MacEuclid 1.1$_{β8}$
User's Guide

Bernard Bernstein

University of Colorado at Boulder

bernard@cs.colorado.edu

BERNARDB@applelink.apple.com

Introduction

MacEuclid is an easy-to-use program for creating, editing and analyzing reasoned arguments. When most individuals think of an "argument", they often expect several participants communicating with each other. In the case of Euclid, however, an argument is any type of reasoned discourse. It may be a single person presenting a case, it could be several people collaborating to create a single case, or it could be a group of people arguing against each other.

A newspaper editorial is a common "argument". One person writes it and a large audience reads it. The argument written in the newspaper is often one-sided, yet articles often reference other articles or books. By referencing others work, the author is effectively presenting several sides of the argument from the author's perspective.

An important feature of Euclid is its ability to

include "sources" with each statement made by the author. When writing an article, the author may make many statements which are actually made by other authors or domains. Since the author wrote the article, they are actually statements from the author's perspective, but with its source being the original author. This will be discussed further in this manual along with examples to clarify what this really is.
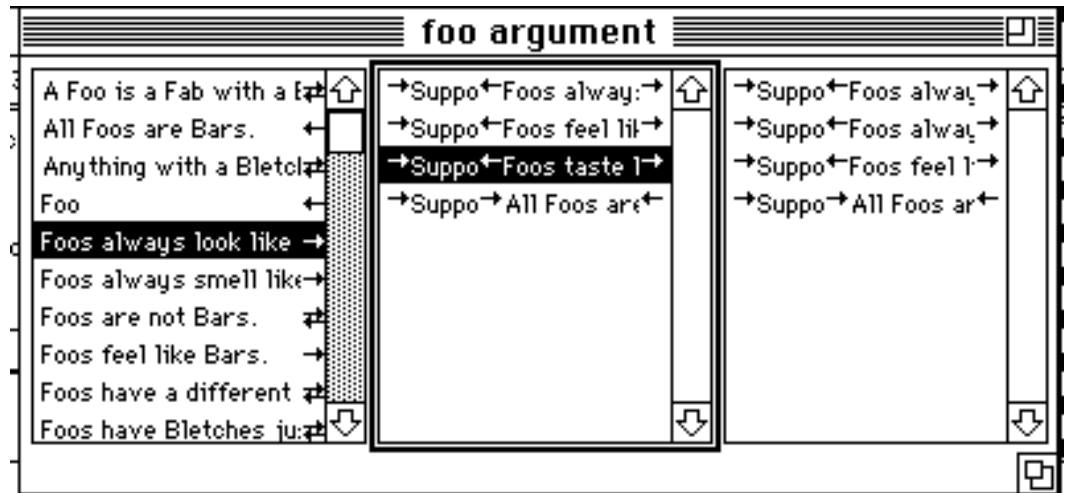
When an author writes an argument, there are various types of text that can be written. The author may make a "claim" to state a fact or a point that the reader might contest. Of course, the author wants the reader to adhere to all claims, so claims should be supported by "facts" or "evidence" or any other type of statement that the user may define.

If the author creates a claim and then creates some evidence to support it, the author will need to connect the two statements with a relation. The author may define a relation, "supports" which will meet that need. Some other relations that the author might use are "refutes", "refers to", "defines", etc.

Another useful construct available is that of "list" objects. A list of claims can be made to represent a complete argument (or sub-argument). Perhaps the author can construct a linear version of the argument for a written paper by creating a list of the statements in their order to appear in hardcopy. Of course, the user may define list types with any desirable name.

So far, I have only talked about the objects that Euclid uses. The structure of the canvas on which the argument is drawn will be discussed before moving on to some more complex functionality.

An argument is stored in a "database". This is a network database management system that can quickly find connections between objects in it. The front end for the database is simply a linear browser. When the user selects an object in the first column, the second column lists all relatives of that object. It is difficult to browse using this, so it is useful only to a small extent. Most people use displays to work on arguments.

Each database may have many displays. One may consider a display as a perspective of the database. A user may create several displays to represent the same argument with different layouts. The displays reference the database to get all the object contents, and in addition, store positioning information about each
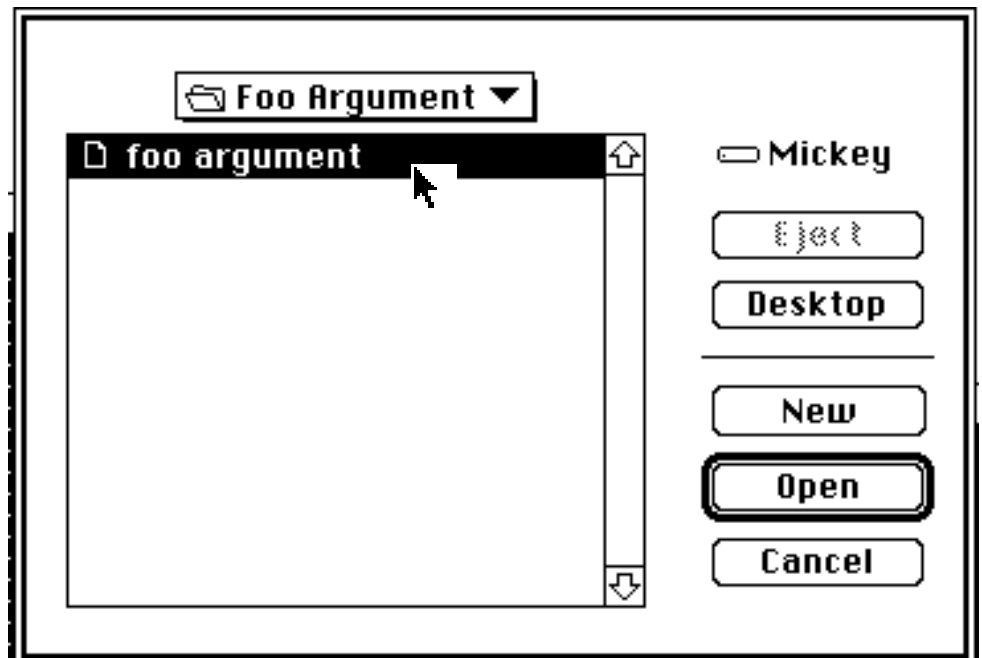
object. The displays each contain a subset of the database contents, so a user may split an argument into several displays, each containing an independent section of the argument.

One can consider the display to be the database browser and editor. As a browser, the display is a hypertext front-end. Users may hide objects from the display, or they may show objects from the database. There are several ways of getting information into a display. One way to get information from the database into a display is by clicking on an object in the database browser, "copying" it, and then "pasting" it into the display. If objects are already in the display, then the database may be "navigated" by following the path of relatives from the displayed objects. If a list object is in the display, then a user may display items in that list.
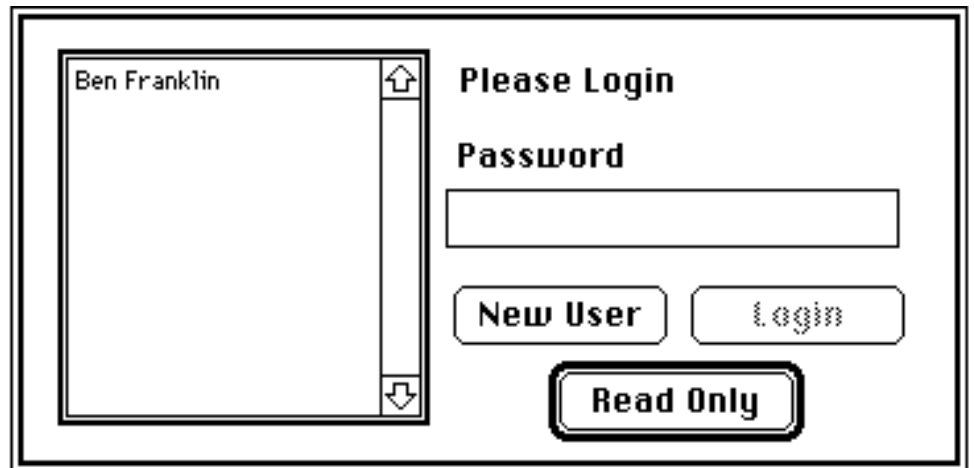
One very important use for list objects is for making queries. If a user writes a large argument and then the person would like to find some of the objects, they may form a query to find them. The author may find all supported claims that contain the word "very". The program puts the results of a query into a new list object. For complex queries, one may search a list to narrow its contents, and another list is created with the results from the new query.

Starting Euclid

To launch Euclid, just double-click on its icon. This will start the program without any data argument. If a file already exists, then that database file may be double-clicked. Launching a display file will result in a message asking for a database to be loaded first. For this tutorial, launch the application directly, so that the program will prompt you for a database file.

When the user launches Euclid, the program will ask the user to open a database or create a new one. To get a feel for how Euclid works, I recommend working with an existing argument. This program includes a simple argument called "foo argument".

The next step is logging into the system. One drawback of having a system that allows multiple users (on a single user machine), is that the program needs to know which user is doing the work during this session.

If the user is simply reading an argument without plans to add to it, the user may log in "read only" which means, the user may only read what has already been written. Create a new user so that you can use add your own information to the database.

