click on it with the Select mouse button.
Extend the selection	Click on it with the Adjust mouse button.
Select everything	Choose “Select All” from the Edit menu.
Unselect an item	Click on it with the Adjust mouse button.
Unselect everything	Click outside of specification with the Select mouse button or choose “Unselect All” from the Edit menu.
Select everything in a mode	Hold down the Shift key and click the Select mouse button on it.
Toggle selections in a user defined box	Drag with the Adjust mouse button to create a selection box.

♦ Is selecting modes and transitions undoable? **No**

How do I undo/redo previous operations?

1. **Select “Undo” from the Edit menu in the work window.**

   Assuming all the operations you have performed are undoable, you can undo all the way back to the state you were in prior to performing those operations. The “Undo” command will be disabled when there are no more operations that can be undone.

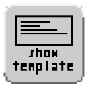
   To help you remember which operation you are undoing, the “Undo” command will be appended with the name of the operation you can currently undo. For example, if you just moved a set of modes, the “Undo” command will be labelled “Undo Move”.

2. **To redo, select “Redo” from the Edit menu.**

   You can redo all operations back to the state that you were in prior to performing any undos. The “Redo” command will be disabled when there are no more undos that can be redone.

   To help you remember which operation you are redoing, the Redo command label will be appended with the name of the operation you can currently redo. For example, if you just performed an “Undo Move”, the redo command will be labelled “Redo Move”.

How do I change mode and transition information?



1. **Select the modes and transitions that you wish to change.**
2. **Click the "Show Template" button in the work window.**

   The information template windows (the same windows used to create modes and transitions) will appear for all items in the current selection set.
3. **Edit the template window.**

   You can edit any of the fields that you could specify when creating the item. For a mode, you can change:
   - the mode name,
   - the mode type (not implemented),
   - the initial mode (with the restriction that it can only be changed to an existing child of the mode being changed),
   - the action associated with a mode and its type and timing parameters (not implemented),
   - and the mode description.

   For a transition, you can change:
   - the transition expression (not implemented), and
   - the transition description.
4. **Click "Apply".**

   The template window will disappear. If you decide you do not want to change a mode or transition, click the "Cancel" button instead.

What can go wrong?

- You changed the mode name to a syntactically invalid mode name. The rules for valid mode names are listed in Appendix 3.
- You changed the mode name to the name of an already existing mode.
- You changed the initial mode name to a syntactically invalid mode name. The same rules which apply to mode names apply to initial mode names.
- You changed the initial mode to a mode which is not an exisiting child of the mode being changed.

◆ Is changing mode and transition information undoable? **Yes**

How do I delete modes and transitions?

1. **Select the modes and transitions that you wish to delete.**
2. **Choose the "Delete" command from the Edit menu in the work window.**

   There may be a few side effects when items are deleted from your specification. Deleting a mode will cause all transitions connected to it to be deleted, regardless of whether or not they are selected. Also, the children of a deleted mode will be "adopted" by their grandparent (parent of deleted mode).

◆ Is deleting modes and transitions undoable? **Yes**

How can I move modes and transitions?

1. **Select the mode(s) and/or transition(s) you want to move.**
2. **Click and hold the Select mouse button inside of one of the selected objects and drag them to their new location.**

   As you move the mouse, an outline of the selected items will follow the mouse. When you release the mouse button, the selected objects will move to their new location and any transitions (selected or not) that are attached to one mode which moved and one which stayed in place will be stretched.

   Currently, you are not allowed to change the mode hierarchy when moving modes. When you perform a move operation, it will be bounded in such a way that you cannot move any of the modes outside of their parents. The only way you can change the existing mode hierarchy is to delete the mode of interest and then add it again in its new hierarchy position.

   You are also not allowed to move a transition so it becomes detached from its source and destination modes. If there is a transition in your selection set and neither its source or destination modes are selected, then the move will be bounded so that you cannot detach the transition.

What can go wrong?

- You moved the mode(s) so at least one overlapped an existing mode.
- You moved the mode(s) so at least one became the child of its sibling.

◆ Is moving a mode undoable? **Yes**

How can I resize a mode?

1. **Select the mode you wish to resize.**
2. **Click and hold the Select mouse button inside of one of the mode handles, and drag the handle to its new location.**

   While dragging the mouse, an outline of the resized mode will be drawn. The corner of the mode opposite to the handle you grabbed will stay fixed. When you release the mouse button, the mode will be resized. Any transitions attached to the resized mode will be stretched. Any other items in the current selection set will not be altered.

   Currently, you are not allowed to change the mode hierarchy when resizing a mode, and the original children (if any) must be kept as the only children. When you perform a resize operation, it will bounded to the parent mode of the mode being resized.

What can go wrong?

- You resized the mode so it overlapped an existing mode.
- You resized the mode so it became the parent of one or more of its siblings.

♦ Is resizing a mode undoable? **Yes**

How can I move a transition point?

1. **Select the transition you wish to alter.**
2. **Click and hold the Select mouse button inside the transition handle you wish to move, and drag the handle to its new location.**

   While dragging the mouse, the outline of the transition point being moved will be drawn. All of the other transition point(s) will stay fixed. When you release the mouse button, the transition point will be moved to its new location. If the transition point being moved is a source or destination point, the point will snap to the mode edge it intersect when the move is done. Any other items in the current selection set will not be altered.

   Currently, you cannot change the source or destination mode of a transition, just the location of a connection to one of those modes.

What can go wrong?

- You did not reconnect the transition to an existing mode.
- You changed the source or destination mode.

◆ Is moving a transition point undoable? **Yes**

How can I add a transition point?

1. **Select the transition you wish to alter.**
2. **Click and hold the Adjust mouse button on the transition between two selection handles, and drag the mouse to the location of the new point.**

   While dragging the mouse, the outline of the transition point being added will be drawn. All of the other transition point(s) will stay fixed. When you release the mouse button, the transition point will be added at its new location. Any other items in the current selection set will not be altered.

◆ Is adding a transition point undoable? **Yes**

How can I delete a transition point?

1. **Select the transition you wish to alter.**
2. **Click the transition handle you wish to delete with the Menu mouse button.**

   The transition will be redrawn without the handle you just deleted. You are not allowed to delete the source or destination handles. The only way to delete the source or destination handles is to delete the entire transition. Any other items in the current selection set will not be altered.

What can go wrong?

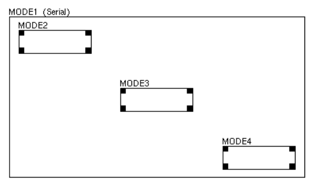
- You tried to delete a source or destination handle.

◆ Is deleting a transition point undoable? **Yes**

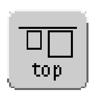
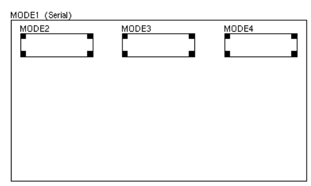
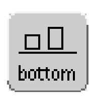
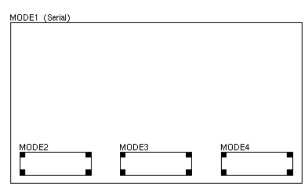
How can I align a set of modes?

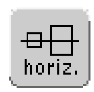
1. **Select the set of modes that you wish to align relative to one another.**
2. **Under the “Align” menu button in the work window, choose the type of align operation you wish to perform.**

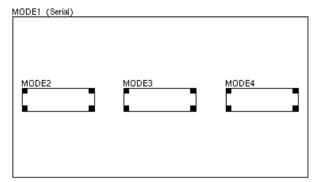
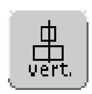
You can align modes to the left, to the right, to the top, to the bottom, horizontally or vertically. To understand how align works, picture an imaginary box around the set of modes you have selected. A left align will try to move the selected modes to the left edge of the imaginary box. A vertical align will try to move the centers of the modes to the vertical center of the imaginary box. Let's consider this example.

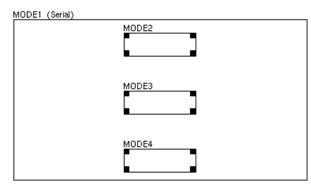
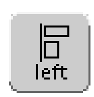


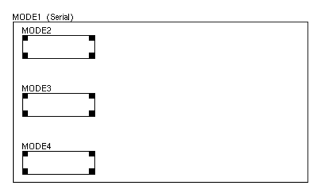
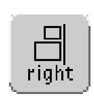
The following table shows the results of performing the different align operations on the given example.

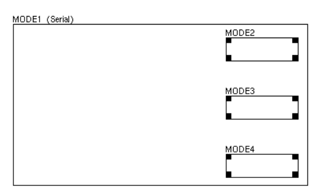
Press:	**Result:**
 Align Top	
 Align Bottom	

<table>
<tr><th>Press:</th><th>Result:</th></tr>
<tr><td>

Align Horizontal</td><td>
</td></tr>
<tr><td>

Align Vertical</td><td>
</td></tr>
<tr><td>

Align Left</td><td>
</td></tr>
<tr><td>

Align Right</td><td>
</td></tr>
</table>

What can go wrong?

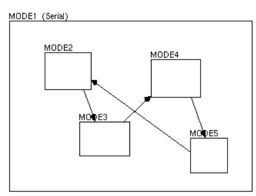
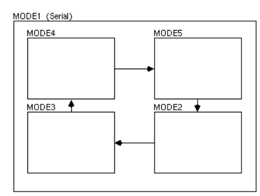
- You requested a set of modes to be aligned that would cause modes to overlap.

♦ Is aligning a set of modes undoable? **Yes**

How can I automatically rearrange a portion of my specification?

The MT has the capability to automatically rearrange all of the modes and transitions contained within the mode(s) you select. The purpose of this feature is to attempt to provide you with a visually pleasing arrangement. To perform this automatic rearrangement:

1. **Select the mode(s) whose contents you wish to have automatically rearranged.**
2. **Choose the "Re-lay" command from the Edit menu in the work window.**

Before:	**After:**
	

♦ Is rearranging a portion of a specification undoable? **No**.

Since "Re-lay" is not an undoable command you may wish to save your specification before you perform this operation in case you do not like the automatic layout. You can revert to the saved version of your specification by choosing the "Revert to Save" command under the Specification menu in the Specification window.

How can I automatically center transitions?

1. **Select the transition(s) you wish to center.**

   If you do not select any transitions, by default, all the transitions will be centered.
2. **Choose the "Center Transitions" command from the Edit menu in the work window.**

♦ Is centering transitions in a specification undoable? **No**.

Since “Center Transitions” is not an undoable command you may wish to save your specification before you perform this operation in case you do not like the new positions of the transitions. You can revert to the saved version of your specification by choosing the “Revert to Save” command under the Specification menu in the Specification window.

How can I make transition(s) snap horizontally or vertically?

1. **Select the transition(s) you want to snap horizontally or vertically.**
2. **Choose the “Snap Trans Hor/Ver” command from the Edit menu in the work window.**

   A transition can only be snapped horizontally or vertically if the positions of it’s source and destination modes will allow it. This command does not move the source and destination modes. Also, this command can only handle straight transitions. If any segmented transitions are selected, they will be ignored.

What can go wrong?

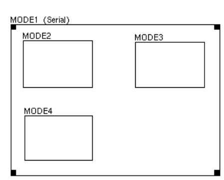
- The positions of the transition’s source and destination mode may not allow the transition to be snapped horizontally or vertically.

♦ Is snapping transitions in a specification undoable? **No**.

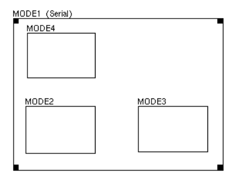
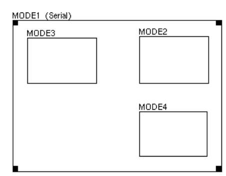
How can I automatically flip the contents of a mode?

1. **Select the mode whose contents you want to flip.**
2. **Under the Edit menu in the work window, choose the type of flip you wish to perform.**

   You can flip the contents of a mode up and down or left and right. For example, let’s say you want to flip the contents of MODE1.



The following table shows the results of performing the different types of flip operations on MODE1.

Choose:	**Result:**
Flip U/D	
Flip L/R	

♦ Is flipping the contents of a mode undoable? **Yes**.

Which display characteristics can I show and hide?

The mode display characteristics that you can hide or show include:

- the mode name,
- the mode type,
- the mode action, and
- the entire mode (graphics and all text).

The transition display characteristics that you can hide or show include:

- the transition expression, and
- the entire transition (graphics and all text).

How can I hide certain display characteristics?

1. **Select the modes and transitions that you want to have partially/entirely hidden.**
2. **Under the "Hide" menu button in the work window, choose the type of hide you wish to perform**.

Hide:	Do this:
All selected modes and transitions †	Choose "Modes and Transitions" from the Hide menu.
All text associated with items in current selection set	Choose "All Text" from the Text Only submenu of the Hide menu.
All selected transitions	Choose "Transitions Only" from the Hide menu.
All transition text of currently selected transitions	Choose "All Transition Text" from the Text Only submenu of the Hide menu.
All selected modes †	Choose "Modes Only" from the Hide menu.
All mode text of currently selected modes	Choose "All Mode Text" from the Text Only submenu of the Hide menu.
The mode names of currently selected modes	Choose "Mode Names" from the Text Only submenu of the Hide menu.
The mode types of currently selected modes	Choose "Mode Types" from the Text Only submenu of the Hide menu.
The mode actions of currently selected modes	Choose "Mode Actions" from the Text Only submenu of the Hide menu.

† When a mode is hidden, all of the transitions attached to it are also hidden, regardless of whether or not they are selected.

♦ Is hiding mode and transition display characteristics undoable? **Yes**

How can I show certain display characteristics?

1. **Select the modes and transitions that you want to perform a show operation on.**

   Currently, there is not a way to select modes or transitions whose graphics are hidden, so any of the show operations that cause the graphics of an object to be shown are performed on all objects that are currently hidden. All other show operations are performed on the current selection set.

2. **Under the “Show” menu button in the work window, choose the type of show you wish to perform**.

Show:	**Do this:**
Everything	Choose “Show All” from the Edit menu.
All hidden modes and transitions	Choose “Hidden Modes and Transitions” from the Show menu.
All text associated with items in current selection set	Choose “All Text” from the Selected Text submenu of the Show menu.
All hidden transitions	Choose “Hidden Transitions” from the Show menu.
All transition text of currently selected transitions	Choose “All Transition Text” from the Selected Text submenu of the Show menu.
All hidden modes	Choose “Hidden Modes” from the Show menu.
All mode text of currently selected modes	Choose “All Mode Text” from the Selected Text submenu of the Show menu.
The mode names of currently selected modes	Choose “Mode Names” from the Selected Text submenu of the Show menu.
The mode types of currently selected modes	Choose “Mode Types” from the Selected Text submenu of the Show menu.
The mode actions of currently selected modes	Choose “Mode Actions” from the Selected Text submenu of the Show menu.

♦ Is showing mode and transition display characteristics undoable? **Yes**

CHAPTER 7

Checking for Consistency and Completeness

What are consistency and completeness checks?

Consistency and completeness checks can help you find instances of incompleteness and undesired behavior in your specification. For example, one of the checks can detect and list the “sink modes” in your specification. (A sink mode is a mode that is the destination of one or more transitions but the source of none.) Note that the checker may indicate behavior that you defined intentionally (such as nondeterministic behavior). It is your responsibility to decide if the indicated behavior is acceptable and, if it is not, to correct it.

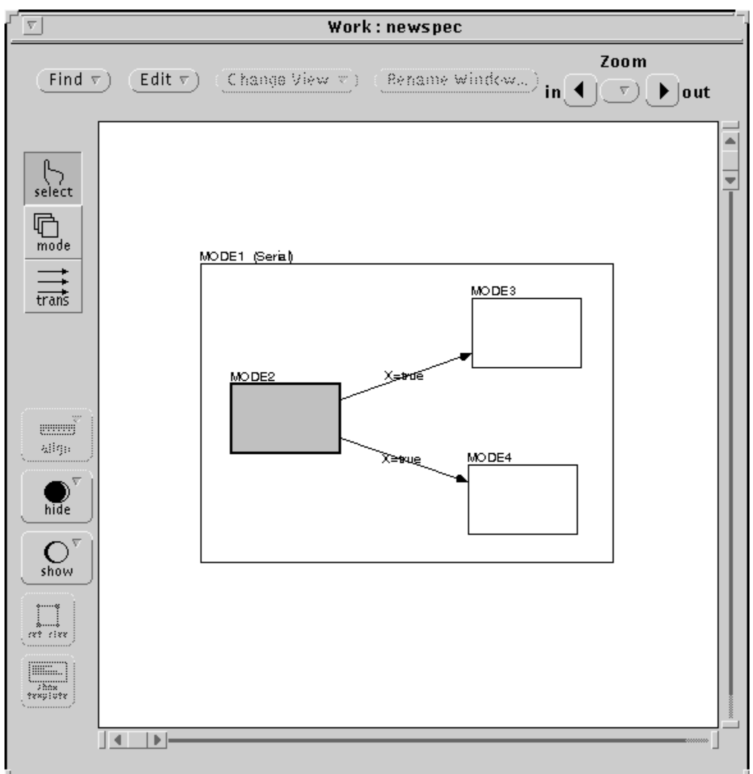
All of the checks are dynamic except where noted. A check is dynamic if after performing the check, you can continue work on your specification, and the check continues to be applied. For example, if a check indicates that MODE1 is a sink mode and then you create a transition out of MODE1, this mode is no longer a sink mode, so the checker will remove it from the list of sink modes. As you continue to work on your specification, you may also create more sink modes. If so, the checker will add each new sink mode to its list of sink modes. The checker will continue to update the list of sink modes until you indicate that you want the sink modes check turned off.

