IDEAL-EDIT: Documentation and Users Guide

Jack Breese
Sampath Srinivas
Jim White
ideal-request@rpal.rockwell.com
Rockwell International Science Center
Palo Alto Laboratory
444 High Street, #400
Palo Alto, CA 94301
(415) 325–7145

November 1993

Beta Test Version .8
Bug reports requested

Technical Memorandum No. 79

Contents

1 Introduction

IDEAL-EDIT is a graphical user interface for IDEAL, based on the CLIM graphics package for Common Lisp. It is a graphical editor that provides commands to construct, edit, and solve belief networks and influence diagrams using the functionality in IDEAL, a LISP system developed for building and evaluating influence diagrams (See [1, 2] for more details). Users of the IDEAL-EDIT package should be familiar with IDEAL.

IDEAL-EDIT (and this manual) was originally written by Denise Draper at University of Washington. It is being maintained and extended by Rockwell International Science Center, the developers of IDEAL. This version of the program and documentation are Beta releases. The system has been tested for CLIM 1.1 on Symbolics and Sun Workstations running Franz Lisp or Lucid Common Lisp. The system was originally written for X-windows on the Sun or Symbolics, and therefore assumes a three-button mouse. We have provided for running the same code on a MacIntosh, using Macintosh Common LISP (MCL) Version 2.0 and CLIM 1.1 provided with MCL. For the Mac, the mouse button mapping is as follows:

Left-button Click ⇒ Single Button Click without modifier keys

Middle-button Click ⇒ Control Click

Right-button Click ⇒ Option Click

For bug reports, send mail to `ideal-bugs@rpal.rockwell.com` or for more information, send mail to `ideal-request@rpal.rockwell.com`.

2 Starting up the Editor

Once the IDEAL-EDIT code and CLIM has been loaded into LISP, CLIM and the editor are initialized with the following functions. These functions are exported from the IDEAL-EDIT package.

`run-editor` *[function]*
Brings up the IDEAL-EDIT window.

`rerun-editor` *[function]*
If an IDEAL-EDIT window has been exited, then this function reinvokes the editor in the same state as prior to the exit.