Name

Abbreviation

Description

Password

Repeat Password

Cancel   Add User

When you add your name to the user-base, you need to provide an abbreviation along with your name. By convention, the abbreviation for a user should be lowercase letters so as not to confuse it with sources. A password is optional, but if you are going to pass the argument on to other users, then it is recommended. If the argument is only for yourself, then it saves time later if you do not need to type a password every time you open the argument. You may change this information later, so don't be too concerned about what you type here.
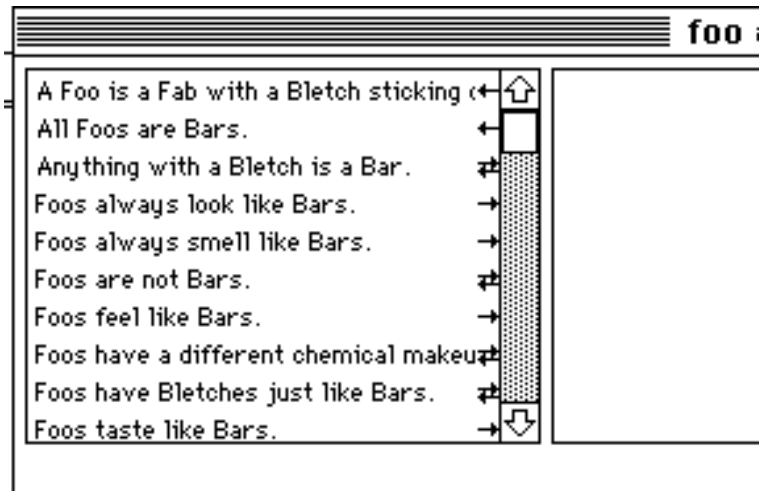
The abbreviation that you provide will be (optionally) displayed above each object that you create. The abbreviation should be up to four characters long, so your initials will suffice unless another user has the same initials. If you are planning on passing the argument to other users, then contact information, like a phone number or network address would be useful in the description box.
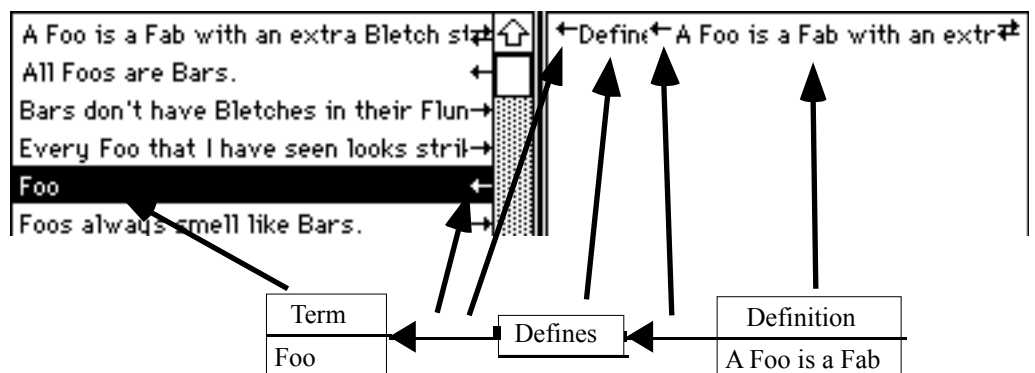
Browsing the Database

After you click "Add User", you will be logged in as that user. Now the program will display the database as a browser. The program only uses the first column of the database browser at first. This column will contain every object in the database. The browser lists the sorted text objects before the relation and list objects. The arrows on the right side of the column signify the direction of connections to each of the objects. A right arrow represents relations connected to the object through an outward link. The browser displays a left arrow if the relation is connecting into the

object. For example, if A defines B, then A will have an outward link representing the connection to "… defines B", and B will have an inward link representing "A defines …".

Click on the object named "Foo". In the next column you will see the connection to the relation which "defines" foo. The arrow to the right of "Foo" means that the object connected to it has an outward arrow connecting it to "Foo". In the next column, "Defines" has a left arrow connecting it with "Foo" with an outward link, and another left arrow representing a link connecting into the definition from the definition itself. If you ever need to start the browser from this state, you may choose **Refresh Browser** from the File menu. This brings the browser in its initial state, with the entire database in the first column.

This illustration should clarify the mapping between the arrows in the database browser and the connections between the objects. The objects on the bottom half of this illustration is your first glimpse of what the display objects will look like.

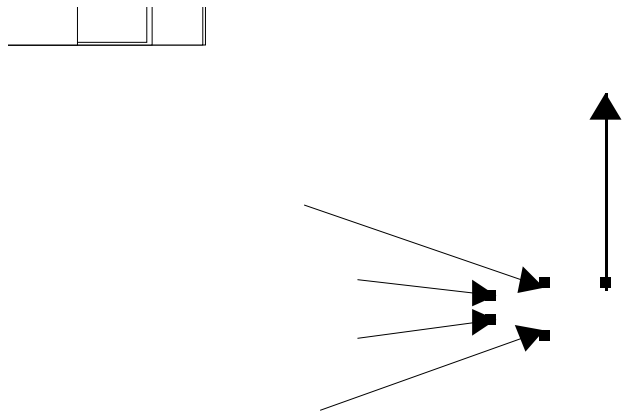Feel free to explore the database using the browser. Since the

browser has limited horizontal space, you will not be able to read many complete objects, so much of the meaning of the claims will be missing. When you click on an object in the last column, the first column will scroll off the screen. Since circular

connections are quite common, it would not be possible to save all previously viewed columns for returning to, so use Refresh Browser to restore the first column.
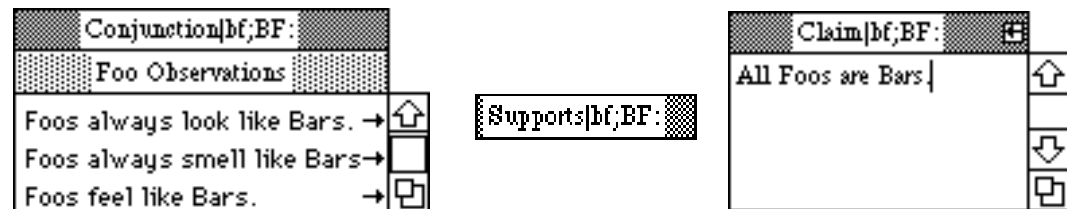
Browsing the Display

Now that you have seen what the database looks like, try moving on to a display. Although only one database may be open at a time, you may have many displays open. The displays always stay coordinated with changes made in the database. If you change the text of an object in one display, the text will become updated in all displays where that object is visible.
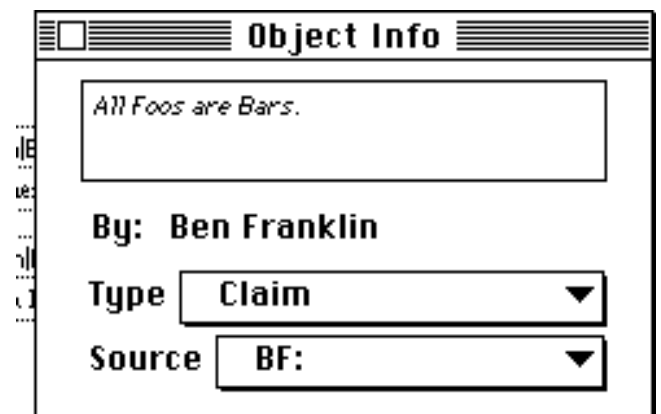
Open a display by choosing **Open Display…** in the File menu. When you are looking at objects, be aware of the top bar above the text. The information in the label is the type of object, the creator and the source. Choosing **Preferences…** in the Edit menu allows you to decide which pieces of information the display shows on each object.

If the selection contains only one object, then it becomes the *active* object. If this object is a text object, then you may edit the text in it or scroll through its contents. If the object is a list object, then you may perform commands on the list objects such as **Show List Members**.

MacEuclid 1.1b7 User's Guide

Introduction

These are three examples of active objects. An object can only be active if it is the only selected object. In the above illustration, all objects are displaying their type, creator and source. The type "Claim" is a text type. The creator of all three objects is "bf", which is the abbreviation for Ben Franklin. The source of all three objects is "BF:" that is an abbreviation for "Ben Franklin says…". We call the box containing this information the "label bar".

The list object has a title in addition to its label. Any object may have a title, but it is optional. The user may change the title of an object, along with other parameters, by double-clicking on the object. When every object is created, it has a default name containing the text in the object. To change the name of an object, select the box in the "object info" window containing the default name and type in a new name. If the user deletes the entire name, then the program substitutes the default name.

By selecting a rectangle around a group of objects or by holding the shift key down while selecting several objects, you may select more than one object. The group of objects that are selected become the "selection". Almost all operations that one performs on individual objects may also be performed on the selection. Selected

objects have a dark box around them to distinguish them from the non-selected objects.

The final component of the display objects is a small arrow that may appear on the right side of the label bar. If there is a right arrow there, then there are "outward" links connecting this object with others that are not visible on this display. A left arrow represents "inward" links. If both arrows are there, then inward and outward links exist in the database that are not in the current display. Clicking on the arrow will cause those missing links to be displayed. This operation is the same as **Show All Relatives** from the Hypertext menu. When you ask to show all relatives, the search traverses two levels of links. Two levels of links is equivalent to one level of relations. For example, if *A supports B*, then if all relatives of *A* are requested, then one level of links connect *supports* and another connect it to *B*.

Creating Objects

You may begin creating objects at any time by choosing **New Text Object**, **New Relation Object** or **New List Object** from the Objects menu. When one creates a new object, the creator is set to the logged in user. The source and type are set to a predetermined default. To change the default source, text type, relation type and list type, the user may change the settings in the Types menu. The Types menu is also used to define new types. This will be described in more detail later.