How can I check for statically nondeterministic transitions in a specification?

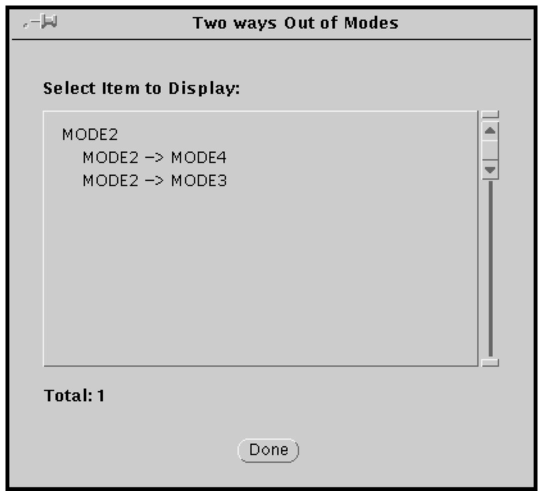
1. **Choose "Modes with nondeterministic transitions" from the CC Checks submenu of the Analysis menu in the specification window.**

   This check finds statically nondeterministic transitions. The checker does this by comparing the transition expressions of all the transitions with the same source mode. If any two or more of the expressions are equal, the source mode has nondeterministic transitions. This check does not take timing constraints into consideration. The verifier (Chapter 9) provides a check for dynamic nondeterminism, i.e. two transitions with the same source mode are eligible to be taken at the same time.

   All of the modes in your specification which have two or more static nondeterministic ways out of them will appear highlighted (filled with gray). It should be noted that you need to use a color or gray scale monitor to see the highlights. In the following example, the transitions out of MODE2 are nondeterministic; i.e. the system will arbitrarily choose which transition to take.



A pop-up window will appear containing a sorted list of all the modes which have two or more non-deterministic ways out of them. The set of transitions which cause the nondeterminism will be listed immediately after the corresponding source mode.



2. **To display a particular mode or transition in the work window, click on the mode/transition name in the list.**

   The lists shown by consistency and completeness checks behave like the lists displayed by the “Find” operations described in Chapter 4.

3. **When you are done checking for nondeterminism, click “Done”.**

   The pop-up window will disappear and all of the highlights in your specification will be turned off.

How can I check for modes with self transitions?

1. **Choose “Modes with self transitions” from the CC Checks submenu of the Analysis menu in the specification window.**

   A pop-up window will appear containing a sorted list of all the modes in your specification which have self transitions.

2. **To display a particular mode in the work window, click on the mode name in the list.**

3. **When you are done checking for self transitions, click “Done”.**

   The pop-up window will disappear and all of the highlights in your specification will be turned off.

How can I check for serial modes with no initial mode specified?

1. **Choose "Serial modes with no initial" from the CC Checks submenu of the Analysis menu in the specification window.**

   A pop-up window will appear containing a sorted list of all the serial modes in your specification for which you have not specified an initial mode. This list will also include any serial modes that identify an initial mode that does not exist.
2. **To display a particular mode in the work window, click on the mode name in the list.**
3. **When you are done checking for serial modes with no initial, click "Done".**

   The pop-up window will disappear and all of the highlights in your specification will be turned off.

How can I check for modes which can be entered but never exited?

1. **Choose "Sink modes" from the CC Checks submenu of the Analysis menu in the specification window.**

   A pop-up window will appear containing a sorted list of all the modes in your specification which can be entered but never exited.
2. **To display a particular mode in the work window, click on the mode name in the list.**
3. **When you are done checking for sink modes, click "Done".**

   The pop-up window will disappear and all of the highlights in your specification will be turned off.

How can I check for modes which can never be entered?

1. **Choose "Unreachable modes" from the CC Checks submenu of the Analysis menu in the specification window.**

   A pop-up window will appear containing a sorted list of all the modes in your specification that are statically unreachable.
2. **To display a particular mode in the work window, click on the mode name in the list.**
3. **When you are done checking for unreachable modes, click "Done".**

The pop-up window will disappear and all of the highlights in your specification will be turned off.

How can I check if two modes can be active simultaneously?

This check indicates whether two modes may be active simultaneously based only on the mode hierarchy (which modes are parallel and which are serial).

1. **Select the two modes that you want to check for being active simultaneously.**
2. **Choose "Can modes be active together" from the CC Checks submenu of the Analysis menu in the specification window.**

   This check is not updated dynamically. A message will appear indicated whether the two selected modes can be active simultaneously or not.
3. **Click "OK" to remove the message.**

How can I check which modes must be active when the system is in a given mode?

This check indicates which modes must be active when the system is in a given mode based only on the mode hierarchy (which modes are parallel and which are serial).

1. **Select the mode that you want to check.**
2. **Choose "Modes that must be active together" from the CC Checks submenu of the Analysis menu in the specification window.**

   A pop-up window will appear containing a sorted list of all the modes in your specification which are active when the selected mode is active.
3. **To display a particular mode in the work window, click on the mode name in the list.**
4. **When you are done with this check, click "Done".**

   The pop-up window will disappear and all of the highlights in your specification will be turned off.

How can I check the values of the state variables when the system is in some particular mode(s)?

1. **Select the mode(s) that you want to check.**
2. **Choose "Derived state variable values" from the CC Checks submenu of the Analysis menu in the specification window.**

   A pop-up window will appear containing a sorted list of the state variable values when the selected modes are active.
3. **When you are done checking the state variable values, click "Done".**

   The pop-up window will disappear.

How can I find any transitions which are implicitly defined?

An implicit transition is a transition that is still defined in your specification, but it does not exist from the user's point of view because its source mode and/or destination mode are no longer defined.

1. **Choose "Implicit transitions" from the CC Checks submenu of the Analysis menu in the specification window.**

   A pop-up window will appear containing a sorted list of the implicitly defined transitions in your specifications.
2. **When you are done finding implicit transitions, click "Done".**

   The pop-up window will disappear.

CHAPTER 8

Simulating a Modechart Specification

Why would I want to simulate a Modechart specification?

Simulating a specification allows you to view an execution of your system. This execution may reveal unintended behavior, may help to record timing properties or collect timing data about the execution of the system, and in general may serve many of the same purposes as a rapid prototype.

It should be pointed out that an execution of a system viewed through simulation is not the same as verification. It is not guaranteed that the behavior of the system observed through simulation will always be the same; the system may behave differently in future executions due to random events, different patterns of external events, different bindings of nondeterministic choices, etc. Behavior certified by verification is behavior true of all executions. Simulation behavior, while certified to be correct behavior under the semantics of Modechart, shows only one possible execution path each time.

What are the prerequisites for simulating a Modechart specification?

In order to simulate a specification, the following must exist:

- *a Modechart specification*

You can simulate either the specification currently displayed by the MT or a previously saved specification. Chapter 3 explains how to save a specification. For more details, the required format of a Modechart specification file is described in Appendix 3.

- *a simulation options file*

  In the following two sections, you will learn what an options file is and how to create one or let the simulator create one for you.

What is a simulation options file?

A simulation options file provides the following information to the simulator:

- How long to run (although you have the option to shorten or extend this time once the simulation begins).
- What values to assume about the Modechart specification if such values were left unspecified. Values that, under some circumstances, may be unspecified include the minimum and maximum execution times for actions, initial values for state variables, minimum separation times for external events, and lower and upper bounds on timed mode transitions.
- What decisions to make when nondeterminism is encountered during execution.
- When to suspend execution so that the user may inspect and modify the system state. A suspension time may be specified as a clock time, a time period, or an event whose occurrence will cause the suspension.
- Which objects in the specification the user is interested in recording information about, and in which order.

The form and contents of a simulation file are described in Appendix 4.

How can I create a simulation options file?

1. **Outside of the MT, invoke "options_maker".**

   This utility program should be installed on the same machine that is hosting the MT and in the same directory. Also, the environmental variable SIMOPTCMD should be set to the location and name of this program. See Chapter 2 for more details. It accepts a Modechart specification file on its standard input and writes a corresponding options file to its standard output. The MT assumes that the name of an options file corresponding with a given specification is the name of that specification appended with ".options". Appendix 4 contains a sample options file.

2. **Edit the generated options file.**

   Once the options file is generated, you should edit it to set the options as desired. The options file is self-documenting and will not be described here.

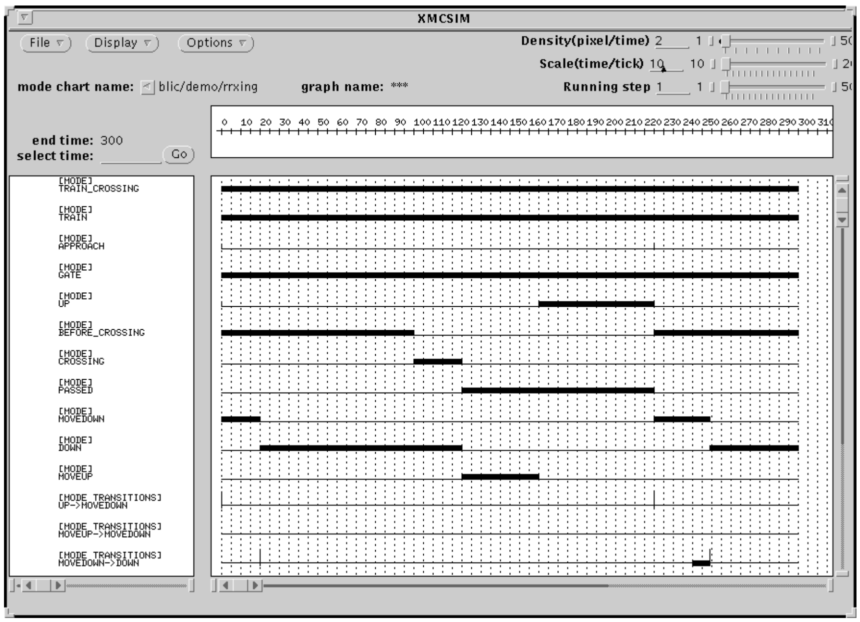
How can I get the simulator to create an options file for me automatically?

When you simulate the specification currently displayed by the MT, there are two cases when an options file will be automatically generated for you: (1) if you have made any un-saved changes to the current specification, and/or (2) if there is no options file associated with the current specification. For case (1), it should be noted that if an options file already exists for the specification to be simulated, it will be ignored. Automatically generated options files are not saved after simulation.

How do I simulate the specification currently displayed in my work window?

1. **Choose "Simulate" under the Analysis menu in the specification window.**

   A shell window will appear that hosts the simulator process; this shell window is used in dialog mode, which is explained later. A simulation window will also appear that displays the results of the simulation so far. The simulation will be suspended when either (a) the run length specified in the options file expires, or (b) a breakpoint event, as specified in the options file, is encountered. In the following example, the simulation is suspended at time 300.



Each object for which a "log" display was requested in Section IV of the options file is given a graph line in the window; the order is that in which the objects were named in the options file. The graph line tells the story of the simulation for each of the different objects..

Graph Line	Modes	Transitions	External events	Actions
thick solid line	active	satisfied	n/a	in progress
thin line	inactive	not satisfied	did not occur	not in progress
vertical bar	n/a	transition taken	event occurred	action aborted
======== open line	n/a	satisfied, not taken	n/a	n/a

The results of the simulation will be written to a log file. The default name of this log file is the specification name appended with ".log".

What can go wrong?

- You have not defined the environmental variables SPAWNCMD, SIMOPTCOMD and/or SIMULATECMD. Refer to Chapter 2 for more details.

How can I continue a suspended simulation?

To continue a simulation until either a breakpoint is encountered, or the simulation run length expires:

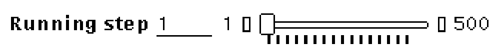
1. **Choose “run” under the Options menu.**

   If the simulation run length has expired, you can step through your simulation or load another options file.

To continue a simulation for some specified number of time units:

1. **Choose “step” under the Options menu.**

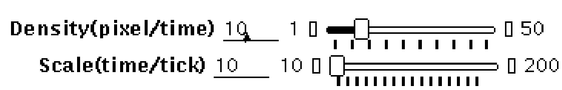
   The step length may be adjusted by using the “Running step” slider:



   or by typing in the desired value in the “Running step” field and hitting return. Remember that the simulation will suspend again before the step increment has expired if a breakpoint is encountered first.

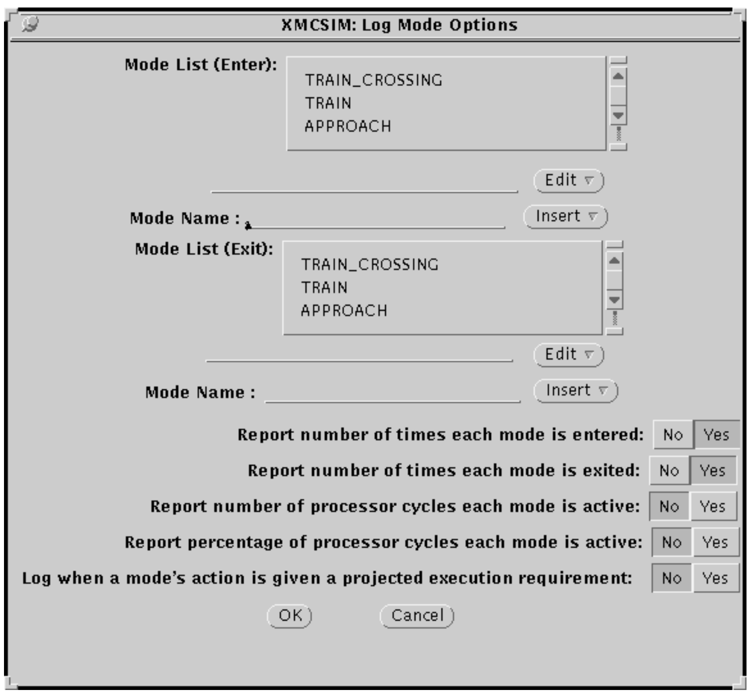
How can I change what is displayed?

- You can scroll and resize the simulation window to show the time units and/or objects which are of interest.
- You can change the “Density” and “Scale” fields which control the amount of information displayed per horizontal unit. This can be done by either dragging the corresponding slider back and forth, or by typing in the desired value and hitting return.



The "Density" value indicates how many display pixels a single time unit is allocated on the display and the "Scale" value determines the scale of the time line across the top of the window. "Scale" values are limited to multiples of 10.

- You can turn the time grid on and off by choosing "Toggle Grid" in the Display menu.
- You can move to the time bar (if displayed) by choosing "Goto Time Bar" in the Display menu. The time bar functionality is described later in this chapter.
- In the log section of the options file, you can specify which Modechart objects you wish to display and the order you wish to display them. You can then effect your changes by choosing "load options file" under the Options menu.
- You can interactively change some of the logging options by choosing "change simulator options" under the Options menu. Currently, you can change mode and transition logging options[1]. The following window appears when you change the mode logging options.



1. A future enhancement is to implement interactive logging options for actions, external events, and state predicates.

This window allows you to change the mode logging options, specifically:

- which modes to log when they are entered,
- which modes to log when they are exited,
- the order which the modes are logged[2], and
- which mode statistics to log.

To add a mode to be logged, type the name of the mode in the “Mode Name:” field, select a mode in the corresponding list, and then choose “Before”, “After”, “Top” or “Bottom” from the Insert menu.

To delete a mode from one of the mode lists, select the mode and then choose “Cut” from the Edit menu.

To change the position of a mode in the list, select the mode, choose “Cut” from the Edit menu, select mode in the list, and these choose “Paste before” or “Paster after” from the Edit menu.