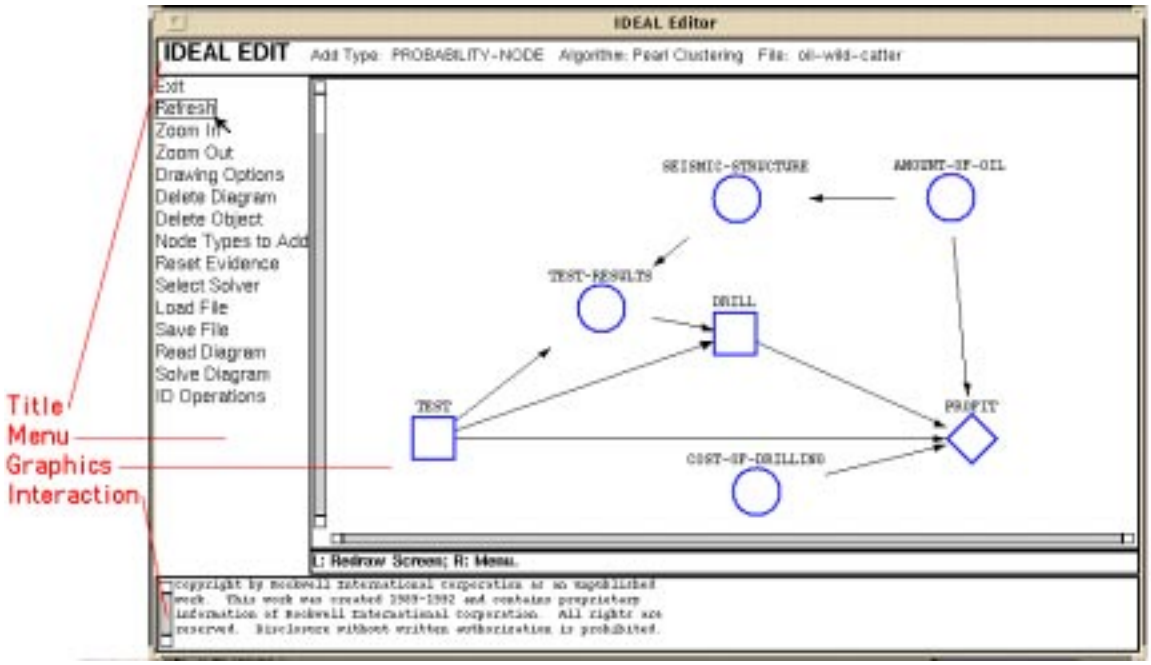

Figure 1: The IDEAL-EDIT application window consists of 4 panes: Title, Menu, Graphics, and Interaction.

3 Pane Descriptions

The main interaction window consists of four panes. Figure 1 shows this screen.

Title: The title pane at the top of the frame shows the currently selected "add node" type (see below), the currently selected solution algorithm, and the currently selected file.

Menu: The menu pane on the left lists a set of mouse sensitive commands. These functions are described in Section 5. The MCL version of CLIM places these menu commands in a pull-down menu from the Apple menu bar rather than in a separate pane on the application window.

Graphics: The graphics pane is where nodes and links are displayed. On some platforms, the bottom of the graphics pane is a mouse documentation line.

Interaction: The interaction pane at the bottom of the screen provides messages to the user and a facility for user typed input.

4 Graphics Pane Interactions

Most actions to build or manipulate a diagram consisting of nodes and links are accessed from the graphics pane using the mouse.

Adding a node: Clicking left on the background of the graphics pane will create a new node. The type of node (probabilistic, deterministic, noisy-or, value, or decision) added is determined by the "add-node type" displayed at the top of the window. Once your diagram is stable, and you no longer wish to add nodes, selecting "add-node-type" of "none" will disable adding nodes.

The basic framework for the noisy-or nodes is described in [3]. The noisy-or is a specialization of a chance node, where an underlying Boolean function (such as an OR function) is modified with independent, uncertain, noise parameters. The benefits of the noisy-or are its ease in assessment. Adding a node will create a binary noisy-or node in the diagram.

Selecting an object: Moving the mouse over a node or a link causes it to be highlighted. Clicking the middle button of the mouse on the highlighted object will select that object (either a node or a link). An exception is when another node is already selected. Then middle clicking will cause a link to be added from the previously selected node to the newly selected node.

Adding a link: When a node is already selected, middle clicking on another node will cause a link to be added from the previously selected node to the newly selected node.

Moving a node: Nodes are dragged by holding the right mouse button down over a node, moving the mouse (and the node), and releasing when the node is at its desired location. Be aware that the CLIM code for dragging presentations can be a bit sluggish.

Editing a node: Node states, names, and probability distributions are accessed through a pop-up editor window that is invoked by clicking left on a node. See section 6 for a description of this interface.

Deleting a node: A node can be deleted by selecting the node using the middle button and then selecting `Delete Object from the menu bar.`

Deleting a link: `A link can be deleted by either 1) selecting the link using the middle button and then selecting Delete Object from the menu bar, or 2) selecting Edit Link with a left click on the link and then selecting Delete Link from the submenu.`

Reversing a link: `A link can be reversed by selecting Edit Link with a left click on the link and then selecting Reverse Link from the submenu. This operation modifies the probabilities and adds links in the network in accordance with Bayes' rule.`

5 Menu Commands

The menu commands are available from the menu pane of the editor (or from a pull-down menu in the case of MCL CLIM). Their effects are documented below.

Exit: Exit the editor. After exiting, invoking `(rerun-editor) will bring back the editor in its previous state.`

Refresh: Redraw the diagram. This is useful for recovery from CLIM redisplay glitches.

