MULTIMEDIA INTERFACES FOR ADVANCED MATHEMATICS: OTTER 3.0 AS A CASE STUDY

John F. Richardson
NCCOSC
Code 44201
53560 Hull Street
San Diego, Ca. 92152
United States
Email: richards@marlin.nosc.mil

ABSTRACT

This paper describes a multimedia interface for the OTTER theorem proving system. This interface is implemented using an advanced multimedia authoring system based upon the SUPERCARD program. SUPERCARD is a Hypercard type system that basically solves most of the problems previously existing in Hypercard. The interface described in this paper is designed for the Macintosh computer although multimedia authoring systems exist for Unix and PC based systems.

The primary purpose of this interface is to provide a “Macintosh look and feel” environment for the OTTER theorem proving system. The interface provides this look and feel through the use of hypermedia objects called cards. These cards serve as backdrops for interactive graphics and controls made available to the user for interacting with the theorem prover.

The secondary purpose of the interface is to provide a natural environment for navigating within the theoretical realms of Automated Reasoning, Logic Programming, Theorem proving and mathematics. The interface seeks to restructure the rules and symbols of the above realms into more “user friendly” and “intuitive” rules and symbols for use by students and general mathematicians. The interface also seeks to make it easier on experts in the field of Automatic theorem proving.

KEYWORDS: Multimedia ; Automatic Theorem Proving ; Expert Systems ; Animation ; Hypermedia

SYSTEM DESCRIPTIONS

The system that the interface is designed to serve is the OTTER [1] theorem proving system developed by the Argonne National Laboratory. The interface is designed to serve OTTER 3.0, which is the latest version. First, some general comments.

OTTER is a computer aid that helps researchers solve problems. Once a problem is defined it is translated into a form that OTTER can manipulate to produce a computer aided proof. This is the easy part for advanced professionals used to automatic reasoning. It may not be quite so easy for students and other mathematicians who are not familiar with computer aided deduction.

Usually this translation involves the production of Prolog style clauses from a starting set of natural language descriptions of the problem to be solved. An example is translating

IF a=b AND b=c THEN a=c to

q,r -> s and then to -s | q | s

Note that OTTER can do the second line but the researcher must reduce the natural language description in the first line to what OTTER calls sequent form.

In general, computer aided deduction is a process that can be very naive and lead to a humongous amount of deductions that are worthless. This sometimes causes the generation of gargantuan amounts of dead end deductions from each of the worthless clauses produced during the search for a proof. OTTER does not rebel at such behavior but most sentient researchers do rebel. In order to curtail such unruly behavior, several sets of additional clauses have to be produced by the researcher.

Generating the additional clauses is the hard part. The classes of these additional clauses are related to the automatic deduction strategies implemented in the automatic deduction system that are used to "guide" the deduction sequence to a successful and useful conclusion. In OTTER 3.0 these clauses are geared towards aiding the following strategies: Demodulation, Knuth Bendix methods, lexical ordering, weight functions, etc.

In order to solve a problem using OTTER, the user provides a file consisting of OTTER system commands. These commands provide for the input of clauses and integer values that initialize system variables and determine parameter and flag settings that affect the strategies that "guide" the proof. OTTER has a small set of a dozen commands. Three of these commands relate to "user friendly" functions. These are the **include** , **op** and **make_evaluable** commands. Three of the commands relate to the "system housekeeping" flags and parameters that control the course of the search for a proof. These commands are **set**, **clear**, and **assign**. There are 70 flags divided into the following groups.

1) Flags for Main Loop control.
2) Flags relating to Inference Rules.
3) Flags relating to Paramodulation.
4) Flags for handling Generated Clauses.
5) Flags relating to Demodulation and Ordering.
6) Flags relating to Input.
7 Flags relating to Output.
8) Flags relating to Indexing.
9) Miscellaneous Flags.

There are 19 parameters divided into the following parameter groups.

1) Parameters for monitoring progress of the search for a proof.
2) Parameters for placing limits on the search.
3) Parameters for placing limits on the properties of generated clauses.
4) Indexing Parameters.
5) Miscellaneous Parameters.

