RESEARCH REPORT IIAS-RR-93-9E

D-ABDUCTOR 2.0 User Manual

Kazuo Misue

May, 1993

International Institute for Advanced Study of Social Information Science,
FUJITSU LABORATORIES LTD.

Numazu office
140 Miyamoto, Numazu-shi, Shizuoka 410-03, Japan
Telephone: +81-559-23-2222 Fax: +81-559-24-6180

Tokyo office
9-3 Nakase 1-chome, Mihama-ku, Chiba-shi, Chiba 261, Japan
Telephone: +81-43-299-3211 Fax: +81-43-299-3075

D-ABDUCTOR

2.0

User Manual

Kazuo Misue

Contents

1. Introduction to D-ABDUCTOR

D-ABDUCTOR is a system to support dynamic thinking processes of humans by sophisticated use of graph drawing algorithms. It is developed to aim at attaining an effective integration of human thinking capability and computer information-processing capability. Diagrams are good media to reflect and organize personal thoughts and to communicate with other persons in collaborative works. D-ABDUCTOR provides several new facilities to deal with diagrams that can represent both adjacency and inclusion relationships among cards. It is the first system that provides the facilities based on automatic drawing of such diagrams.

Features of D-ABDUCTOR

D-ABDUCTOR provides the following facilities.

Compound Graphs

D-ABDUCTOR provides an environment to deal with diagrams based on compound graphs that have both adjacency and inclusion edges. The environment includes a direct manipulation interface and menus for simple operations.

Automatic Diagram Drawing

D-ABDUCTOR provides an automatic drawing facility for compound graphs, using an algorithm based on cognitive criteria. Certain operations selected by the users can trigger the invocation of this facility for visual response. This facility also supports many of other facilities mentioned below.

Collapse and Expand Operation

The collapse operation collapses a group of vertices into a vertex to make an outline of a diagram. The expand operation expands a vertex that has been collapsed to a group to make return the detail to the diagram.

Abridgment

Abridgment changes size of vertices according to their importance. More important vertices become larger, and less important vertices become smaller or omitted. Structure of diagrams, semantics of vertices and the specific user's viewpoint influence importance of vertices.

Display with Animation

D-ABDUCTOR provides display of changes with animation. The animation reduces the instantaneous visual change so that the changes preserve the user's mental map.

Communication

D-ABDUCTOR can communicate with other systems to share text, images and others. D-ABDUCTOR can also communicate with other processes of D-ABDUCTOR on other workstations to share information.

Getting Started

System Requirements

Software Environment

D-ABDUCTOR works under UNIX operating system and the X window system version 11 release 5. Thus you need UNIX workstations. It has confirmed that D-ABDUCTOR works on Sun-4/110, SPARCstation 1, 1+ and 2. You also need XView version 3 to compile and execute the D--ABDUCTOR program.

Window Manager

D-ABDUCTOR provides a user-interface following OPEN LOOK. You are recommended using D-ABDUCTOR with the window managers providing OPEN LOOK environment such as *olwm* and *olvwm.*

Text Editor

D-ABDUCTOR does not have the facilities to edit text. D-ABDUCTOR invokes an external text editor to exploits the text editing facilities. If you would like to enter and edit text while working with D-ABDUCTOR, you should prepare a text editor for example, *Emacs* in the same environment. If you hope dealing with Japanese, the text editor should also be able to deal with Japanese.

Installing D-ABDUCTOR

To prepare files

1. Make a directory, for example, **ABD2** in an appropriate directory.

% mkdir ABD2

2. Visit the new directory.

% cd ABD2

3. Extract files from the archive of the D-ABDUCTOR programs.

This creates three subdirectories in the new directory.

To compile the D-ABDUCTOR programs

1. Visit a subdirectory **abductor2.**

% cd abductor2

2. Make a **makefile** from the Imakefile.

% xmkmf -a

You see some error messages because there is not a file "sysvar.hh," which is used by checking dependency among source files.

3. Make a file "sysvar.hh"

% make sysvar.hh

4. Make a **makefile** from the imakefile again.

% xmkmf -a

5. Compile the D-ABDUCTOR program.

% make

To compile the Message Transmitter program

1. Visit a subdirectory abdtrans.

% cd ../abdtrans

2. Make a **makefile** from the Imakefile.

% xmkmf -a

3. Compile the Message Transmitter program.

% make

To compile the Card Base program

1. Visit a subdirectory cardbase.

% cd ../cardbase

2. Compile the Card Base program.

% make

2. Basic Use of D-ABDUCTOR

This manual has two chapters describing how to use of D-ABDUCTOR. One is this chapter and the other is the next chapter, "Advanced Use of D-ABDUCTOR." D-ABDUCTOR provides many facilities. However you do not need to learn all of them to work with D-ABDUCTOR. This chapter focuses on only basic skills and concepts you will need for using D-ABDUCTOR. In this chapter, you will learn the followings:

- Basic operations for the mouse, menus, dialog boxes.
- Starting and quitting D-ABDUCTOR.
- Creating and editing diagrams.
- Loading and saving diagrams.

Operations for the Mouse

To work with D-ABDUCTOR, you mainly use the mouse. You have to master operations for the mouse.

Mouse Buttons

D-ABDUCTOR requires a mouse with three buttons. These buttons are called the select button, the adjust button, and the menu button, respectively.


Select button

The Select Button The left button is called the select button. The select button is used to select an element, press command buttons and others.


Adjust button

The Adjust Button The middle button is called the adjust button. The adjust button is used to adjust the status of selection of elements.


Menu button

The Menu Button The right button is called the menu button. The menu button is used to open menus and choose a menu item.

Basic Operations

There are three basic operations often used working with D-ABDUCTOR. They are pointing, clicking, and dragging.


Pointer

Pointing Moving the mouse to place the pointer on an item is called pointing.

Clicking Pointing to an item and then quickly pressing and releasing a mouse button is called clicking.

Dragging Holding down a mouse button as you move the pointer is called dragging.

Operations for Menus

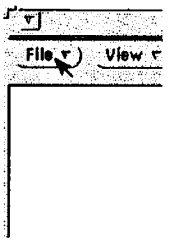
D-ABDUCTOR provides menus. You can execute commands by choosing menu items. This section describes operations for the menus.

D-ABDUCTOR has two kinds of menus. Most menus are placed in the upper side of the main window, and are pull-down menus. Each node has the other kind of menu, which is called the node menu, and is pop-up menu. The node menu has a submenu.

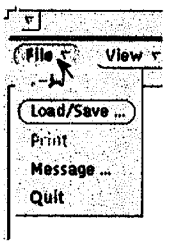
Choosing Menu Items

To choose a menu item

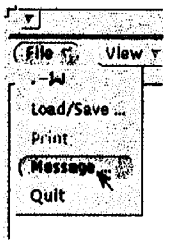
1 Point the menu you want to open.



2 Press the menu button on the menu to open the menu.



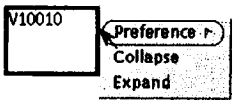
3 Drag to the menu item you want to choose, and release the menu button.



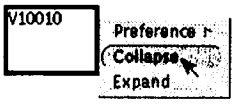
To choose a menu item of the node menu

1 Point a boundary line of a node whose menu you want to open.

2 Press the menu button on a boundary line of the node to open the node menu.



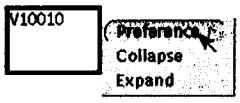
3 Drag to the menu item you want to choose, and release the menu button.



To choose a submenu item

The node menu has a submenu. The menu item whose label is followed by a triangle has a submenu.

1 Open the node menu, and drag to the menu item whose label is followed by a triangle.

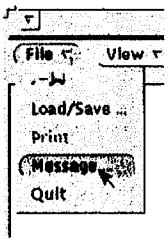


2 Drag to the triangle to open the submenu, choose a submenu item, and release the menu button.

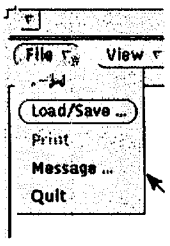
Canceling Choosing Menu Items

To cancel choosing a menu item

1 You are choosing a menu item.



2 Drag to the outside of the menu, and release the menu button to close the menu.



Operations for Dialog Boxes

D-ABDUCTOR provides some dialog boxes. Dialog boxes have buttons, check boxes, and sliders. This section describes how to open and close dialog boxes and basic operations for buttons, check boxes, and sliders.

Opening and Closing Dialog Boxes

To open dialog boxes

All dialog boxes are opened by using menus. You see some menu items whose labels are followed by three dots, for example "Load/Save..." in the File menu. The dots mean that choosing the menu item opens a dialog box.

To close a dialog box

Stuck push pin

Unstuck push pin

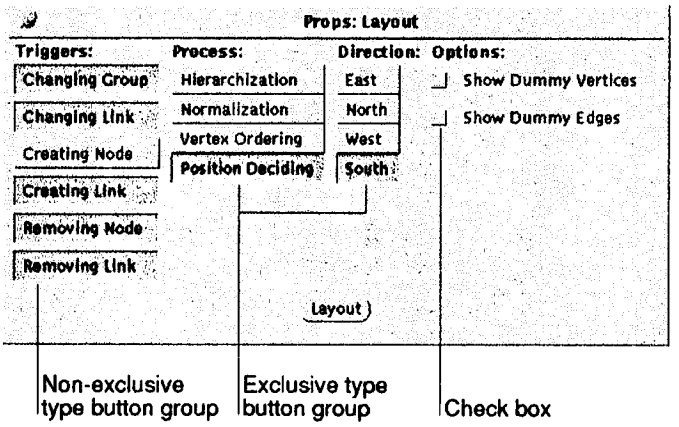
When you are using a window manager that follows OPEN LOOK, each dialog box has a push pin to leave the dialog box open. Unsticking the push pin closes the dialog box. To stick an unstuck pin or to unstick a stuck pin, you click them.

Operating of Buttons, Check Boxes, and Sliders

To operate buttons

Dialog boxes have some software buttons. In this manual, they are also called buttons. Pushing or clicking a software button mean clicking the select button (that is, a hardware button) on the software button.

Some (software) buttons consist groups. There are two kinds of button groups: the exclusive type and the non-exclusive type. Buttons in an exclusive type group have no space among them, while buttons in a non-exclusive type group have some space among them. With a button group of exclusive type, you can push only one button at once. To release a pushed button, you push another button. With a button group of non-exclusive type, you can push zero or more buttons at once. To release a pushed button, you push the button again.



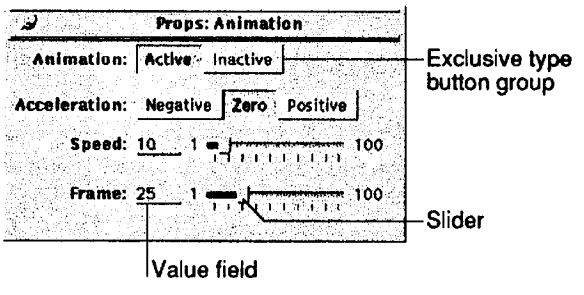
Example of dialog box (1)

To operate check boxes

Check boxes are similar to button groups of non-exclusive type. To check or uncheck a check box, you click the select button on the check box.

To operate sliders

Dialog boxes have some software sliders. To change their values, you drag the sliders. Each slider has a value field, which displays its value in digital. You can type a value in the value field directly. When you press return key after typing a value, the slider moves automatically.



Example of dialog box (2)

Starting D-ABDUCTOR

Before starting the D-ABDUCTOR program, X window system has to be working.

To start D-ABDUCTOR

- Execute command **abd** under UNIX

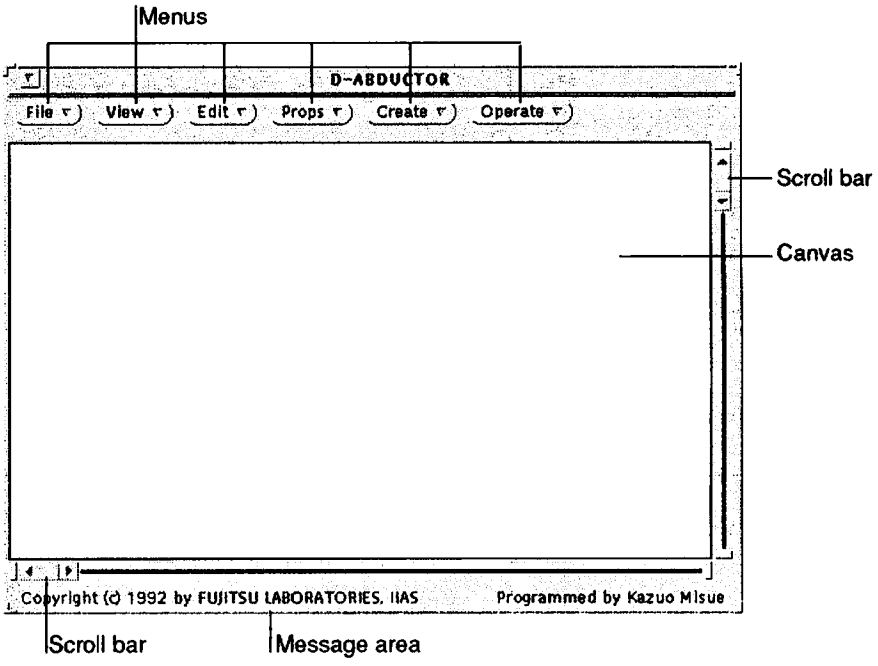
```
% abd
```

This starts D-ABDUCTOR and opens a D-ABDUCTOR main window.