Zoom In, Zoom Out: Scale the diagram so that more or less of it is visible in the display area.

Drawing Options: This invokes a submenu which controls the setting of certain options. Currently the options are:

Display Values: Controls whether or not the computed probabilities of each node are shown in the graphical display of the diagram.

Display Link Messages: Controls whether or not the λ and π messages are shown on the links in the graphical display of the diagram (pertinent only for poly-tree).

Delete Diagram: Clears the diagram.

Delete Object: Deletes the currently selected object (node or link) from the diagram.

Node Types to Add: Invokes a sub-menu to choose the type of node to create when building a diagram. This ''add node type'' is displayed in the title pane. Allowable node types are probability-node, deterministic-node, decision-node, value-node, and noisy-or-node.

Reset Evidence: Remove all evidence on the diagram.

Select Solver: Invokes a sub-menu to choose which solver to invoke. This algorithm name is displayed in the title pane. The choices correspond to the following IDEAL functions:

Poly Tree: poly-tree-infer

Pearl Clustering: clustering-infer

Jensen Clustering: jensen-infer

Conditioning: conditioning-infer

Simulation: simulation-infer

Schacter Evaluation: shac-eval

IDES Evaluation: ides-eval

See the IDEAL documentation for specifics on the algorithms.

Load File, Save File: Load or save the diagram to a file. Loading provides a pop-up menu showing file names from the IDEAL diagram directory. Saving a file generates a prompt for a file name in the textual window - you must supply the filename without the extension. This actually creates/reads two files, one of which is the standard IDEAL file, the other of which contains additional information needed by IDEAL-EDIT. If only the IDEAL file is available, it will load the file anyway, and the initial location of the nodes on the screen will be generated based on ordering the nodes in the network.

Read Diagram: Prompts for a LISP expression in the textual window, which is evaluated to return a new diagram. When this expression is evaluated, the variable ideal:*diagram* is bound to the diagram in the window. Examples of useful expressions (assuming that you have done (use-package 'ideal):

```
(create-random-diagram 10)
(reduce-probabilistic-node '#n(cost-of-drilling) *diagram*)
*diagram*
```

The last is useful for forcing a complete resynchronization between the editor and IDEAL.

Solve Diagram: Invoke the selected solver on the diagram. If there are error messages, they will be displayed in the textual window.

Note that diagram set-up or initialization does not need to be done explicitly; IDEAL-EDIT keeps track of when these need to be done and does them for you. (However, it is possible to confuse the editor if you do operations on your diagrams outside the editor, and then re-enter the editor; in this case you can cause the diagram to be re-initialized by using ``Read Diagram'' with the argument ideal:*diagram*, as mentioned above.)

The influence diagram evaluations take place on copies of the diagram, so they do not affect the displayed diagram. When an influence diagram solution has been done, the expected utility of the value node, and the decisions on any decision node, if unique, are displayed.

ID Operations: Invoke a sub-menu which displays many of the primitive IDEAL operations on influence diagrams, such as absorb-chance-node. Select one of the entries, and then select a node or link as the operand, if necessary. These operations do act on the diagram directly, resulting in a changed diagram. See the IDEAL documentation for an explanation of the individual operations.

6 Node Editor Interface

The internal state of a node may be edited by invoking its node editor, accomplished by clicking on the node with the right mouse button. If you have the `multiprocessing` package, the node editor will run in parallel with the diagram editor, allowing you to switch back and forth between the main diagram and as many node editors as you wish. Note that keeping many node editors open tends to make things run very slowly. The node editor display depends on the type of node being displayed, but in general it consists of a display indicating the parents of the node, its distribution, any evidence on the node, and the last belief calculated for that node. If the value of the node is conditional on the value of its parents, the node editor will also display the probability distribution or value for a single instance of the parent values at a time. In addition, there is a window for textual input and error messages, as in the main display window. The editor window is shown in Figure 2.

