Internal Use Only (非公開)

TR-I-0041

The Formalization of a Knowledge Base for English

Donald GEDDIS, Naomi INOUE, Tsuyoshi MORIMOTO
ドナルド・ゲダィス, 井ノ上 直己, 森元 逞

1988.9.16

Abstract

This paper describes a graphical editor that modifies a knowledge base of semantic concepts, as well as a small English knowledge base created with the aid of the program.

The English knowledge base is derived from the first five pages of the English translation of the ATR dialogues about international conference registration. The major concepts include Bring, Fill Out, Send, and Pay.

The graphical editor, KBE, incorporates an understanding of knowledge base structures developed at ATR. Major sections of the program include adding, deleting, and modifying nodes and links; drawing the knowledge base in a graphical format; file input and output in two formats; and a routine to check the global consistency of the knowledge base for possible errors in input.

This work was done as an internship at ATR Interpreting Telephony Research Laboratories.

ATR Interpreting Telephony Research Laboratories
ATR 自動翻訳電話研究所

The Formalization of a Knowledge Base for English

Donald Geddis   Naomi Inoue   Tsuyoshi Morimoto

ATR Interpreting Telephony Research Laboratories

September 16, 1988

Contents

Internship Report

Researcher	Donald F. Geddis
Position	Intern student June 29, 1988 through September 16, 1988
Research	The Formalization of a Knowledge Base for English
Company	ATR Interpreting Telephony Research Laboratory Knowledge and Database Department Twin 21 MID Tower, 2-1-61 Shiromi Higashi-ku, Osaka 540, Japan
Affiliation	Stanford University Stanford, California 94305 USA
Postal Mail	Post Office Box 4647 Stanford, California 94309 USA
Internet Electronic Mail	Geddis@Score.Stanford.Edu

Introduction

The Knowledge and Database Department has developed a theory of knowledge in the form of a semantic network, with specified types of nodes and links. Using this format, the Department is entering a substantial knowledge base of concepts in Japanese, in the domain of registrations for international conferences.

As raw material for the concepts, the Department has written simulations of people registering for conferences over a telephone. There also exist English language translations of these transcriptions, and the first six dialogues in the translations served as the raw material for my knowledge base of English.

The English knowledge base and the KBE (for Knowledge Base Editor) program developed simultaneously, with problems on one side leading to solutions on the other. The English knowledge base defines the relations between some of the main concepts in international conference registration, from specific instances in the world (like a time of 6 o'clock in the evening) to very general concepts (like the process of paying for something). KBE provides a graphical view of the developing knowledge base, and a user-friendly human interface for editing it. In addition, the program "understands" the meanings of the node types and link types, and has the capability to check the input data for errors in entry.

The first section of this paper describes the knowledge base, and the second section uses it as an example for demonstration of the editor. The major concepts of the knowledge base include Pay, Send, Fill-Out, and Bring.

The KBE editor incorporates an understanding of knowledge base structures developed at ATR. Major sections of the program include Adding, Deleting, and Modifying Nodes and Links; Drawing the knowledge base in a clear graphical format; File input and output in two formats; and a routine to check the global consistency of the knowledge base for possible errors in input.

In this text, the Roman typeface of the Times font in which these words are set is used for normal running text. Phrases or words which are meant to be exactly the same as in an example on the program screen are in *italics*.

This work was done as an internship at ATR Interpreting Telephony Research Laboratories.