Some command line options are available similar to other client programs of the X window system.

For more information about the files ".abductor_pref" and ".abductor_init", see Chapter 3, "Advanced Use of D-ABDUCTOR."

Note You see some error messages, unless you have two files ".abductor_pref" and ".abductor_init" in your home directory. If you dislike to see these error messages, make two empty files with these names.



D-ABDUCTOR main window

Note If you are using the window manager that does not follow OPEN LOOK, you will see another style of window frames.

Selecting and Unselecting Elements

Unselected node

Selected node

Selecting Elements

To select a node

- Pointing the node you want to select, click the select button.

This makes the node selected, and eight handles appear on its boundary lines. If you select a node in this way, all other elements will be unselected.

Note To point a node, you have to point a boundary line of the node. Inside of nodes are not regarded as themselves because nodes can include other nodes.

To select a link

- Pointing the link you want to select, click the select button.

This makes the link selected, and some handles appear on its line segments. If you select a link in this way, all other elements will be unselected.

To select two or more elements

- Pointing the nodes and the links you want to select, click the adjust button.

Even if you click the adjust button pointing a node or a link, other elements that have been selected will not be unselected.

To select all elements

- From the Edit menu, choose **Select All**.

Handles appear on all elements.

Unselecting Elements

To unselect an element

- Pointing the element you want to unselect, click the adjust button.

You adjust the status of selection of elements by using the adjust button. If you click the adjust button pointing a selected element, the element will be unseleced. You can select it by clicking the adjust button again.

To unselect all elements

- At empty space of the canvas, click the select button.

Note When no elements (nodes and links) are selected, clicking the select button at empty space of the canvas causes redrawing of diagrams.

Creating Diagrams

Creating New Diagrams

To create a new diagram

- From the Edit menu, choose **New**.

This deletes the old diagram and cleans the canvas.

Note You do not need perform this operation just after starting D-ABDUCTOR.

Creating New Nodes

To create a new node

- From the Create menu, choose **Node**.

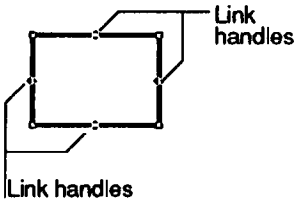
This creates a node and places it on the default position. The node just after created is selected. If system variable **textedit_options** is 2 or 3, a text editor is opened at the position of the new node.

For more information about system variables, see Chapter 5, "Language Simple."

For information about editing text, see the following section, "Editing Diagrams" in this chapter.

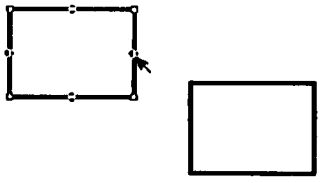
Creating New Links

To create a link

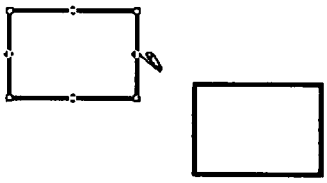



Pencil

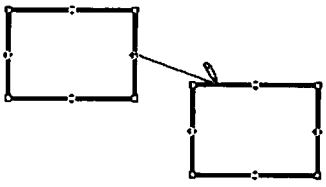
1 Select the tail node of the new link. Eight handles appear on the boundary lines of the tail node. The four handles on the middle points of four boundary lines are called link handles.



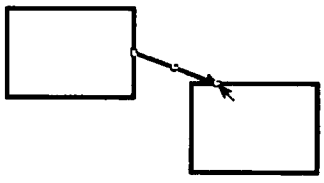
2 Press the select button pointing a link handle. This changes shape of the pointer to a pencil.



3 Drag to the head node of the new link. When you are dragging the mouse, if you points a node, handles appear on the node



4 Release the select button pointing the head node. This create a new link from the tail node to the head node. The link just after created is selected.



Creating New Groups

To create a group

1 Select one or more nodes.

2 From the Create menu, choose **Group**.

This create a new group that includes all selected nodes. A group node just after created is the selected group.

For more information about system variables, see Chapter 5, "Language Simple."

If system variable **textedit_options** is 2 or 3, a text editor is opened at the position of the new group node.

Note A group is a node that includes one or more other nodes. It is called "group node" if it should be distinguished from the nodes that include no nodes. Group nodes can be also connected with other nodes by links.

Editing Diagrams

Deleting Elements

You choose **Cut** from the Edit menu to delete selected elements.

To delete elements

1 Select the elements you want to delete.

2 From the Edit menu, choose **Cut**.

This deletes all the selected elements.

Bugs When a lot of elements are deleted at once, D-ABDUCTOR might stop.

Editing Text of Nodes

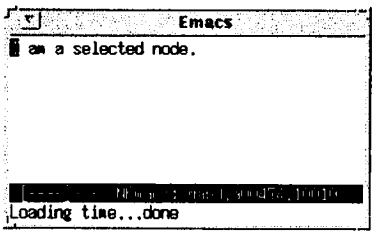
You choose **Text** from the Edit menu to open a window of a text editor.

To edit text of a node.

1 Select the node whose text you want to edit.

2 From the Edit menu, choose **Text**.

This opens a window of a text editor. If the node has text, the text editor initially loads the text.



Window of text editor (Emacs)

3 Edit or enter text with the text editor.

4 Save the text and quit the text editor.

This updates the text of the node.

Note You should refer the manuals of the text editor you are using to know about it detail.

Bugs Even if you have selected two or more nodes when you choose **Text** form the Edit menu, only one window of a text editor appears

Moving Nodes

To move a node

1 If two or more elements have been selected, unselect all elements or select only the node you want to move.

2 Press the select button on the node you want to move.

Note When a node is selected, the node has eight handles. But you do not have to press the select button on these handles to move the node.

3 Drag the mouse slightly.


Hand

This changes the shape of the pointer to a hand.

4 Drag to the objective position.

5 Release the select button.

Note The moved node is selected even if the node has never been selected. If you move some group nodes, all nodes included by them are also moved.

To move two or more nodes

1 Select the nodes you want to move.

2 Press the select button on one of the selected nodes.

3 Move the mouse slightly.

This changes the shape of the pointer to a hand.

4 Drag to the objective position.

5 Release the select button.

Changing Groups

To add existing nodes to a group

- Move the added nodes inside of the group node.

When you are dragging the mouse, if pointer is inside of a node, handles appear on the node

It is also possible to make a node including no other node (that is, non-group node) a group node.

Note When you move the pointer to add nodes to a group, you should release the select button inside of the group node. All added nodes do not need to be inside of the group node.

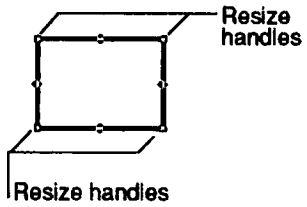
To delete a node from a group

- Move the deleted nodes outside the group node.

When you are dragging the mouse, if pointer is outside a group node, handles on the group node disappear.

Note When you move the pointer to delete nodes from a group, you should release the select button outside the group node. All deleted nodes do not need to be outside the group node.

Resizing Nodes



Resize mark

To resize a node

1 Select the node you want to resize.

Eight handles appear on the boundary lines of the tail node. The four handles on the corners are called resize handles.

2 Press the select button pointing a resize handle.

This changes shape of the pointer to a resize mark.

3. Drag the resize handle.

An outline of the node changes with the pointer.

4. Release the select button.

The node is its new size and is selected.

Drawing Diagrams Automatically

D-ABDUCTOR provides the facility of automatic drawing diagrams. You can change the layout of diagrams automatically by this facility.

To lay out the diagram automatically

- From the Operate menu, choose **Layout**.

This lays out the diagram automatically. The diagram changes to new layout with animation.

Note New layout of the diagram decided automatically depends on the layout before application of the automatic drawing facility. Try to apply automatic drawing facility after change layout of diagrams manually.

Bugs Do not perform other operations before the animation stops, or the diagram on the canvas will get out of order. If the diagram have gotten out of order, choose **Layout** from the Operate menu again.

Loading and Saving Diagrams

For more information about language Simple, see Chapter 5, "Language Simple."

You can load data of diagrams and save diagrams to files. Data of diagrams are described in language Simple.

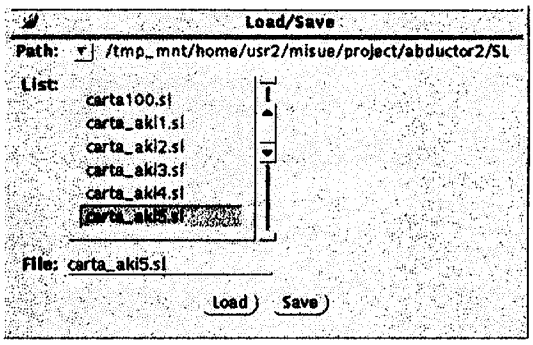
Opening the Load/Save Dialog Box

To load and save diagrams, you use the Load/Save dialog box.

To open the Load/Save dialog box.

- From the File menu, choose **Load/Save**.

This opens the Load/Save dialog box.



Load/Save dialog box

Specifying Files

You specify the file name to load or save. You can specify an existing directory and an existing file by choosing it from the Path menu and the List of files.

To visit a super directory

- From the Path menu, choose the directory you want to visit.

This updates the List of files.

To visit a subdirectory

- From the List of files, choose the directory you want to visit.

This updates the List of files and the Path menu.

To choose an existing file name

- From the List of files, choose the file name you want to load or save.

By this, the name of the chosen file appear in the File field.

Note When you first open the Load/Save dialog box, or when you update the List of files, a file name seems chosen. But you have to click the file name at least once to choose it. You can confirm what file is chosen by seeing the File field.

Loading Diagrams

If you have a file of diagrams, you can load the file to continue works on the diagram.

To load data of a diagram from a file

1 From the List of files, choose the file you want to load.

– or –

Type the file name you want to load in the File field.

2 Click the Load button.

This loads data of diagrams in the specified file. When the data are loading, message "Loading..." is displayed in the message area. When loading has finished, message "Loading...done." is displayed.

Saving Diagrams

You can save data of the diagram on the canvas to a file. But information about the positions of elements is not saved.

To save data of the diagram to a file

1 From the List of files, choose the file you want to save.

– or –

Type the file name you want to save in the File field.

2 Click the Save button.

This creates a file with the specified name, and saves data of the diagram on the canvas into the file. When the data are saving, message "Saving..." is displayed in the message area. When saving has finished, message "Saving...done." is displayed.

Note If you choose an existing file name, the file is over written without confirmation.

Quitting D-ABDUCTOR

To stop D-ABDUCTOR

- Form the File menu, choose **Quit**.

This closes the main window and all dialog boxes, and terminates the D-BDUCTOR program.

3. Advanced Use of D-ABDUCTOR

This chapter describes all facilities of D-ABDUCTOR except what described in the chapter 2, "Basic Use of D-ABDUCTOR." If you have never read chapter 2, read it before reading this chapter. In this chapter, you will learn the followings:

- Details of automatic drawing facility.
- Changing properties of elements and view of nodes.
- Collapsing and expanding
- Abridgment facility
- Animation facility
- Sending messages to the D-ABDUCTOR system
- Getting Information about diagrams
- Communication facility
- Using system files
- Customization

Automatic Drawing Details

For more information about automatic drawing algorithm, refer the paper "Visualization of Structural Information: Automatic Drawing of Compound Digraphs" in IEEE SMC, Vol. 21, No. 4, 1991.

D-ABDUCTOR provides an automatic drawing facility that draws compound graphs (area-net diagrams) by using an algorithm based on cognitive criteria. The automatic drawing facility draws diagrams hierarchically, that is, lays out nodes on one of nested parallel bands, and orients most links to the same direction orthogonal to the nested bands.

Some editing operations may trigger the invocation this facility for visual response. You can select what operations trigger it. The direction to which most links are oriented is called the layout direction. You can choose four layout directions. The algorithm to draw compound graphs consists of four steps. You can select a step as the final step to see output of each step. The algorithm adds some dummy nodes and links to normalize compound graphs. You can see these dummy elements by choosing some options.