There are two commands for the input of the clauses that guide the search for a proof. These are the **list** and the **formula_list** commands. There is a command for input of weight lists (the **weight_list** command). The **lex** command assigns an ordering on symbols. The **skolem** command identifies skolem functions and the **lrpo_multiset_status** command sets the status for lrpo which is a property in OTTER that guarantees termination.

The SUPERCARD [2] system basically produces a series of ordered and interrelated graphical objects that can be selected using a point and click strategy. By using the mouse and keyboard as user friendly input devices a series of messages are generated that can be interpreted and processed by the SUPERTALK scripting language. The SUPERTALK scripting language is basically a high level multimedia programming language. It can be thought of as the multimedia equivalent of C that is in an **interpreted** natural language format. SUPERCARD allows the user to rapidly create Macintosh windows, menus and graphics. A SUPERCARD program is called a project. The windows of the interface live inside the project and each window consists of a set of **cards**. Cards should be thought of as a sort of blackboard upon which the various classes of SUPERCARD graphical objects can be placed. SUPERCARD windows and cards can communicate among different projects.

SUPERCARD has a set of built in graphical objects for Graphical User Interface (GUI) design. The primary classes of objects in SUPERCARD are **buttons**, **fields**, **menus**, **drawing objects** and **paint objects**. These objects are made available to the designer via a tool palette. The Interface designer simply clicks on an object in one of the palettes and drags the object onto a card. Each object can have a script consisting of a sequence of keywords and statements from the SUPERTALK script language. These statements are user friendly from the standpoint of natural language. An example is the statement "**put the third word of xyz into card field abc of card alphabet of window fairy_tail"**. SUPERTALK scripts are executed when a message matching what could be considered a "subroutine name" in a language like C is dispatched to a card, window or project. These messages are generated by user actions or by statements within a script. As an example, Suppose the user clicks the mouse button on a button that has the phrase "shuffle letters" displayed inside the button. Then if the above example statement is enabled to execute when a mouse click occurs the third word in the text string contained in xyz will be placed into the field abc in the alphabet card of the fairy_tale window.

Buttons are used by the interface designer to perform some action when the user clicks on the graphical object representing the button. Buttons can be used to display the familiar Macintosh "check boxes" and "radio buttons". These check boxes and radio buttons serve to set and clear flags in many interface designs. Fields are used to display information and to serve as forms or templates for user input of text or values for parameters. Draw objects are used in conjunction with buttons and fields to provide a wide variety of labels. Fields can also be used as labels that are interactively changed in response to user actions. Draw objects and paint objects can be used to provide pure graphic embelishments for the user interface. Fancy graphics can also be "clicked on" and thus perform interface functions by executing scripts. SUPERTALK provides for the creation of menus similar to the menus in the apple menu bar that are displayed when executing word processing programs, etc. These menus can be modified interactively in response to user actions.

It should be noted that the intrinsic design capabilities of SUPERCARD are geared towards 2 dimensional color graphics design. SUPERCARD has extensive facilities for animation and execution of sound and video files. For sound, the files are .snd resources produced by many commercial sound editing systems(SoundPEdit Pro, etc.). For video and 3 dimensional graphics design the files are PICT files produced by many image processing programs (Adobe Photoshop, etc.) and video programs. Most 3-D design programs (Ray Dream Designer, etc.) produce PICT files.

INTERFACE DESIGN

The design of the interface is bases upon three goals. The first goal is to provide basic user friendly I/O. The second goal is to provide user friendly input to the flags and parameters used in OTTER. The third goal is to provide a rudimentary interface geared towards providing clever clauses for use in guiding the deduction process.

The OTTER interface consists of a series of menus and windows that have the look and feel of general Macintosh menus and windows. The menus are the OTTER menu, the flags menu, the parameters menu, the Expert menu and the Help menu. The windows consist of the OTTER3.0 I/O window, OTTER3.0 Settings window, OTTER3.0 Control window, Expert window and the Help window.