New Text Object creates a new object with a mini editor. The editor will allow changing of text font, style and size for any selection of text. The styles of text are saved in the database, not in the display, so a change in one display is reflected in all displays of that object.

New Relation Object creates a new relation. The relation that one creates does not have any links to other objects until the user creates them. The quickest way to create a relation between two objects is by holding down the option key while dragging from one object to another. If both ends of the drag are not relations, then a
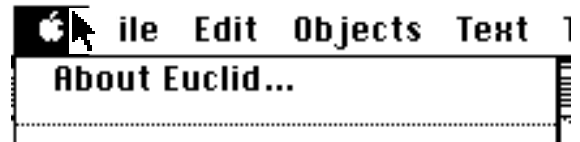
new relation is created between them and then links are created connecting the relation to its endpoints. If one wants to create a relation connecting a relation to another object, then one needs to create the relation first and then drag lines between the objects using the option-drag routine.

New List Object creates a list object containing the selected objects. First you would select the objects to put in the list and then choose this command to create a list with those objects. A list object behaves like a text object in that it has a label and a content. The objects in the list object are called its "members". Another method of creating list objects is by choosing **Query Database…** from the Hypertext menu. This allows you to choose what to put in the list by forming an expression.

Reference

This section describes every operation available in Euclid. First each menu item is described, then each mouse and keyboard gesture is described.

Apple Menu

About Euclid… provides the version number of this release and gives credit for the programs supporters.

File Menu

```
 File  dit  Objects  Te
 New Display      ⌘N
 Open Display...  ⌘O

 Close            ⌘W
 Save Display     ⌘S
 Save Display As...
 Revert to Saved

 New Database...
 Open Database...

 Close Database
 Save Database
 Save Database As...

 Refresh Browser
 Merge Database...

 Page Setup...
 Print...

 Quit             ⌘Q
```

New Display creates a new empty display. The display will be associated with the current database. Subsequent openings of this display must have the same database open. This command is only available when a database is open.

Open Display… prompts the user to choose a display that already exists. The display that is chosen must have been created with the current database. This command is only available when a database is open.

Close closes the frontmost window. If the window is the database browser, then this command is passed to all open displays before closing the database. If there have been changes made in the
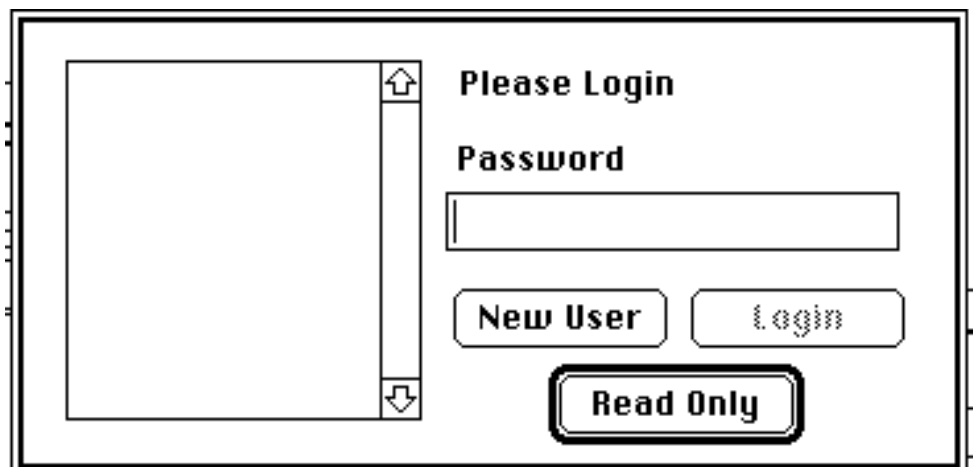
window, then program asks the user if changes should be saved.

Save Display saves the frontmost display if there have been any changes to it since it was last saved. If the display is new and has not been named yet, then this is identical to Save Display As…. If the database has changes in it at the time the display is saved, it will be saved also. This is a precaution because the database is where all the data from the display is stored, so the program needs to be sure that its database is as up-to-date as its latest display.
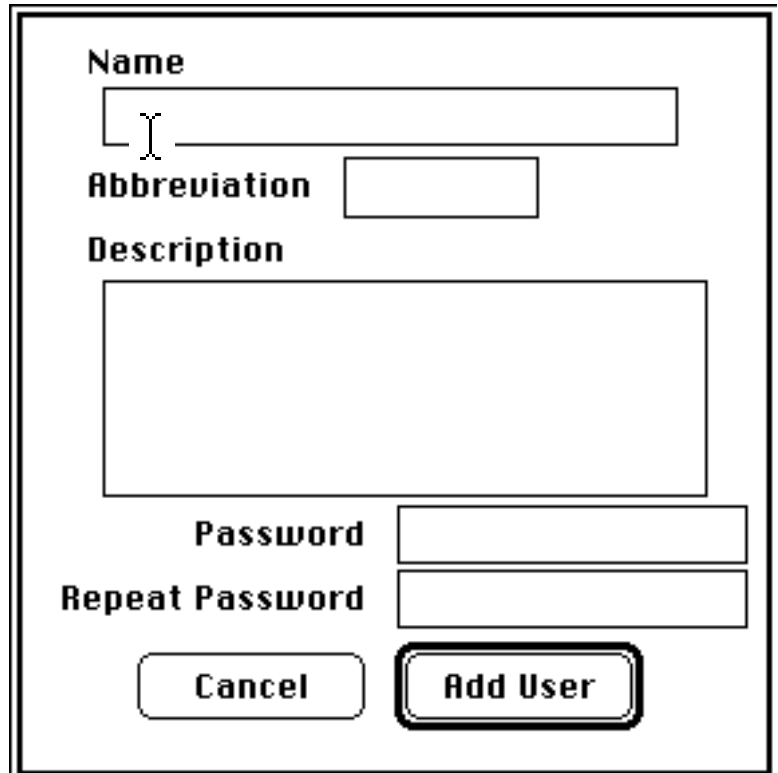
Save Display As… prompts the user for a new name for the display and then saves the current state of it to the specified file.

Revert to Saved discards any changes made since the display was last saved and refreshes the display to its state before it was saved.

New Database… creates a new database from scratch. It asks the user to name the new database and for the new user to log in. This is the command that is selected when the program launches without a database.

After the user names the new database, this dialog box prompts for the user to log in. **Read Only** would be useless since a new database contains nothing to "read". Since this is a new database, **New User** should be chosen. The user will then fill out some information about him/herself.

The user provides a name that will be used as an identity for logging in. The abbreviation that the user provides should be a short identifier to be displayed in the label bar of objects and by convention, it should be lower case. The users' initials or a short "nickname" may be used here. The description can hold any additional information that the user may want to list. If many people will be using the database, then the description could hold contact information for the user.
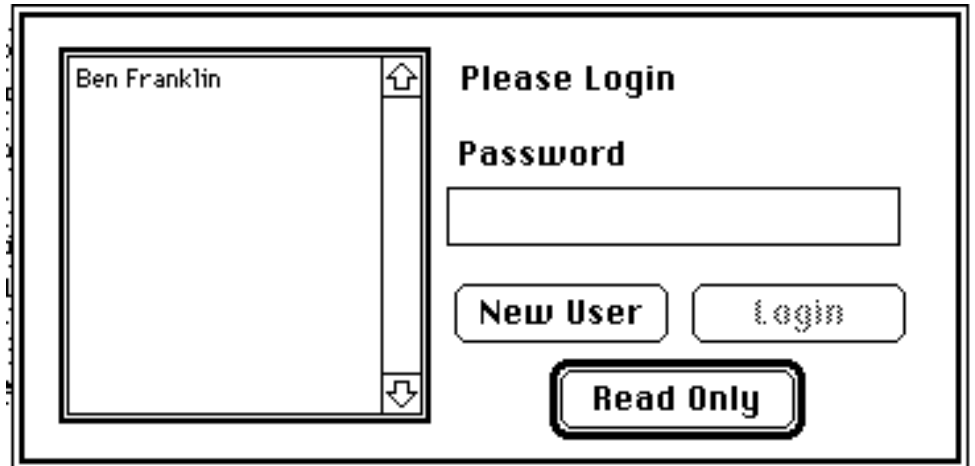
The password field does not echo the characters typed in. A password is optional, but if it is given, then it must be typed in twice. This is a safeguard against spelling errors.

When all the information is given, click **Add User** and the information will be stored in the database. Next, the program presents an empty database browser and the user will be ready to begin work on the new database.

Open Database… opens an existing database. It prompts the user

to choose the database to open. This is the command that is selected when the program launches with a database.