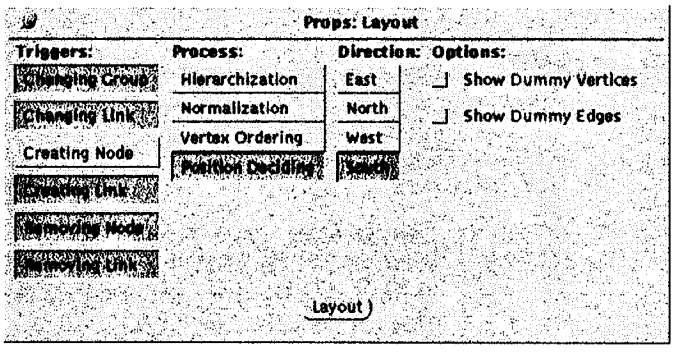
Opening the Layout Dialog Box

You use the Layout dialog box to select the triggers, the layout direction and the final step, and to choose some options.

To open the Layout dialog box

- From the Props menu, choose **Layout**.

This opens the Layout dialog box.



Layout dialog box

Selecting Layout Triggers

Visual response of editing operations sets you at ease. The editing operations that change the structure of diagrams (that is, compound graphs) may trigger the invocation the automatic drawing facility for visual response. The editing operations trigger it are called the layout triggers. You can select one or more operations as the layout triggers.

To select layout triggers

- From the Triggers buttons, choose some of them.

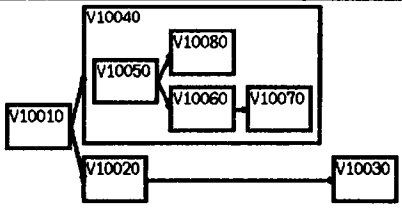
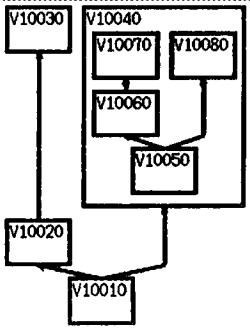
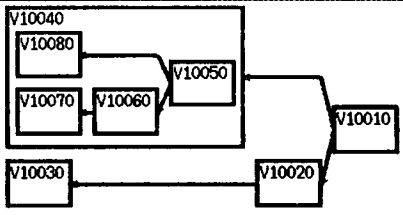
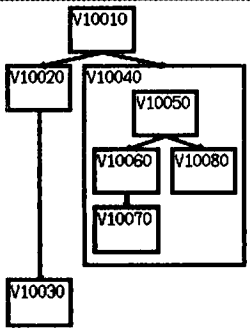
Choose	To lay out when
Changing Group	a group is changed
Changing Link	a link is changed
Creating Node	a node is created
Creating Link	a link is created
Removing Node	a node is removed
Removing Link	a link is removed

Changing Layout Directions

The automatic drawing facility orients most links to the same direction, which is called the layout direction. Available directions are four: right-ward, left-ward, upward, and downward. When the layout direction is right-ward or left-ward, nodes are stuffed to left side. When the layout direction is upward or downward, nodes are stuffed to upper side.

To choose a layout direction

- Choose one of the Direction buttons.

Choose	To orient links to
EAST	 right-ward
NORTH	 upward
WEST	 left-ward
SOUTH	 downward

Selecting Subprocess of Automatic Drawing

The algorithm to draw compound graphs consists of four steps: hierarchization, normalization, vertex ordering, and position deciding. You can stop the algorithm at a step you like, and see the output of the step. It is useful to know the work of each step.

To select subprocess of automatic drawing

- Choose one of the Process buttons.

Choose	To perform until that
Hierarchization	Levels that represent nested parallel bands are assigned to all nodes.
Normalization	Some dummy nodes and links are added to normalize the compound graph.
Vertex Ordering	The order of nodes in each level is decided to reduce the number of crossing links.
Position Deciding	The position of each node is decided to reduce the length of links and to symmetries links.

Note These buttons are used to debug of the automatic drawing facility or to show the work of each step of the algorithm. Ordinal users should choose **Position Deciding**.

Options of Automatic Drawing

The algorithm adds some dummy nodes and links to normalize compound graphs. For ordinal use, these dummy elements are invisible. But, you can see these dummy nodes and dummy links by selecting options.

Show Dummy Vertices Select this option to show dummy nodes used by the automatic drawing algorithm.

Show Dummy Edges Select this option to show dummy links used by the automatic drawing algorithm.

Note Nodes and links are respectively called vertices and edges to emphasize they are elements of compound graphs. These options are mainly used for debugging.

Performing Automatic Layout

To lay out the diagram

You have two ways to perform automatic layout.

- Click the Layout button on the Layout dialog box.

  – or –

  From the Operate menu, choose **Layout**.

Changing Properties

Elements of diagrams have some visual properties. Visual properties you can change are shape style, line style, line width, and color. You can change these properties of each existing element and default properties that are used for new elements.

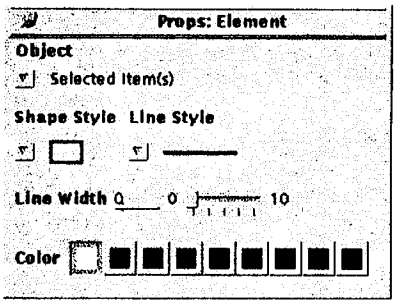
Opening the Element Dialog Box

To change properties of elements, you use the Element dialog box.

To open the Element dialog box

- From the Props menu, choose **Element**.

This opens the Element dialog box.



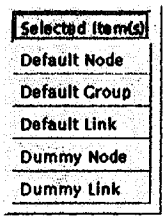
Element dialog box

Note According to the options of installation, the number of color buttons can be different from the above figure.

Selecting Objects

You can change properties of existing elements and default properties. Before selecting properties, you select objects whose properties you want to change.

To choose an object whose properties are changed



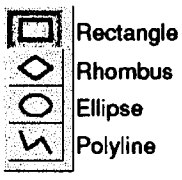
- From the Object menu, choose one of the followings:

Choose	Objects
Selected Item(s)	selected elements
Default Node	default properties of nodes
Default Group	default properties of group nodes
Default Link	default properties of links
Dummy Node	default properties of dummy nodes
Dummy Link	default properties of dummy links

When you choose Selected Item(s) and one or more elements are selected, you see the results of operations to change properties immediately.

Selecting Shape Styles

You can change the shape style of selected elements and default shape style of nodes and links. Available shape styles are rectangle, rhombus, ellipse, and polyline.



To change shape styles

1 Select elements whose shape style you want to change.

You do not need to do this when you have chosen except **Selected Item(s)** from the Object menu.

2 From the Shape Style menu, choose one of them.

Note You cannot choose polyline for any nodes and cannot choose rectangle, rhombus, and ellipse for any links. You can choose rhombus and ellipse for group nodes, but it is possible that the group nodes do not include their members.

Selecting Line Styles

You can change the line style of selected elements and default line style of boundary lines of nodes and links.

To change line styles

1 Select elements whose line style you want to change.

You do not need to do this when you have chosen except **Selected Item(s)** from the Object menu.

2 From the Line Style menu, choose one of them.

Selecting Line Width

You can change the line width of selected elements and default line width of boundary lines of nodes and links.

To change line width

1 Select elements whose line width you want to change.

You do not need to do this when you have chosen except **Selected Item(s)** from the Object menu.

2 Drag the Line Width slider.

Selecting Colors

You can change the color of selected elements and default color of boundary lines of nodes and links.

To change colors

1 Select elements whose color you want to change.

You do not need to do this when you have chosen except **Selected Item(s)** from the Object menu.

2 Click one of the Color buttons.

Note According to the options of installation, different color sets might be available.

Changing View of Nodes

Node with text

Node with image

Each node can have data of text and data of an image. Thus each node has two kinds of view, that is, displayed with text or displayed with an image. You can select a preference of view of each node. A node is displayed in the preferable view if the node has the data you prefer.

Changing View of a Node

To change preference of view of a node, you use the Node menu of the node. Menu item **Preference** in the Node menu has a submenu with two menu items: **Text** and **Image**.

To change preference of view of a node

1 Open the Node menu on the node whose view you want to change.

2 From the Preference submenu of the Node menu, choose **Text** or **Image**.

Choose	When you prefer
Text	displayed with text
Image	displayed with an image

Changing View of All Nodes

To change preference of view of all node, you use the View dialog box.

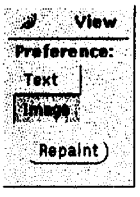
To change preference of view of all nodes

1 From the Props menu, choose **View**.

This opens the View dialog box.

2 From the Preference buttons, choose one of them.

Choose	When you prefer
Text	displayed with text
Image	displayed with images

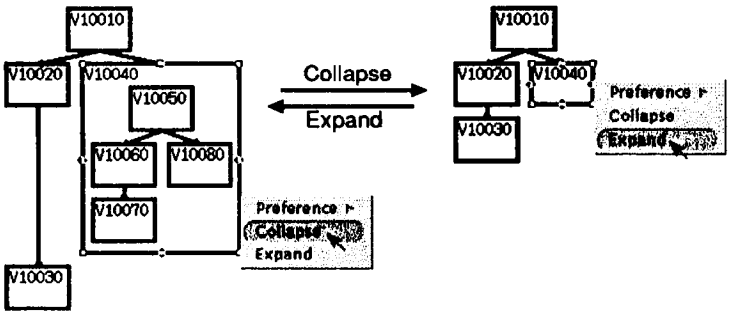


View dialog box

Collapsing and Expanding

You can collapse group nodes to make diagrams outline. A collapsed group node is drawn as a node that seems to include no other nodes. All the nodes

included by a collapsed node are omitted. Expanding is the reverse operation of collapsing. You can expand collapsed groups to restore them.



Collapsing and Expanding a Group

To collapse or to expand a group node, you use the Node menu of the node.

To collapse a group

1 Press the Menu button on the node you want to collapse.

This opens the Node menu.

2 From the Node menu, choose **Collapse**

To expand a group

1 Press the Menu button on the node you want to expand.

This opens the Node menu.

2 From the Node menu, choose **Expand**

Collapsing and Expanding Groups

To collapse or to expand selected group nodes, you use the Operate menu.

To collapse groups

1 Select group nodes you want to collapse.

2 From the Operate menu, choose **Collapse**

To expand groups

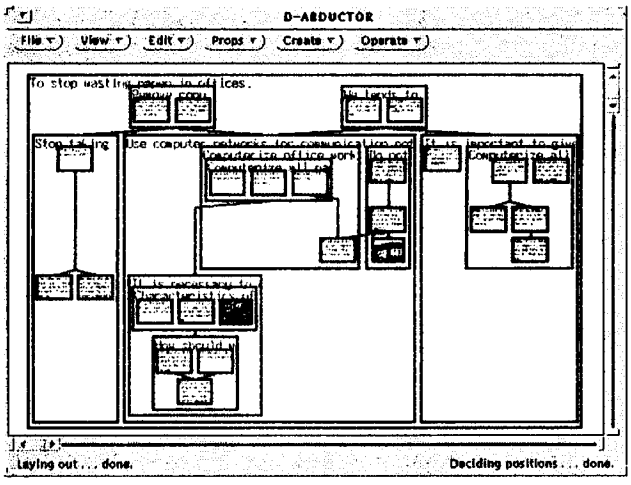
1 Select collapsed nodes you want to expand.

2 From the Operate menu, choose **Expand**

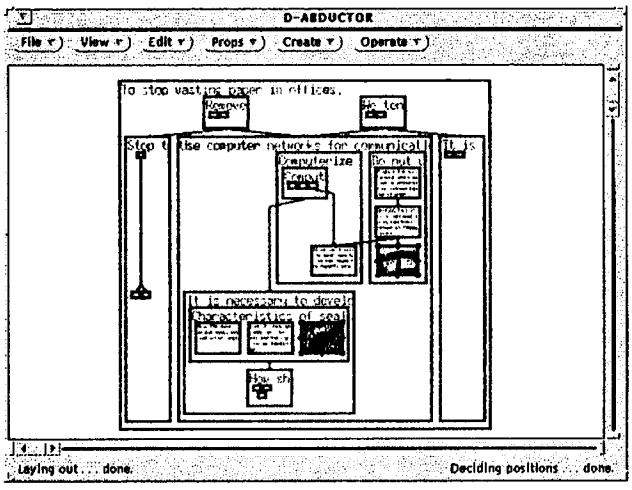
Note When you collapse a node, if the node includes no other nodes, the node is not changed. When you expand a node, if the node has never been collapsed, the node is not changed.