The OTTER menu displays cards that provide graphical interfaces for displaying lists of "canned" OTTER input files and for creating input files to hold the information generated by the other menu functions available to the user within the OTTER interface. The OTTER menu also allows for the entry of clauses into a text field. The contents of the text field are then deposited into OTTER input files created by the cards and functions available from the OTTER menu. Commands generated in response to the user's actions for setting and clearing flags and also for assigning parameters can be inserted anywhere in the generated input files. There is a facility for the display of previously generated error files.

The flags menu displays cards that provide graphical interfaces for setting or clearing the flags that are used by OTTER to “guide” the search for a proof. The function of setting and clearing OTTER flags is implemented via check boxes or 3D "buttons" on the various cards that can be selected by the user via the settings menu. The results of executing the scripts associated with the check boxes/buttons is a set of OTTER **set** and **clear** commands that are deposited in the OTTER input files created by the interface.There is a menu item for each of the 9 classes of OTTER flags. The interface checks for the status of the check boxes and generates the required **set** and **clear** commands. Generation of the OTTER commands is performed when the user chooses to submit a file.

The parameters menu displays cards that provide graphical interfaces for assigning integer values to the OTTER parameters that are used by OTTER to “guide” the search for a proof. The function of assigning OTTER parameters is implemented via SUPERCARD fields on the various cards that can be selected by the user via the parameters menu. The results of executing the scripts associated with the fields is a set of OTTER **assign** commands that are deposited in the OTTER input files created by the interface. There is a menu item for each of the 5 classes of OTTER parameters. The interface displays information on the default value of the individual parameters and the range of values that can be entered. The interface performs range checking and translates the input into OTTER **assign** commands. Generation of the OTTER commands is performed when the user chooses to submit a file.

The Help menu provides for the display of information about the OTTER interface. The help information displayed ranges from general help to specific information on the menu selections. Much of the help information consists of excerpts from the OTTER users manual.

The Expert menu provides for access to a rudimentary Expert system shell [3] that is written in Open Prolog which is the public domain (or at least shareware) Prolog from Trinity College in Dublin. This expert system shell is devoid of rules and must be filled up with rules related to “guiding” automatic reasoning programs such as OTTER. Users are encouraged to deposit their “rules” with the Association for Automated Reasoning. The interface translates the rules from “natural language form” into a form acceptable to the Expert system. There is a scrolling field that contains a list of the entered rules in both “natural language form” and it’s associated PROLOG form. The rules are output to an output file with the file extension .rule. These rule files are searched by the expert system when it starts up. An example is the following:

IF flag is ur_res and flag is for_sub
THEN max_weight is 10

is translated to the following form:

```
Parameter ( max_weight, 10 ) :-
        flag ( ur_res),
        flag (for_sub).
```

INTERFACE HYPERMEDIA

The SUPERCARD authoring system has built in support for animation and grapics input. Acceptable input sources are any 3D design or animation system capable of producing PICT or PICS files. PICT file input is used to create 3D clickable buttons. PICS file input is used for animation sequences. All of the figures below are just the last in an animation sequence. Producing 3D multimedia "buttons" is just a matter of using the SUPERCARD transparent button tools and tracing the outline of the button.

The 3D PICT and PICS files are produced using VIRTUS VR and Ray Dream Designer. These systems are used to produce the realistic textured scenes used in the interface. More sophisticated models for the animation function in VIRTUS VR can be produced using VIRTUS WALKTHROUGH PRO. The Virtus software is basically a virtual reality system for architects. These and other 3D design systems have sophisticated surface modeling functions and a wide array of texture generators. This interface makes use of "canned" scenes that are supplied with the Virtus and Ray Dream systems.

INTERFACE FIGURES

Figures 1 and 2 show examples of a cards that are selected from the OTTER menu. Figure 3 shows an example of a card that is selected from the Flags menu.Figure 4 shows an example of a card that is selected from the Parameters menu. Figure 5 shows an example of a card that is selected from the Help menu. Figure 6 shows an example of a card that is selected from the Expert menu.

It should be noted that many of the graphic objects in the figures are hypermedia objects. If you click on them something happens. Technically they are SUPERCARD buttons and SUPERCARD "hot text" that is equivalent to hypertext in the HYPERCARD system.