After the user has selected the database, this dialog asks the user to log into it. The user selects his/her name, and if there is a password, the user types it in the password field. When the user has typed the proper password, the **Login** command becomes available and the default.
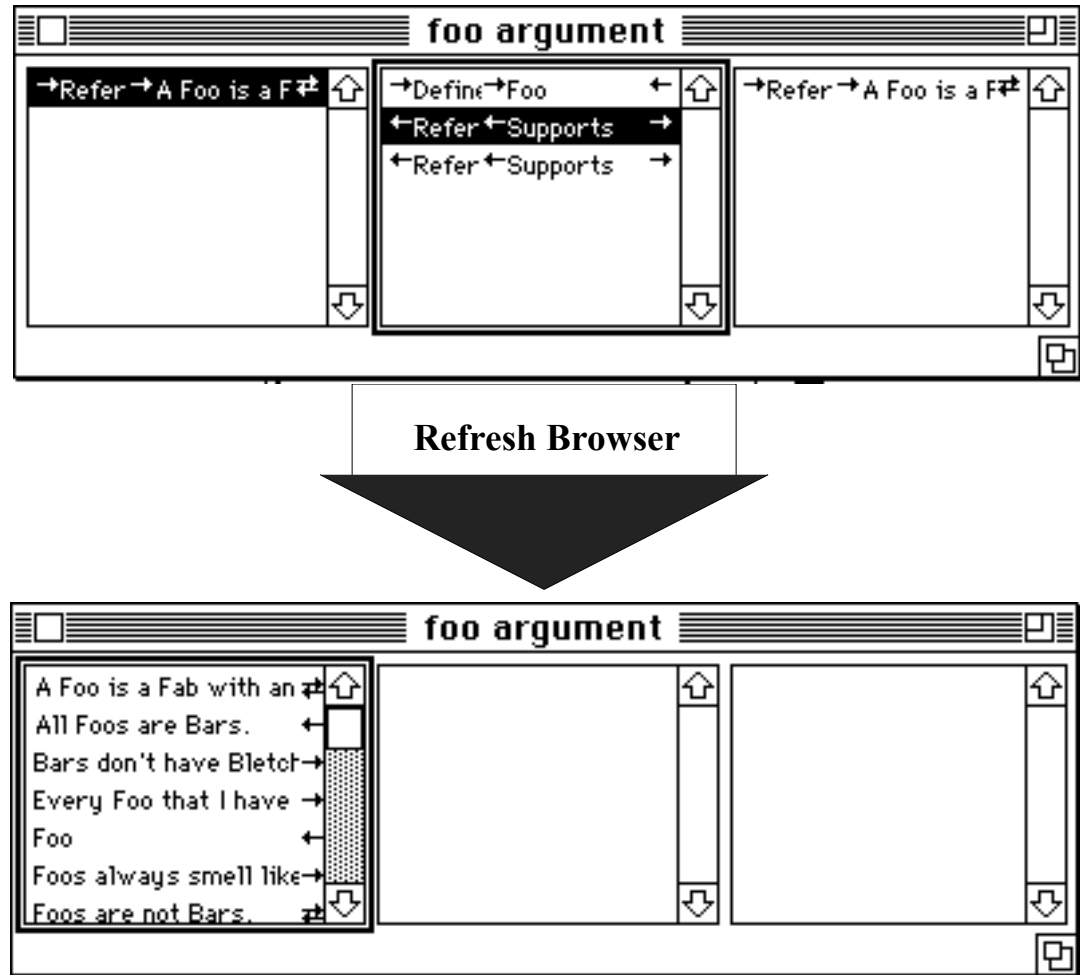
If the user chooses **Read Only**, then several menu options in the program become unavailable. The user may not add objects or edit the objects that exist in the database. The database may not change at all while in read-only mode. The user may, however, create displays and layout the database. Objects may be manipulated, as long as the contents of the objects do not change.

Close Database closes the current database. It also closes all displays that are using the database. After this command is done, the user may only open a database or create a new one.

Save Database saves the current database.

Save Database As… prompts the user for a name for the database and saves it using that new name.

Refresh Browser puts the database browser into its initial state. It sorts the database objects and puts that sorted list into the first column of the database browser.

After the user has scrolled off the right side of the browser, this is the only way to restore its initial state.

Merge Database… merges another database into the current one. All the objects in the other database are transferred into the open database. If there is a conflict between objects types or sources, then the most recently modified one is used.

Page Setup… is used to set various attributes about the printing of the windows. The contents of the dialog box given depends on the printer that is being used.

Print… prints the contents of the frontmost window. If the window is a display window, then the content of the display is printed. When printing, the page will not show the state of the
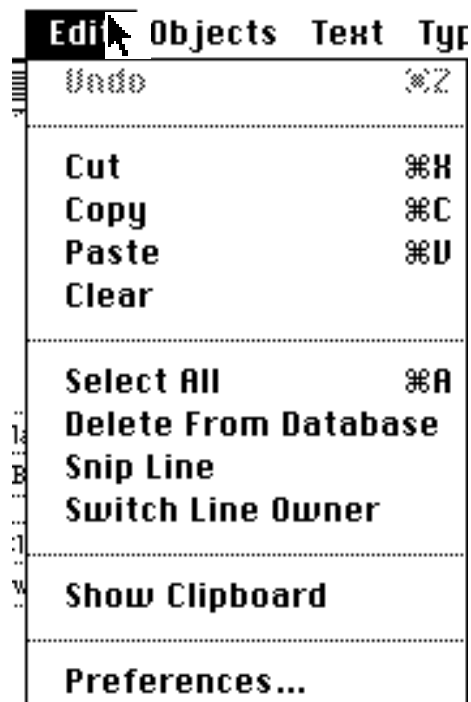
selected or active object. It also does not print the arrows in the title bars. The page breaks are visible in the display as gray

lines extending around each page. These lines can not be removed and do not interfere with the contents of the displays. They do not appear on the printed version. Page numbers and file information is printed at the top of each page in the default font and style.

Quit exits the program. If there are any windows with data that has changed, the user will be asked to verify if the program should save the changes first.

Edit Menu

Undo restores the state of the last edited object to its state before the editing occurred.

Cut puts a copy of the current selection into the clipboard and then clears the object. More information about Copy and Clear are given in their respective command descriptions.

Copy puts a copy of the current selection into the clipboard. The clipboard can hold selected text, database objects, pictures or display objects. Copy can be performed on a selection of objects in

a display, or on a selection or text in an object, or on an object in the database browser.

When a group of objects in a display is copied, it may be Pasted into another display or into a program that can read the standard 'PICT' or 'TEXT' format, such as a drawing program or word processor.

The PICT version of a selection includes all lines between the objects but does not include the boxes indicating selection nor the arrows in the label bar.

The TEXT version of the selection includes the complete contents of the objects. Text objects contain their text, relations contain their name (or type) and list objects contain the text of their members. A list object has start and end delimiters to separator the members of the list from the other objects.

When one copies an object from the database browser, it may be pasted into a display. This creates a new display object of the appropriate type and fills the content from the database.

Paste enters the data that was previously put into the clipboard from Cut or Copy.

Clear removes the selection. If the selection is text, then clear simply deletes it. If the selection is a group of objects in a display, then clear does the same as **Hide Object** from the Hypertext menu. Hiding objects removes them from the display without affecting their database representation.

Select All puts all objects in the current display into the selection. If a text object is active, then it selects all the text for that object.

Delete From Database deletes the selected objects from the database. This is irreversible and should only be used if the objects are never to return. This is not the same as Clear that simply removes the objects from the display. This is the only way to delete objects from the database. The object will also disappear from all other displays. If a display is opened which contained a deleted object, the user will be alerted to that fact when it is loaded.
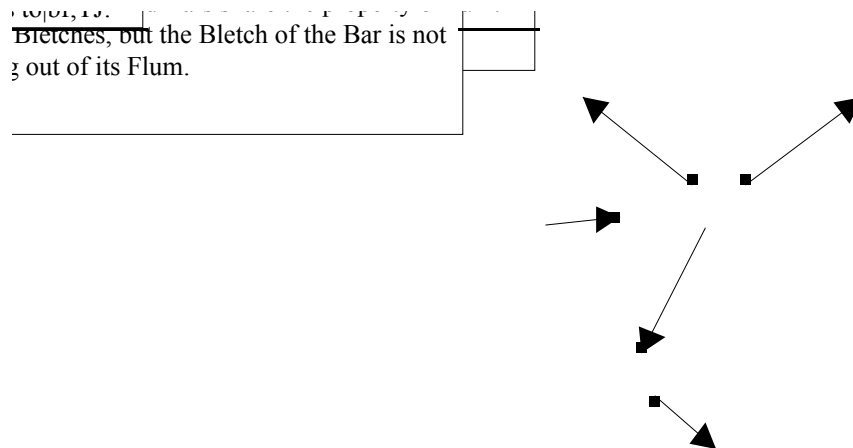
Introduction

Snip Line is used to delete lines. The cursor will become a pair or scissors and when it is over a line, will look like they have closed. The user may click the mouse when it is over a line and the line will be deleted. This deletes the line from the database. There is no way to have a pair or related objects in a display

without showing the line connecting them, so anytime a line is deleted, it is also removed from all other displays.

Switch Line Owner changes the owner of a line. A small box appears on one end of every line to represent the ownership of the line. The owner is always a relation. The relation that "owns" a line is the one that the line represents. This is necessary because relations can be "related" to each other.