All of the changes you make will take effect when you click “OK” (but the changes will not affect the contents of the options file you specified at the beginning of the simulation). This procedure is similar to the one for changing transition logging options.

How can I get information about the state of my system when the simulation is suspended?

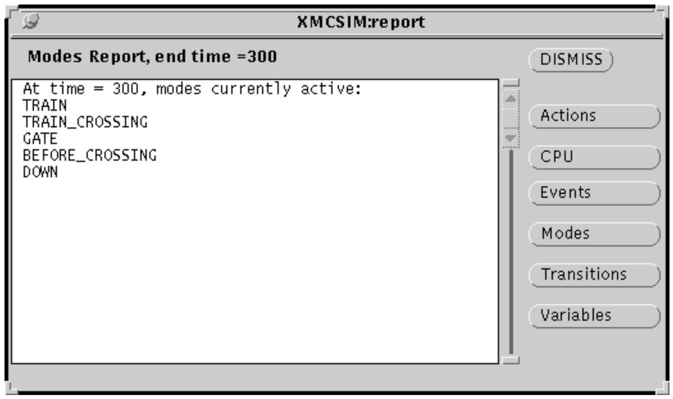
1. Choose “report ...” from the Options menu.

A pop-up window will appear which will allow you to perform several types of system queries.

Click:	**To get a report on:**
Actions	What actions are in progress
CPU	The current state of the CPU (idle or active)
Events	What external events occurred at this time
Modes	Which modes are currently active
Transitions	Which transitions are currently enabled
Variables	The current value of all state variables

2. The order of the mode lists also corresponds to the order that the modes are displayed in the graph. If the order of the two lists conflicts, the mode entry list has priority. A pending change is to merge these two lists.

The following is an example of an active modes report.

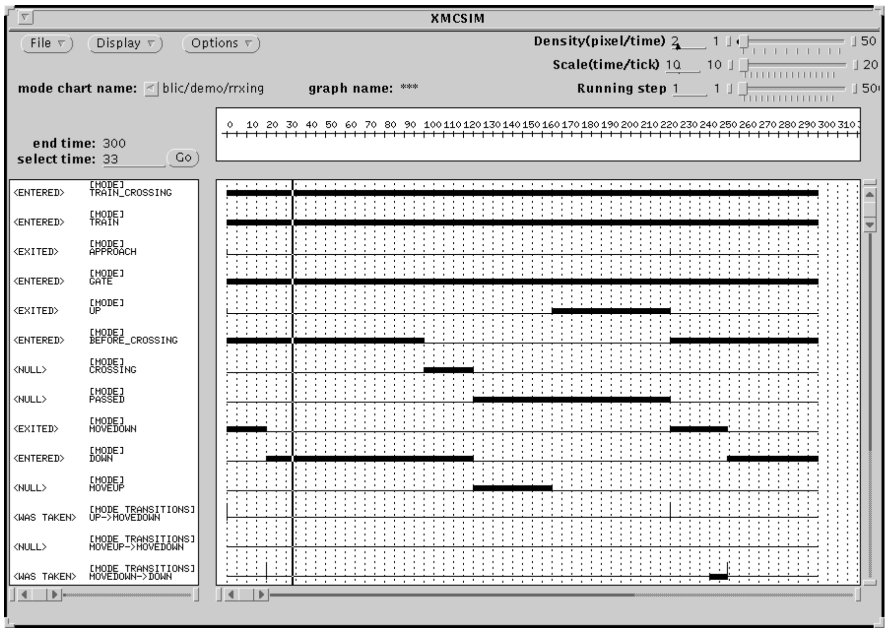


2. **To end your report queries, click "DISMISS".**

How can I get information about my simulation at selected times?

1. **Click the Select mouse button (in the graph) at the time of interest OR type the time in the "select time:" field in the simulation window and click "GO".**

   A vertical time bar will appear at the time indicated. If you did not type a time in the "select time:" field and then clicked "GO", the time bar will appear at time 0. The state of the system at the time indicated by the time bar is shown to the left of the object names. The following example shows the state of the system at time 33.



2. **To move the time bar, click and hold the Select mouse button on the time bar and drag the bar to its new location.**

   The time bar can also be moved using the methods described in step #1. Once displayed, the time bar may not be deleted.

What is dialog mode and how can I enter it?

You have a few more options to control the state of the simulation when in dialog mode. Dialog mode is text-based, but in a future version of the simulation software used in MT, it will be window-based.

1. **Choose "dialog ..." under the Options menu.**

   When dialog mode is entered, the simulation writes to the shell window which appeared when you invoked the simulator; you can type commands in that window.

2. **To see a list of commands available in dialog mode, type "help".**

   Note that several of these commands are redundant with features available via the graphical user interface.

```
Simulation suspended at time 50.  Type 'help' for list of commands.
>>help

Commands/abbreviations are:
     abort:       abort a specified action now in execution
     action,a,|-: see what actions are currently in execution
     break:       add an event or a time to the breakpoint list
     cpu:         inquire which action has the cpu
     event,e,^:   see what events have just occurred
     help:        display this message
     mode,m,{}:   see what modes are currently active
     nobreak:     remove an event or a time from the breakpoint list
     options:     reset all simulator options from a file
     preserve:    write current options to a named options file
     quit,q:      in MCSIM, exit simulator; in XMCSIM, exit dialog mode
     restart:     restart the simulation from a snapshot file
     run,r:       resume simulation
     signal:      cause a specified, eligible external event to occur now
     snap:        save a snapshot of the simulation, for later startup
     step:        cause the simulation to take a set number of steps
     take:        cause a specified, eligible transition to be taken
     time,t:      print the simulated time
     trans,->:    see what transitions are currently eligible
     var,v:       see what the current values of state variables are
Command keywords are not case sensitive.
Enter 'help' and a command keyword for specific information.

>>
```

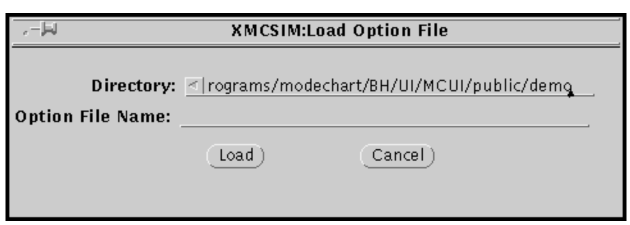
3. **To exit dialog mode, type "quit".**

WARNING: When you are in dialog mode and the simulator is waiting for input, clicking on or entering data in another window will freeze up your computer. If this happens, kill the simulator process. You will lose any work you were doing with the simulator but your specification should be unchanged otherwise. If you kill the process running the MT, you will lose any changes you have made to your current specification.

How can I load a new options file?

1. **Choose "Load options file ..." from the Options menu.**

   A window will appear:.



2. **Type in the directory and the name of the options file.**

3. **Click “Load”.**

   If you decide that you do not want to load a new options file, click “Cancel”.

What can go wrong?

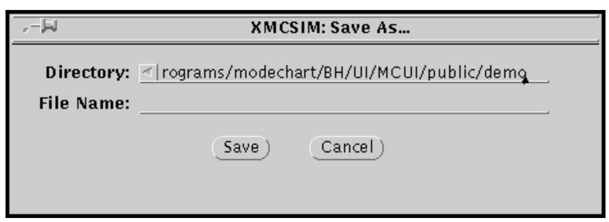
- The options file referenced by the given directory and filename does not exist.
- The file does not have the correct permission for you to read it.
- The options file you specified does not correspond to the specification you are simulating.

How can I save a simulation graph for later viewing?

This operation allows you to save a simulation graph which may be loaded again later for viewing. A saved simulation graph may only be viewed; it may not be used to re-trace or re-start simulation..

1. **Choose “Save Graph to file ...” under the File menu.**

   A window will appear asking for a directory and a filename.



2. **Type in the directory and the name to save the graph under.**
3. **Click “Save”.**

   If you decide that you do not want to save the simulation graph, click “Cancel”.

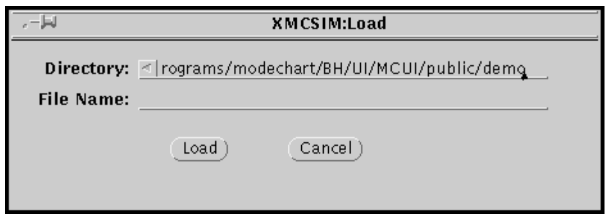
What can go wrong?

- You entered a directory which does not exist.
- You do not have write permission in the directory you specified.
- The graph name you specified already exists and you do not have permission to write over it.

How can I view a previously saved simulation graph?

1. **Choose "Load Graph from file ..." under the File menu.**

   The following window will appear:



2. **Type in the directory and the name of the graph to be loaded.**
3. **Click "Load".**

   If you decide that you do not want to load the graph, click "Cancel". Loading a simulation graph has the effect of quitting the simulation of the current modechart, as described below.

What can go wrong?

- The graph referenced by the given directory and filename does not exist.
- The file does not have the correct permission for you to read it.

How can I quit simulating a specification?

1. **Choose "quit simulation" under the Options menu.**

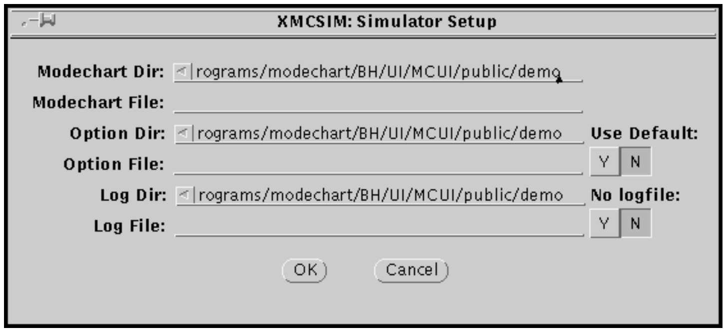
   At this point, you can either load another specification and options file, or you can exit the simulator and return to the MT. Prior to quitting the simulation, you will be given the option to cancel.

How can I simulate a different specification?

1. **Choose "Load Modechart files ..." from the File menu.**

   The following simulator file setup window will appear:

   Loading a new Modechart specification to be simulated does not change the specification that is currently displayed in the work window.



2. **Type in the name of the Modechart specification**
3. **Either type in the name of the corresponding options file, or choose “Y” under “Use Default”.**
4. **Either type in the name of the corresponding log file, or choose “Y” under “No logfile”.**
5. **Click “OK”.**

   If you decide you don’t want to simulate another specification, click “Cancel”.

What can go wrong?

- The Modechart specification file you specified cannot be opened because either it cannot be read, it does not exist or it is not in the correct format (see Appendix 3).
- The options file you specified cannot be opened because either it cannot be read, it does not exist or it is not in the correct format.
- You do not have permission to write in the directory where you specified the log file to be written.
- You do not have permission to write in the log file.

How can I exit the simulator?

1. **Choose “Exit XMCSIM” under the File menu.**

   Before exiting, you will be given the option of exiting the simulator or cancelling this operation.

CHAPTER 9

Verifying a Modechart Specification

What is verification?

Verification allows you to answer particular questions about the behavior of a specified system. In the MT, you can verify the specification that you are currently working on. Verification works by generating a computation graph that captures all possible run-time behaviors of the specified system. Once this graph is generated, you can make queries about the behavior of the system, which the verifier answers by checking the computation graph.

Overview of the verification process

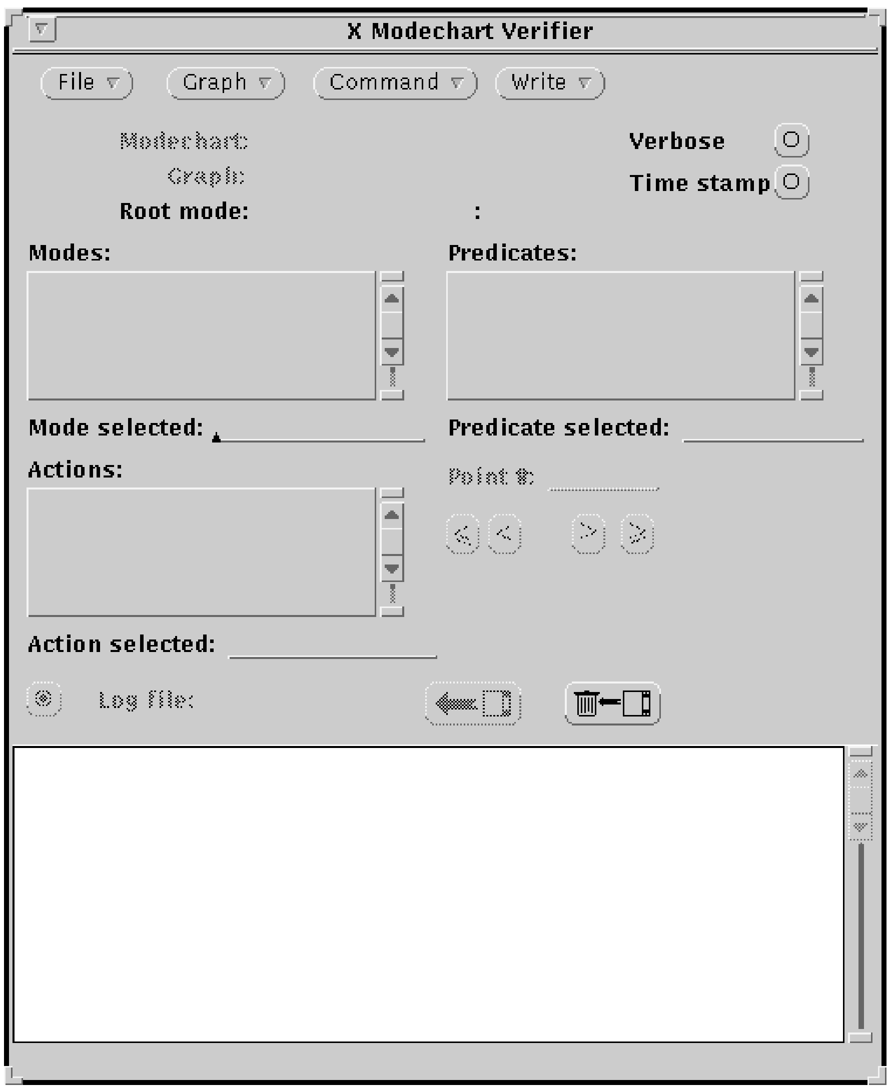
Each verification passes through three stages.

1. The verifier generates a computation graph for the system you wish to verify, and may, at your discretion, generate intermediate output to allow you to monitor the creation of the graph.
2. The verifier checks the graph to ensure that it satisfies certain correctness requirements, and determines which queries are supported for the particular graph.
3. The verifier provides a number of query templates which represent classes of queries it can answer about the graph, together with several input/output operations. You interact with the verifier to build queries by inputting parameters (typically, names of Modechart objects and timing values) in the appropriate query template(s).

How can I invoke the verifier?

1. **Choose "Verify" from the Analysis menu in the specification window.**

   A window titled "X Modechart Verifier" will appear:



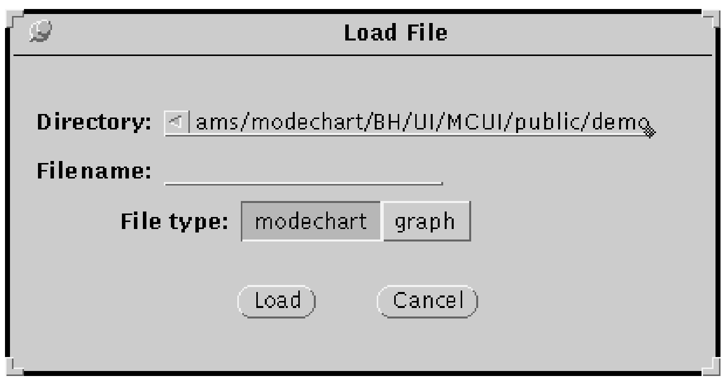
What can go wrong?

- You have not defined the environmental variables SPAWNCMD and/or VERIFYCMD. See Chapter 2 for more details.
- If the verifier window appears and then immediately disappears without a message appearing to indicate the problem, it is very likely that the computer you are using to run the MT does not have enough memory to run the verifier.

How do I load a specification into the verifier?