Abridgment

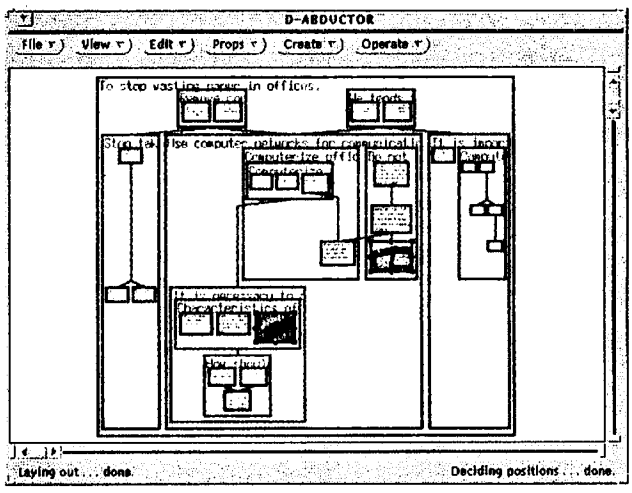
Abridgment is a rather automatic collapsing and expanding facility. Abridgment changes size of nodes according to their importance. More important nodes become larger, and less important nodes become smaller or omitted.



Abridgment inactive



Abridgment active (hybrid type)



Abridgment active (proportional type)

The abridgment facility consists of the importance function and the weighted drawing. The importance function gives a value called importance to each node. The weighted drawing draws nodes sized according to their importance.

The Importance Function

The importance function gives importance to each node by linearly combining three primitives of importance: the structural importance, the semantic importance, and the focal importance.

For more information about language Simple, see Chapter 5, "Language Simple."

The structural importance of a node is defined by its depth of the nesting level. A node enclosed by fewer nodes is more important. The semantic importance is defined by you. You can give some value to each node as the semantic importance. Only a way to give the semantic importance to a node is by using language Simple. The focal importance is defined as structural closeness to the focal nodes on the compound graph. The selected nodes are used as the focal nodes.

The Weighted Drawing

The weighted drawing draws diagram by using importance of nodes. D-ABDUCTOR provides two types of weighted drawing: the hybrid type and the proportional type.

The hybrid type weighted drawing uses two thresholds. All nodes with less importance than the lower threshold are omitted. All nodes whose importance values are less than the higher threshold and greater or equal to the lower threshold are drawn in small size. Other nodes are drawn in their original size. But size of group-nodes may not follow the rule.

The proportional type weighted drawing draws nodes in size proportional to their importance value and their original size. But group nodes may not follow the rule.

Opening the Abridgment Dialog Box

To exploit the abridgment facility, you use the Abridgment dialog box.

To open the Abridgment dialog box

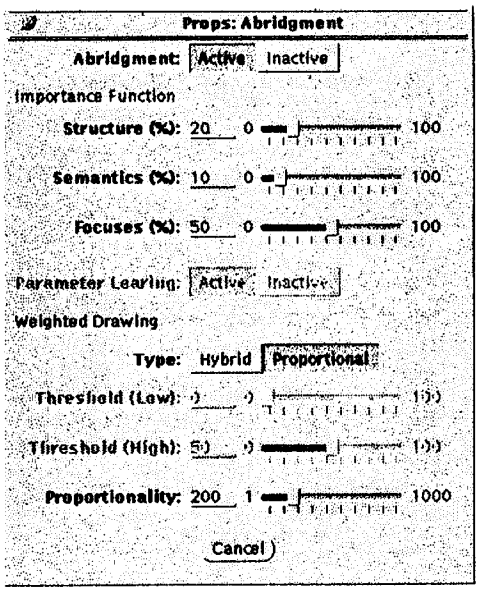
- From the Props menu, choose **Abridgment**.

This opens the Abridgment dialog box.

To close the Abridgment dialog box

- Click the **Cancel** button.

Note Clicking the Cancel button does not cancel any operations you have done since opening the Abridgment dialog box.



Abridgment dialog box

Making Abridgment Active and Inactive

To make abridgment active and inactive

- Choose one of the Abridgment buttons.

Choose	To do
Active	make abridgment active
Inactive	make abridgment inactive

When you choose Inactive, other buttons and sliders in the Abridgment dialog box are not available.

Changing the Importance Function

You can customize the importance function as you like by changing weight coefficients of linear combination of three primitives. Changing weights causes the change of view of the diagram immediately.

To change the weight of the structural importance

- Drag the Structure slider.

To change the weight of the semantic importance

- Drag the Semantics slider.

To change the weight of the focal importance

- Drag the Focus slider.

Note Three weight coefficients are normalized to be totally one.

Changing the Weighted Drawing

To select a weighted drawing

- Choose one of the Type buttons.

Choose	To select
Hybrid	the hybrid type weighted drawing
Proportional	the proportional type weighted drawing

When you select the hybrid type weighted drawing, the Threshold (Low) slider and the Threshold (High) slider are available. When you select the proportional type weighted drawing, the Proportionality slider is available.

To change the lower threshold for the hybrid type

- Drag the Threshold (Low) slider.

To change the higher threshold for the hybrid type

- Drag the Threshold (High) slider.

To change the proportional constant for the proportional type

By the Proportionality slider, you choose the proportional constant used by the proportional type weighted drawing. The nodes with the highest importance value are drawn in the size of the constant percentage of their original size.

- Drag the Proportionality slider.

Note In the default mode, diagrams larger than the canvas are reduced to fit to the canvas. Thus the proportional constant does not make sense.

Changing Your Focuses

When the weight of focal importance is positive, your focuses influence the view of the diagram. You can change your focuses dynamically by using mouse. Changing your focuses changes the view of the diagram immediately.

To change your focuses

- Select the nodes you want to put your focuses.

Animation

Automatic drawing possibly changes diagrams completely. Suddenly and drastically changes of diagrams often destroy your mental map of the diagram and make efficiency of works lower. D-ABDUCTOR provides display of changes with animation. The animation reduces the instantaneous visual change so that the changes preserve your mental map.

You can change configuration of animation: activeness, acceleration, speed, and the number of frames.

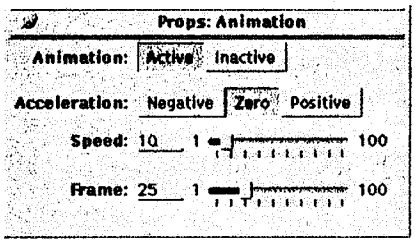
Opening the Animation Dialog Box

To change configuration of animation, you use the Animation dialog box.

To open the Animation dialog box

- From the Props menu, choose **Animation**.

This opens the Animation dialog box.



Animation dialog box

Making Animation Active and Inactive

To make animation active and inactive

- Choose one of the Animation buttons.

Choose	To do
Active	make animation active
Inactive	make animation inactive

When you choose Inactive, other buttons and sliders in the Animation dialog box are not available.

Changing Parameters of Animation

To change acceleration of animation

You can select one of three changing patterns of animation speed.

- Choose one of the Acceleration buttons.

Choose	To make animation speed
Negative	slower and slower
Zero	constant
Positive	faster and faster

To change speed of animation

By the Speed slider, you change the interval time between two frames of the animation.

- Drag the Speed slider

To change the number of frames of animation

By the Frame slider, you change the number of frames consisting the animation.

- Drag the Frame slider

Sending Messages

For more information about language Simple, see Chapter 5, "Language Simple."

You can send messages (statements) described in language Simple to the D-ABDUCTOR system. The language Simple can describe all commands and operations of D-ABDUCTOR.

Opening the Message Dialog Box

Through the Message dialog box, you can send messages to the D-ABDUCTOR system.

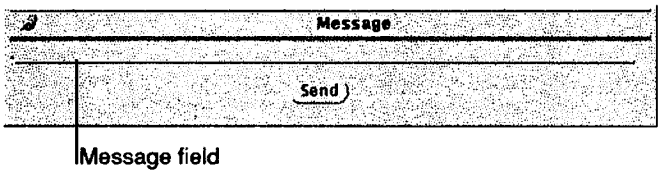
To open the Message dialog box

1 From the File menu, choose **Message**.

– or –

Pointing in the canvas, begin typing a message.

This opens the Message dialog box.



Message dialog box

To send a message to the D-ABDUCTOR system

1 Type a message (statement) in the message field.

2 Click the Send button.

This sends the message to the D-ABDUCTOR system.

Getting Information about Diagrams

You can get low level information about elements. The information is displayed on the terminal window that invokes the D-ABDUCTOR program.

Displaying Attributes

To display information about elements

1 Select elements whose information you want to display.

2 From the Props menu, choose **Attribute**.

This displays low level information about the selected elements.

Note Ordinary users do not need to know this low level information. They are mainly used for debugging.

Communications

You may hope to use D-ABDUCTOR for a group work. D-ABDUCTOR provides facilities to communicate with processes on other workstations. You can share diagrams with some other persons working with other workstations.

Note When you use communication facilities of D-ABDUCTOR, two or more processes of D-ABDUCTOR may not run on the same display.

Preparation for Communications

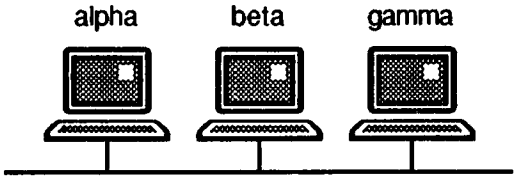
To use the communication facilities, you need setting of an environment variable before starting D-ABDUCTOR.

To prepare for communications

- Set the environment variable **ABDUCTOR_MEMBER** to the name list of the displays of workstations.

Assume you want to communicate among three workstations (displays): alpha, beta, and gamma. You set the variable **ABDUCTOR_MEMBER** to "alpla:0.0 beta:0.0 gamma:0.0".

% setenv ABDUCTOR_MEMBER "alpla:0.0 beta:0.0 gamma:0.0"



Workstations on a network

Note When you want to communicate among three workstations, all of you have to set the environment variable to the same value.

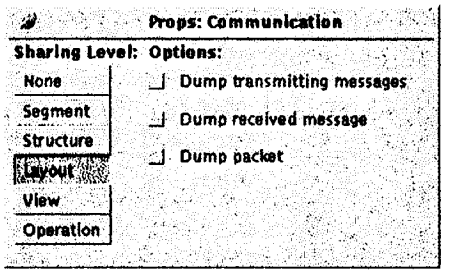
Opening the Communication Dialog Box

Through the Communication dialog box, you can select information sharing-level and can choose some options of communications.

To open the Communication dialog box

- From the Props menu, choose **Communication**.

This opens the Communication dialog box.



Communication dialog box

Selecting Sharing Level

You can select sharing-levels of information among the D-ABDUCTOR processes on different workstations.

To change sharing level

- Choose one in the Sharing Level buttons.

Choose	To share information about
None	none
Segment	nodes with text, images and attributes
Structure	compound graphs (nodes and edges)
Layout	triggers of automatic drawing
View	select information of elements
Operation	move and resize operations

Options to Dump Packets

You can monitor information exchanged among workstations.

Dump transmitting messages Select this option to monitor messages that the D-ABDUCTOR process you are using is transmitting.

Dump received message Select this option to monitor messages received by the D-ABDUCTOR process you are using.

Dump packet Select this option to monitor all packets through the D-ABDUCTOR process you are using.

Using System Files

For more information about language Simple, see Chapter 5, "Language Simple."

When D-ABDUCTOR is starting, it reads two system files including statements described in the language Simple. D-ABDUCTOR executes the statements in these files.

The Preference File

The preference file is the file ".abductor_pref" in your home directory. D-ABDUCTOR reads the preference file before creating the canvas. Thus, the preference file cannot include statements that cause drawing on the canvas. The preference file is used to change configuration of D-ABDUCTOR.

The Initial File

The initial file is the file ".abductor_init" in your home directory. D-ABDUCTOR reads the initial file after creating the canvas. The initial file can include any statements. The initial file is used to describe data, operations, and commands.

Customization

Environment variables and X resources are available to customize the configuration of D-ABDUCTOR.

Environment Variables

`ABDUCTOR_PATH`

This variable is used to define the path to the system files. If this variable is undefined, the path to your home directory is used.

`ABDUCTOR_~~INIT~~ PREF`

This variable is used to define the name of the preference file. If this variable is undefined, the file ".abductor_pref" is read as the preference file.

`ABDUCTOR_~~PREF~~ INIT`

This variable is used to define the name of the initial file. If this variable is undefined, the file ".abductor_init" is read as the initial file.

`ABDUCTOR_SREC_FILE`

If this variable is defined as a file name, all command you execute and all operations you perform are recorded in the file.

`ABDUCTOR_TEXTEDIT`

This variable is used to define the text editor. If the variable ABDUCTOR_TEXTEDIT is defined, choosing **Text** from the Edit menu invokes the command represented by the variable. The command of a text editor has to make a window. For example, when you want to use vi, you should define a shell script combining xterm and vi, and define the variable ABDUCTOR_TEXTEDIT as the shell script. The default command is "xnemacs".

`ABDUCTOR_TEXTEDGE`

This variable is used to specify the option tag for geometric parameters of the command of the text editor. If the position of the text editor can be specified by using command line options, and you define the variable ABDUCTOR_TEXTEDGE as the option tag, the text editor is opened at the position of the node.