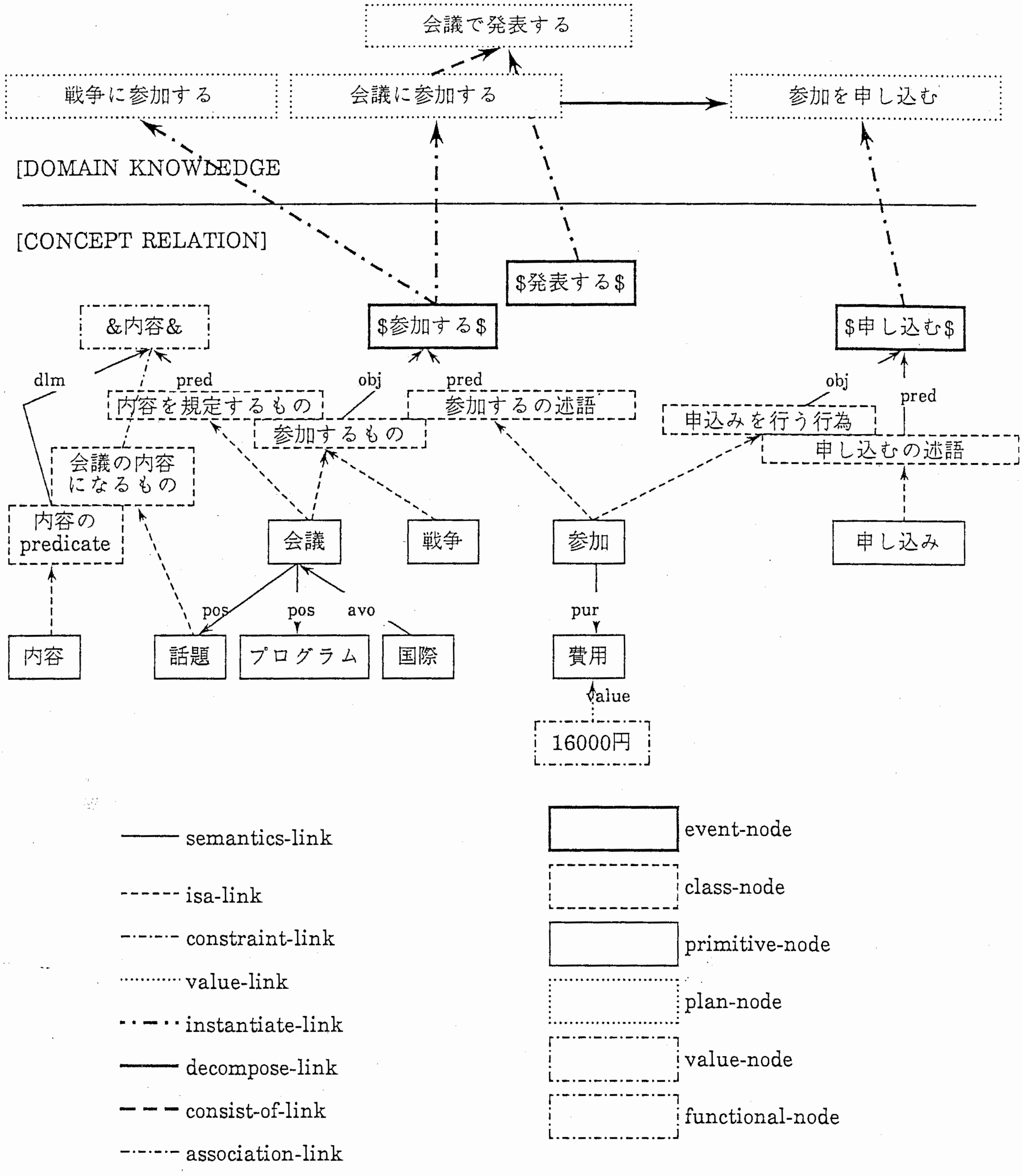
Knowledge Base Concepts

Nodes come in six types, which (in the order from top to bottom that they are usually seen) are: *Plan*, *Functional*, *Event*, *Class*, *Primitive* and *Value*. Class nodes group similar concepts; primitive nodes correspond to actual words in the language, and value nodes are specific instances of things in the world.

There are many types of links than can connect nodes, and the allowed types depend on nodes that are being connected. There are two non-directional links, *Association* and *Constraint*. All others define a hierarchy of nodes. The others are *Isa* links, *Semantic* links, and *Value* links. Plan nodes are special; from an event node to a plan node there are *Instantiation* links, and between plan nodes are either *Consist* or *Decompose* links. Plan nodes also have special semantics indicating the purpose of the plan that the node represents. Plan node semantics consist of semantic relations with corresponding nodes.