1. **Choose "Load File..." from the File menu in the verifier window.**

   A dialog box will appear asking you for a directory, a filename, and a file type:



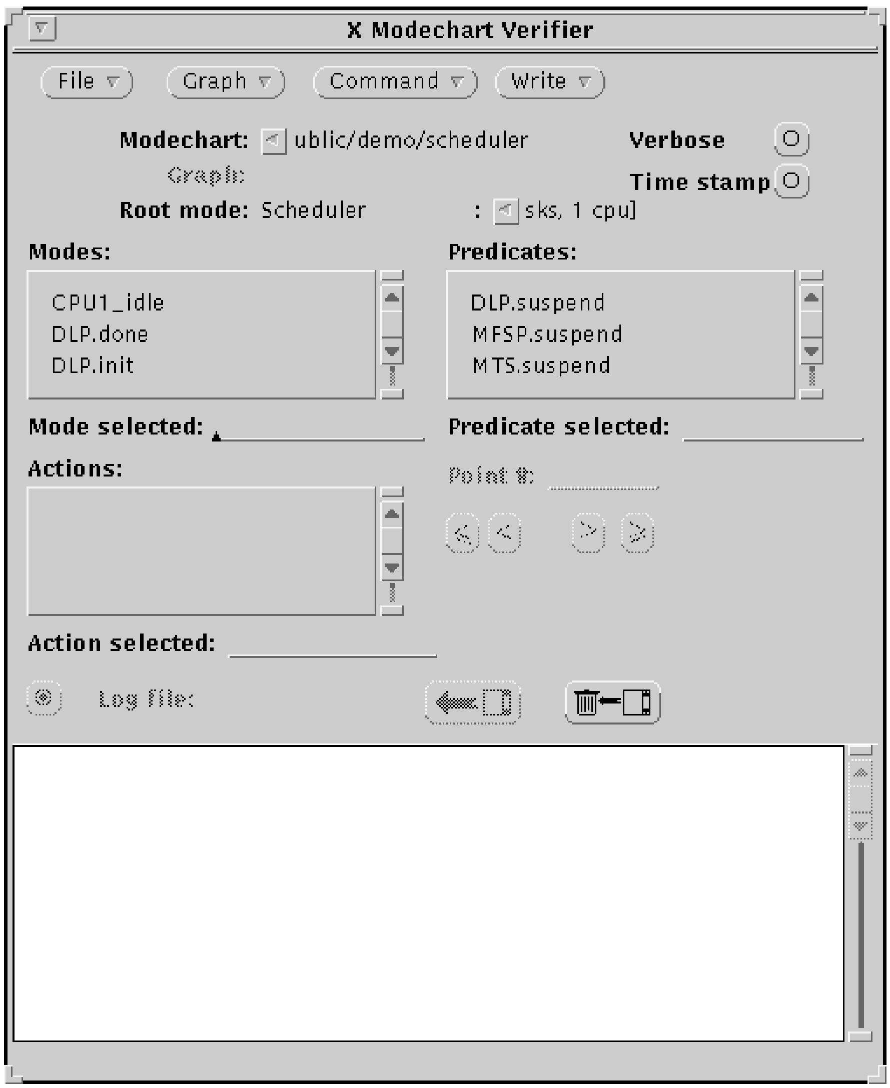
   The values default to the values that you entered previously. Initially, the default directory is the directory where the MT was invoked. If you want to verify the specification that you have been working on in the MT, make sure that you save it to a file first.

2. **Type in the directory and the filename of the specification you want to verify.**

   The file can either be a modechart specification (as saved by the MCTool, for instance) or a computation graph saved from a previous verifier session.

3. **Indicate the file type.**
4. **Click "Load".**

After a successful load, the verifier window will look like the following:



The specification objects (modes, predicates and actions) are listed in alphabetical order in separate, scrollable lists[1]. The large scrollable area at the bottom is a read-only output area where certain verifier output is directed. The name of the specifications's root mode is displayed, along with the contents of its description field, if any.

What can go wrong?

- The file referenced by the given directory and filename doesn't exist.
- The file does not have the correct permissions for you to read it.
- The filename and file type do not match.

  In addition, the current version of the verifier places several restrictions on the databases that it can handle. A message wil appear indicating the reason that the verifier cannot handle the database you specified. These restrictions include:
- The modechart structure does not meet current restrictions. The modechart must have a single root mode that is parallel. Its children must be serial. Its grandchildren must each either be (a) atomic; or (b) parallel, with serial children and atomic grandchildren.
- An action does not obey current restrictions. Actions may only be associated with atomic or parallel modes (other than the root).
- A mode transition does not obey current restrictions. A transition source or destination may only be an atomic or non-root parallel mode.
- A transition expression does not obey current restrictions. Mode-related events may only refer to atomic or non-root parallel modes.
- An external event is part of the modechart. External events are not supported by the verifier.
- The modechart does not obey size restrictions. The number of modes, transitions, conditions, actions, state predicates, functions, and entries in a predicate list are all limited.
- The modechart does not obey width restrictions. The maximum fanout (number of children of a mode) at each level is limited.

  If size or width constraints are violated, you may have the verifier re-compiled with larger capacities and try again.

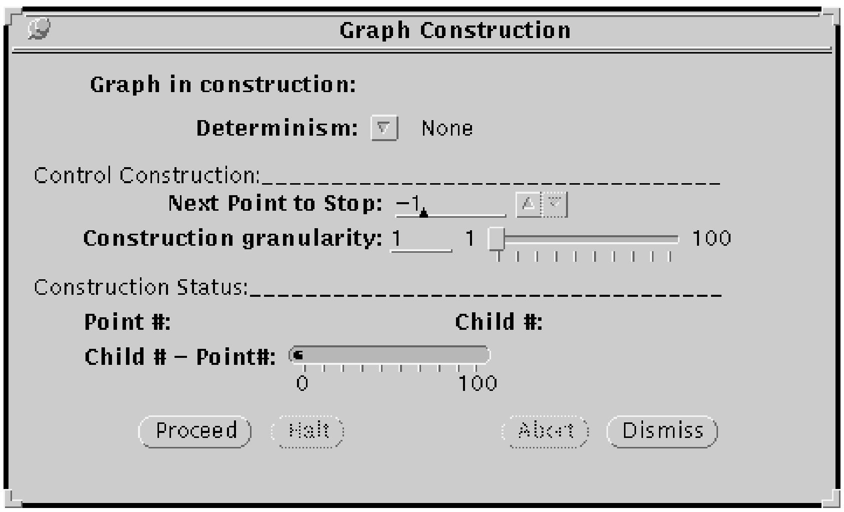
How do I generate a computation graph for my specification?

Before you can start verifying your specification, a complete computation graph must be generated, assuming one was not already loaded[2].

1. A future version of the verifier will also include external events.

1. **Choose "Construct graph" from the Graph menu.**

   A dialog box will appear asking you to input some information about the graph you want to construct.



2. **Set the type of determinism to exploit during construction.**

   Exploiting determinism means not generating paths in the computation graph that will never be taken because inferences can be made about how a mode will be exited. For example, if there are two transitions out of a mode, one is labelled (5,5) and the second is labelled (10,10), then no computations need be considered in which the second transition is taken.

   Determinism options are "None", "Trigger", "Timing" and "All". Timing determinism checks determinism only for transition with lower/upper bound conditions. Trigger determinism checks for determinism based on transitions with trigger conditions. "All" does both.

   Checking for trigger or timing determinism may yield noticably slower graph construction but may also yield a significantly smaller computation graph, with commensurate gains in query analysis. Checking for both may take the longest but produce the smallest graph and fastest queries.

3. **Set the next point at which to stop construction**.

2. A small number of ancillary commands are available without a complete graph being present. These commands will be described in subsequent sections.

The graph construction will stop when it generates (fully expands) a point whose index is greater than the number you enter. You can disable the auto-stop feature by setting the next point to stop to "-1" (default).

4. **Set the construction granularity by typing in a value or by moving the slider.**

   During construction, you can halt the process to inquire about the current state of the graph. Construction granularity controls the responsiveness of the verifier to your interruptions. A granularity of 1 makes the verifier as responsive as possible, but it may result in a noticeably longer time to build the graph.

5. **Click "Proceed".**

   This starts the graph construction. For non-trivial modecharts this step may take several minutes, or even hours. Graph construction works by generating a point, and then generating all possible successors to that point. During construction, the verifier displays the difference in indices between a point and its successors; when that difference goes to zero, construction is complete.

   Clicking "Halt" during graph construction causes the graph construction to stop so that you may examine the (incomplete) graph using the small number of commands that work for incomplete graphs. These commands include Reachability and a few commands that simply report information about the modechart specification and the graph. You can save an incomplete graph by choosing "Save computation graph to file" from the File menu. An incomplete graph can be re-loaded later and construction resumed.

   When construction is halted (whether the graph is complete or not), Abort may be chosen. Clicking "Abort" cancels the graph construction, deletes the incomplete graph, and re-loads the original file.

   When the graph is complete, the graph is marked complete in the verifier window.

6. **If desired, choose "Action pruning" from the Graph menu.**

   If the generated computation graph is invalid, action pruning may produce a valid graph by eliminating those computation paths corresponding to unsatisfiable timing constraints imposed by the timing of actions. Select "Action pruning" from the Graph menu. A window will appear that gives information about the points in the graph that action pruning will affect, and offers buttons to prune either selected points in the list, or the entire set.

7. **Check for busy-wait modes by choosing "Check busy-wait modes" from the Graph menu (optional).**

   Busy-wait modes are modes that the system can remain in forever while the system continues to progress. You are not required to check for busy-wait modes to perform most verification queries. In some cases it takes a significant amount of time to perform this check. However, you must check for busy-wait modes before you can use the "Cover Mode...", "Under Mode..." and "Exclusive Modes..." queries. If busy-wait modes have been checked, a note to that effect appears in the verifier window.

What can go wrong?

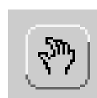
- The storage capacity of the verifier may be exceeded by a very large computation graph, either because of the number of nodes or the number of edges required by the graph. If this happens, a message will appear describing the problem. You will need to re-install the verifier with the appropriate capacity increased.
- The resulting graph may not be valid because of action timing. The timing generated by actions may yield unsatisfiable timing constraints. For example, a computation may initiate an action when a mode is entered, but the mode is not active long enough for the action to complete, and hence under Modechart semantics must never have been initiated. Performing action pruning (see next step) removes these semantically anomalous points, making the graph valid.
- The resulting graph may not be valid because zero-cycles are present in the modechart. If zero-cycles are detected in the modechart, the graph will be rendered invalid. This cannot be remedied except by modifying the modechart specification itself.

What can I verify about my specification?

Once you have a complete computation graph, several verification commands are available from the Command menu.

Each of these commands will be explained in turn. A query template window will appear each time you invoke one of these commands. You will formulate your queries by providing input parameters to these query windows. All query windows (with a few exceptions) have the following functionality in common.

- You can clear the window, i.e. restore it to its initial blank configuration by clicking “Clear”.
- You can start verifying the query by clicking “Evaluate”.
- You can ask the verifier to calculate any offset constants so that they will make the query you have built true. If there are no such constants, the query result will be “False”. (Any constants you specify will be ignored.)
- You can close the window by clicking “Dismiss”. The window will retain its parameters until you display it again, unless you construct a new graph in the interim.
- You can do the equivalent of clearing the window and then closing it in one step by clicking “Clear & Dismiss”.
- You can input parameters by either typing them in the appropriate blanks, or you can select them from the object lists in the verifier window. To do the latter, click on the object of choice in the verifier window. The selected object will then appear as the Mode Selected, Action Selected, or Predicate Selected, as appropriate, in the query window. From there, you may click on one of the hand icons to insert the selected object into the corresponding blank in the template.

  Repeatedly clicking the hand icon will scroll through all appropriate forms of the selected object(s): mode entry, mode exit, action start, action completion, etc.

  If the formula is such that the same parameter must appear in more than place, the tool will automatically supply it in the other required places when you install it in any one.
- In several of the formulas, you can change the relation signs (e.g., you can change “<“ to “>”, etc.). The presence of a menu button next to a relation symbol indicates that the relation can be changed. Click on the button to scroll through the choices.
- The formulas return True, False, or Error. If Error, an alert window will identify the problem.

There are several common reasons why a verification command might not be able to be performed. . Any additional reasons are documented accordingly.

What can go wrong?

- You constructed an improper formula. For example, you may have left some of the input fields blank or typed in names of objects that are not in the current modechart. When this happens, the formula will return Error, and a message will appear identifying the problem.

- The graph is too big to be handled by the formula. In particular, a path explored by the formula has a length which exceeds the maximum that the verifier can handle. If this happens, you may have the verifier re-compiled with a larger path length capacity and try again.

Inner Universal

This query checks the validity of formulas of the form:

$$\forall i \exists j : [\ @(\text{endpoint}_3, j) + c_1\ R_1\ @(\text{endpoint}_1, i)\ \&$$

$$@(\text{endpoint}_1, i) + c_2 < @(\text{endpoint}_2, i)\ \&$$

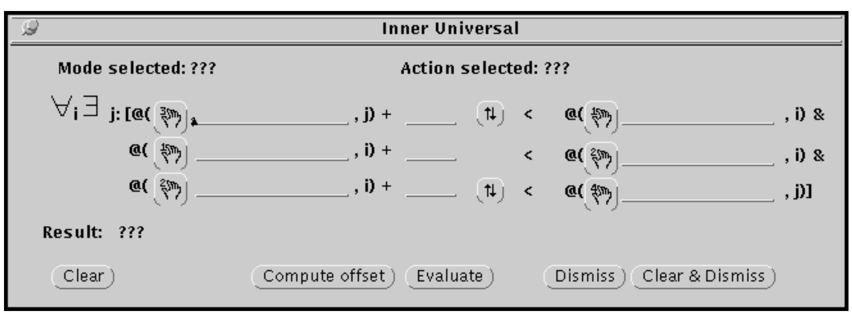
$$@(\text{endpoint}_2, i) + c_3\ R_3\ @(\text{endpoint}_4, j)\ ]$$

where c_1, c_2 and c_3 are non-negative integers, and R_1 and R_3 are either "<" or "≤". This formula is subject to the constraint that all simple paths in the graph (i.e. paths without cycles) and all cycles in the graph preserve both intervals, meaning that the mentioned events occur the same number of times in each simple path or cycle.

To verify an inner universal formula:

1. **Choose "Inner Universal..." from the Command menu.**

   The inner universal query window will appear.



2. **Enter the four endpoints and the relations.**
3. **Click "Calculate Offset" or enter the constant offsets and click "Evaluate".**

What can go wrong?

- One of the endpoint events names an unmatched universal endpoint. The events that share the same index variable in the formula (either i or j) define an interval. Hence, the formula defines two intervals: the interval formed by the events $endpoint_1$ and $endpoint_2$ that are indexed by i, and the interval formed by the events $endpoint_3$ and $endpoint_4$ that are indexed by j.

  It may be the case that the interval defined by the index variable to which the universal quantifier "for all" applies does not always exist; this will occur if the endpoint events do not occur in paired lockstep with each other in every path and cycle of the computation graph. If there is a place in the graph where one of the endpoint events occurs but there is no corresponding occurrence of the other endpoint event, then the message "n is an unmatched universal endpoint" will appear, where "n" is the point number of the unmatched point.

Outer Universal

This query checks the validity of formulas of the form:

$$\forall j \exists i : [\ @(\mathrm{endpoint}_3,j)+c_1\ R_1\ @(\mathrm{endpoint}_1,i)\ \&$$

$$@(\mathrm{endpoint}_1,i)+c_2 < @(\mathrm{endpoint}_2,i)\ \&$$

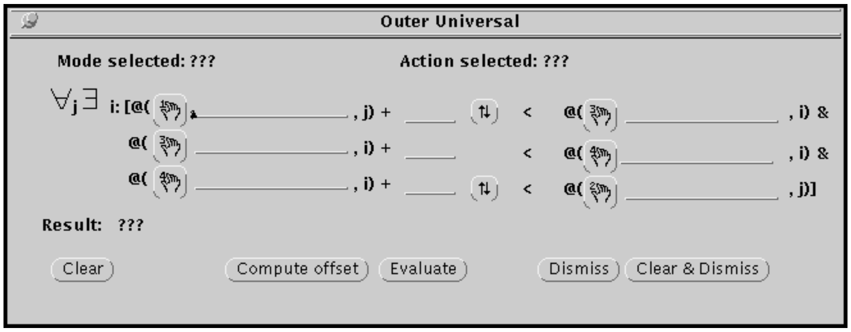
$$@(\mathrm{endpoint}_2,i)+c_3\ R_3\ @(\mathrm{endpoint}_4,j)\ ]$$

where c_1, c_2 and c_3 are non-negative integers, and R_1 and R_3 are either "<" or "≤". This formula is subject to the constraint that all simple paths and all cycles in the graph preserve both intervals, meaning that the mentioned events occur the same number of times in each simple path or cycle.

To verify an outer universal formula:

1. **Choose "Outer Universal..." from the Command menu.**

The outer universal query window will appear:



2. **Enter the four endpoints and the relations.**
3. **Click “Calculate Offset” or enter the constant offsets and click “Evaluate”.**

What can go wrong?