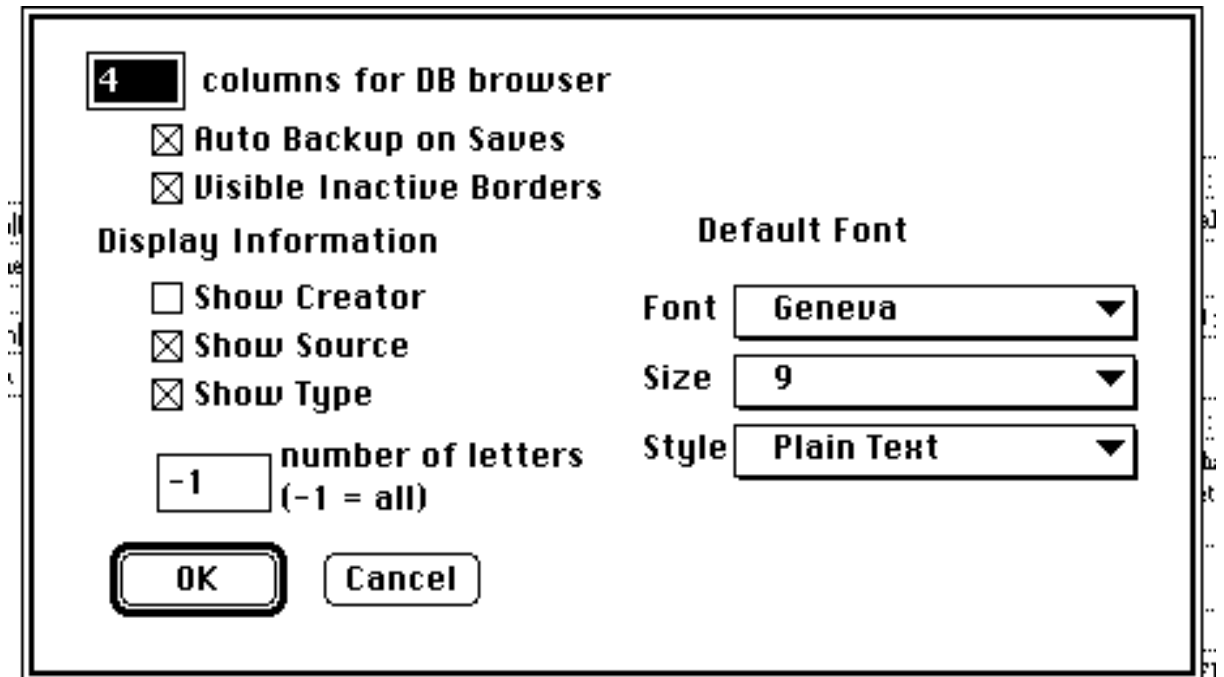Bletches, but the Bletch of the Bar is not
out of its Flum.

In this example, one claim is supporting another and the relation between them is referring to a definition that warrants the support. If there were no concept of ownership, then a reader would not be able to distinguish between the two arrows leaving the supports relation. The *supports* is connecting the two claims and the *Refers to* is connecting the *supports* to the definition.

When the user creates a line, the owner always goes to the relation if the other end is another object, but in ambiguous cases, the program may guess wrong. This operation switches the ownership to the other end of the line. The operation works similarly to the snip line operation. The cursor is an arrow with a box on one end. When the cursor is over a line, the box switches to the other end. That is when the user may press the button to do the operation.

Show Clipboard opens a window that shows the contents of the clipboard. The clipboard contains the last object that was cut or copied.

Preferences… allows the user to change several attributes of the program.

Number of columns for DB browser is the number of columns that the browser should have. Smaller numbers allow wider columns, but more columns allow more depth of browsing to be visible.

Auto Backup on Saves means whenever the database is saved, a backup of the same database is also saved. The program gives the backup file the name of the original file with "backup" appended to it. This file is overwritten each time the program saves the database. This could be used as a safeguard against disk failures. Having two copies of the file are on a disk double the chances of recovering the file when the disk fails crashes. An extra safeguard for file safety is a temporary file that is saved every time the database is saved. The temporary file only exists while the database is being saved, and contains a copy of the previous version of the database. If there is an error while the file is being
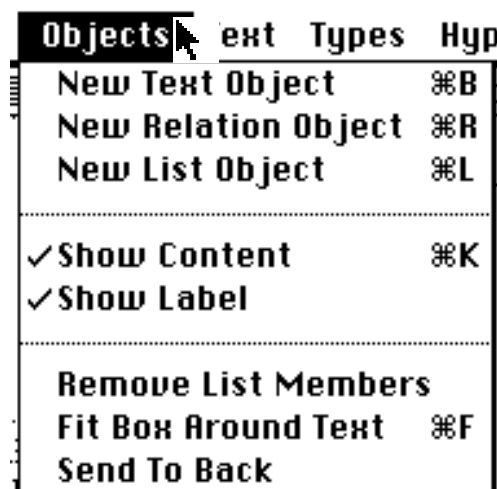
saved, then the user may need to revert to the previous version that is stored in the temporary file.

Visible Inactive Borders is whether or not the borders of objects should be shown when the objects in the display are inactive. This may be removed from subsequent versions of the program. It is recommended that it remain on. This may be turned off if the user does not like having boxes around the objects.

Display Information allows the contents of the label bar to be changed. The creator, source and type may be shown or hidden. If the type is shown, then the user may decide the number of characters to show of the types. If the user wants to see the entire type names, then -1 may be given.

Default Font sets the font that is used in the label bar as well as the default font in the text objects in the displays.

Objects Menu

New Text Object creates a new empty text object in the database and in the current display. The type is set to the default text object type and the source is set to the default source. The creator of the object is the current user.

New Relation Object creates a new relation. The type is set to the default relation type and the source is set to the default source.

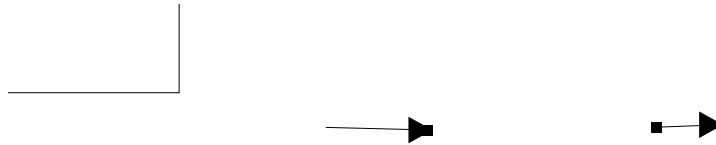Lines may be drawn from this relation to other objects by holding

down the option key and dragging from the start object to the relation or from the relation to another object.

A quicker way of creating a relation between two non-relation objects is by option-dragging from the start object to the other. This gesture creates a new relation and connects the lines from the start object to the relation and from the relation to the end object.

New List Object creates a new list object containing the current selection. The type is set to the default relation type and the source is set to the default source. If the user wishes to associate a set of objects, then a list type may be created to represent the association and a list may be created containing those objects.

Show Content toggles the displaying of the content of the selected objects. The content is the text of the text objects and the list in the list objects. When the content is hidden, the name of the object is displayed regardless of whether it is the default name.

Above are objects with the content shown. Below are the same objects with their contents hidden.

If an object contains a name other than its default, then the content is simply hidden. If the name is the default, then it is used below the label bar.

Show Label toggles the display of the label bar of the selected object.
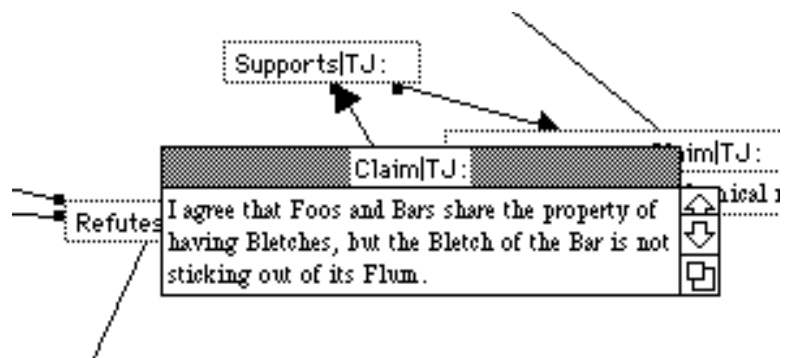
Above are the same three objects with the label not being shown but the content shown. Below the objects have neither the content nor the label shown.
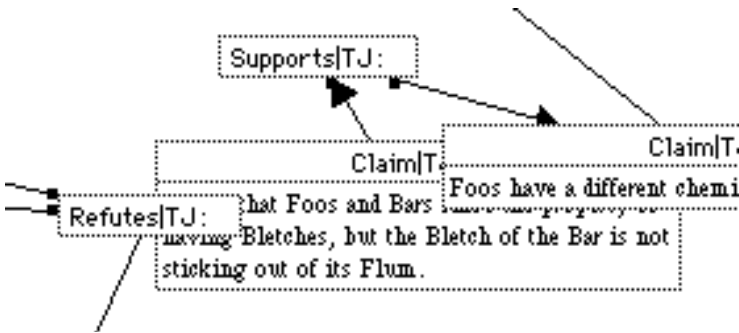
Remove List Members removes selected members from a list. The change takes place in the database as well as in the display. Several items in the list may be selected by holding the shift key while selecting a range of items or by holding the option key while selecting individual items. This command is only active when a list is the active object and items are selected in the list.