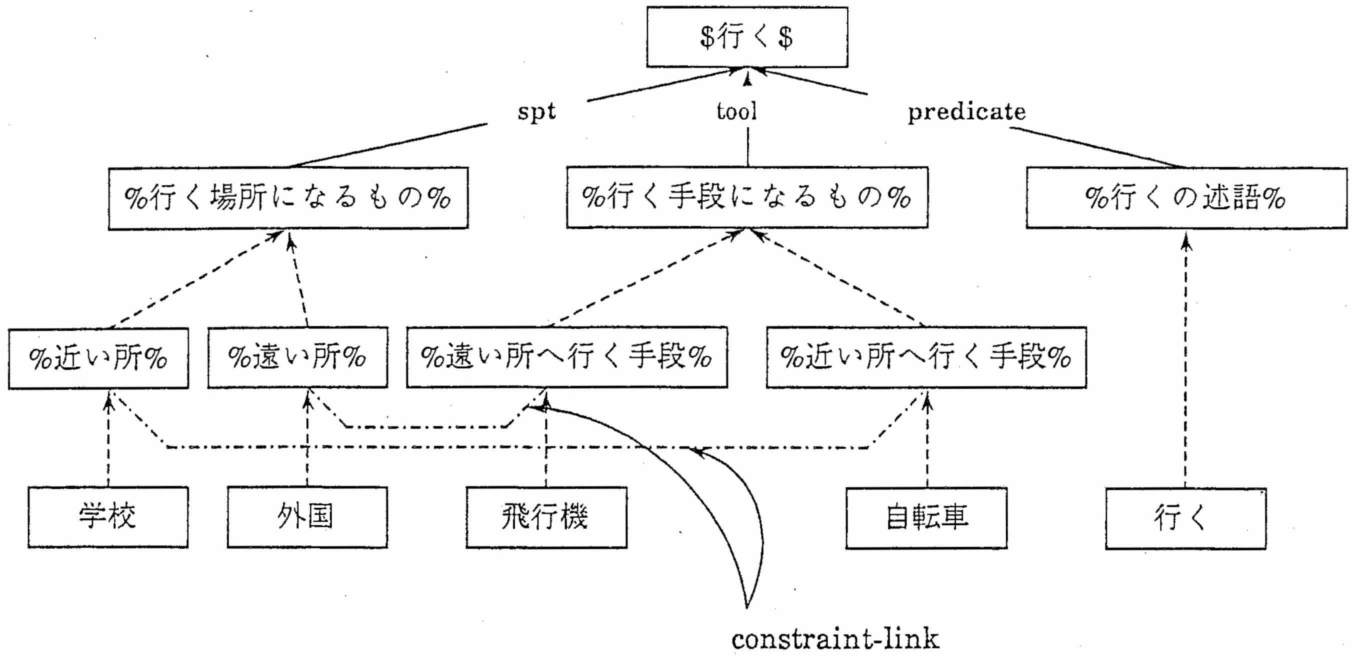
There are many different types of semantic relations between nodes, and the number and kinds keep changing as time goes on. They can be divided into (at least) two categories: those between a noun and a verb, and those between two nouns. The important semantic relations between a noun and a verb include: *Agent* (Taro studies), *Material* (I made the castle of sand), *Object* (Melon which is cut by knife), *Space At* (The ship goes to Hakodate), *Time At* (Work ends at 6pm), and *Tool* (I cut the melon with the knife). The important relations between two nouns include: *Author* (The sword made by Masamune), *Delimiter* (That kind of person), *Object Of Attribute* (The water temperature), *Part* (Foreigner who has blue eyes), *Possessor* (Taro's book), and *Whole* (The leg of a dog).

For some examples of a knowledge base, please see Figures 1 and 2.



N.B.:constraint-link is not in this figure.
as for constraint-link, refer to the another figure

Figure 1: Construction of Knowledge Base



The path between "学校" and "自転車"、 and the path between "外国" and "飛行機" are connected with constraint-links. Gate control is done by constraint-links

Figure 2: Example of Constraint-Link Usage

English Knowledge Base

The English Knowledge Base is a small network that associates concepts in the domain of international conference registration. The raw data came from translated simulations of human telephone conversations in the domain. The appendix shows the English Knowledge Base in both the KBE and Inoue file formats.

The knowledge base was created in the same way as the Japanese knowledge base was created. There are twenty eight primitive nodes and twenty two value nodes in the English knowledge base, as well as twelve class nodes and four event nodes. In addition there are many connections between all of the nodes.

The primitive nodes correspond to actual words used in the original text. The value nodes are specific individuals or items in the real world that were mentioned. Class nodes associate similar groups of words together, while event nodes define major concepts.

The four event nodes defined in the knowledge base are Bring, Fill-Out, Pay, and Send. Each is immediately connected to typical class nodes below it, namely a predicate (i.e. the verb or action), the object the action is directed toward, and perhaps the agent performing the action.

The event node *$Bring$* has a class of objects and a class of predicates. The only possible object is *Students*, reflecting the sentence in the dialogues where a professor wanted to bring some students to a conference. The predicate is the simple English verb *Bring*.