ABDUCTOR_MEMBER

This variable is used to define the display names for D-ABDUCTOR to communicate with other processes on them.

X resources

abd.asciiFont

This resource is used to define ASCII fonts of text in nodes.

abd.kanjiFont

This resource is used to define kanji fonts of text in nodes.

Bugs You are recommended to use fonts that size is smaller than or equal to 16 pixels. If you use larger fonts, text may be overlapped because D-ABDUCTOR does not use the font height to draw text.

4. Summary of Operations

This chapter summarizes mouse operations and menus.

Summary of Mouse Operations

Mouse operations are classified by contexts where a mouse button is pressed.

On Menus

On menus, the select button and the menu button are available.

The Select Button

Clicking Clicking chooses default item of the menu.

Dragging Pressing shows the default item on the menu name. Releasing there chooses the default item. Releasing outside the menu chooses no item. You can cancel to choosing the default item to release the button outside the menus.

The Menu Button

Clicking Clicking opens the menu.

Dragging Pressing pulls down the menu. Releasing chooses the pointed item.

Dragging outside the menu closes the menu and releasing the menu button there chooses no item. You can cancel to choosing menu item to release the button outside the menus.

On the Canvas

On the canvas, only the select button available. The operations described in this section are begun at an empty space in the canvas. The operations on elements are described below.

The Select Button

Clicking When some elements are selected, clicking unselects these elements. When no elements are selected, clicking re-draws the diagram on the canvas

Dragging Dragging shows a rubber rectangle. Handles appear on elements completely enclosed by the rectangle. Releasing selects the enclosed elements.

On Nodes

On the nodes, three buttons are available. Operations of the select button on link handles and resize handles have special semantics, and are described below.

The Select Button

Clicking Clicking selects the node and unselects all other elements.

Dragging Dragging moves the selected nodes and nodes that are included by the selected nodes recursively. When no elements are selected, dragging moves the pointed node and nodes that are included by the pointed node.

When you release the select button if the pointer is inside a node, the moved nodes become members of the node.

The Adjust Button

Clicking When the node is unselected, clicking selects the node. When the node is selected, clicking unselects the node.

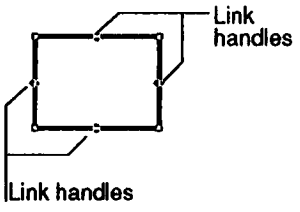
The Menu Button

Clicking Clicking opens the menu.

Dragging Pressing pull down the menu. Releasing chooses the pointed item.

Dragging outside the menu closes the menu and releasing there chooses no item. You can cancel to choosing menu item to release the button outside the menus.

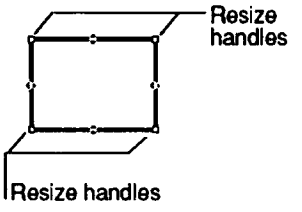
On Link Handles of Nodes



The Select Button

Dragging Dragging displays a rubber line segment that shows the new link, and show handles on a pointed node. You release the select button when handles appear on a head node.

On Resize Handles of Nodes



The Select Button

Dragging Dragging displays a rubber rectangle that shows new size of the node. You release the select button when size of the rubber rectangle is what you want.

On Links

The Select Button

Clicking Clicking selects the link and unselects all other elements.

The Adjust Button

Clicking When the link is unselected, clicking selects the link. When the link is selected, clicking unselects the link.

Summary of Menus

The File Menu

Load/Save...

Choosing **Load/Save** opens the Load/Save dialog box. In the Load/Save dialog box, you save data of diagrams to files and load the files including diagram data.

Print

The **Print** item is not available.

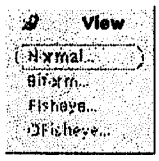
Message...

Choosing **Message** opens the Message dialog box. In the Message dialog box, you send some messages in language Simple to the D-ABDUCTOR system.

Quit

Choosing **Quit** closes the main window and all dialog boxes, and stops D-ABDUCTOR.

The View Menu



The View menu has four items but all of them are not available now.

Normal...

The **Normal** item is not available.

Biform...

The **Biform** item is not available.

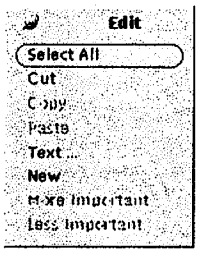
Fisheye...

The **Fisheye** item is not available.

Ofisheye...

The **Ofisheye** item is not available.

The Edit Menu



Select All

Choosing **Select All** selects all elements on the canvas. Handles appear on all elements.

Cut

Choosing **Cut** deletes all the selected elements.

Copy

The **Copy** item is not available.

Paste

The **Paste** item is not available.

Text...

Choosing **Text** opens a window of a text editor to enter or edit text of the selected node. After entering or editing text, you quit the text editor. Quitting the text editor updates the text of the selected node.

New

Choosing **New** cleans the canvas and creates a new diagram.

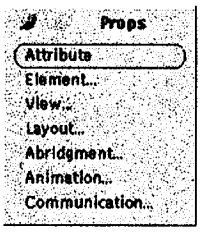
More Important

The **More Important** item is not available.

Less Important

The **Less Important** item is not available.

The Props Menu



Attribute

Choosing **Attribute** displays information about the selected elements on the window where the D-ABDUCTOR program is invoked.

Element...

Choosing **Element** from the Props menu opens the Element dialog box. In the Element dialog box you changes shape styles, line styles, line width, and colors of selected elements and of default.

View...

Choosing **View** opens the View dialog box. In the View dialog box, you select your preference views.

Layout...

Choosing **Layout** opens the Layout dialog box. In the Layout dialog box, you select the triggers for the automatic layout facility, final process of automatic layout facility, and direction of the layout. You can also choose two options to show dummy elements.

Abridgment...

Choosing **Abridgment** opens the Abridgment dialog box. In the Abridgment dialog box, you make abridgment active and inactive. When abridgment is active, you can also change parameters of the importance function and type and parameters of weighted drawing.

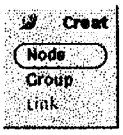
Animation...

Choosing **Animation** opens the Animation dialog box. In the Animation dialog box, you make animation active and inactive. When animation is active, you can also change acceleration, speed, and the number of frames of animation.

Communication...

Choosing **Communication** from the Props menu opens the Communication dialog box. In the Communication dialog box, you change sharing levels of the communication among the D-ABDUCTOR processes. You can also a few options to show messages used by communication.

The Create Menu



Node

Choosing **Node** create a new node.

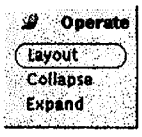
Group

Choosing **Group** create a new group node that includes all the selected nodes.

Link

The **Link** item is not available.

The Operate Menu



Layout

Choosing **Layout** lays out the diagram on the canvas.

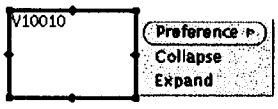
Collapse

Choosing **Collapse** collapses the selected nodes. If a selected node is not a group node, the node is not changed.

Expand

Choosing **Expand** expands the selected nodes. If a selected node has never been collapsed, the node is not changed.

The Node Menu



There is no button for the Node menu. To open the Node menu, you press the menu button on a node.

Preference

The Preference item is a submenu that has two items: image and text. Choosing one of them changes preference of the view of the node on which the Node menu is opened.

Collapse

Choosing **Collapse** collapses the node on which the Node menu is opened.

Expand

Choosing **Expand** expands the node on which the Node menu is opened.

5. Language Simple

Overview

The language Simple is designed to describe compound graphs, commands, and operations. D-ABDUCTOR uses the language Simple to save diagrams, to communicate with D-ABDUCTOR processes on other workstations, to communicate with other tools. You and other programs can also use the language Simple to describe diagrams. Furthermore, the language Simple has capability to record your works with D-ABDUCTOR.

Diagrams are described as compound graphs in the language Simple. Elements of compound graphs are called differently from diagrams on the canvas of D-ABDUCTOR. In the language Simple, nodes are called vertices, links are called adjacency edges, and inclusion relationships of groups are called inclusion edges.

Statements

A statement consists of one line, which ends with end of line or CR. A statement begins with a percent symbol (`%`). A line begins with another character is ignored. Thus you can use it for comments.

There are four kinds of statements.

1. description of compound graphs
2. description of operations
3. description of commands
4. description of controls

Evaluation

D-ABDUCTOR can be regarded as an interpreter of the language Simple. You have three ways to evaluate statements in the language Simple.

1. Loading a file of statements in the language Simple by using the Load/Save dialog box.
2. Sending a message (that is, a statement) in the language Simple by using the Message dialog box.
3. Writing statements in the language Simple to a property of the D-ABDUCTOR window.

Description of Compound Graphs

This section explains the statements to describe compound graphs. D-ABDUCTOR uses these statements to save diagrams. You can also write these statements by using text editors to describe diagrams.

Reference

A compound graph has vertices, adjacency edges, and inclusion edges as its elements. Each element is refereed in some way.

Identifiers are available to refer elements. Names are also available to refer vertices. Names are offered for humans to be easy to read and to write statements. Most of you are not interested in the identifiers managed by the system. However the identifiers are important for communication among two or more processes of D-ABDUCTOR to share the same compound graphs.

There are three styles of references.

Existing Reference assumes that the refereed element has existed. If the element does not exist, the reference causes an error.

Creating Reference assumes that the refereed element has never existed. If the element has existed, the reference causes an error.

Conditional Reference does not assume existing of the refereed element.

Existing Reference

Existing references use sharp symbols (#).

Existing reference with identifier

 # identifier

A reference in this form refers the element with the identifier. If the element has never exists, this reference causes an error. This is the only form to refer an adjacency edge.

Creating Reference

Creating references use exclamation symbols (!).

Creating reference with name and identifier

name ! identifier

A reference in this form newly creates a vertex with the name and gives the identifier. If a vertex with the identifier has existed, this reference causes an error.

Creating reference with identifier

! identifier

A reference in this form newly creates a vertex without name and gives the identifier. If a vertex with the identifier has existed, this reference causes an error.

Creating reference without identifier

!

A reference in this form newly creates a vertex without name and gives an arbitrary identifier.

Conditional Reference

Conditional references use at symbols (@).

Conditional reference with name

name

A reference in this form refers the vertex with the name. If the vertex has never existed, a vertex with the name is created and given an arbitrary identifier.

Conditional reference with name and identifier

name @ *identifier*

A reference in this form refers the vertex with the identifier. If the vertex has never existed, a vertex with the name is created and given the identifier.

Conditional reference with identifier

@ *identifier*

A reference in this form refers the vertex with the identifier. If the vertex has never existed, a vertex without name is created and given the identifier.

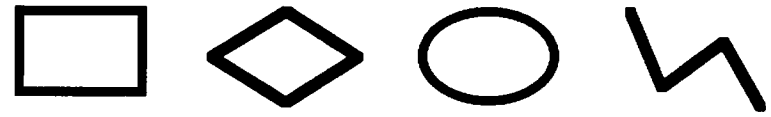
Attributes

A description of an attribute value consists of an attribute tag followed by a colon and a value. One or more descriptions of attribute values must be separated by white spaces and put into brackets.

Shape type

`stype` : *shape_type*

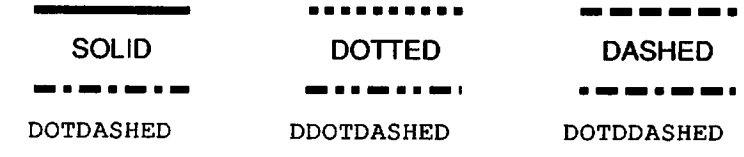
Shape types are the same as shape styles. Available shape types are RECTANGLE, RHOMBUS, ELLIPSE, and POLYLINE.



Line type

`ltype` : *line_type*

Line types are the same as line styles. Available line types are SOLID, DOTTED, DASHED, DOTDASHED, DDOTDASHED, DOTDDASHED.



Color

`color : color`

Two sets of colors are prepared and available set depends on installation options. One is large set, which includes 81 colors, and the other is small set, which includes nine colors.

Large Set

black	blue	blue2
blue3	brown	brown2
brown3	cadetblue	cadetblue2
cadetblue3	cyan	cyan2
cyan3	gold	gold2
gold3	gray13	gray25
gray38	gray50	gray63
gray75	gray88	green
green2	green3	khaki
khaki2	khaki3	magenta
magenta2	magenta3	maroon
maroon2	maroon3	mediumpurple
mediumpurple2	mediumpurple3	olivedrab
olivedrab2	olivedrab3	orange
orange2	orange3	orchid
orchid2	orchid3	pink
pink2	pink3	plum
plum2	plum3	purple
purple2	purple3	red
red2	red3	royalblue
royalblue2	royalblue3	salmon
salmon2	salmon3	seagreen
seagreen2	seagreen3	skyblue
skyblue2	skyblue3	tan
tan2	tan3	tomato
tomato2	tomato3	white
yellow	yellow2	yellow3

Small Set

black	olivedrab	orange2
purple3	royalblue	salmon
skyblue3	white	yellow3

Line width

`width` : ***integer***

Line width is specified by an integer.