Fit Box Around Text resizes the objects in the selection so that the boundaries of the boxes containing the text are large enough to display the entire text. The box changes vertically always and if the object contains only one line of text, then it will also shrink horizontally/
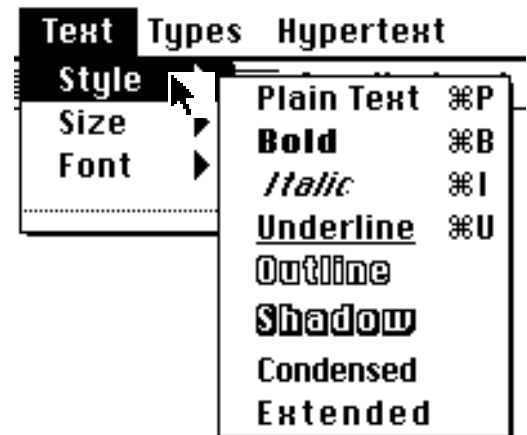
Send To Back moves the selection behind any other objects which is overlapping. If an object is covering other objects, then this command will move it so that the other objects are "on top of" it, obscuring parts of it.
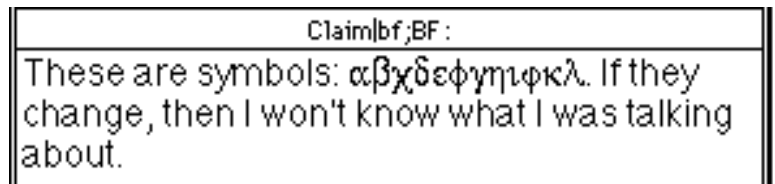
Above is an object obscuring some others. After Send To Back is used, the other objects are above the original one.
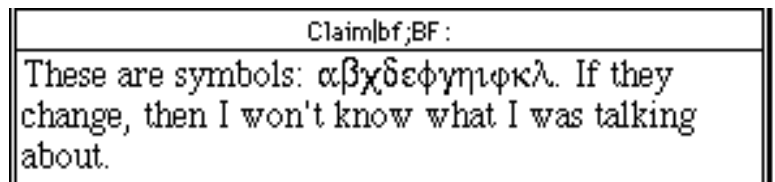
MacEuclid 1.1b7 User's Guide

Supports|TJ:

Claim|T

Claim|T

Refutes|TJ:

hat Foos and Bars

Foos have a different chemi

having Bletches, but the Bletch of the Bar is not

sticking out of its Flum.

Introduction

Text Menu

Text Types Hypertext
Style   Plain Text   ⌘P
Size   **Bold**   ⌘B
Font   *Italic*   ⌘I
             Underline   ⌘U
             Outline
             Shadow
             Condensed
             Extended

Font changes the font of the current text selection in the active text object. Text objects have styleable text. Any selection of text may have a different font, size and style. If more than one object is in the current selection, then all selected text objects will have their fonts changed to the selected font.

Claim|bf;BF :
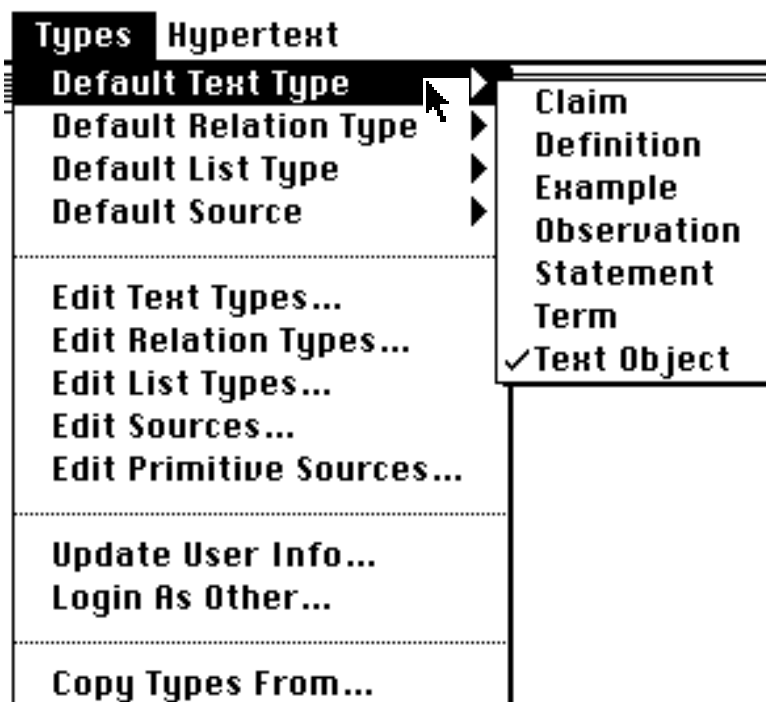These are symbols: αβχδεφγηιφκλ. If they change, then I won't know what I was talking about.

Above is an object where the main text font is Helvetica. If it is part of a selection whose font changes to Times, then the object will look like the one below. If the option key is held down while the menu item is selected, then the symbols in the text will change to the font along with the other characters. The main font is determined by the font of the first character in the object.

Claim|bf;BF :
These are symbols: αβχδεφγηιφκλ. If they change, then I won't know what I was talking about.

Size changes the size of the current text selection in the active text object. It changes the size of every character in the selection or the selected objects.

Style changes the style of the current text selection in the active text object. It changes the style of every character in the selection or the selected objects.

Types Menu

```
 Types  Hypertext
 Default Text Type        ►
 Default Relation Type    ►    Claim
 Default List Type        ►    Definition
 Default Source           ►    Example
                               Observation
 ..........................    Statement
 Edit Text Types...            Term
 Edit Relation Types...      ✓ Text Object
 Edit List Types...
 Edit Sources...
 Edit Primitive Sources...
 ..........................
 Update User Info...
 Login As Other...
 ..........................
 Copy Types From...
```

Default Text Type changes the type of the next new text object to be created. The checked type is the current default. The text types are defined in **Edit Text Types…**.

Default Relation Type changes the type of the next new relation object to be created. The checked type is the current default. The relation types are defined in **Edit Relation Types…**.

Default List Type changes the type of the next new list object to be created. The checked type is the current default. The list types are defined in **Edit List Types…**.

Default Source changes the source of the next new object to be created. A source is associated with every object. The checked source is the current default.

Sources are expressions consisting of a set of "primitive sources". A primitive source is a person or perspective that may make a statement. A source is an expression consisting of "a says x" where

x may be another source or an object. A more complex source may be "a says b says c says x".

In this menu, the abbreviation of sources are displayed. The expression "a:b:c:" is the shorthand version of the above example of a complex source. When primitives are created, a basic source is created automatically. The source "a:" is equivalent to "a says x", where x is the object.

Sources and primitives are defined in **Edit Sources…** and **Edit Primitive Sources…**.

Edit Text Types… allows the user to add, modify or delete text object types. The name is what is used to identify this type in all menus. The description can contain relevant information about the type.
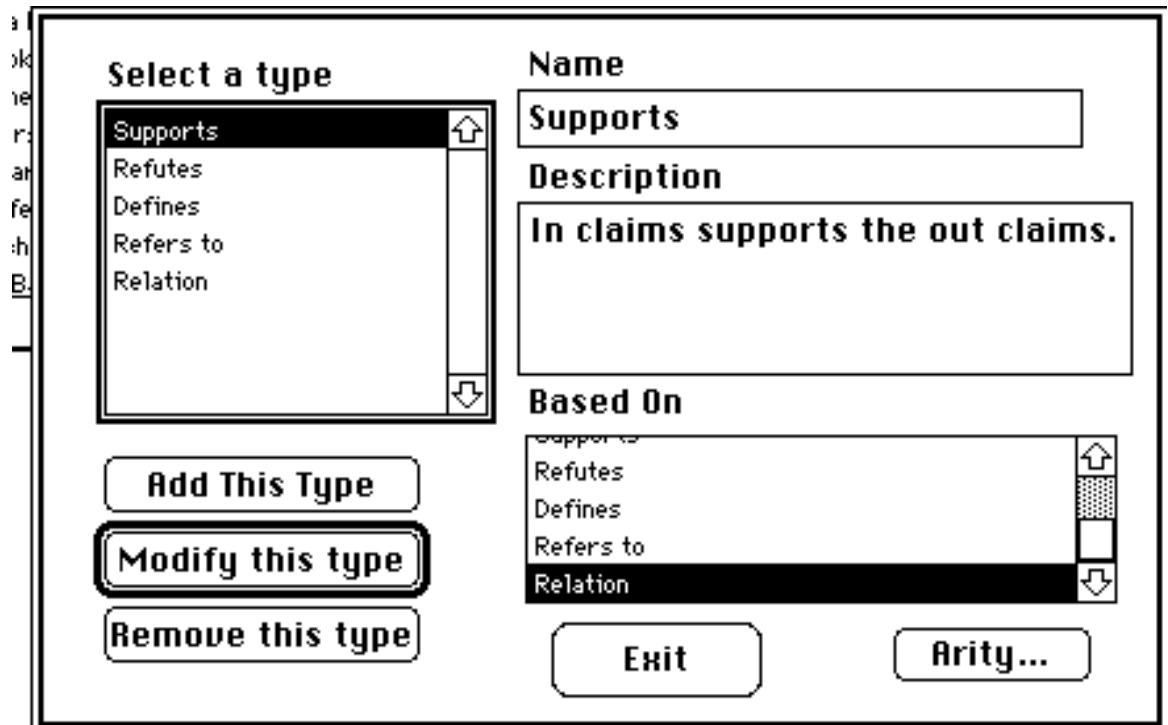