The event node *$Fill-Out$* also has classes of objects and predicates. The objects can be either an *Application* or a *Registration-Form*. The predicate is the simple verb *Fill-Out.*

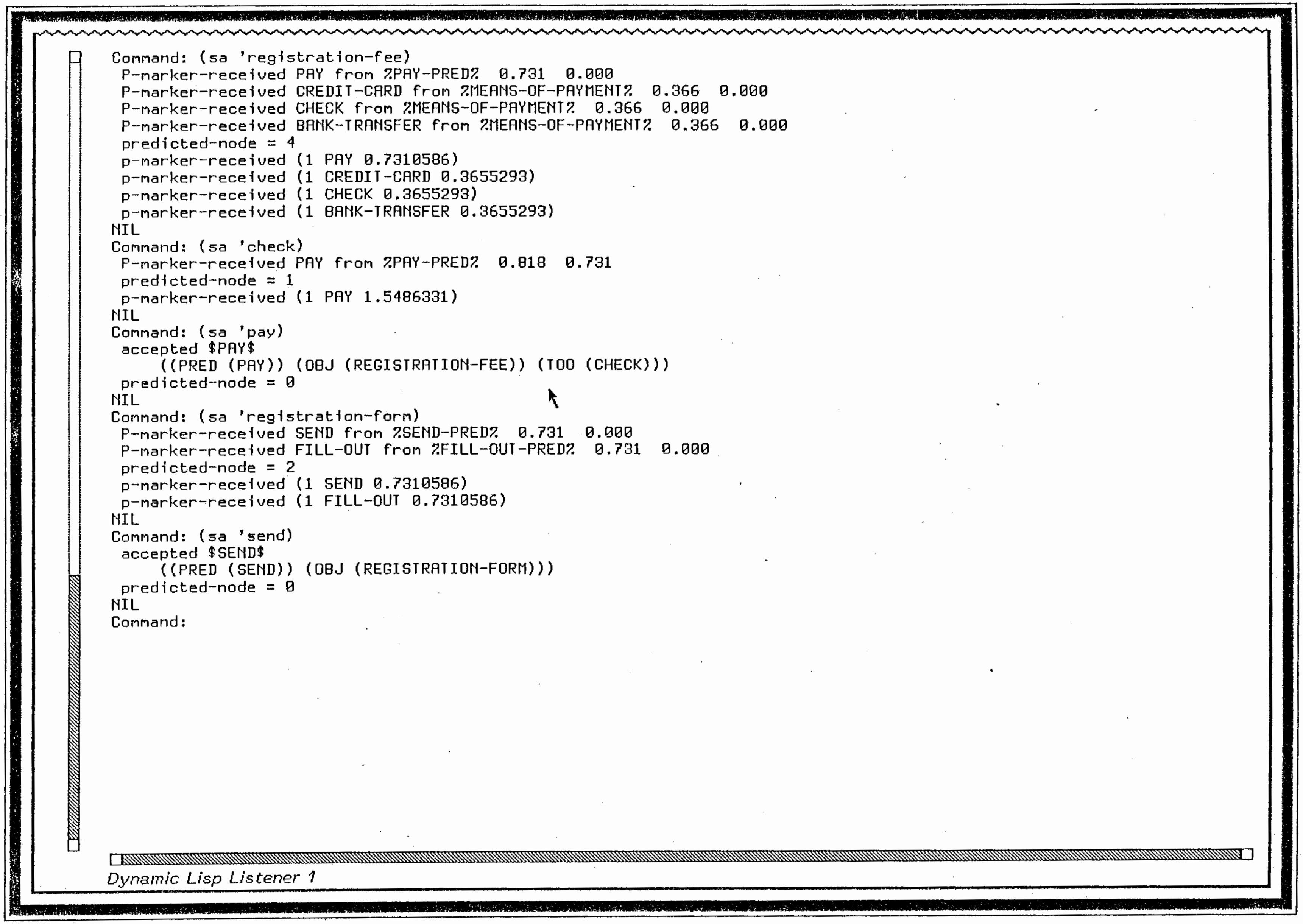
The event node *Pay* is more complicated, with classes of means of payment, objects, and predicates. The means of payment can be either *Bank-Transfer*, personal *Check*, or *Credit-Card*. The possible objects are *Attendance-Fee*, *Manuscript-Fee*, *Money*, or *Registration-Fee*, and the predicate is the verb *Pay*.

The event node *$Send$* has objects and predicates. The object is a *Registration-Form*, and the predicate is *Send*.

The English knowledge base also contains other structures. For example, a registration fee has as a characteristic, the amount of the fee. For the particular dialogues used, five specific amounts were mentioned: 5,000 yen, 10,000 yen, 16,000 yen, 50 dollars, and 100 dollars. These are all value nodes. Other value nodes are connected to the primitives *Time*, *Dates*, *Name*, *Number*, *Section* (a part of an address) and *City*.