Position

`xpos` : ***integer***
`ypos` : ***integer***

Position is specified by x coordinate and y coordinate separately. Each of them is specified by an integer.

Size

`xsize` : ***integer***
`ysize` : ***integer***

Size is specified by size in x direction (width) and size in y direction (height) separately. Each of them is specified by an integer.

Ordering hint

`order` : ***integer***

Ordering hint is specified by an integer. It is used by the vertex ordering step of the automatic drawing algorithm. When there are two or more vertices whose barycenters are the same on a band, vertices with smaller values of the ordering hint come to left-hand side.

For more information about the drawing algorithm, refer the paper of "Visualization of Structural Information: Automatic Drawing of Compound Digraphs" in IEEE T.SMC, Vol. 21, No. 4, 1991.

Semantic importance

`semiv` : ***float***

Semantic importance is specified by a float number. It is used by the abridgment facility.

For more information about the abridgment facility, see chapter 3, "Advanced Use of D-ABDUCTOR."

Ptext

`ptext` : ***string***

Text used as labels of vertices is called ptext. Ptext is specified by a string. The string can include not only ASCII code but also EUC kanji code and some escape sequences.

Image file

`xpmfn` : ***file_name***

Image files are specified by strings of the file names. The file names should include full path form the root directory.

Default Attributes

Description of attribute values of vertices

`%V [` ***attributes*** `]`

A statement in this form is used to set default attribute values of vertices.

Description of attribute values of adjacency edges

`%A` [*attributes*]

A statement in this form is used to set default attribute values of adjacency edges.

Vertex

Statements begin with `%V` are used to describe vertices.

Simple description of a vertex

`%V` *reference*

A statement in this form is used to create a new vertex. Thus the reference should be creating reference or conditional reference. Existing reference in this form may cause no error, but it has no sense.

Description of a vertex with attributes

`%V` *reference* [*attributes*]

A statement in this form is used to set attribute values of a vertex. All reference styles are available and make sense.

Description of two or more vertices

You can enumerate two or more references may be followed by attribute descriptions. This means that two or more vertices can be described in a statement.

Adjacency Edge

Statements begin with `%A` are used to describe adjacency edges.

Simple creation of an adjacency edge

`%A` *tail* : *head*

A statement in this form is used to create a new adjacency edge. The adjacency edge is given an arbitrary identifier. The tail and head must be references of vertices.

Creation of an adjacency edge

`%A` *tail* : *head* [> *identifier*] [[*attributes*]]

The statement in this form is used to create a new adjacency edge, to give it the identifier, and to set attribute values of it. The symbol > with the identifier can be omitted. If it is omitted, the edge is given an arbitrary identifier. The attribute descriptions can also be omitted. If it is omitted, default attribute values are set. The tail and head must be references of vertices.

Creation of two or more adjacency edges

You can enumerate two or more heads may be followed by attribute descriptions in the above two forms. This means that two or more adjacency edges whose tails are the same can be described in a statement.

Modification of attributes of an adjacency edge

`%A` *reference* [[*attributes*]]

A statement in this form is used to set attribute values of an adjacency edge. Reference style in this form must be existing reference. The attribute

descriptions can be omitted. But the statement without attribute descriptions has no sense.

Modification of attributes of two or more adjacency edges

You can enumerate two or more references followed by attribute descriptions in the above form. This means that attribute values of two or more adjacency edges can be described in a statement.

Inclusion Edge

Statements begin with `%I` are used to describe inclusion edges.

Simple creation of an inclusion edge

`%I` ***tail*** `:` ***head***

A statement in this form is used to create a new inclusion edge. If there is an inclusion edge whose head is the same as the new inclusion edge, the older one is removed. For a vertex there can be only one inclusion edge whose head is the vertex, because the subgraph with only all inclusion edges must be a tree.

Creation of two or more inclusion edges

You can enumerate two or more heads in the above form. This means that two or more inclusion edges whose tails are the same can be described in a statement.

Description of Operations

The language Simple has facilities to describe all operations for compound graphs on the D-ABDUCTOR system.

General Form

Statements begin with `%O` or `%o` are used to describe operations. The operations begin with `%O` are called "global operations," and the operations begin with `%o` are called "local operations." The D-ABDUCTOR system sends the global operations to all other processes of D-ABDUCTOR.

Operation descriptions have the following general forms.

`%O` ***op_name*** [(***param_list***)] [***elem_list***]
`%o` ***op_name*** [(***param_list***)] [***elem_list***]

Each operation has a unique name. The name may be followed by a parameter list put in parenthesis and an elements list. Both or either of the parameter list and the elements list may be omitted.

Operations

The select operation

`%O SELECT` [***elem_list***]

The select operation makes specified elements (or all elements, if the elements list is omitted) selected.

The unselect operation

`%O UNSELECT` [***elem_list***]

The unselect operation makes specified elements (or all elements, if the element list is omitted) unselected.

The move operation

`%O MOVE (x, y) [ elem_list ]`

The move operation requires two parameters, which specify a vector. This operation moves specified vertices (or selected vertices, if the element list is omitted) according to the vector.

The moveabs operation

`%O MOVEABS (x, y) [ elem_list ]`

The moveabs operation requires two parameters, which specify a position. This operation moves specified vertices (or selected vertices, if the element list is omitted) to the position.

The resize operation

`%O RESIZE (x, y) [ elem_list ]`

The resize operation requires two parameters, which specify size in x-coordinate and size in y-coordinate. This operation changes size of specified vertices (or selected vertices, if the element list is omitted) according to the parameters.

The cut operation

`%O CUT [ elem_list ]`

The cut operation cuts specified elements (or selected elements, if the element list is omitted).

The collapse operation

`%O COLLAPSE [ elem_list ]`

The collapse operation collapses specified vertices (or selected vertices, if the element list is omitted). All vertices included by the collapsed vertices are disappeared.

The expand operation

`%O EXPAND [ elem_list ]`

The expand operation is the reverse operation of the collapse operation. This operation expand specified vertices (or selected vertices, if the element list is omitted). Vertices included by the collapsed vertices are appeared.

The layout operation

`%O LAYOUT [ forcibly ]`

The layout operation lays out the diagram on the canvas. You can give an optional Boolean parameter *forcibly*. When it is TRUE, this operation lays out the diagram forcibly, even if the current diagram does not need re-layout.

Description of Commands

The language Simple has facilities to describe all commands of the D-ABDUCTOR system.

General Form

Statements begin with `%X` or `%x` are used to describe commands. The commands begin with `%X` are called global commands, and the commands begin with `%x` are called local operations. The D-ABDUCTOR system sends the global commands to all other processes of D-ABDUCTOR.

Command descriptions have the following general forms.

`%X` ***cmd_name*** [***param_list***]
`%x` ***cmd_name*** [***param_list***]

Each command has a unique name. The name can be followed by a parameter list

Commands

The new command

`%X NEW`

The new command creates a new compound graph. If there is an old one, it is abandoned.

The redraw command

`%X REDRAW`

The redraw command draws current compound graph again.

The set command

For more information about system variables, see the section of "System Variables" in this chapter.

`%X SET` ***variable expression***

The set command requires two parameters: a variable and an expression. This command sets the variable to the value of the expression.

The push command

`%X PUSH` ***variable***

The push command requires a parameter of variable name. This command pushes the value of the specified variable to the stack.

The pop command

`%X POP` ***variable***

The pop command requires a parameter of variable name. This command pops out the top value of the stack to set the specified variable to it.

The load command

`%X LOAD` ***file_name***

The load command requires a file name as a parameter. This command reads the file and interprets text in the file as in the language Simple.

The save command

`%X SAVE` ***file_name***

The save command requires a file name as a parameter. This command saves the current compound graph into the file. The compound graph data is described in the language Simple.

The exec command

`%X EXEC` ***string***

The exec command creates a subprocess and executes UNIX commands described the string. D-ABDUCTOR does not wait finish of the subprocess.

The connect command

`%X CONNECT` ***display_name***

The connect command is used to begin communication with D-ABDUCTOR processes on specified displays. When D-ABDUCTOR is starting, it sends this command with the name of the display on which the process is to other processes.

The quit command

`%X QUIT [` ***confirm*** `]`

The quit command closes the main window and all dialog boxes, and terminates the D-ABDUCTOR program. You can give an optional Boolean parameter *confirm*. When it is TRUE, the command confirms you that you wish to quit D-ABDUCTOR.

Description of Controls

Control descriptions control the action of D-ABDUCTOR reading text in the language Simple. Thus control descriptions are available only in text described in the language Simple. They may not work out of text.

General Form

Statements begin with `%%` are used to describe control commands. Control descriptions have the following general form.

`%%` ***ctl_name*** `[` ***param_list*** `]`

Each control command has unique name. The name can be followed by a parameter list.

Controls

The end command

`%% END`

The end command is used to stop to read statements in the text including the end command. All lines following the end command are ignored.

The include command

`%% INCLUDE` ***file_name***

The include command is used to insert lines of another file into there. This command is similar to the macro command "#include" of the language C. It is useful for two or more text files to share the same statements.

The trace command

`%% TRACE [` ***Boolean*** `]`

The trace command changes the mode. If the command has TRUE as a parameter, the mode will become the trace mode. If the command has FALSE, the mode will become the normal mode. If the parameter is omitted, the mode will become the other mode. In the trace mode, D-ABDUCTOR notices you trace information of every statement.

The debug command

%% DEBUG [*Boolean*]

The debug command changes the mode. If the command has TRUE as a parameter, the mode will become the debug mode. If the command has FALSE, the mode will become the normal mode. If the parameter is omitted, the mode will become the other mode. In the trace mode, D-ABDUCTOR notices you debug information of some statements.

System Variables

D-ABDUCTOR has many system variables. You can set these variables to some values by the set command. In this section, all variables are explained.

Variables are listed in alphabetical order. For each variable, the name, sharing level, default value, and semantics are described. Sharing level of system variables means the variables are shared when the sharing level of D-ABDUCTOR is it or more. The variables with sharing level of PRIVATE (P) are not shared. The variables with sharing level of COMMON (C) are always shared.

Variables for the abridgment facility.

Variable Name (Sharing Level)	Default	Semantics
abridg_active (V)	FALSE	If TRUE, the abridgment facility is active.
abridg_const_prop (V)	200	The proportional constant of the proportional type weighted drawing.
abridg_h_threshold (V)	50	The higher threshold of the hybrid type weighted drawing.
abridg_l_threshold (V)	0	The lower threshold of the hybrid type weighted drawing.
abridg_wdraw_type (V)	0	The type of weighted drawing. The hybrid type is 0, the proportional type is 1.
abridg_weight_focif (V)	50	The weight coefficients of the focal importance.
abridg_weight_semif (V)	50	The weight coefficients of the semantic importance.
abridg_weight_strif (V)	50	The weight coefficients of the structural importance.

Variable for the animation facility.

Variable Name (Sharing Level)	Default	Semantics
animation_accele (P)	1	Acceleration of animation. Negative (slower and slower) is 0, zero (constant speed) is 1, and positive(faster and faster) is 2.
animation_active (P)	TRUE	If TRUE, the animation facility is active.
animation_frames (P)	25	The number of frames.
animation_options (P)	0	Unless 0, loci of animation is preserved.
animation_speed (P)	10	The speed of animation.

Variable for the vertex ordering step of the automatic drawing facility.

Variable Name (Sharing Level)	Default	Semantics
bc_global_loop (L)	1	The number of iterations in each vertex.
bc_local_loop (L)	1	The number of iterations in each level.
bc_reverse_mode (L)	TRUE	If TRUE, the order of vertices with the same barycenter in the end level is reversed in each iteration step.

Variables for the communication facility.

Variable Name (Sharing Level)	Default	Semantics
comm_level (C)	3	The sharing level. Only this variable is always shared in spite of the sharing level (that is, this variable). 0: COMMON, 1: SEGMENT, 2: STRUCTURE, 3: LAYOUT, 4: VIEW, 5: OPERATE, 255: PRIVATE.
comm_options (P)	0	Summation of the followings. a: To dump transmitted messages, 2: To dump received messages, 4: To dump all packets .

Variables for dummy reverse (that is, the direction is different from the level direction) edges.

Variable Name (Sharing Level)	Default	Semantics
dmyrev_edge_color (P)	olivedrab	The default color of dummy reversed edges.
dmyrev_edge_ltype (P)	DOTTED	The default line type of dummy reversed edges.
dmyrev_edge_stype (P)	POLYLINE	The default shape type of dummy reversed edges.
dmyrev_edge_width (P)	2	The default width of dummy reversed edges.