All types are defined hierarchically with multiple inheritance. In the Based On field, the user chooses the parent types that this object should be based on. The top of this hierarchy is "Text Object" that is the basic type of text object.
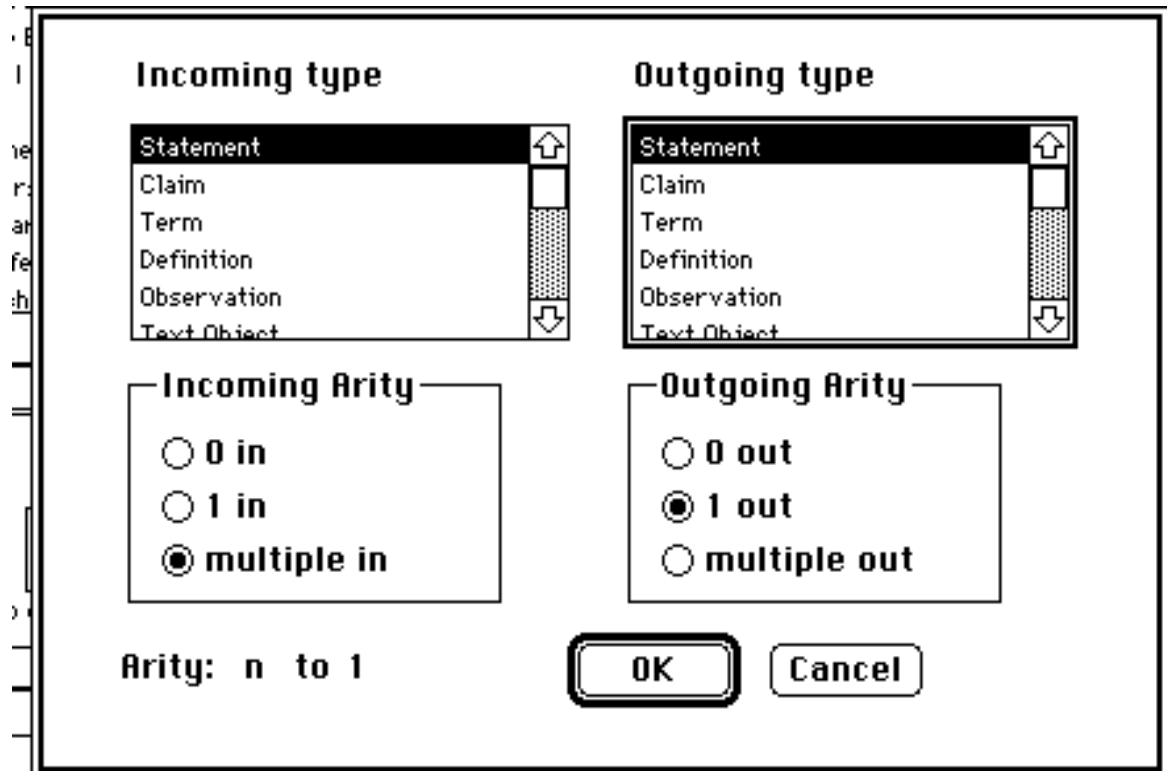
If the user wants to have a type that is based on more than one type, hold the command (apple) key down while selecting the

types in the Based On field.

Edit Relation Types… allows the user to add, modify or delete relation object types. The program uses the name to identify this type in all menus. The description can contain relevant information about the type.
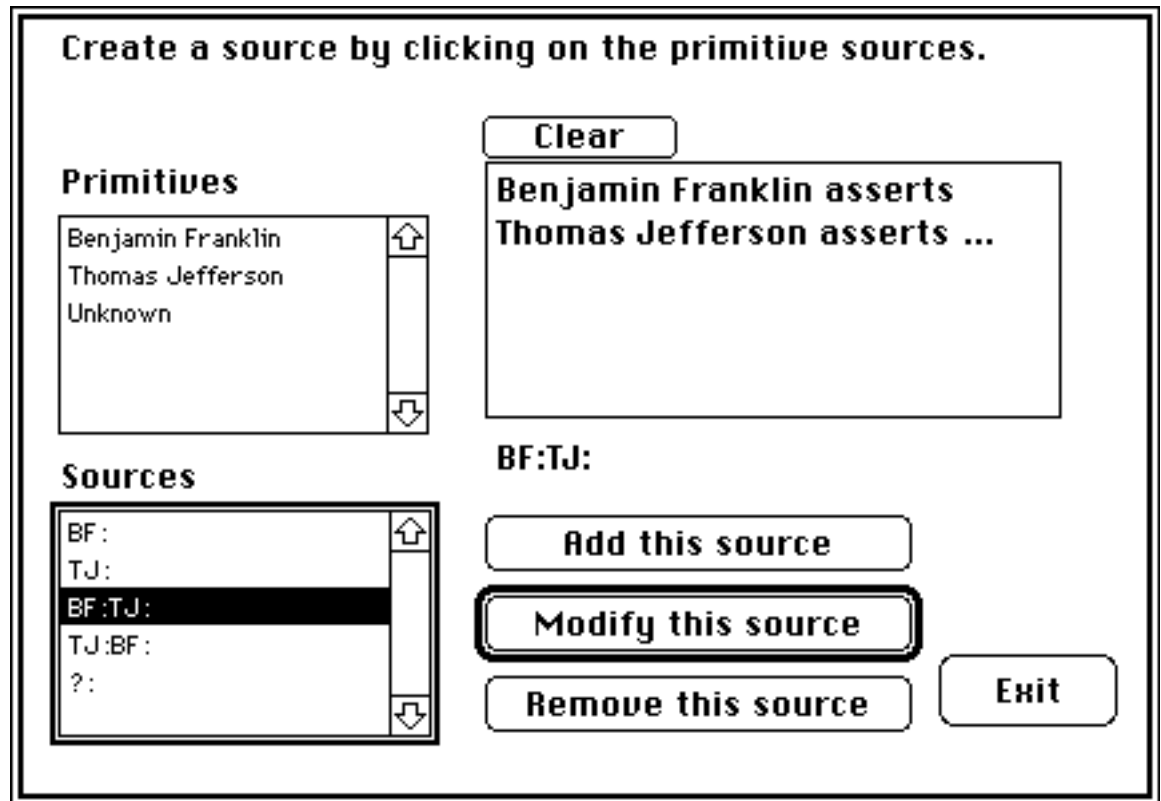
The only difference between the description of a relation and the other types is the Arith… button. The arity dialog box is used to determine the objects that are connected to the relation. At this time, no arity checking is done, so the settings of the Arity are ignored.

Incoming type

Statement
Claim
Term
Definition
Observation
Text Object

Outgoing type

Statement
Claim
Term
Definition
Observation
Text Object

Incoming Arity

○ 0 in
○ 1 in
◉ multiple in

Outgoing Arity

○ 0 out
◉ 1 out
○ multiple out

Arity: n to 1

[OK]   [Cancel]

The arity settings include the type for the inward and outward links. The arity is the number of connections that may enter or exit the relation.

Edit List Types… allows the user to add, modify or delete list object types. The name is what is used to identify this type in all menus. The description can contain relevant information about the type. The type description works identically to **Edit Text Types…**.

Edit Sources… is used for creating source expressions. A source expression is a representation of a "context" or "perspective" of an object. If the user creates an object, the source may be from a perspective other than the users. A quote or a reference to work by other authors work are cases for using a source other than the users own.

The sources are the individual perspectives. They may be people or books or political opinions or any other type of perspective. The primitive sources are defined using **Edit Primitive Sources…**.

Edit Primitive Sources… is used to define the primitive sources. Primitives are not used directly, but are accessed through Sources. When a primitive is created, a basic source containing only that primitive is created automatically.

The abbreviation for the primitive is used for the source expression in all menus. By convention, they should be uppercase letters so they are not confused with users. If users wish to make objects using their own perspective, then they need to make sources with their own name in addition to logging in with their name.

Update User Info… is used to change information about the user who is logged in. The user may not change information about any other user.
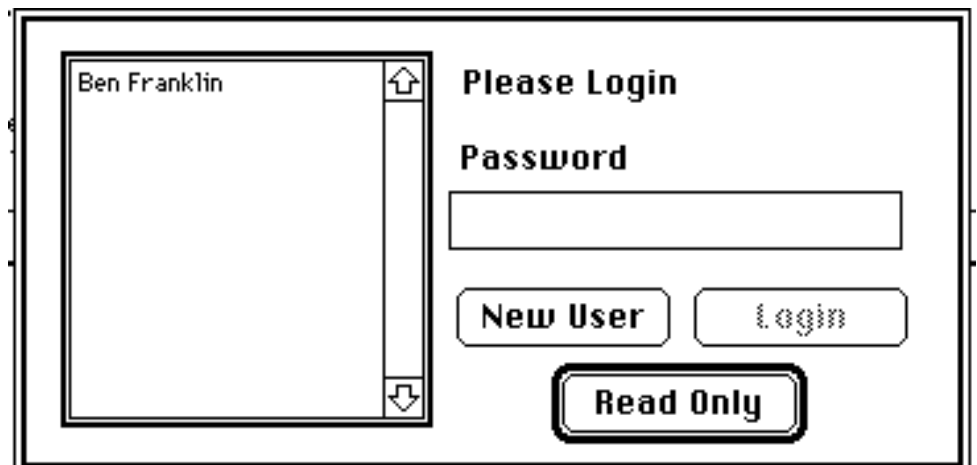
The user needs to type a password twice to change it or to keep it. If the user does not provide a password, then the password is removed.

Login as Other… is used if a different user wants to use the system. The other user will get the same prompt as the opening of the database. The new user would select a name and type their password.
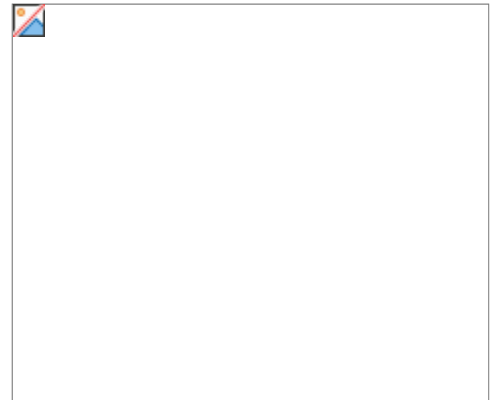

Introduction

The objects in the database are protected from modification by other users. If a user did not create an object, then they will be unable to edit the text in it or change any connections to relations that the user did not create.
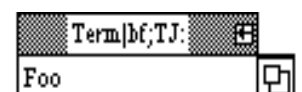
Copy Types From… is a means of reading all types from another database. If a user creates a database that needs a set of types that already exist in another database, then the user may read them in using this command. The user may opt to create template databases with types for specific purposes. This command copies all types, sources and users from the other database.