- One of the endpoint events names an unmatched universal endpoint. Refer to the corresponding <u>Inner Universal</u> section on page 73 for details.

<u>All Universal</u>

This query checks the validity of formulas of the form:

$$\forall i : [\ @(\text{endpoint}_{1,1},i)+c_1\ R_1\ @(\text{endpoint}_{1,2},i)\ \&$$

$$@(\text{endpoint}_{2,1},i)+c_2\ R_2\ @(\text{endpoint}_{2,2},i)\ \&\ \ldots$$

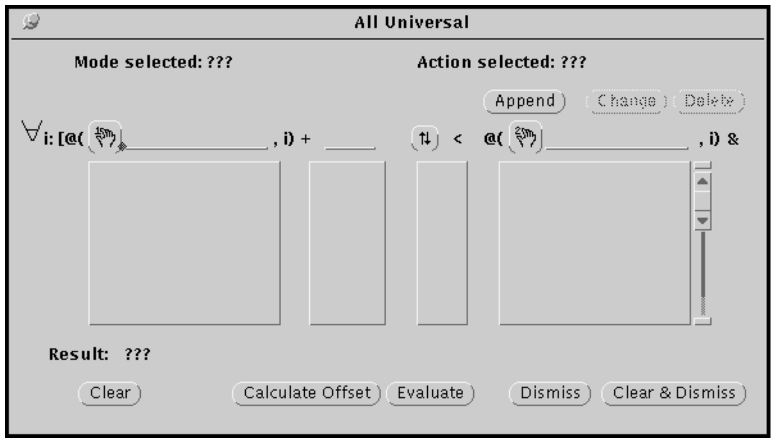
$$@(\text{endpoint}_{n,1},i)+c_n\ R_n\ @(\text{endpoint}_{n,2},i)\ ]$$

where c_1, c_2 ... c_n are integer constant, n is a positive integer, and each R_i is either “<“ or “>”. This formula is subject to the constraint that none of the endpoints can overtake any of the others.

To verify an all universal formula:

1. **Choose “All Universal...” from the Command menu.**

The all universal query window will appear:



This window is slightly different from the other Universal formulas, in that the query is a conjunction of individual All Universal assertions. Each conjunct is built on the top line of the window, and then added to the formula via the Append button. A line (conjunct) in the formula can be selected by clicking on it, in which case it appears on the top line. To delete the selected conjunct, click "Delete". To change the selected conjunct, make your changes and then click "Change".

For each line (conjunct) in the formula:

2. **Specify two endpoints and the relation.**
3. **Enter the constant offsets, if desired.**
4. **Click "Evaluate" or click "Calculate Offset".**

   If you specified the constant for every conjunct in the formula, you can choose "Evaluate". Otherwise, you must choose "Calculate Offset", and any constants you may have entered will be ignored.

What can go wrong?

- One of the endpoint events names an unmatched universal endpoint. Refer to the corresponding Inner Universal section on page 73 for details.

Separation

This query checks the validity of formulas of the form:

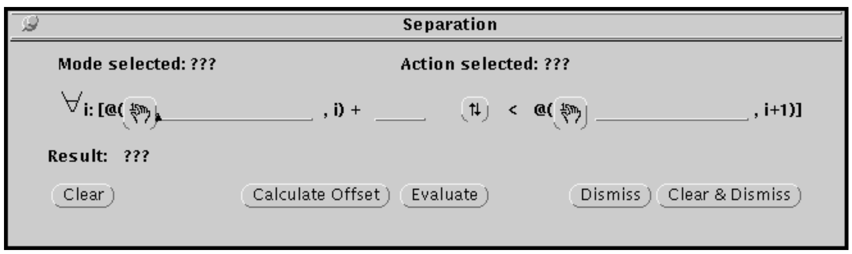
$$\forall i : @(\text{endpoint}_1,\text{i}) + \text{c R } @(\text{endpoint}_1,\text{i}+1)$$

where c is an integer constant, and R is either “<“ or “>”. It asserts the minimum (maximum) time between two consecutive occurrences of the same event.

To verify a separation formula:

1. **Choose “Separation...” from the Command menu.**

   The separation query window will appear:



2. **Specify the endpoint and the relation.**
3. **Click “Evaluate” or enter the constant offset and click “Calculate Offset”.**

Cover Mode

This query checks the validity of formulas of the form:

$$\forall i \exists j : [\ @(\text{->mode}_1,\text{i}) + \text{c}_1 \le @(\text{->mode}_2,\text{j})\ \&$$

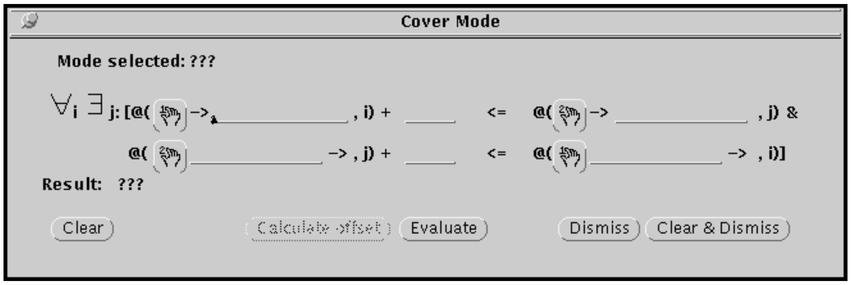
$$@(\text{mode}_2\text{->},\text{j}) + \text{c}_2 \le @(\text{mode}_1\text{->},\text{i})\ ]$$

where c_1 and c_2 are non-negative integers. This formula is subject to the constraint that mode_1 and mode_2 are not busy-wait modes. It asks whether every activity interval of the first mode *covers* an activity interval of the second mode, subject to the “overhang” intervals defined by the constants.

To verify a cover mode formula:

1. **Choose “Cover Mode...” from the Command menu.**

The cover mode query window will appear:



2. **Enter the names of the two modes.**
3. **Enter the two offsets.**[3]
4. **Click "Evaluate".**

Note: Busy-wait modes must be checked for prior to this operation, as discussed on page 69.

Under Mode

This query checks the validity of formulas of the form:

$$\forall i \exists j \ [\ @(\text{->}mode_2,j) + c_1 \le @(\text{->}mode_1,i) \ \&$$

$$@(mode_1\text{->},i) + c_2 \le @(mode_2\text{->},j) \ ]$$

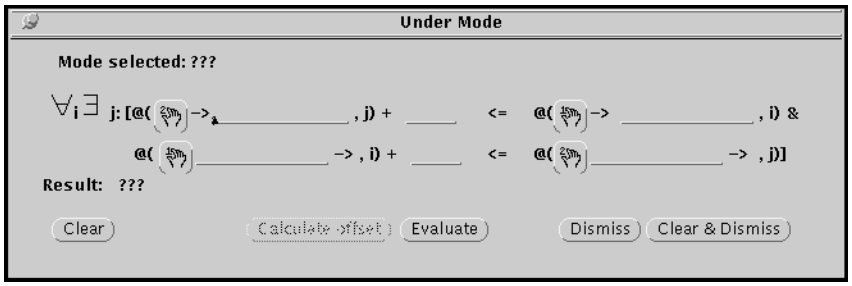
where c_1 and c_2 are non-negative integers. This formula is subject to the constraint that $mode_1$ and $mode_2$ are not busy-wait modes. It asks whether every activation interval of the first mode is *under* (covered by) an activation interval of the second mode, subject to the "overhang" defined by the constants.

To verify an under mode formula:

1. **Choose "Under Mode..." from the Command menu.**

3. Calculate Offsets is not yet implemented for this command.

The under mode query window will appear:



2. **Enter the names of the two modes.**
3. **Type in the two offsets.[4]**
4. **Click "Evaluate".**

Note: Busy-wait modes must be checked for prior to this operation, as discussed on page 69.

Exclusive Modes

This query checks the validity of formulas of the form:

$$\forall i \forall j : [\ @(mode_2\text{->},j) + c_1 \leq @(\text{->}mode_1,i)\ |$$

$$@(mode_1\text{->},i) + c_2 \leq @(\text{->}mode_1,j)\ ]$$

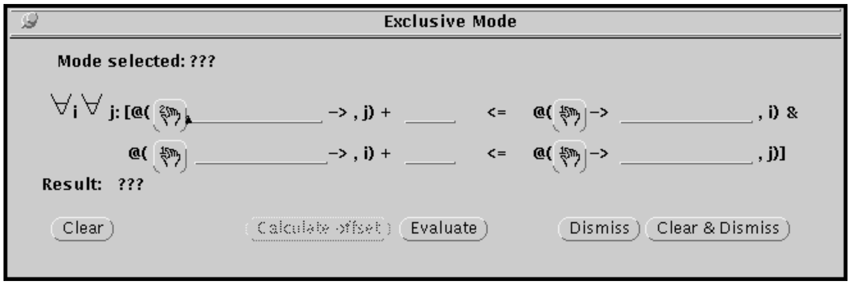
where c_1 and c_2 are non-negative integers. This formula is subject to the constraint that $mode_1$ and $mode_2$ are not busy-wait modes. This formula asserts that the activation intervals of the two modes, modified by the "overhang" of the constants, never overlap in time.

To verify an exclusive mode formula:

1. **Choose "Exclusive Mode..." from the Command menu.**

4. Calculate Offsets is not yet implemented for this command.

The exclusive mode query window will appear:



2. **Enter the names of the two modes.**
3. **Type in the two offsets.**[5]
4. **Click “Evaluate”.**

Note: Busy-wait modes must be checked for prior to this operation, as discussed on page 69.

Zwischen

This query checks the validity of expressions of the form:

$$\forall i \forall j : @(\text{endpoint}_1,\text{i}) + \text{c R } @(\text{endpoint}_2,\text{j})$$

where c is an integer constant and R is either “<“ or “>”.

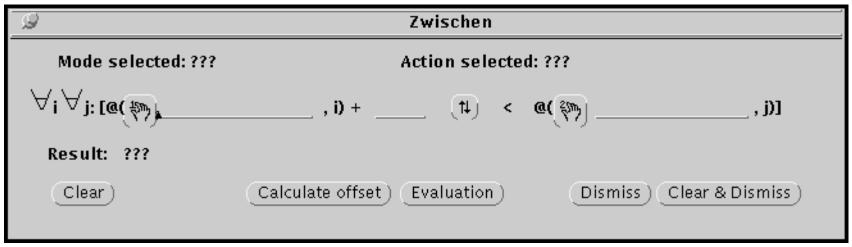
Strictly speaking, Zwischen does not verify a formula with respect to the specification, because for every instance of the first endpoint it finds, it searches only successors in the graph to find a point corresponding to the second endpoint. This procedure may miss an occurrence of the second endpoint that precedes the first endpoint in the graph, which is simultaneous with it in time.

To check zwischen expressions:

1. **Choose “Zwischen...” from the Command menu.**

5. Calculate Offsets is not yet implemented for this command.

The zwischen query window will appear:



2. **Specify the two endpoints and the relation.**
3. **Click “Evaluate” or enter the constant offset and click “Calculate Offset”.**

Reachability

This query reports all the points in the computation graph that match the restrictions that you provide. The restrictions include:

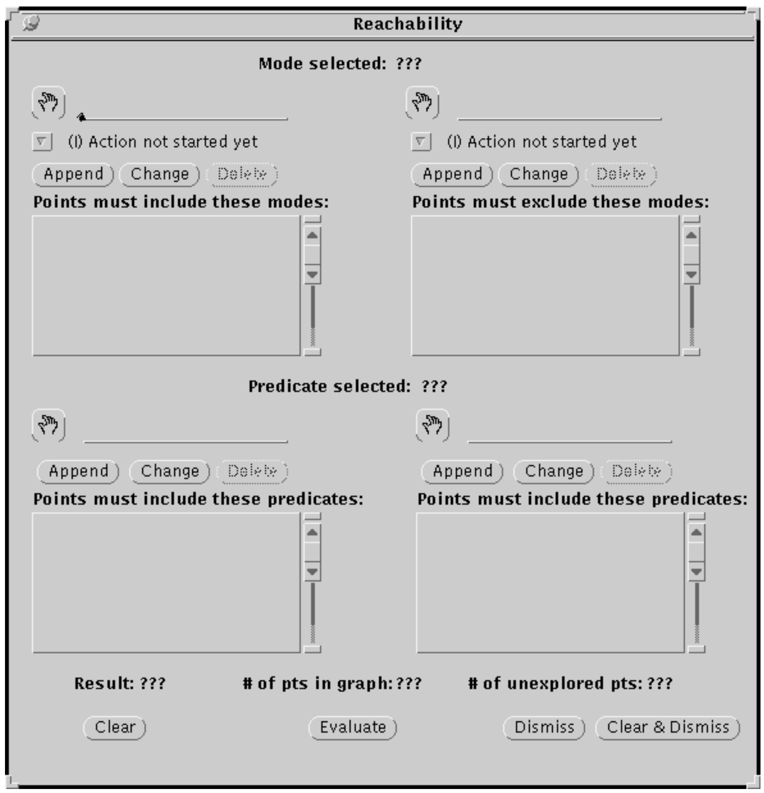
- modes that must be active
- modes that must not be active
- predicates that must be true
- predicates that must be false

In addition, for modes you specify, you may specify whether the action it hosts, if any, has started, has not started, has completed, or has been discarded. An action is discarded at a point if that point represents a mode activation that is predicted to be too short-lasting for the action to complete, and hence the action is never begun.

To perform a reachability query:

1. **Choose “Reachability...” from the Command menu.**

The reachability query window will appear:



2. **Enter the modes that must be active, the modes that must be inactive, the predicates that must be true and the predicates that must be false, to qualify a point.**

You can type in the fields or select items from the lists in the verifier window. To add an item to a list, click “Append”. To delete an item from one of the lists, select it and click “Delete”. To edit an item, select it, edit it in the type in field above the list and click “Change”.

If you enter a mode that has an action associated with it, you may also specify the status of that mode’s action, if any, that must hold in order for a point to qualify. You can choose any of the following action statuses:

I	Action has not started yet
S	Action started
C	Action completed

D Action discarded, meaning that along this path in the computation graph, the hosting mode is predicted to be active for too short a time to complete its action; hence, the action is not initiated.

N Matches all action statuses

3. **Click "Evaluate".**

   The number of points that match the parameters of the query will appear at the bottom of the window. The list of matching points, if any, will appear in the verifier window's output area.

   If used during graph construction, the number of unexplored points refers to the number of unexplored points that match the query, and the number of points refers to the number of explored points in the graph that match the query. An unexplored point is one in the computation graph for which children have not yet been generated.

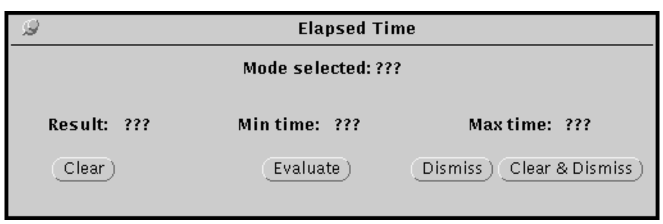
Elapsed Time

This query prints the minimum and maximum time that may be spent in the given mode.

To check the elapsed time:

1. **Choose "Elapsed Time..." from the Command menu.**

   The elapsed time query window will appear:



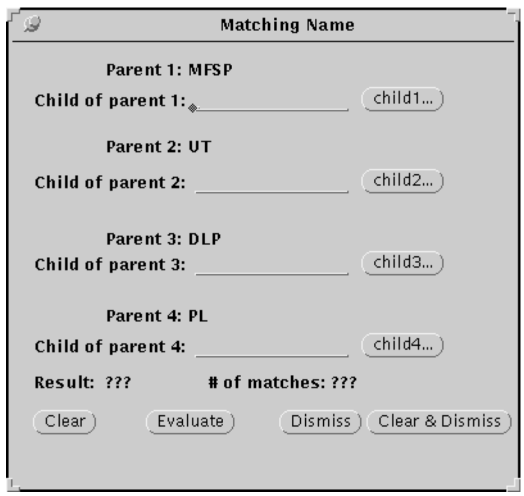
2. **Specify the mode.**
3. **Click "Evaluate".**

Matching Name

This command is primarily of interest to implementors. It is used to print the point indices of the points whose labels correspond to the modes supplied as arguments to the command. To use the command:

1. **Choose "Matching Name..." from the Command menu.**

   The matching name query window will appear:



2. **Enter the name of a child mode for each of the specified parent modes.**

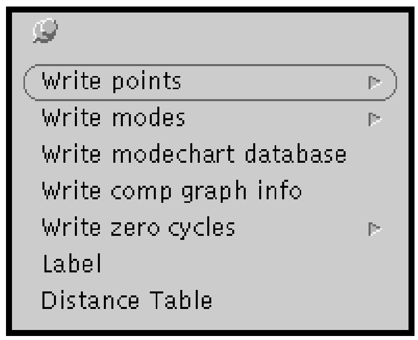
   You can either type in the name of the child or you can click the button to the right of the corresponding text field. If you click the button, a list of all the children of that parent mode will appear and you can select one from the list.