For example, in figure 1, clicking on the large screen TV graphic starts the interface. Clicking on the bed quits the interface. In figure 4, clicking on an asteroid sets or clears a paramodulation flag. Clicking on one of the numbers in figure 5 results in help text being displayed. Clicking on the conference table in figure 2 will open a new OTTER3.0 file, clicking on the computer screen submit button will sumbmit an OTTER3.0 file to the theorem prover and execute the theorem prover.

REFERENCES

1.Mclune, W., *OTTER 3.0 Reference Manual and Guide*, Argonne National Laboratory Technical Report ANL-94/6, January 1994.

2. Himes, A., Ragland, C., *Inside Supercard*, Microsoft Press, 1990

3. Merritt, D., *Building Expert Systems in Prolog*, Springer Verlag, 1989.

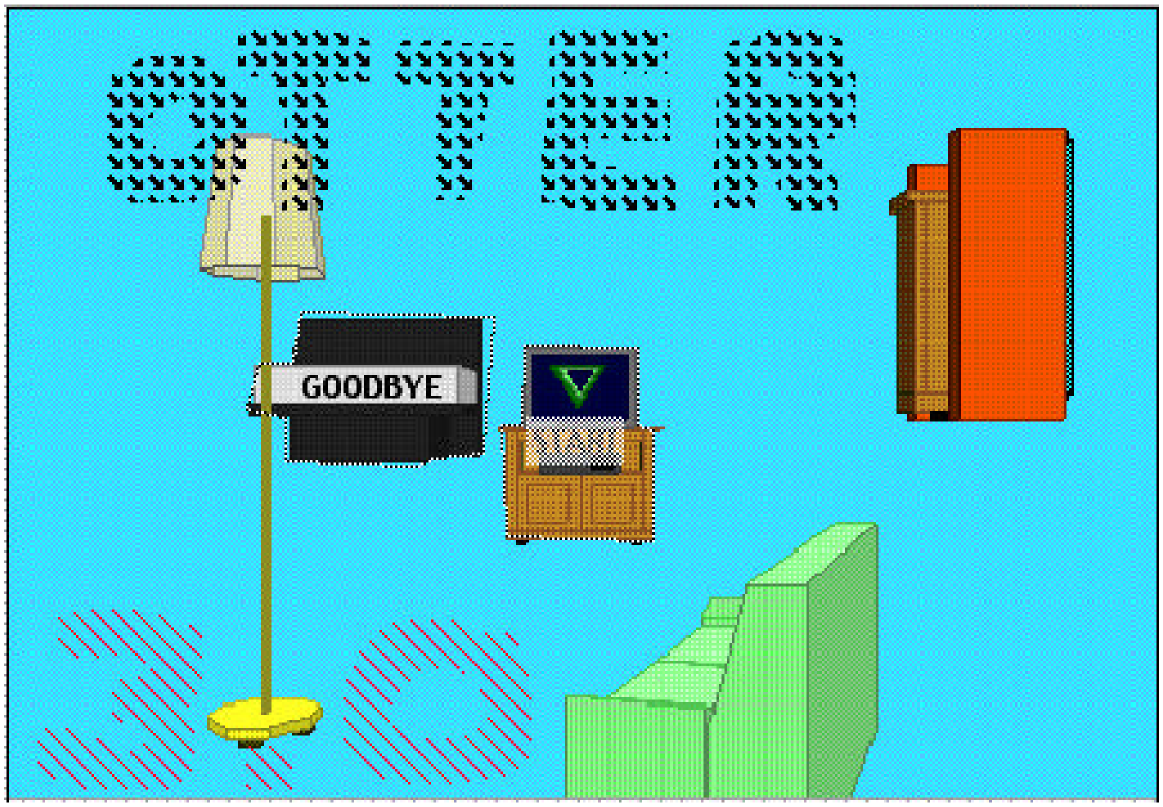


Figure 1

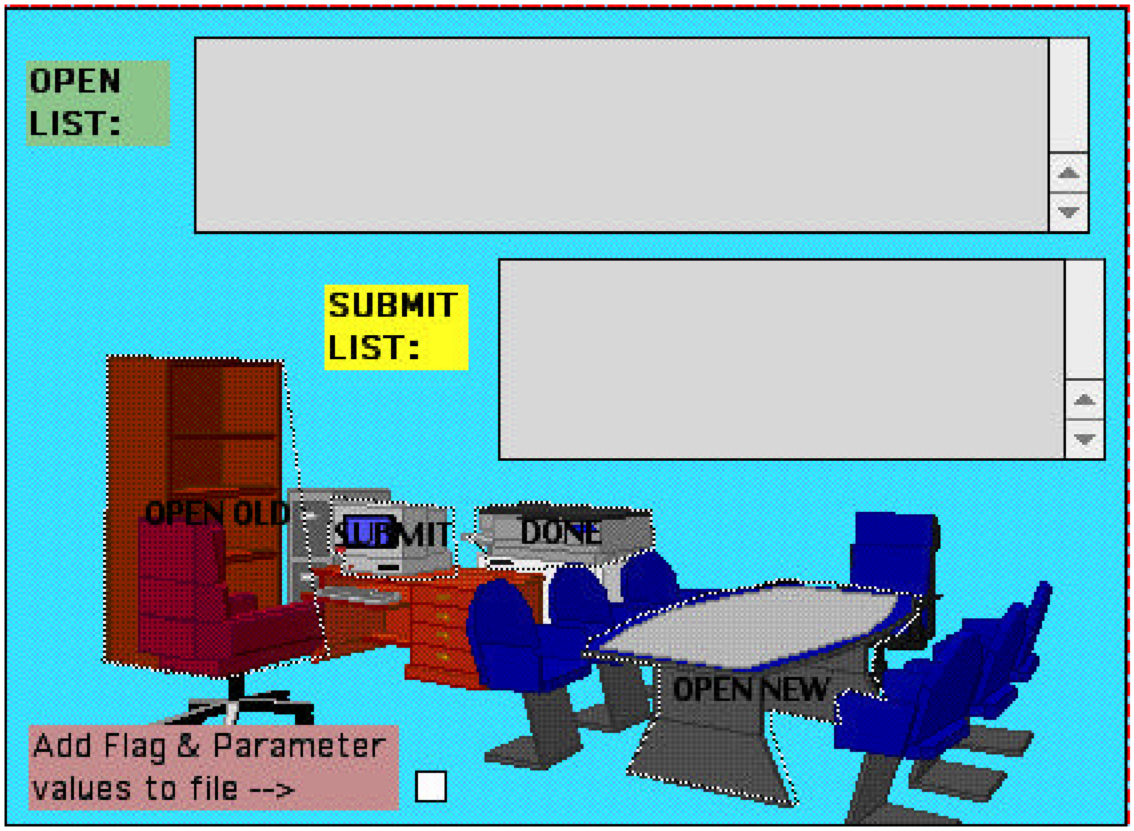
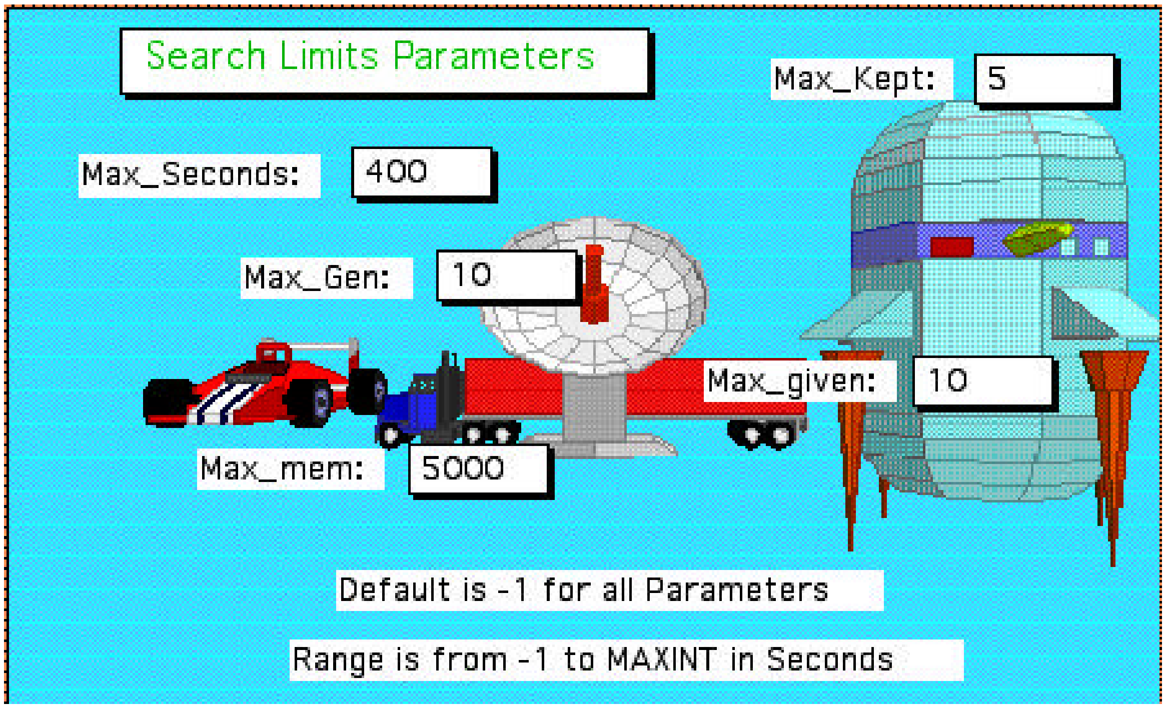