Variables for direct manipulations.

Variable Name (Sharing Level)	Default	Semantics
dm_minmove (P)	3	The minimum number of pixels for dragging to move nodes.

Variables for dummy edges.

Variable Name (Sharing Level)	Default	Semantics
dummy_edge_color (P)	orange2	The default color of dummy edges.
dummy_edge_ltype (P)	DOTTED	The default line type of dummy edges.
dummy_edge_stype (P)	POLYLINE	The default shape type of dummy edges.
dummy_edge_width (P)	2	The default width of dummy edges.

Variables for dummy vertices.

Variable Name (Sharing Level)	Default	Semantics
dummy_vert_color (P)	purple3	The default color of dummy vertices.
dummy_vert_ltype (P)	DOTTED	The default line type of dummy vertices.
dummy_vert_stype (P)	RECTANGLE	The default shape type of dummy vertices.
dummy_vert_width (P)	2	The default width of dummy vertices.

Variables for group vertices (that is, group nodes).

Variable Name (Sharing Level)	Default	Semantics
group_vert_color (p)	skyblue3	The default color of group vertices.
group_vert_ltype (p)	SOLID	The default line type of group vertices.
group_vert_stype (p)	RECTANGLE	The default shape type of group vertices.
group_vert_width (p)	3	The default width of group vertices.

Variables for the automatic drawing facility.

Variable Name (Sharing Level)	Default	Semantics
layout_direction (L)	SOUTH	The layout direction which most adjacency edges are oriented to.
layout_options (L)	0	The total value representing options for the automatic drawing facility. The option to draw dummy vertices is 1, and the option to draw dummy edges is 2.
layout_proc (L)	4	The final subprocess. Hierarchization is 1, normalization is 2, vertex ordering is 3, and position deciding is 4.
layout_triggers (L)	63	The triggers of the automatic drawing facility. Total triggers are represented by the summation of the following values. 1: changing group, 2: changing link, 4: creating node, 8: creating link, 16: removing node, 32: removing link.

Variables for the level assignment process of the automatic drawing facility.

Variable Name (Sharing Level)	Default	Semantics
la_reverse_edge (L)	FALSE	If TRUE, when the compound graph has cycles of edges, some edges are reversed.

Variables for the size of handles.

Variable Name (Sharing Level)	Default	Semantics
mark_dsize (P)	4	Size of link handles. The unit is pixels.
mark_rsize (P)	5	Size of resize handles. The unit is pixels.

Variables for normal edges.

Variable Name (Sharing Level)	Default	Semantics
normal_edge_color (P)	salmon	The default color of normal edges.
normal_edge_ltype (P)	SOLID	The default line type of normal edges.
normal_edge_stype (P)	POLYLINE	The default shape type of normal edges.
normal_edge_width (P)	3	The default width of normal edges.

Variables for normal vertices.

Variable Name (Sharing Level)	Default	Semantics
normal_vert_color (P)	olivedrab	The default color of normal vertices.
normal_vert_ltype (P)	SOLID	The default line type of normal vertices.
normal_vert_stype (P)	RECTANGLE	The default shape type of normal vertices.
normal_vert_width (P)	3	The default width of normal vertices.

Variable for normal reverse (that is, the direction is different from the level direction) edges.

Variable Name (Sharing Level)	Default	Semantics
nrmrev_edge_color (P)	purple3	The default color of normal reverse edges.
nrmrev_edge_ltype (P)	SOLID	The default line type of normal reverse edges.
nrmrev_edge_stype (P)	POLYLINE	The default shape type of normal reverse edges.
nrmrev_edge_width (P)	3	The default width of normal reverse edges.

Variables for the position deciding process of the automatic drawing facility.

Variable Name (Sharing Level)	Default	Semantics
pr_dummy_prio (L)	TRUE	If TRUE, dummy nodes are given high priority.
pr_final_level (L)	-2	The level which the final step of the deciding process is begun with. Two negative numbers have special mean. -1: local maximal level from bottom; -2: maximal level.
pr_global_loop (L)	2	The number of iterations in each vertex.
pr_local_loop (L)	2	The number of iterations in each level.

Variables for the root vertex (that is, a dummy vertex that includes all other vertices).

Variable Name (Sharing Level)	Default	Semantics
root_vert_color (P)	white	The default color of root vertex.
root_vert_ltype (P)	DOTTED	The default line type of root vertex.
root_vert_stype (P)	RECTANGLE	The default shape type of root vertex.
root_vert_width (P)	0	The default width of root vertex.

Variables for view control on the canvas of D-ABDUCTOR.

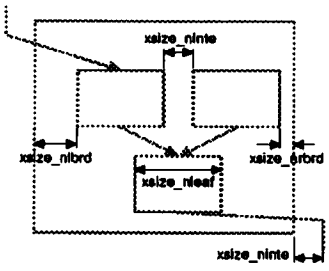
Variable Name (Sharing Level)	Default	Semantics
screen_gravity (V)	CENTER	Gravity on the canvas.
screen_scale_large (V)	0	The scale factor of when the diagram is larger than the canvas. If the value is 0, the diagram is similarly reduced to fit the canvas. If the value is -1, the diagram is reduced to just fit the canvas.
screen_scale_small (V)	100	The scale factor of when the diagram is smaller than the canvas If the value is 0, the diagram is similarly magnified to fit the canvas. If the value is -1, the diagram is magnified to just fit the canvas.
screen_vtype_large (V)	CARTESIAN	The view type of when the diagram is larger than the canvas.
screen_vtype_small (V)	CARTESIAN	The view type of when the diagram is smaller than the canvas.

Variables for text editing.

Variable Name (Sharing Level)	Default	Semantics
textedit_options (P)	0	A summation of the followings. 1: for resizing the node when whose text is edited; 2: for opening a window of a text editor, when a node or a group is created.

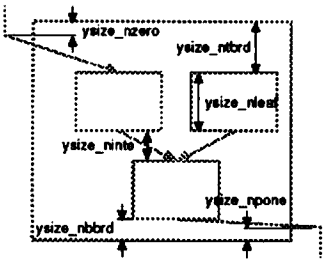
Variables for preferable view.

Variable Name (Sharing Level)	Default	Semantics
view_shprio (P)	1	Preferable view of default nodes. 0: text; 1: images.



Variables for layout of diagrams (in x-direction).

Variable Name (Sharing Level)	Default	Semantics
xsize_ninte (P)	20	Normal interval clearance between two nodes.
xsize_nlbrd (P)	15	Normal left hand side clearance in a group node.
xsize_nleaf (P)	60	Normal width of leaf nodes (non-group nodes).
xsize_nrbrd (P)	15	Normal right hand side clearance in a group node.



Variables for layout of diagrams (in y-direction).

Variable Name (Sharing Level)	Default	Semantics
ysize_nbbrd (P)	20	Normal bottom clearance in a group node.
ysize_ninte (P)	20	Normal interval clearance between two nodes.
ysize_nleaf (P)	40	Normal height of leaf nodes (non-group nodes).
ysize_npone (P)	10	Normal bottom dummy clearance in a group node.
ysize_ntbrd (P)	40	Normal top clearance in a group node.
ysize_nzero (P)	10	Normal top dummy clearance in a group node.

Summary of Syntax

Token

Boolean ::=
```
TRUE|true|FALSE|false
```

integer ::=
```
[0-9]+
```

float ::=
```
([0-9]+(\.[0-9]*)?)|([0-9]*\.[0-9]+)
```

name ::=
```
[-+#$&*,./;<=?^_|~A-Za-z0-9\E]+
```

file_name ::=
    `[-+._/A-Za-z0-9]+`

display_name ::=
    `[-+._/A-Za-z0-9]+`

string ::=
    `"([^"\]|\\"|\\\\)*"`

Common

param ::=
    boolean
  | *integer*
  | *float*
  | *name*
  | *file_name*
  | *display_name*
  | *string*

param_list ::=
    param_list param
  | *param*

identifier ::=
    integer

Reference

reference ::=
    existing_reference
  | *creating_reference*
  | *conditional_reference*

existing_reference ::=
    # *identifier*

creating_reference ::=
    name ! *identifier*
  | ! *identifier*
  | !

conditional_refernece ::=
    name
  | *name* @ *identifier*
  | @ *identifier*

Attributes

attr_list_bk ::=
    [*attr_list*]

attr_list ::=
    attr_list attribute
  | *attribute*

attribute ::=
    stype : *shape_type*
  | ltype : *line_type*
  | color : *color*
  | width : *integer*
  | xpos : *integer*
  | ypos : *integer*
  | xsize : *integer*
  | ysize : *integer*
  | order : *integer*
  | semiv : *float*
  | ptext : *string*
  | xpmfn : *file_name*

Vertex

attr_vert ::=
    reference
  | *reference attr_list_bk*

attr_vert_list ::=
    attr_vert_list attr_vert
  | *attr_vert*

desc_vert ::=
    %V *attr_vert_list*
  | %V *attr_list_bk*

Adjacency Edge

plain_tail ::=
    reference

assign ::=
    > *identifier*

attr_head ::=
    reference
  | *reference assign*
  | *reference attr_list_bk*
  | *reference assign attr_list_bk*

attr_head_list ::=
    attr_head_list attr_head
  | *attr_head*

attr_adja ::=
    existing_reference
  | *existing_reference attr_list_bk*

attr_adja_list ::=
    attr_adja_list attr_adja
  | *attr_adja*

desc_adja ::=
    %A *plain_tail* : *attr_head_list*
  | %A *attr_adja*
  | %A *attr_list_bk*

Inclusion Edge

plain_head ::=
    reference

plain_head_list ::=
    plain_head_list plain_head
  | *plain_head*

desc_incl ::=
    %I *plain_tail* : *plain_head_list*

Operation Description

op_name ::=
    SELECT
  | UNSELECT
  | CUT
  | COLLAPSE
  | EXPAND
  | LAYOUT
  | MOVE
  | RESIZE

operation ::=
    op_name param_list_bk elem_list
  | *op_name param_list_bk*
  | *op_name elem_list*
  | *op_name*

desc_operation ::=
    %O *operation*
  | %o *operation*

Command Description

cmd_name ::=
    NEW
  | REDRAW
  | UNDO
  | SET
  | PUSH
  | POP
  | LOAD
  | SAVE
  | PRINT

command ::=
    cmd_name param_list
  | *cmd_name*

desc_command ::=
    %X *command*
  | %x *command*

Control Description

ctrl_name ::=
    END

| CONNECT
| INCLUDE
| TRACE

control ::=
ctrl_name param_list
| *ctrl_name*

desc_control ::=
%% *control*

Statement

statement ::=
desc_vert
| *desc_adje*
| *desc_incl*
| *desc_operation*
| *desc_command*
| *desc_control*

6. Message Transmitter

D-ABDUCTOR does not wait command from standard input since it is an event driven system. However, it is convenient that D-ABDUCTOR can read commands from the standard input. It makes easy to use D-ABDUCTOR as a tool to draw diagrams by combining with other application systems.

Message Transmitter is a support program of D-ABDUCTOR. It reads character strings from the standard input and sends them to a D-ABDUCTOR process. The D-ABDUCTOR process executes received messages as statements or control commands. All thing to do for application systems to communicate with D-ABDUCTOR is writing out messages and control commands to the standard output.

Command

Synopsis

`abd_tx` [***options*** ...]

Description

Message Transmitter abd_tx reads character strings from the standard input and sends them to a D-ABDUCTOR process on the same display.

Options

`-display` ***display***

Display on which the D-ABDUCTOR process should receive the messages sent by this command. The way to specify the display name is as same as the other X11 clients.

Local Command

Message Transmitter regards strings of characters begin with double sharp symbols (##) as commands for itself. Thus Message Transmitter does not send these character strings to D-ABDUCTOR.

`## quit`	Terminates Message Transmitter. Message Transmitter also terminates when it reads an end of file.
`## echo`	Enables or disables echo backs and prints new status. If echo back is enabled, "TRUE" is printed. Otherwise "FALSE" is printed.
`## ?echo`	Prints the status of echo back.

`## debug` Changes the mode for debugging and prints new status. When it is in the debugging mode, "TRUE" is printed. Otherwise "FALSE" is printed.

`## ?debug` Prints the current mode for debugging.

`## ?receiver`

Prints the window identifier of the D-ABDUCTOR process receiving messages. When there is no process of D-ABDUCTOR receiving character strings, Message Transmitter prints "None."

Control Command

Message Transmitter regards strings of characters begin with sharp and dollar symbols (#$) as commands to control D-ABDUCTOR. The Message Transmitter sends these control commands to a D-ABDUCTOR process.

`#$ Open` Opens the main window of D-ABDUCTOR. When the window has opened, the command does not work.