3. **Click "Evaluate".**

   The number of points that match your query will appear at the bottom of the window. The corresponding point indices, if any, will appear in the output area of the verifier window.

How do I get information about my specification and its computation graph?

There are several commands that you can perform to get information about your specification and its corresponding computation graph. These commands can be found in the Write menu.



Each of these commands will be explained in turn. All of the results of these commands are displayed in the output area of the main verifier window.

Write points

1. **Choose the type of point labels you wish to write from the Write points submenu.**

Write:	**Do this:**
All point labels	Choose “Write all point labels” from the Write points submenu. A further submenu lets you choose the format to print the labels. If you choose “clean format”, only the point labels of points in the graph are printed. If you choose “long format”, the labels of the points in the graph plus any unreachable points, pruned points and/or replaced points are printed.
Pruned-action point labels	Choose “pruned points” from the Write points submenu.
Labels of points in which system can reach a persistent state (halt)	Choose “wait points” from the Write points submenu.
Labels of points in which system must halt, i.e. points that have no successors	Choose “deadlock points” from the Write points submenu.
Labels of points that were pruned by exploiting determinism during graph construction	Choose “deterministically pruned points” from the Write points submenu.
Labels of points with deterministically pruned children	Choose “points w/ deterministically pruned children” from the Write points submenu/
Labels of points that were not pruned by exploiting determinism during graph construction	Choose “non-deterministic points” from the Write points submenu.

Point labels are described in the Label section on page 88.

Write modes

1. **Choose the type of modes you wish to write from the Write modes submenu.**

Write:	Do this:
Busy-wait modes	Choose "busy-wait modes" from the Write modes submenu. This command is only enabled if busy-wait modes have been checked for in your computation graph.
Deterministic modes	Choose "deterministic modes" from the Write modes submenu.
Non-deterministic modes	Choose "non-deterministic modes" from the Write modes submenu.

Write modechart database

1. **Choose "Write modechart database" from the Write menu.**
   This command prints the Modechart specification.

Write comp graph info

1. **Choose "Write comp graph info" from the Write menu.**
   This command prints the following statistics about the computation graph:

Statistic	Description
Points	Total number of points in data structure
Checked Points	Number of points explored during graph construction
Good Points	Number of points in computation graph (Point status G)
Replaced Points	Number of replaced points (Point status R)
Pruned Points	Number of pruned points (Point status P)
Stable Points	Number of points with no successors
Edges	Number of objects in data structure that represent timing constraints of one sort or another

Statistic	Description
Names, Names Reused, Longest Namelist, Most Used Name	These report information of interest primarily to the implementors concerning internal representation of modechart data.
Persist	Number of points the system can stay in indefinitely
Busy-wait	Number of busy-wait modes (modes the system can remain in indefinitely while making progress)

Write zero cycles

A zero cycle is a symptom of an illegal modechart. Briefly, it is an anomalous condition in which, under the semantics of Modechart, an infinite number of transitions are required to be taken in zero time.

Suppose a transition from A to B is predicated on event E occurring. Suppose also that a transition from B to A is predicated on the same event. If A is active at time t, and E occurs, then the transition to B is required to occur (assuming there is no other exit from A available at the time). However, the transition from B to A is also required to occur, and so forth. This is the simplest of zero cycles, but others are much more subtle. Printing zero cycle information uncovers zero cycles in the modechart that have been identified by the verifier.

To print zero cycle information:

1. **Choose the type of zero cycles you wish to write from the Write zero cycles submenu.**

Write:	Do this:
Zero cycle points (writes the zero cycle ending at the indicated point)	Choose “zero cycle points” from the Write zero cycles submenu, after selecting a point in the Point field of the main window.
Zero cycles in graph (lists points that have been identified as end points of zero cycles)	Choose “zero cycles in graph” from the Write zero cycles submenu.
Zero cycles including replaced points	Choose “zero cycles included replaced points” from the Write zero cycles submenu.

Label

Given a point in a computation graph, this command prints its *label*, or summary of relevant information, in the output area of the verifier window. An example of a label, with its fields identified is:

2	**G**	**(1,3)**	**BC S UP I**	**NEAR**	**TRAIN**	**{ ->BC }**	**1:**	**4**
Point index	Point status	Component number that created the point	Modename and action status for the elements in the component numbert pair	Actions in progress at this point	State variables true at this point	Set of events occurring at this point	Number of children	Point indexes of children

A point status is one of the following:

G	Good point; suffix ! for persistent.
U	Unreachable point. A point is unreachable if it is not part of any legal computation.
R	Replaced point; equivalent to a point already in graph
N	New point for which children have not yet been generated; only present if a complete graph has not yet been constructed.
P	Pruned point; eliminated by action pruning for timing faults. For more information on pruning for time faults, see: D. Stuart and P. Clements, "*Clairvoyance, Capricious Timing Faults, Causality, and Real-Time Specifications*," Proceedings, Twelfth Real-Time Systems Symposium, San Antonio, Texas (1991).
T	Deterministically pruned point.

A component number is an ordered pair defines a mode in the verifier's data structures. The first element of the ordered pair indicates a parallel child of the root; the second refers to the parallel (or atomic) child of the active serial child of the mode indicated by the first component. If the second element is zero, it refers to the active serial mode itself.

An action status is one of the following:

I	Initial; either there is no action or the action has not started yet
S	Action started
C	Action completed
D	Action discarded, meaning that along this path in the computation graph, the hosting mode is predicted to be active for too short a time to complete its action; hence, the action is not initiated.

To print the label of a given point index:

1. **Specify a point index in the verifier window.**

   This may be done by scrolling through the valid point numbers by clicking the "≤", "<", ">", or "≥" buttons, or by typing in the desired point number.

2. **Choose "Label" from the Write menu.**

Distance Table (dt)

Given a point in a computation graph, this command will print the adjacency matrix and the distance table for that point. The first column and the first row of these tables are headings. These heading are the same for both types of tables, only the values and the meaning of the tables differ.

	15	**16**	**3**	**12**	**10**	**9**
15	0	-9000	-9000	-9000	-9000	-9000
16	-9000	0	-9000	-9000	-9000	-100
3	-9000	-9000	0	0	-9000	-9000
12	300	0	0	0	-9000	-9000
10	-9000	-9000	-9000	0	0	-9000
9	-9000	20	-9000	-9000	100	0

In an adjacency matrix, the value (i,j) is the the number of ticks that must elapse between event i and the next succeeding occurrence of j. If i and j can occur at the same time, the value is zero; a representation of negative infinity appears in the matrix if j cannot occur after i. This value takes into account all direct constraints on j with respect to i. In a distance table, the value (i,j) is as in the adjacency matrix, but it takes into account all constraints taken or pending by the end of the path being considered. The value

9000 is used to represent infinity; this may be changed by re-compiling the verifier with a different value.

To print the adjacency matrix and the distance table for a given point index:

1. **Specify a point index in the verifier window.**
2. **Choose "Distance Table" from the Write menu.**

The point chosen must be one whose point status is "G", "R", or "N" (see Label).

How can I control and save the output printed in the verifier window?

Log files

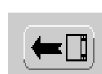
You can save all of the information that is written in the verifier window's output area to a file. To save this output to a file:

1. **Choose "Open log file" from the File button.**

   A dialog box will appear prompting you for the name of a log file.
2. **Type in the name of the log file.**

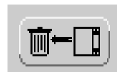
   If the log file you specify already exists, you will be given the option of either appending your output to it or overwriting it.
3. **To control whether output is being recorded in the log file, toggle the Log file button just above the output area in the verifier window.**

   By default the Log file button is turned on, i.e. output is being saved to the log file.
4. To print the current contents of the output area to the log file, click the echo output button:

   This can be done regardless of whether or not the Log file button is on or off. However, echoing the current output to the log file when the Log file button has been on will result in some information be duplicated in the log file.

5. **When you want to clear the contents of the output area, click the clear output area button:**



Verbose

To cause commands to generate additional output (the amount of additional output depends on the command in question):

1. **Click the Verbose button in the verifier window.**
2. **To exit verbose mode, click the Verbose button again.**

Time stamp[6]

To begin recording in the output window when subsequent commands start and finish:

1. **Click the Time stamp button in the verifier window.**
2. **To stop attaching time stamps to commands, click the Time stamp button again.**

How do I quit the verifier?

1. **Choose Exit from the File menu.**
   Remember to save your computation graph before exiting the verifier, if desired.

What features will be added to the verifier in the future?

The following features are being considered for addition to the verifier:

6. This feature is not implemented yet.

1. **Auto-save:** Graph construction may take hours to complete. This feature would allow a user to specify a periodic interval (e.g., a number of points) after which the verifier would automatically save the computation graph under construction. In this way, an interruption or a system crash would not force the user to start over.
2. **Command abort:** Currently, graph construction may be aborted. This is useful if it is taking longer than the user had in mind. However, once a command is begun, there is no way to interrupt it. We plan to add abort capabilities to the commands so that a query taking longer than the user cares to wait may be stopped.
3. **Tighter integration with MCTool:** In the future, the verifier will read the Modechart specification directly from memory, and not from a file. In this way, the modechart that the user has just constructed will be the one verified.
4. **Specify preferences with X resources**: By allowing the user to specify an X resource file, many options (such as graph construction granularity, type of determinism to exploit, size and color and initial placement of windows, etc.) will be set according to user preferences.
5. **More informative output**: Many of the queries produce output that is poorly formatted at best, and esoteric at worst.
6. **Scripts**: Allowing users to specify scripts of queries to run will facilitate rapid "regression verification," i.e., running the same verification tests when a specification has been modified.

How do I find out more about the theory behind verification?

For more information about Modechart verification, see any of the following papers:

F. Jahanian and A.K. Mok, "Safety Analysis of Timing Properties in Real-Time Systems," *IEEE Transactions on Software Engineering*, Vol. SE-12(9), pp. 890-904 (September 1986).

F. Jahanian, A. Mok, and D. Stuart, "Formal Specification of Real-Time Systems," *University of Texas Technical Report (TR-88-25)*, Austin, Texas (Jun 1988).

F. Jahanian and D. Stuart, "A Method For Verifying Properties of Modechart Specifications," *Proceedings, 9th IEEE Real-Time Systems Symposium*, (1988).

D.A. Stuart, "Implementing a Verifier for Real-Time Systems," *Proceedings, 1990 Real-Time Systems Symposium*, pp. 62-71 (5-7 December 1990).

C. L. Heitmeyer and B. G. Labaw, “Requirements Specification of Hard-Real-Time Systems: Experience with a Language and a Verifier,” *Foundations of Real-Time Computing Formal Specifications and Methods*, A. van Tilborg and G. Koob, Eds., Kluwer Academic Publishers, Norwell, MA, pp. 291-313 (1991).

P. C. Clements, C. L. Heitmeyer, B. G. Labaw, and A. T. Rose, “MT: A Toolset for Specifying and Analyzing Real-Time Systems,” *Proceedings, 1993 Real-Time Systems Symposium* (December 1993).

APPENDIX 1

Using OpenWindows

How to set up your environment for OpenWindows

To run OpenWindows, you must execute the following four steps.

% setenv OPENWINHOME <*OpenWindow directory*>

% set path = ($OPENWINHOME/bin $path)

% setenv LD_LIBRARY_PATH $OPENWINHOME/lib:<*X library path*>:<*system library path*>

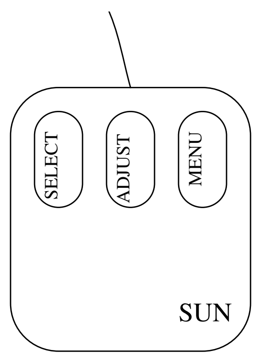
% setenv FONTPATH $OPENWINHOME/lib/fonts

% openwin

It might be a good idea to include these steps as part of the initialization sequence performed when you login. If you are having difficulty running OpenWindows, consult your local system administrator. Refer to the OpenWindows start-up documentation for details on how to customize your window environment. Once in OpenWindows, MT can be executed following the steps described in Chapter 2.

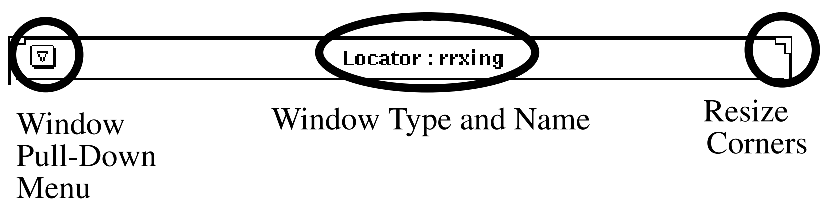
Using the Mouse Buttons

Sun Workstations use three-button mice. Each button has a specific name and use that is consistent across applications. The left button (Select) is used to select objects on the screen. Selection of items include point-and-click operations, as well as point-click-and-drag operations. The middle button (Adjust) is used to extend the selection created with the left button. The right button (Menu) is used to open menus. Menus are identified with a small triangle pointing down or to the right.



Window Components

All windows in the MT have some common parts that help identify their use. The title bar of a win-



dow contains the window type and the name of the window. The name refers to the specification to which the window belongs. In the example above, the locator window belongs to a specification named ‘rrxing’.

The small icon on the left of the title bar is a pull down menu with window related actions. These actions are provided by OpenWindows and are not specific to the MT. For a description of these, consult the OpenWindows documentation.

All four corners of a resizable window in OpenWindows contain a resize handle, as shown in the diagram above. This icon allows resizing the window from any of the four corners.

Moving a Window

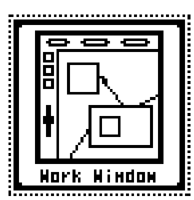
To move an OpenWindows window, click and hold the Select (left) mouse button on the window header or border and drag the window to the desired location. As you move the mouse, an outline of the window will follow the mouse. Upon release of the button, the whole window will be moved to the new location.

Resizing a Window

Windows that can be resized, contain the resize handles (or corners) shown in the diagram in the previous page. To resize a window, click and hold the Select (left) mouse button, and drag the window corner to the new desired size. As you move the mouse, an outline of the window will follow the mouse. Upon release of the button, the whole window will be resized.

Iconifying a Window

Any MT window can be iconified. An iconified window is displayed as a small icon on the desktop. For example, the icon below is that shown for the work window when iconified.



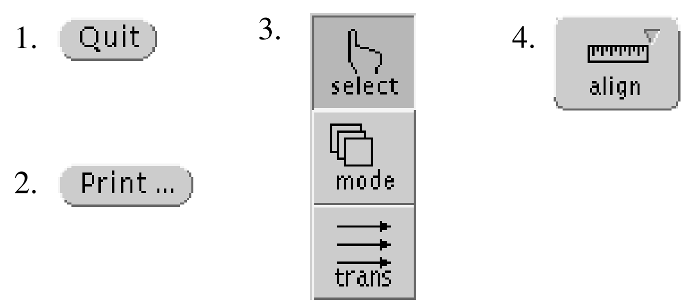
To iconify a window, click and release the Select (left) button on the window pull-down menu (top-left corner of the window). A different way of obtaining the same results would be to click and hold the Menu (right) button on the window pull-down menu. When the window menu appears, select the menu item Close, and then release the button.

Opening an Iconified Window

Once a window has been iconified, there are two ways of opening it. The easiest is to double-click the Select button on the icon for the window. The other way is to use the Menu button on top of the icon. A window menu will appear, select the item Open from it.

Buttons

OpenWindows provides different types of buttons. In the MT, there are four different uses for buttons. The following diagram shows examples, from the MT, of each type of button and their use.