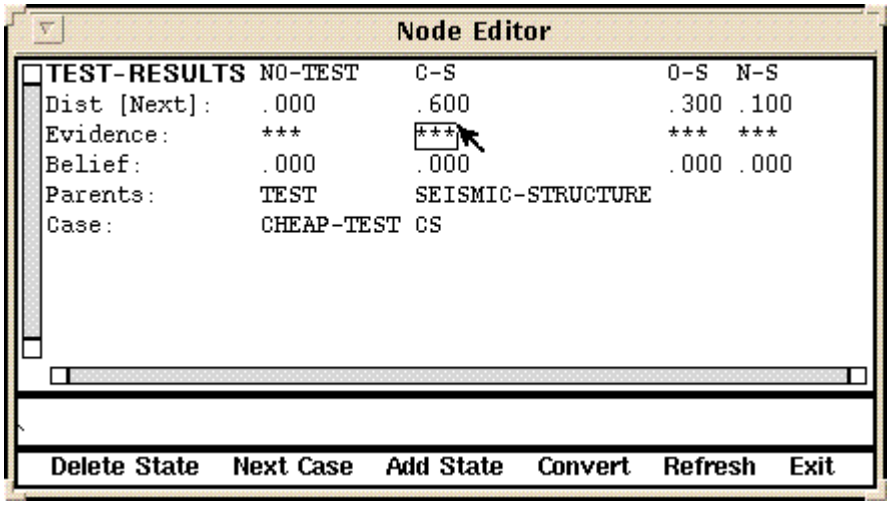

Figure 2: The IDEAL-EDIT Node Editor window.

Many node editing operations are accessed through the node editor. Each is described below:

Changing a node name: Click on the bold node name entry in the upper left of the display and edit the node name directly (in this example, the node name is "Test-Results"). The name must be a Lisp symbol.

Changing the name of a node state: Click on the node state name in the top row, and enter the new name. The name must be a Lisp symbol.

Adding a new node state: Click on the menu entry `Add State` and enter the new state name in response to the prompt. The name must be a Lisp symbol. The new state will be pushed onto the beginning of the list of states; it is initialized to have a conditional distribution of 0.

Deleting a node state: Click on the menu entry `Delete State`, then click on the name of the state you wish to delete. The distribution of the deleted state is spread over the remaining states; the distribution of the child nodes is also changed. [Note that deleting and then adding a state will have a different effect than renaming a state because of the way in which the distribution is marginalized out in the node and its children.] If the node is a decision node, the node editor will display the current policy if the influence diagram has been evaluated to obtain an optimal decision.

Changing the distribution or values: Click on the item you wish to change, and enter the new value. The displayed distribution is for a particular configuration of the predecessors to a node - the case. To change the case, click `Next Case` in the menu bar or on the `Dist [Next]` entry. For probability distributions, the editor ensures that the distribution sums to one; it modifies the distribution in a left-to-right fashion.

Entering evidence: IDEAL supports two different kinds of evidence: 'concrete' evidence, which states that a particular state of the node is definitely

true, and 'virtual' evidence, which gives odds on the different node states. Note that most of the inference algorithms operate only with concrete evidence; only poly-tree-infer and clustering-infer can handle virtual evidence.

IDEAL-EDIT allows you to enter both kinds of evidence. No evidence at all is indicated by all fields containing ''***''. To indicate concrete evidence, click on the evidence field which has been observed; this will display as ''xxx'', while the other fields display as ''- - -''. When either concrete or virtual evidence is present, clicking on the ''Evidence:'' label clears it. To indicate virtual evidence, click on the ''Evidence:'' label again; this will switch to 'virtual mode', where the evidence is initially displayed as all 1's (even odds). Clicking on one of the numeric fields allows you to change it to other values. The following table summarizes the state transitions:

evidence type	click on Evidence button	click on field
no evidence	virtual	concrete
virtual	no evidence	virtual
concrete	no evidence	concrete