`#$ Close` Closes the main window of D-ABDUCTOR. When the window has closed, the command does not work.

`#$ Map` Maps the main window of D-ABDUCTOR. When the window has mapped, the command does not work.

`#$ Unmap` Unmaps the main window of D-ABDUCTOR. When the window has never mapped, the command does not work.

`#$ Shutdown`

Terminates D-ABDUCTOR.

Functions of Message Transmitter

Message Transmitter reads lines from the standard input and then checks the first two characters of each line. If the two characters are two sharp symbols (##), the message transmitter regards the line as a local command. If the two characters are a sharp symbol and a dollar symbol (#$), Message Transmitter regards the line as a control command. Otherwise Message Transmitter regards the lines as an ordinal command in the language Simple.

Local Command

Local commands are only effective for Message Transmitter itself. Thus Message Transmitter executes these commands locally, and sends them nowhere.

Control Command

Message Transmitter writes control commands on a window property named "_GRIPS_DA_CONTROL" of the root window. D-ABDUCTOR processes are watching the window property named "_GRIPS_DA_CONTROL" of the root window. When some processes

change the property, the D-ABDUCTOR processes read the string and regard it as a control command.

Ordinal Command

Message Transmitter makes an ordinal command a packet form and writes it on a window property named "__ABDUCTOR_SELF" of the D-ABDUCTOR window. An ordinal packet has information about sender and receiver of the packet and a message described in the language Simple. A D-ABDUCTOR process is watching the window property named "__ABDUCTOR_SELF" of itself. When someone changes the property, the D-ABDUCTOR process reads the string and regards it as a packet.

Structure of Packets

Message Transmitter transmits ordinal command in the language Simple as a packet. A packet has the following form.

! [***sender***] ! [***receiver***] ! ***length*** ! ***message***

An exclamation mark (!) is a separator of fields. The sender and receiver fields are used in inter communication among D-ABDUCTOR processes on different displays. The length field represents the length of character string in hex-decimal. The message field is the character strings. Empty fields of sender and receiver are acceptable. Message Transmitter uses the following form as a packet.

!!! ***length*** ! ***message***

Atom Names

Message Transmitter uses the following atoms. These atom names must be consistent with D-ABDUCTOR.

`__ABDUCTOR_SELF`

A D-ABDUCTOR process is watching the window property whose atom name is "__ABDUCTOR_SELF" of itself. Message Transmitter writes a character string representing a packet to the window property.

`_GRIPS_DA_CONTROL`

A D-ABDUCTOR process is watching the window property whose atom name is "_GRIPS_DA_CONTROL" of the root window. Message Transmitter writes a character string of a control command to the window property.

7. Card Base

Card Base is a database management system but it offers only facilities to retrieve cards. You give a keyword expression to Card Base, and Card Base retrieves every card whose key text matches the keyword expression. Card Base returns some statements in the language Simple to create nodes on the canvas of D-ABDUCTOR.

Database File

The Card Base may use some kinds of files. One is the master file. The others are data files.

Master File

The master file includes a unique number and key text of every card. A line, which ends with '\n', of the master file represents a record. A record of the maser file corresponds a card, and has two fields. First field includes a unique number of a card, and the second field includes key text according to the card.

A physical line has the following form.

number : ***key_text_of_card***

The ***number*** is digits, for example "001," representing a unique number. The ***key_text_of_card*** is a character string it may include EUC kanji code. A colon separates these two fields.

You can use arbitrary name for the master file. The default name is "carta.cb".

Data Files

The data files include some data of all cards. A file includes data corresponding to only one card. Thus you have to prepare the same number of files as cards for one kind of data. All files of a kind of data must be in the same directory.

The name of files must be constructed by using the number of cards by the following C statement.

```
sprintf(name, format, number);
```

Where, the name and the format are arrays of **char**'s and the number is an **int**.

Image files

The image files include image data of all cards. The image data is represented in XPM format. The default format to construct the names is "%03d.xpm". Thus the names are "001.xpm", "002.xpm", ..., "100.xpm".

Text files

The text files include text data of all cards. The text may include EUC kanji code. The default format to construct the names is "%03d.txt". Thus the names are "001.txt", "002.txt", ..., "100.txt".

Script files

The script files include script in language Simple corresponding to all cards. The default format to construct the names is "%03d.sl". Thus the names are "001.sl," "002.sl", ..., "100.sl".

Command

Synopsis

`cardbase` [`-x` ***xpos***[+***xinc***]] [`-y` ***ypos***[+***yinc***]]
[`-d`***master_file***]
[`-I`[***image_path***]] [`-i`[***image_form***]]
[`-T`[***text_path***]] [`-t`[***text_form***]]
[`-S`[***script_path***]] [`-s`[***script_form***]]
[`-em`] [`-debug`] [`-trace`] ***keyword_expression***

Options

`-x` ***xpos***[+***xinc***]

The x coordinates of cards on the canvas of D-ABDUCTOR. The x coordinate of n-th card becomes ***xpos*** + (n - 1) ***xinc***. The ***xinc*** option may be omitted. Card Base uses a default value when ***xinc*** is omitted.

`-y` ***ypos***[+***yinc***]

The y coordinates of cards on the canvas of D-ABDUCTOR. The y coordinates of n-th card becomes ***ypos*** + (n - 1) ***yinc***. The ***yinc*** option may be omitted. Card Base uses a default value when ***yinc*** is omitted.

`-d`***master_file***

The name of master file. It can include absolute path.

`-I`[***image_path***]

Card Base uses image files. When you also specify the path name ***image_path***, Card Base uses image files in the directory represented by the path.

`-i`[***image_form***]

Card Base uses image files. When you also specify the format ***image_form***, Card Base constructs the names of image files by using the format.

`-T[`***text_path***`]`

Card Base uses text files. When you also specify the path name ***text_path***, Card Base uses text files in the directory represented by the path.

`-t[`***text_form***`]`

Card Base uses text files. When you also specify the format ***text_form***, Card Base constructs the names of text files by using the format.

`-S[`***script_path***`]`

Card Base uses script files. When you also specify the path name ***script_path***, Card Base uses script files in the directory represented by the path.

`-s[`***script_form***`]`

Card Base uses script files. When you also specify the format ***script_form***, Card Base constructs the names of script files by using the format.

`-em`

This option is used to emphasize keywords in the text. On the canvas of D-ABDUCTOR, all keywords that are used to retrieve the cards are emphasized.

`-debug`

This option is used to set debug mode. In debug mode, Card Base dumps some information for debugging. Ordinal users do not need to use this option.

`-trace`

This option is used to set trace mode. In trace mode, Card Base dumps some trace information. You can use this option to confirm the structure of keyword expressions and retrieved cards.

Keyword Expression

Syntax

A keyword expression is described in a Polish notation. Three prefix operations are available. Syntax of a keyword expression is described as follows.

keyexp	:=	***keyword***	(1)
	|	-a***n*** ***keyexp1 keyexp2 ... keyexpn***	(2)
	|	-o***n*** ***keyexp1 keyexp2 ... keyexpn***	(3)
	|	-n ***keyexp*** (4)	

The operations "-a" and "-o" are followed by digits that specifies the number of operands. You can omit the digits if the number is 2.

Semantics

A keyword expression corresponds a set of cards. The system retrieves all cards in the set by the keyword expression. The set is defined recursively by the followings.

A keyword expression in the form (1), that is, a keyword corresponds a set of every card whose key text includes the keyword. A keyword expression in the form (2) corresponds intersection of every set that the keyword expression ***keyexpi*** ($1 \leq i \leq n$) corresponds. A keyword expression in the form (3) corresponds union of every set that the keyword expression ***keyexpi*** ($1 \leq i \leq n$) corresponds. A keyword expression in the form (4) corresponds compliment set of the set that the keyword expression ***keyexp*** corresponds.

Examples

SPRING

This keyword expression corresponds a set of every card whose key text includes word "SPRING."

-a2 SPRING SNOW

This keyword expression corresponds a set of every card whose key text includes both words "SPRING" and "SNOW." You can omit the digit "2."

-o3 SPRING SUMMER AUTUMN

This keyword expression corresponds a set of every card whose key text includes either words "SPRING," "SUMMER" or "AUTUMN."

-o3 -a2 SPRING SNOW SUMMER AUTUMN

This keyword expression corresponds a set of every card whose key text includes words "SPRING" and "SNOW," or "SUMMER" or "AUTUMN." You can omit the digit "2."

-n AUTUMN

This keyword expression corresponds a set of every card whose key text does not include word "AUTUMN."

Functions of Card Base

Generating Statements

Card Base generates a statement in the following form for every retrieved card.

`%V @`***number*** `[`***attributes***`]`

For more detail information about *conditional reference*, see Chapter 5, "Language Simple."

The ***number*** is a unique number of the card. Card Base might retrieve the same two or more cards by different users or by different keyword expressions. On the canvas of D-ABDUCTOR, however, the same two or more cards are not necessary. A conditional reference (`@`***number***) is used not to create the same two or more cards.

The ***attributes*** included the statement depends on the options.

`ptext:` ***string***

Whenever the attribute includes key text. The ***string*** is the key text enclosed by double quotation marks.

`xpos:` ***xpos***

When you specify the x coordinate by using `-x` option, the attribute includes x coordinate ***xpos*** of the card.

`ypos:` ***ypos***

When you specify the y coordinate by using `-y` option, the attribute includes y coordinate ***ypos*** of the card.

`xpmfn:` ***image_file***

When you specify the `-I` option or `-i` option, the attribute includes a file name ***image_file*** of image data according to the card.

`txtfn:` ***text_file***

When you specify the `-T` option or `-t` option, the attribute includes a file name ***text_file*** of text data according to the card.

Additionally, when you specify the `-S` option or `-s` option, Card Base generates a statement in the following form for every retrieved card.

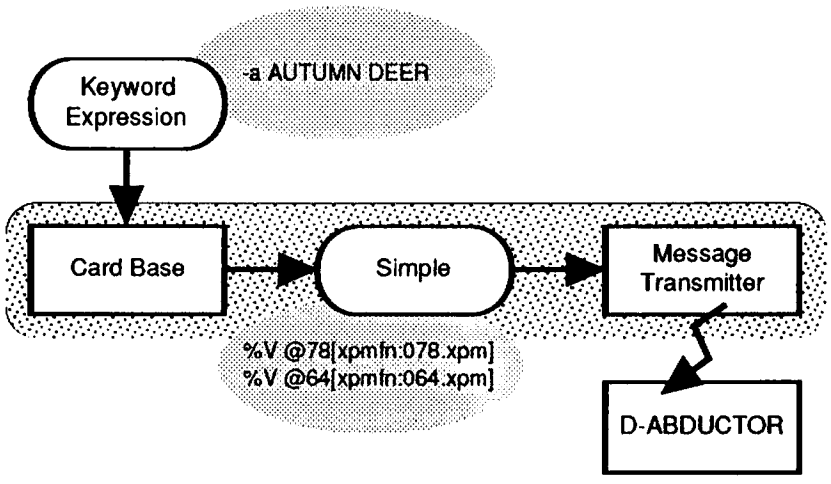
```
%x LOAD script_file
```

The name ***script_file*** is a name of script file according to the card.

Communication with D-ABDUCTOR

Card Base cannot communicate with D-ABDUCTOR directly. One easiest way is to connect Card Base with Message Transmitter by a UNIX pipe. In this way, Card Base communicates with D-ABDUCTOR by the following procedure.

1. You invoke Card Base with a keyword expression and Message Transmitter connected by a pipe.
2. Card Base retrieves some cards from its card database by using the keyword expression, and generates some statements to create retrieved cards on the canvas of D-ABDUCTOR. The statements are described in the language Simple. Card Base writes the statements to the standard output.
3. Message Transmitter reads strings from the standard input, and writes them to a property of D-ABDUCTOR window.
4. D-ABDUCTOR regards the string of its window property as statements, and executes them to create some cards retrieved by the keyword expression.



Data flow between Card Base and D-ABDUCTOR

Customization

The following environment variables are available to customize the configuration of data files.

CARDBASE_DATA_FILE

This variable is used to specify the master file. It may include absolute path. The default file is "./carta.cb".

CARDBASE_IMAGE_PATH

This variable is used to specify the path to the image files. The default path is ".", that is the current directory.

CARDBASE_IMAGE_FORM

This variable is used to specify the format to construct the image file names. The default format is "%03d.xpm".

CARDBASE_TEXT_PATH

This variable is used to specify the path to the text files. The default path is ".", that is the current directory.

CARDBASE_TEXT_FORM

This variable is used to specify the format to construct the text file names. The default format is "%03d.txt".

CARDBASE_SCRIPT_PATH

This variable is used to specify the path to the script files. The default path is ".", that is the current directory.

CARDBASE_SCRIPT_FORM

This variable is used to specify the format to construct the script file names. The default format is "%03d.sl".