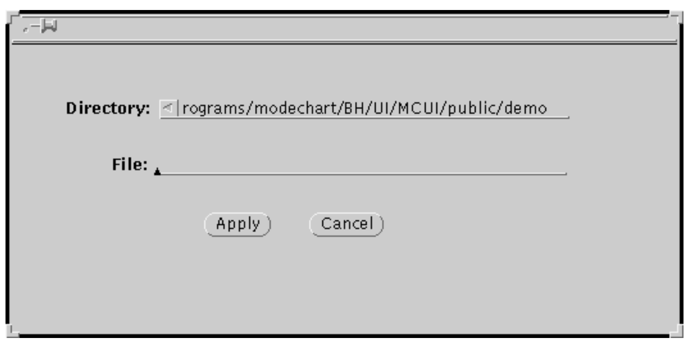


1. Simple Action Button -This type of buttons perform the action identified by its label as soon as the mouse button is released. Use the Select mouse button to press this type of button.
2. Complex Action Button -This type of button, when pressed, produces a window prompting the user for other information required to complete the command. They are easily identified by the ellipsis (...) at the end of the name of the button. The windows generated by these buttons always have a Cancel button which allows for a quick exit from the action without making any changes. Use the Select mouse button to press this type of button.
3. Moded Buttons - In the moded buttons, one of the choices is always selected (shown with a bold outline in the diagram). Selection of one of the other buttons in the group, changes the current mode of selection in the group. No action is performed as a direct result of these buttons; rather these buttons enable certain classes of actions and disable others. To activate them, use the Select button.

4. Pull-Down (Menu) Buttons - Some buttons are used as pull-down menus. These always have a small downward pointing triangle in the right side of the button. To activate these buttons, you can use the Menu button on the mouse. Press and hold the Menu button, a menu will appear, then make a selection from the menu by releasing the mouse button over the desired item. All menus also have a default item which can be accessed by click and release of the Select button (see the description of the window pull-down menu).

Type-In Fields

Type-in fields are fields used to enter textual information into the system. They are graphically represented with a label followed by a single line (fill-in blank). To enter text, you must use the keyboard. The mouse is used to identify the type-in field to which the text is entered. In the window below, a small black diamond identifies the type-in field in which text is being entered.



To change the active type-in field, move the mouse to a type-in field, and click the Select button. Also, the Tab or Return key will cycle through all the type-in fields available in a window.

APPENDIX 2

Using X Windows

How to setup your environment for X Window

To run X Windows, you must execute the following steps.

% setenv XHOME <X *directory*>

% set path = ($XHOME/bin $path)

% setenv LD_LIBRARY_PATH $XHOME/lib

% xinit (or appropriate X initialization routine)

It might be a good idea to include these steps as part of the initialization sequence performed when you login. If you are having difficulty running X Windows, consult your local system administrator. Refer to the X Windows start-up documentation for details on how to customize your X environment. Once in X Windows, MT can be executed following the steps described in Chapter 2.

There are several window managers that can be used with X Windows. The behavior described in this chapter pertains to the twm window manager. If you are using a different window manager, see your window manager documentation for details.

Using the Mouse Buttons

Within the MT, the OpenWindows mouse button functionality has been adopted.

Mouse Button	Function
Left	Select
Middle	Adjust
Right	Menu

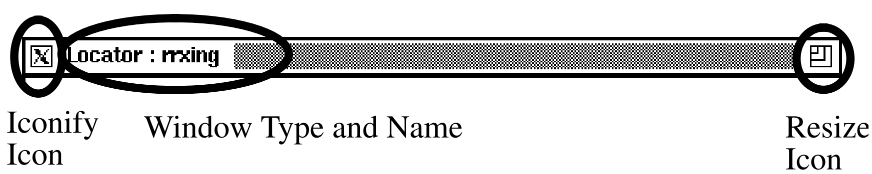
Appendix 1 (Using OpenWindows) describes the meaning of the mouse button functions in the table above.

Window Creation

Unlike in OpenWindows, windows in X do not simply appear without any user interaction. In X windows, when a window is created, an outline of that window is attached to your mouse, as you move the mouse this outline will follow. This allows you to control the initial window position. Simply click the Select mouse button where you want the window to appear.

Window Components

All windows in the X environment have some common components.



The title bar, just like in OpenWindows, contains the window type and the name of the specification to which it belongs. The small icon on the left side of the title bar allows you to iconify the window and the icon on the right side allows you to resize the window.

Moving a Window

Windows are moved the same way in X windows and in OpenWindows. See Appendix 1 (Using OpenWindows) for more details.

Resizing a Window

Click and hold the Select (left) mouse button in the resize icon, and drag it to its new location. The bottom left corner of the window will stay fixed. As you drag the mouse, an outline of the window will follow the mouse. When you release the mouse button, the window will be resized.

Iconifying a Window

To iconify a window, click the small icon to the left of the title bar with the Select (left) mouse button.

Opening an Iconified Window

To open an iconified window, click the icon with the Select (left) mouse button.

Buttons

In the MT, buttons behave the same in X windows and in OpenWindows. Appendix 1 (Using OpenWindows) discusses how to use different buttons.

Type-In Fields

Type-in fields behave the same in X windows and in OpenWindows. Appendix 1 (Using OpenWindows) discusses how to enter information in type-in fields.

APPENDIX 3

Modechart Database File Syntax

What is a Modechart database file?

A Modechart database file is the type of ascii file that the MT expects to find when you load an existing specification. It is automatically generated by the MT when you save a specification (in addition to a file containing graphical information about the specification), but you can also create it manually. This file captures the essential semantic information about a modechart (the objects in the chart, how they are related to each other, and their attributes) but it does not capture any graphical information about the modechart (e.g., the layout of a group of modes, the path of transition arrows, etc.).

It is postulated that for some applications, it will be more efficient to create this file manually rather than by using the direct manipulation facilities of the MT. For example, an especially systematic modechart might be quickly defined by using a text editor's copy and paste facilities. For this reason, this chapter will explain the format and meaning of Modechart database files, so that you can edit (and even create) them as desired.

What do I need to know about the format of Modechart database files?

Certain MT operations require text input from the user, and this input must follow the rules defined in this chapter. The minimum syntax knowledge that is necessary to use the MT constitutes the following:

- mode transition expression (MTE) syntax, and
- mode name syntax.

All other rules of syntax may be ignored, especially by beginning users. The MT will read and write the necessary files, in the necessary formats, for you.

General Syntax Rules

A database file consists of a series (in any order) of definitions. The different types of definitions include:

- primitive action definitions,
- composite action definitions,
- external event definitions,
- state variable definitions,
- mode definitions,
- mode transition definitions,
- function definitions, and
- typeclass definitions.

A section of this appendix is devoted to each type of definition. These sections will adhere to the following conventions:

- *italics* denotes non-terminals that will be explained later, unless their name obviates the description.
- [] denotes symbols which may appear zero times or once.
- [][+] denotes symbols which may appear one or more times.
- [][*] denotes symbols which may appear zero or more times.

The following syntax rules apply to more than one section:

1. Input is free-form, in that unlimited white space (blanks, tabs, newlines) separate tokens.
2. Comments, which are unrestricted text delimited by /* and */, may appear anywhere in the file. As in C, they may not nest, they may cross line boundaries, and they are ignored by the software that processes the file.
3. A semicolon terminates each definition.
4. For keywords, case is not significant; they may be given in all upper-case or all lower-case (or even mixed). For the names of user-defined objects, case is significant: “NAME” will be stored differently from “name”, for example.
5. Each definition may include a description just before the terminating semicolon. A description is free-form text enclosed in square brackets. Whereas the MT ignores comments, descriptions are stored as part of the modechart data. Descriptions are optional and may be omitted.
6. A syntactic unit described as a list of some sort is enclosed in { braces }, with the items separated by commas.
7. Identifiers follow the rules for C identifiers, except that periods are allowed in identifiers as other than the first character. Keywords are reserved.
8. Operators, where called for, follow C syntax. In particular, the assignment operator is “=”, and the boolean equality operator is “==”.
9. The symbol “?” is a wild-card value that may appear in the file in place of any value token whose value is not known. It stands for “to be determined.” For example, “?” may appear instead of an integer giving the maximum execution time of an action.

   The MT expects mode and transition definitions to be complete; definitions using the wild-card value or referencing undefined modes are considered incomplete and may cause undefined behavior.
10. The MT expects tokens in certain positions to be names of particular types (mode, action, etc.). If these expectations are violated, the behavior of the tool is undefined, and errors may result.

Mode Definitions

A serial mode definition consists of:

SM *name*, CO *parent-mode*, IM *initial-mode*, *action-definition*;

Atomic and parallel mode definitions consist of:

mode-type name, CO *parent-mode*, *action-definition*;

- The valid *mode-type*s are SM (serial), PM (parallel) and AM (atomic).

NOTE: When you create modes using MT, there is no way to declare a mode as atomic. However, if you create a specification (using a text editor) that has atomic mode definitions, MT can load it.

- *parent-mode* is the name of the parent of mode *name*; if *name* is a root mode, *parent-mode* is given by a *.
- If *mode-type* is SM (serial), then *initial-mode* is the name of the mode initially entered when *name* is entered.
- The *action-definition* defines the native action associate with the mode. If there is no action, action-definition is given by a *. Otherwise the *action-definition* consists of:

  action-type name(*deadline* , *time2*);

  The valid *action-type*s are PA (periodic action) and SA (sporadic action). *deadline* and *time2* are an ordered pair of positive integer literals where *deadline* <= *time2. time2* is the period if the action is periodic and the minimum separation is the action is sporadic.

Examples include:

```
PM m1, CO M2, PA a21(/* deadline */ 150, /* period */ 300);
AM m1, CO M2, PA a21(/* deadline */ 150, /* period */ 300);
SM m3, CO *, IM I1, SA a23(/* deadline */ ? , /* min. sepn. */ +50);
SM m3, CO ?, IM *, SA a23(1, 10);
SM m3, CO *, IM ?, SA a23(1, ?);
SM m3, CO ?, IM ?, SA a23(1,10);
```

Mode Transition Definitions

A mode transition definition consists of:

source-mode-name -> *destination-mode-name* : *transition-expression*;

- The *transition-expression* is a set of disjuncts separated by "|". Each disjunct is either a trigger or a lb-ub pair. A trigger is a set of conjuncts, separated by "&". Parentheses may be used to group. The different kinds of conjuncts are described in the following table.

Conjunct form	**Event or condition**	**Interpretation**
^name	E	name is interpreted as name of external event
|-name	E	start action: name is interpreted as name of action
-|name	E	stop action: name is interpreted as name of action
name=true	E	name is interpreted as name of boolean state variable; this event refers to the occurrence of the variable's value becoming true when it wasn't true before
name=false	E	name is interpreted as name of boolean state variable; this event refers to the occurrence of the variable's value becoming false when it wasn't false before
->name	E	name is interpreted as the name of a mode; this event refers to the occurrence of the named mode being entered
name->	E	name is interpreted as the name of a mode; this event refers to the occurrence of the named mode being exited
$name_1$->$name_2$	E	$name_1$ and $name_2$ are interpreted as names of modes; the event refers to the transition from $name_1$ to $name_2$ being taken
name==true	C	name is interpreted as the name of a mode; this condition is satisfied if and only if the system is currently in the named mode
{(m_1,m_2,...,m_n)}	C	m_i are interpreted as names of modes; true at time t if and only if the system is in at least one of the modes at t.
{(m_1,m_2,...,m_n]}	C	m_i are interpreted as names of modes; true at time t if and only if the system entered at least one of the modes at or before time t and exited at least one of the modes at time t

Conjunct form	Event or condition	Interpretation
$\{<m_1,m_2,...,m_n)\}$	C	m_i are interpreted as names of modes; true at time t if and only if at lease one of the modes is active at time t, and was most recently entered before time t.
$\{<m_1,m_2,...,m_n]\}$	C	m_i are interpreted as names of modes; true at time t if and only if the system entered at least one of the modes at or before t and exited at least one of the modes at time t.
$\{[m_1,m_2,...,m_n)\}$	C	m_i are interpreted as names of modes; true at time t if and only if at least one of the modes was entered at time t.
$\{[m_1,m_2,...,m_n]\}$	C	m_i are interpreted as names of modes; true at time t if and only if the system entered at least one of the modes at time t, and the system exited at least one of the modes at time t.
name	C	name is interpreted as the name of a boolean state variable; condition is true if and only if the variable is true.
!name or !(name)	C	name is interpreted as the name of a boolean state variable; condition is true if and only if the variable is false.

A lb-ub pair is one of the following:

- (*n1*, *n2*) where *n1* <= *n2* and *n1* and *n2* are non-negative integer literals,
- (alarm *n*) where *n* is a non-negative integer literal,
- (deadline *n*) where *n* is a non-negative integer literal or infinity,
- (delay *n*) where *n* is a non-negative integer literal, or
- (*n* , infinity) where *n* is a non-negative integer literal.

Examples include:

```
m1->m2: {(m1,m2,m3)} | ({<m4]})&-|c | |-b&-|c | (alarm 1) | (1, 3);
m3->m4: (alarm 50) | (deadline 25) | (1, infinity) | (delay 500);
m4->m5: ->m1 & s2=false & m3->m4 & pred1=true & !pred2 & ^ee1;
m5->m6: (ac_airborne == false & on_carrier == true) | {<m4>};
```

Primitive Action Definitions

A primitive action definition consists of:

action-keyword name (*positive-integer* , *positive-integer*) : *state-variable* = *function-name*() [*{ predicate-list }*] ;

- An *action-keyword* is ! or AC.
- The *positive-integer*'s denote the minimum and maximum execution times of the action, respectively.
- A *predicate-list* names predicates whose value may change as the result of the assignment of the function value to the variable. This is as opposed to predicates whose value may change simply as the result of any side-effects of executing the function, independently of the assignment of its value. The predicate list, if given, is a brace-enclosed comma-separated list of terms and values. The values are given in parentheses, and may be given in one of five ways:

  (1) true;

  (2) false;

  (3) ?, meaning that the value is TBD;

  (4) *, meaning that the value is computed nondeterministically; and

  (5) <>, meaning that the value depends deterministically upon the values of the data input to the function.

Examples include:

```
!A1(?,?): SYSALT = compute_altitude() /* comment */ { GT10000(?), AC_AIRBORNE(true), AC_ON_-SHIP(<>) } [description];
AC A2(?,250 V2 = turn_on() [description];
!A3(125,125): V3 = turn_off();
```

Composite Action Definitions

A composite action definition consists of:

action-keyword name : *action-expression;*

- An *action-keyword* is ! or AC.

- An *action-expression* is composed of parallel ("||") and sequential (";;") operators linking other action names (either composite or primitive). The operators have equal precedence so parentheses must be used to disambiguate.

  An expression may also include synchronization points, which are denoted by <n>, where n is any unsigned nonzero integer. Synchronization points come in sets; two sync operators with the same integer tag belong to the same set. They constitute a rendezvous mechanism in the composite action definition. Each sync point is a "virtual" action that sets no variable values and takes zero time to execute, but which will suspend the action or group in which it appears until all of its set partners have been reached.

Examples include:

!actn: (a1 || b1 || c1) ;; (d1 || (f1;;g1)); !actn: (a1 || b1 || c1) ;; (? || (f1;;?)) [description];
!actn: (a1 || b1 || c1) <1> ;; (? || <1> (f1;;?)) [description];
!actn: (a1 ;; a2 ;; a3) <2> || (b1 ;; <2> b2 ;; b3) !act13: g2 ;; /* comment */ h2;

External Event Definitions

An external event definition consists of:

event-keyword name [*non-negative-integer*];

- An *event-keyword* is ^ or EV.
- A *non-negative-integer* denotes the minimum separation between two consecutive occurrences of the event.

Examples include:

^ E1 [description] ;
EV E2 10 [description] ; /* comment */

State Variable Definitions

A state variable definition consists of:

variable-type name = initial-value;

- A *variable-type* is either real, integer, boolean or a string literal.

- For booleans, an *initial-value* is either “true” or “false”. For strings, it is a double-quoted string; string literals may extend across line boundaries, and may contain escaped double quotes. The *initial-value* may be “?’ (unknown) or “*” (don’t care).

Examples include:

```
real SYSALT = 100.756 [units = feet]; /* init value is altitude of base */
integer iv1 = -145;
integer iv2 = 0;
integer iv3 = +145;
boolean b1 = false;
boolean b2 = true;
othertype o1 = ?;
othertype o2 = *;
altitude ALT = “best system estimate of current altitude” [rules TBD] ;
```

Function Definitions

A function definition defines a function that defines a primitive action. It consists of:

name() : “*string-literal” [{ predicate-name-list }];*

- A *string-literal* defines the function body. It might be C code, but the MT does not examine it.
- An optional *predicate-name-list*, each given with a boolean value. These are the predicates that might change value as the result of evaluating the function. As in the predicate list for primitive actions, the boolean values may be “true”, “false”, “?”, “<>”, or “*”.

Examples include:

```
f1(): “ c code defining the parameterless function “ { p1(*) };
f2(): “ c code defining the parameterless function “ { p2(<>), p3(true) };
f3(): “ c code defining the parameterless function “ [description] ;
```

Typeclass Definitions

You may want to define some new typeclasses in your specification to which some state variables belong. We distinguish two kinds: numeric and enumerated. An enumerated typeclass is one whose variables may take on a finite and enumerated set of values, and for which the normal mathematical operations are not defined.