Convert: This item is used to convert probabilistic and deterministic nodes into NOISY-OR nodes and vice-versa [3]. If a node is a noisy-or node, Convert transforms it into a standard probabilistic node with the table entries calculated from the noisy-or relation. If the node is a probabilistic node, Convert creates the appropriate :nary or :binary noisy-or node based on the number of states that it has. If the node to be converted is a deterministic node, then the deterministic function is used as the basis for the noisy-or definitions. See [3],[2] for details.

Refresh: This item refreshes the data and display.

A node editor display will not be updated when information changes from outside the editor. Warning: this can be very confusing. For example, if a node editor display is open when "Solve Diagram" is selected at the diagram level, the node editor will not display the updated belief. Clicking "Refresh" in the node editor, or closing and opening it, will cause it to display the new data correctly. Also, changes to noisy-or node distribution parameters are not updated until the the editor is closed.

7 Accessing the diagram

The following function provides access to the current IDEAL-EDIT diagram.

`diagram` *[function]*
Returns the IDEAL diagram currently displayed in the current IDEAL-EDIT window. This function is exported for the IDEAL-EDIT package.

In addition, the variables `ideal-edit:*diagram-cliques*` and `ideal-edit:*diagram-join-tree*` get set by solving the diagram with Pearl Clustering or Jensen Clustering respectively.

8 Installation/Implementation Notes

Loading and installation instructions are included with the IDEAL and IDEAL-EDIT distributions. IDEAL-EDIT assumes that IDEAL is in a package named "IDEAL" – if this is not the case, probably the easiest thing to do is make "IDEAL" a nickname for whatever package does contain IDEAL.

IDEAL-EDIT traps all errors from IDEAL and will generally do the right thing. However, in some cases IDEAL issues a warning message but no error, which might result in the diagram displayed by IDEAL-EDIT becoming out of sync with `ideal:*diagram*`. It is possible to re-synchronize the two as explained under the "Read Diagram" command in Section 5.

The node editors use multiprocessing functionality if it is available in the underlying Common Lisp. If multiprocessing is not present, the node editors will be run as subroutines, i.e., you will have to exit them before returning to the main diagram editor.

A feature of CLtL2 that is widely used in the code is the error handler `handler-case`. This functionality is essential for the IDEAL-EDIT code; if `handler-case` does not exist, IDEAL-EDIT will have to be changed to use something equivalent.

9 Bug Reports and more information

For bug reports, please send mail to `ideal-bugs@rpal.rockwell.com`. For more information, please send mail to `ideal-request@rpal.rockwell.com`. Alternately, you can use the contact information below.

If you port IDEAL-EDIT to a platform other then Allegro CL or Lucid CL on Suns, Symbolics Genera 8.X, or MCL 2.0, please let us know what changes were necessary so that they can be incorporated in the general distribution.

Sampath Srinivas
James White
(415) 325–7145
Rockwell International Science Center
Palo Alto Laboratory
444 High Street, #400
Palo Alto, California 94301

References

[1] Srinivas, S. and Breese, J. S. IDEAL: Influence Diagram Evaluation and Analysis in LISP. Documentation and user's guide. Technical Memorandum No. 23, Rockwell International Science Center, Palo Alto Lab, Palo Alto, CA 94301. January 1989.

[2] Srinivas, S. and Breese, J. S. IDEAL: A software package for analysis of influence diagrams. Proceedings of Sixth Workshop on Uncertainty in Artificial Intelligence, Cambridge, MA. August 1990.

[3] Srinivas, S. Generalizing the Noisy-Or concept to non-binary variables. Technical Memorandum No. 79, Rockwell International Science Center, Palo Alto Lab, Palo Alto, CA 94301. May 1992.

Manual formatted on November 10, 1993.