Figure 2



Search Limits Parameters

Max_Kept: 5

Max_Seconds: 400

Max_Gen: 10

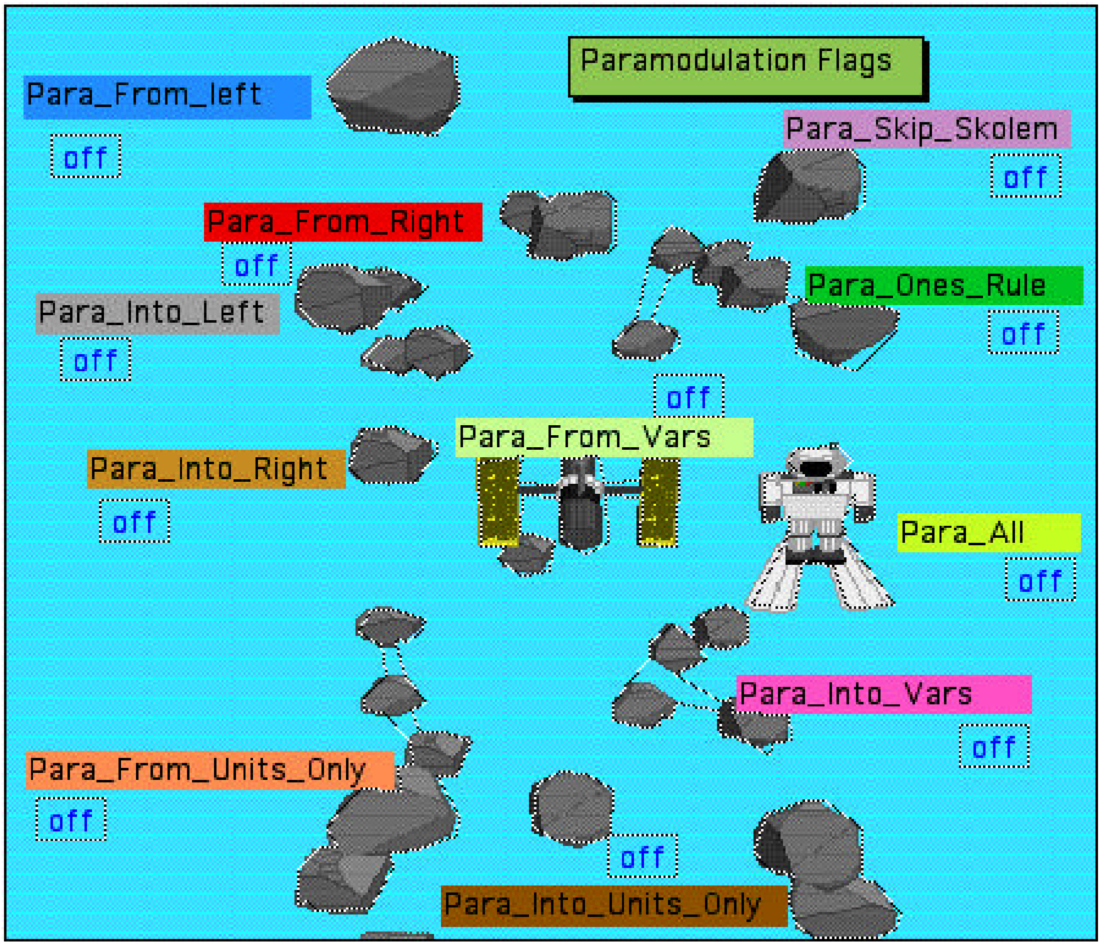
Max_given: 10

Max_mem: 5000

Default is -1 for all Parameters