A numeric typeclass definition consists of:

NTC *name* [{ *physical-units-list* }];

An enumerated typeclass definition consists of:

ETC *name* { *domain-values* };

Examples include:

```
NTC real; /* no units needed for real */
NTC distance {feet, mi, km, inches};
NTC altitude {feet} [min=-1000, max=50000];
ETC boolean {true, false};
ETC master_arm_switch {boc, bocoffset, norm_attack, norm_offset, off};
```

Keywords

The following keywords are used in the preceding definitions:

mode definitions:	sm pm am co im pa sa
transition definitions:	infinity alarm deadline delay true false
action definitions:	ac
external event definitions:	ev
typeclass definitions:	real integer boolean ntc etc

Using the C Preprocessor

Modechart users need the capability to define templates for mode substructures and then instantiate those templates throughout the global structure at will. To provide a quick implementation of this capability, you can first run the database file through the C preprocessor (cpp), which gives you a macro facility via the #define command.

For example, suppose that you want to define a canonical structure that consists of a serial mode m1, with two children m2 and m3. To do this, you might include the following in your file:

```
 #define template(a) SM a.m1, CO a, IM a.m2, *;
      AM a.m2, CO a.m1, SA a24(150,300);
      AM a.m3, CO a.m1, SA a24(150,300);
      a.m2 -> a.m3: (alarm 50)|(deadline 25)|(1,infinity)|(delay 500);
template(X)
template(Y)
```

The first instantiation would create three modes X.m1 (child of X), X.m2 (child of X.m1), and X.m3 (child of X.m1). It would also define the transition from X.m2 to X.m3. The second instantiation would do the same for Y.

APPENDIX 4

Sample Simulation Options File

This appendix defines the syntax and semantics of the Modechart Simulator options file. By editing this file, the simulator user can select which options are in effect during a run of the Modechart Simulator.

The options file consists of five sections. Each will be explained in turn. The options are divided into several sections. The order of options within some sections is arbitrary, and prescribed in others. The order of the sections themselves is fixed.

The user is free to insert C-style comments /* like this */ anywhere in the options file; they will be ignored by the simulator. Input is free-form unless otherwise indicated: tokens may begin in any column, and must be separated by any form of "white space": blanks, tabs, or newlines. The following is an example of a legal Modechart Simulator Options file. The individual options are explained within the file in the form of comments.

```
/***************************************************************
 *
 * Sample Modechart Simulator User Options File
 *
 ***************************************************************/



/***************************************************************
 *
 * SECTION I: Simulation run length
 *
 * This value specifies the number of time units the simulation is to run.
 * A negative integer means "infinity", presumably when the simulator is to
 * be stopped by an external interrupt.
 *
 * This section consists of a single integer. Giving zero or a non-integer is
 * an error.
 *
 ***************************************************************/

2500



/***************************************************************
 *
 * SECTION II: Initialization options
 *
 * Under certain circumstances some timing parameters and state variable initial
 * values may be left unspecified in the loaded modechart. However, the simulator
 * must use values for all properties in order to produce a valid execution run.
 * The initialization options tell the simulator how to assign values left
 * undetermined in the loaded modechart.
 *
 * If all values are specified in the loaded modechart, this set of options
 * will have no effect and may be left empty.
 *
 * The order of the seven options within this section is fixed. Newlines
 * are not significant.
 *
 ***************************************************************/
```

```
/* II A. Action deadlines */

MIN     /* If no deadline for an action is given, what to use?
        * Options are:
        *     "n", an unsigned integer
        *     "MIN", meaning to use the action's maximum execution time;
        *     "*n", where n is an unsigned positive integer, defining
        *             the interval to be n times the action's maximum
        *             execution time;
        *     "EXP n", where n is an unsigned integer, meaning a random
        *             number with negative exponential distribution having
        *             mean "n". The random number is added to the action's
        *             maximum execution time.
        */


/* II B. Action execution times */

10      /* If an action's minimum execution time is not defined, what to use?
         * Option must be given by an unsigned integer. */
25      /* If an action's maximum execution time is not defined, what to use?
         * Option must be given by an unsigned integer. */


/* II C. State variables' initial values */

T       /* What initial value should a state variable be given, if none is
         * specified? Options are:
         *    "T", meaning all uninitialized variables are set to TRUE;
         *    "F", meaning all uninitialized variables are set to FALSE;
         *    "UNIF", meaning each uninitialized variable is set randomly
         *            (with uniform distribution) to TRUE or FALSE. */


/* II D. External events' minimum separation */

25      /* If an event's minimum separation is not defined, what to use?
         * Option must be given by an unsigned integer. */


/* II E. Mode transitions' timing */

20      /* If a mode transition has a (lb,ub) clause with an unspecified
         * lower bound value, what to use? */

100     /* If a mode transition has a (lb,ub) clause with an unspecified
         * upper bound value, what to use? */
```

```
/***************************************************************
 *
 * SECTION III: Next-time options
 *
 * Many choices during a simulation are nondeterministic; for example, the
 * actual execution times of actions, or the next time that an external event
 * will occur. The Simulator requires guidance about how to predict
 * these dynamic time values. Some options provide for random number
 * generation; when chosen, the simulator will generate a new random number
 * each time one is needed.
 *
 * Next-time options may be specified for an entire class of objects,
 * or for individual objects. The options in this section may be given
 * in any order. An option applying to a particular object may be given
 * more than once; the last one applies.
 *
 * Next-time options are available for actions, external events, and
 * mode transitions. Each option must appear on a separate line.
 *
 * This entire section may be left empty, in which case the simulator will
 * choose default options for each case. The defaults are described below.
 *
 * There are three types of next-time options: those governing actions,
 * timed mode transitions, and events.
 *
 ***************************************************************/
```

```
/**************************************************************
*
* III A. ACTION NEXT-TIME OPTIONS
*
*        1. WHAT IT MEANS: Determines the actual execution time for an action
*
*        2. HOW IT'S GIVEN: |- action_name -| value
*
*         where "value" is
*              "MIN" for minimum execution time;
*              "MAX" for maximum execution time;
*              "UNIF" for a random number (with uniform distribution) of
*                     cycles between the minimum and maximum exeuction times
*              "n" (unsigned positive integer) actual number of time units
*                     If a number less than the minimum execution time is
*                     given, the minimum execution time will be used.
*                     If a number greater than the maximum execution time is
*                     given, the maximum execution time will be used.
*
*        where "action_name" is the name of an action, or a pattern (see
*         below).
*
*        3. WHAT IF NO OPTION IS GIVEN? The effect is as though "|-*-| MAX"
*        had been given.
*
*        The following sequence sets all action times to uniform-random, except
*         A1 to 14 time units and all actions whose names begin with "B" to the
*         minimum execution time. Note that order is important; had the
*        order been reversed, all actions would be uniform-random, because if
*         more than one option applies to an object, the last one holds.
*
**************************************************************/
|-*-|  UNIF
|-A1-| 14
|-B*-| MIN
```

```
/****************************************************************
 *
 * III B. MODE TRANSITION NEXT-TIME OPTIONS
 *
 *        1. WHAT IT MEANS: When a transition has a (lower-bound,upper-bound)
 *         clause in its mode transition expression, and the source mode is
 *        active, when should the transition fire?
 *
 *         2. HOW IT'S GIVEN: mode_name -> mode_name value
 *
 *         where "value" is
 *               "LB", meaning always at the lower bound;
 *                "UB", meaning always at the upper bound; note that choosing
 *                        this option for a transition whose upper bound is
 *                         infinity causes the transition to never occur.
 *                "RAND", if upper bound is not infinity, then a random number
 *                        (with uniform distribution) between the lower and
 *                        upper bound. If the upper bound is infinity,
 *                         then a random number (with exponential distribution)
 *                        with mean equal to the default initial value for the
 *                         upper bound for (lb,ub) mode transitions.
 *               "n" (unsigned positive integer) actual number of time units
 *                        If a number less than the lower bound is given, the
 *                         lower bound will be used.
 *                        If a number greater than the upper bound is given, the
 *                        upper bound will be used.
 *
 *        where "mode_name" is the name of an mode, or a pattern (see below).
 *
 *        3. WHAT IF NO OPTION IS GIVEN? UB is used for each transition not
 *        otherwise specified.
 *
 *
 ****************************************************************/
m1->m2 LB
M4385 -> M43*4 20
```

```
/*****************************************************************
*
*III C. EVENT NEXT-TIME OPTIONS
*
*         1. WHAT IT MEANS: When should an external event occur?
*
*         2. HOW IT'S GIVEN: ^ event_name value_list
*
*        where "event_name" is the name of an event, or a pattern (see below).
*        where "value_list" is a list of "value"'s, separated by white space
*
*        where "value" is
*               "+n" (n an unsigned integer) Specifies that the named event(s) will
*                       occur at the minimum separation PLUS n time units after the last
*                       occurrence (or after time 0 if there hasn't yet been an occurrence).
*               "*n", (n unsigned positive integer), meaning that the named event(s)
*                       will occur at time equal to the time of the last occurrence (or 0 if
*                       no occurrence yet) plus n times the minimum separation.
*                       the minimum separation for the named event(s);
*               "EXP n", (n unsigned positive integer) meaning at a time obtained by
*                       summing the time of the last occurrence (or 0 if no occurrence
*                       yet), a random number (with negative exponential distribution of
*                       mean "n"), and the minimum separation for the named event(s).
*               "n n ..." (list of unsigned positive integers separated by white space)
*                       specifying actual times of occurrence. Up to 500 times may be
*                       given for each event.
*               0      meaning the named event(s) will never occur.
*
*        All of these values are mutually exclusive for a given event; if an event is given
*         more than one option, the last one given is the one used.
*
*        Also, in no case will an event be schedule in a way to violate its minimum
*         separation criterion. If the next-time option for an event would violate that
*        criterion, then the event will occur as soon as its minimum separation
*        constraint allows.
*
*        3. WHAT IF NO OPTION IS GIVEN? The effect is as though "^* 0" had
*         been given, cancelling all external events.
*
*        In the following, the last line has the effect of cancelling any occurrence of
*         ^E1, although "^E1 0" would have been a shorter way of doing the same
*         thing.
*
*****************************************************************/
^button_pressed @14 @27 @84
^target_sighted 0
^*launched MIN
^E1 EXP 42 +16 @84 0
```

```
/***************************************************************
 *
 * SECTION IV: Statistics and Log Entry options
 *
 * These options control what the simulator reports to the user.
 * The options within this section must be given in the order specified.
 * A “log” request causes an output during the run each time a particular
 * item of interest occurs.
 *
 * A “report” request causes a summary report to be printed at the end of
 * the log file, summarizing how often a particular item of interest occurred.
 *
 * Options must be given in the specified order. Where a list of items is called for,
 * no object whose name is “T” or “F” may be a member of the list.
 *
 ***************************************************************/

/* Processor utilization and clock */
 F       /* Log each time the simulation clock is incremented whether anything
          * happens at that time or not; if false, the clock time will be
          * reported only if some action is logged at that time. */
 F       /* Report number of idle processor cycles */
 F       /* Report percentage of idle processor time */
 F       /* Log “PROCESSOR IDLE” each time the processor is idle */


/* Modes */
 MOVEUP MOVEDOWN
 APPROACH BC
 TRAIN[12]
         /* Log entry and exit for each mode in list; “*” means all; empty
          * list means none. The list and order of the modes shown on the
          * graphical display is taken from this list. */

 T       /* Report number of times each mode is entered */
 T       /* Report number of times each mode is exited */
 F       /* Report number of processor cycles each mode is active */
 F       /* Report percentage of processor cycles each mode is active */
 F       /* Log “MODE mode_name’S ACTION action_name SCHEDULED TO
          * TAKE n TIME UNITS” each time a mode’s action is given a projected
          * execution requirement. */


/* Mode transitions */
 A->B   C->D
 MOVE* -> TRAIN[123]
         /* Log “modename->modename TAKEN” when any transition in the
          * list is taken; “*->*” means all; empty list means none. The list and
          * order of the transitions shown on the graphical display comes from
```

```
          * this list. */

T        /* Report number of times each transition is taken */

A->B C->D
         /* Log “modename->modename SATISFIED” when any transition in the
          * list is satisfied, whether taken or not; “*->*” means all;
          * empty list means none. */
T        /* Report number of times each transition is satisfied, but not
          * necessarily taken */
F        /* Log “modename->modename TIMED TRANSITION SCHEDULED FOR
          * n TIME UNITS AFTER ENTRY” when a (lb,ub)-based mode transition is
          * scheduled by the Simulator */


/* State variables */
V1 V2 V3
         /* Log “VARIABLE varname ASSIGNED boolean_value” when a variable
          * in this list is assigned a value during the simulation; log “VARIABLE
          * varname INITIALLY ASSIGNED boolean_value” when a variable is given a
          * value during initialization. “*” means all; empty list means none. The
          * list and order of state variables shown on the graphical display comes from
          * this list.

T        /* Report number of times each state variable is assigned a value
          * (excluding initialization at time zero) */

V3 V4 V5
         /* Log “VARIABLE varname CHANGED TO boolean_value” when a
          * variable in this list is assigned a new value during the simulation; log
          * “VARIABLE varname INITIALLY ASSIGNED boolean_value” when a
          * variable is given a value during initialization. “*” means
          * all; empty list means none. */

T        /* Report number of times each state variable is assigned a new value
          * (excluding initialization at time zero) */


/* Actions */
T        /* Log “ACTION action MINEXECTIME := n” when the Simulator discovers
          * an unspecified minimum execution time for an action and assigns one */
T        /* Log “ACTION action MAXEXECTIME := n” when the Simulator discovers
          * an unspecified maximum execution time for an action and assigns one */
T        /* Log “ACTION action of MODE mode HAS PROCESSOR” when an action is
          * given cpu time at the current time */

ALPHA BRAVO CHAR*
         /* Log “ACTION action OF MODE mode STARTED” when an action in
          * this list begins. “*” means all actions; empty list none. The list and order
```

```
          * of actions shown on the graphical display comes from this list. */

T        /* Report number of times each action is started */

A B C
         /* Log "ACTION action OF MODE mode COMPLETED" when a named
          * action finishes; "*" means all; empty list is none. */

T        /* Report number of times each action is completed */

A B C
         /* Log "ACTION action OF MODE mode ABORTED" when an action in the
          * list is aborted; "*" means all; empty list is none. */

T        /* Report number of times each action is aborted */
T        /* Report number of processor cycles consumed by each action */
T        /* Report percentage of processor cycles consumed by each action */
T        /* Report percentage of active processor cycles (i.e., cycles used by
          * any action) consumed by each action */


/* External events */
T        /* Log EVENT eventname MINSEP := n" when Simulator discovers an
          * unspecified minimum separation for an event and assigns one */

E1 E2 Event4891 Pilot_pushed_button
         /* Events for which to log "EVENT eventname OCCURRED" when it
          * occurs. "*" means all; empty list means none. The list and order of
          * external events shown on the graphical display comes from this list.*/

T        /* Report number of times each external event occurred */
T        /* Log EVENT eventname SCHEDULED FOR TIME t" when an event is
          * assigned a simulated occurrence time in the future. */
```

```
/**************************************************************
*
 * SECTION V: Breakpoint information
 *
 * By setting breakpoints, the user can cause the simulation to stop when the
 * clock reaches certain values or when certain events occur. At that point,
 * the user may examine the state of the simulation. The options in this
 * section can occur in any order. Options are separated by white space.
 *
 * If no breakpoint options are present, the simulator will run until the
 * simulation run length option is satisfied.
 *
 * Options are of the form:
 *         ->modename break after the named mode has been entered
 *         modename-> break after the named mode has been exited
 *         modename->modename break after the named transition has been taken
 *         |-actionname break after the named action has started
 *         -|actionname break after the named action has finished
 *         ^externaleventname break after the named event has occurred
 *         statevarname:=t break after the named variable has become true
 *         statevarname:=f break after the named variable has become false
 *         statevarname:=* break after the named variable has been assigned
 *         n (n unsigned integer) break at time n
 *         *n (n unsigned integer) break at every time divisible by n
 *
 * In all cases, the simulator stops AFTER all work has been done for the
 * time value at which the event occurs. All names may be given by
 * patterns (see below).
 *
 * The following sequence requests breakpoints at time 10, 20, 30, at any time
 * that is a multiple of 6 or 7, whenever any action whose name begins with "A"
 * starts, and whenever mode M1 or M2 is entered.
 *
 **************************************************************/
10
 20
 30
 *6
 *7
 |-A*
 ->M[12]


/* END OF FILE */
```

Index