Hypertext Menu

Hide Object removes the selected objects from the display. This operation does not effect the object in the database. This is identical to **Clear** in the Edit menu.

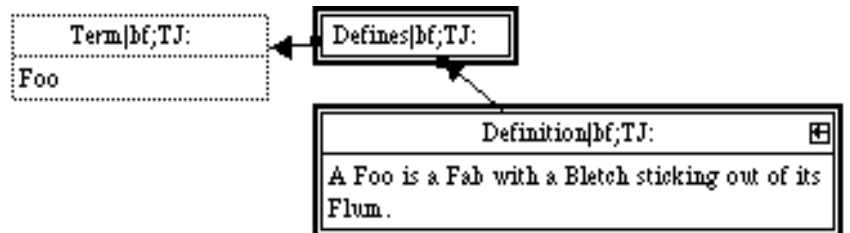**Show All Relatives** displays all relatives of the selected objects. This is similar to clicking on the arrow in an objects label bar. This command may work on more than one object in the selection that the arrow click may not do.

Above is an object before selecting this command and below is the display after the command. In this case, the menu operation has the

same effect as clicking on the arrow on the object.

This operation traverses two sets of links for every relative. The first link connects the object to a relative and the second links the object to the object on the other end of the relative.

Show Some Relatives uses a dialog box to ask which relatives should be displayed. The user may choose more than one object by holding the command (apple) key while selecting each object in the list. Holding the shift key allows selection of a contiguous set of objects from the list.

Show Hidden Objects gives the user a list of the entire database from which any number of objects may be chosen to show in the display.

Show List Members is used to display objects that are members of a list object. The user may select some objects from the list and have those objects displayed. If none of the listed objects are selected, then all the objects are displayed. If the option key is held down while choosing this command, no new objects are brought into the display, but the members which are already displayed are selected.

In the above illustration, two members of a list are selected. Below is what the display will look like after this operation completes.

```
┌─────────────────────────────────┐
│  Conjunction|bf;BF:        ⊞     │
├─────────────────────────────────┤
│        Foo Observations          │
├─────────────────────────────────┤
│ Foos always look like Bars.   →  │
│ Foos always smell like Bars→     │
│ Foos feel like Bars.          →  │
│ Foos taste like Bars.         →  │
└─────────────────────────────────┘
```

```
┌─────────────────────────────────┐
│  Observation|bf;BF:        ⊞     │
├─────────────────────────────────┤
│ Foos always smell like Bars.     │
└─────────────────────────────────┘
```

```
┌─────────────────────────────────┐
│  Observation|bf;BF:        ⊞     │
├─────────────────────────────────┤
│ Foos taste like Bars.            │
└─────────────────────────────────┘
```

Query Database… Finds objects from the database that match the users' description. The result of the operation creates a list object of the default list type that contains the result of the query.

```
┌─────────────────────────────────────┐
│  Find objects containing text:      │
│  ┌───────────────────────────────┐  │
│  │                               │  │
│  │                               │  │
│  └───────────────────────────────┘  │
│                                     │
│  Of Type  [ Claim            ▼ ]    │
│  By Creator [ Ben Franklin    ▼ ]   │
│  With Source [ TJ:            ▼ ]    │
│  That [ have      ▼ ]               │
│    Relation [ Supports      ▼ ]     │
│    Going [ into it        ▼ ]       │
│   (( Search ))    ( Cancel )        │
└─────────────────────────────────────┘
```
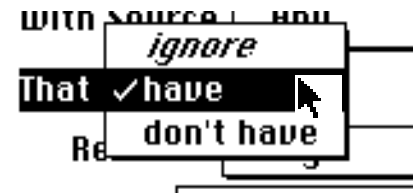
The user may specify some text that objects may contain. If this field is blank, then no text is searched. The search will look for any substring of any text object containing the given string.

The type may match either the exact type or a subtype of it. For example, if the user searches for type "Statement", then all Statements, Claims, and any other types that are based on statement or its subtypes will be found.
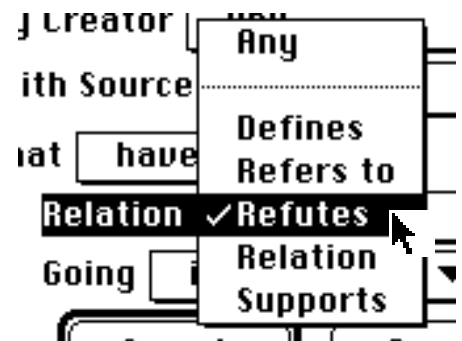
The creator and source must match exactly.

Any of the three specifications: Type, Creator and Source may use the specifier "Any" that will cause the search to ignore the attribute.
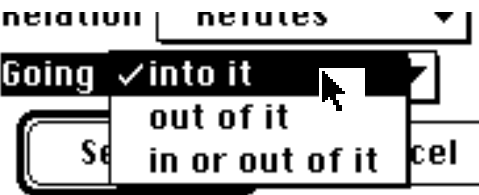
The relation specification is for finding objects that may or may not be connected to various relations in the database. The user may ignore this specification by selecting "ignore" from the relation specification.

If the user selects "have" then the user intends to include the given relation in the query. "don't have" looks for objects that are not connect by the given relation or those that have no relations.

The search uses the type specification to select the type of relation to find (or exclude) in the search. "Any" will match any relation.

"Going into it" asks the query to find objects that have (or don't have) the given relation connected to them through an inward link. "Going out of it" is for outward links and "in or out of it" is for any links.

Query From List… is used to perform a query using a list as the universe. A query may be performed in two phases using this command. The first query of the database may find all object matching some criteria and a second query may specify more criteria to narrow the results of the first search. This command is available only when the active object is a list object.

Keyboard/Mouse Gestures

Selecting objects may be done in two ways. The user may drag a rectangle to surround a group of objects, or the user may click on each object while holding the shift key down.

Shift-clicking on an object toggles the object for selection or deselection, so it may be used to deselect objects that are already selected. When used in conjunction with the rectangle drag, the user may select a set of objects and then deselect a subset of them.

Adding objects to lists is done by dragging a selection of objects into the list object. If the user is moving a selection and the mouse cursor is over a list object, then the program hilights the list object indicating that the objects will be added to the list. Adding objects to a list does not effect the objects that were moved, so the objects remain where they were in the display.

Changing order of list members is done by selecting a set of items in a list object and then dragging them to their desired location. One may use a list object to represent an ordered list of objects. One example of an ordered list is a written document. A user may enter a document as a set of individual statements, and then create a list object (of type *document* perhaps) representing the original document by sorting the statements in order of their

appearance in the paper.

Connecting objects is performed by holding the option key down while dragging from the start object to the end object. This is what the cursor will look like while the user is pressing the option key.
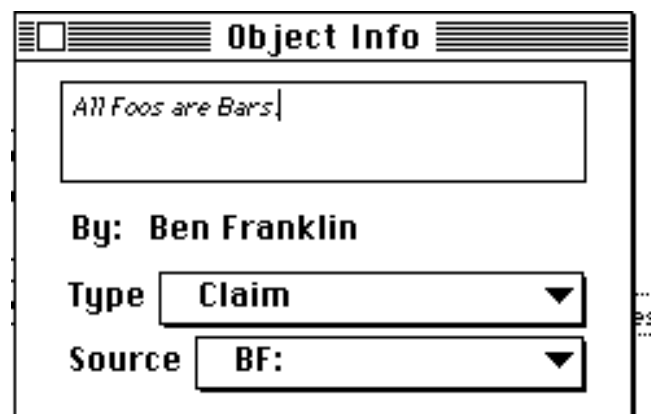
If neither endpoint is a relation, then this method creates a new relation between the objects. If either object is a relation, then it creates a link between the objects. If only one endpoint is a relation then the relation receives ownership of the line. If both ends are relations, then the starting relation receives ownership. If the owner is wrong, then the user may change it using the **Switch Line Owner** command in the Edit menu.

Quick Scrolling, a shortcut for scrolling the display is performed by holding down the Control key while dragging the mouse. This is what the cursor looks like when the user presses the control key.

The metaphor for using this gesture is using a hand to slide a piece of paper around a desktop. Similarly, clicking the mouse and dragging the display will move it under the window so that the user may view different parts of it.

Object Info may be accessed by double-clicking on the label bar of an object in the display. The program will display this window, showing information about the object and allowing the user to modify the attributes of the object.

Once the window is visible, other objects may be modified by double clicking on them. The information in the window represents

the last object to be double-clicked and not necessarily the currently active object.

To change the name of an object, select the current name and type the new one to replace it. If the current name is the default name (which is part of the text of the object), then it will be italicized.

The user may change the type and source by selecting from the appropriate menus. The changes take effect immediately, so there is no operation that needs to follow the changes. The creator is permanent and is not subject to change.

Adding a page to the display is done by dragging an object off the current page to the right or the bottom. The program automatically adjusts the drawing area according to the boundaries around the objects. When the user moves an object off the current page, the display adds a new page. If all objects are removed from a page, then the page will disappear. There is always a first page in which to draw.