Range is from -1 to MAXINT in Seconds

Figure 3



Paramodulation Flags

Para_From_left off

Para_Skip_Skolem off

Para_From_Right off

Para_Into_Left off

Para_Ones_Rule off

off

Para_From_Vars

Para_Into_Right off

Para_All off

Para_Into_Vars off

Para_From_Units_Only off

off

Para_Into_Units_Only

Figure 4

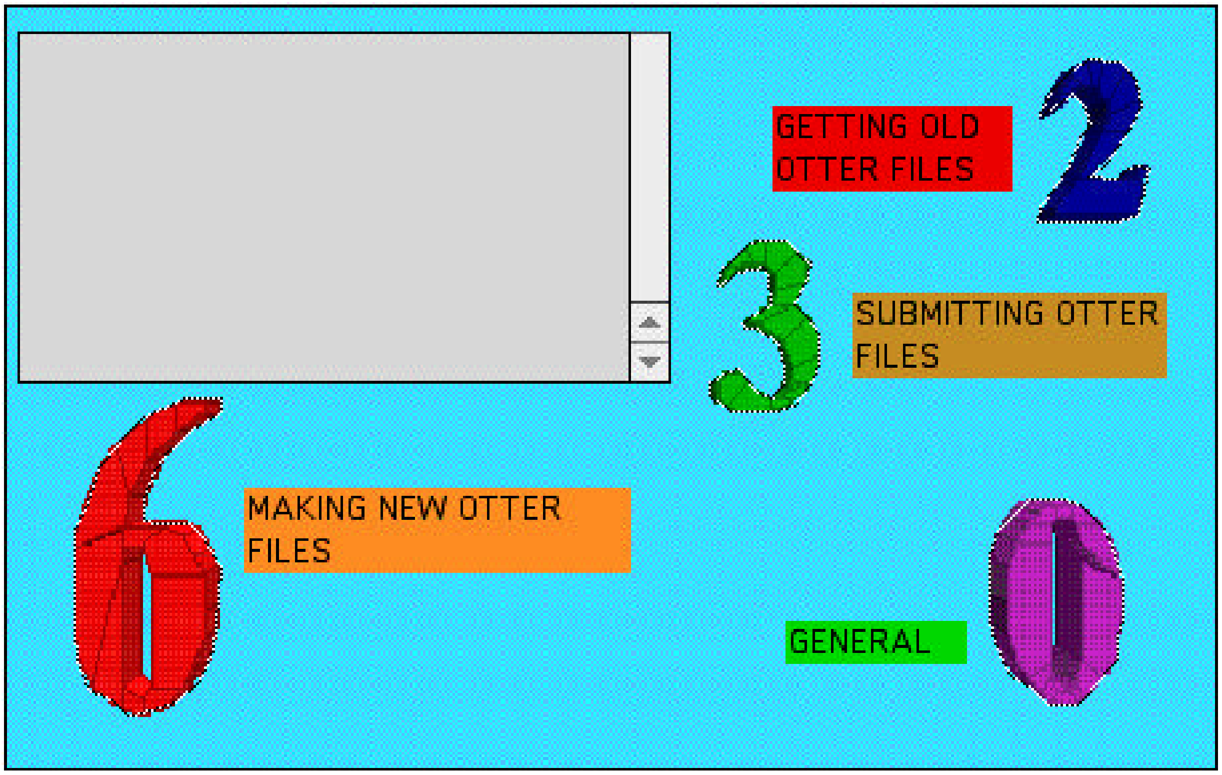


Figure 5

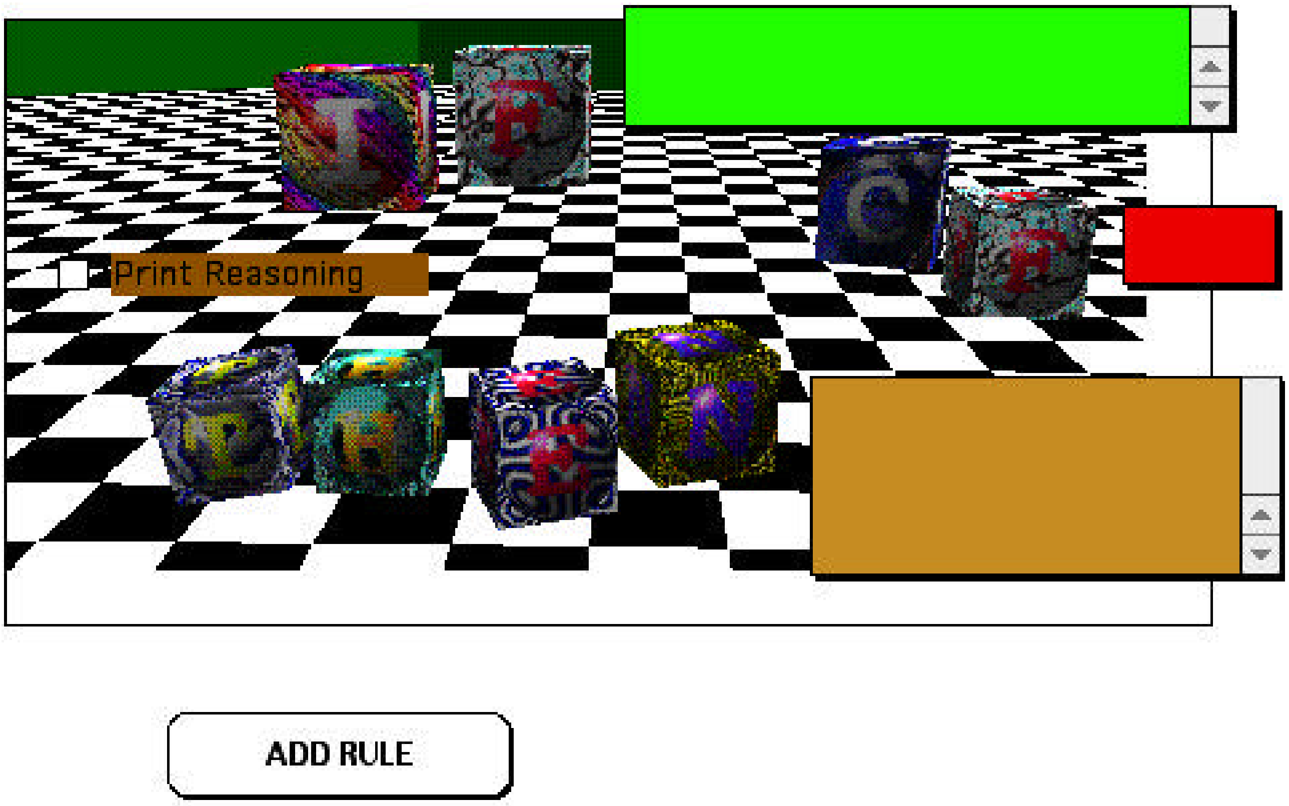


Figure 6