After the creation of the knowledge base, the associational mechanism used on the Japanese knowledge base was tested on the English knowledge base. An example can be seen in Figure 3. Although the English knowledge base is quite small, there seem to be no problems using the associational mechanism with it. However, testing should probably be done on a larger knowledge base of English to confirm this result.

This success with the Japanese association mechanism on the English knowledge base helps to show that the structure of the knowledge bases for the two different languages are really very close. The main differences between the knowledge bases come from the syntatic differences in the languages, although that influence seems to be small. It is mostly idioms that cause differences, since phrases that do not translate well of course would not have matching structures in the other language.



```
Command: (sa 'registration-fee)
 P-marker-received PAY from %PAY-PRED%  0.731  0.000
 P-marker-received CREDIT-CARD from %MEANS-OF-PAYMENT%  0.366  0.000
 P-marker-received CHECK from %MEANS-OF-PAYMENT%  0.366  0.000
 P-marker-received BANK-TRANSFER from %MEANS-OF-PAYMENT%  0.366  0.000
 predicted-node = 4
 p-marker-received (1 PAY 0.7310586)
 p-marker-received (1 CREDIT-CARD 0.3655293)
 p-marker-received (1 CHECK 0.3655293)
 p-marker-received (1 BANK-TRANSFER 0.3655293)
NIL
Command: (sa 'check)
 P-marker-received PAY from %PAY-PRED%  0.818  0.731
 predicted-node = 1
 p-marker-received (1 PAY 1.5486331)
NIL
Command: (sa 'pay)
 accepted $PAY$
     ((PRED (PAY)) (OBJ (REGISTRATION-FEE)) (TOO (CHECK)))
 predicted-node = 0
NIL
Command: (sa 'registration-form)
 P-marker-received SEND from %SEND-PRED%  0.731  0.000
 P-marker-received FILL-OUT from %FILL-OUT-PRED%  0.731  0.000
 predicted-node = 2
 p-marker-received (1 SEND 0.7310586)
 p-marker-received (1 FILL-OUT 0.7310586)
NIL
Command: (sa 'send)
 accepted $SEND$
     ((PRED (SEND)) (OBJ (REGISTRATION-FORM)))
 predicted-node = 0
NIL
Command:
```

Figure 3: Example of Association Mechanism

KBE Program Documentation

KBE, for "Knowledge Base Editor", runs on the Symbolics 3600 family of Lisp machines. It was written in Symbolic Common Lisp, making heavy use of the Symbolics graphics and window extensions. At last count the source code had 3,251 lines made up of 115,058 characters, while the object code was more than 40K bytes of compiled Lisp functions.

The program structure looks like the picture in Figure 4. The editor maintains an internal description of the knowledge base, which can be modified with routines to change node data and link data. Show Knowledge Base presents the knowledge base in a clear graphical format. Check Validity, under the Options menu, searches the knowledge base for errors. File routines manage the interaction between the KBE editor and external disk files. Inoue format external files can then be loaded using other software in order to create a knowledge base network.

When the program is first loaded[1], the screen is filled with five horizontal window panes. Figure 5 shows the screen after loading a knowledge base file. The top pane is the title pane, identifying the program name, the laboratory and department, and in the lower right corner, the program version number. The second pane is the command menu. The third pane is a short status line, giving the format of the external file being edited and the file pathname. After the pathname can appear the words *(new file)*, if the knowledge base has never been saved to disk, or *(modified)*, if the knowledge base has been changed since last saved to disk. The fourth pane is the large display pane, with both horizontal and vertical scroll bars. This is where the graphical drawings of the knowledge base appear. The bottom pane is a Lisp listener, where various notifications from the program appear and input prompts are written.

The commands in the command menu (the second horizontal pane from the top) can be grouped into a number of categories. The leftmost six commands all change the knowledge base, by adding, deleting or modifying some structure. The upper three commands operate on nodes, while the lower three commands operate on links. In the middle pair, the top command (*Set Plan Semantics*) also modifies the knowledge base, namely the semantic information for a plan node. The bottom command, *Show Knowledge Base*, displays all or a portion of the knowledge base in a clear and useful graphic format, with nodes as ovals and links as arrows.

[1] At this time, the source directory is "LM01:>Geddis>KBE.*", although the exact names and locations are subject to change. KBE may be started by typing "(load "lm01:>geddis>kbe")" to a Lisp Listener on the Symbolics, and then <select>-K after it loads.

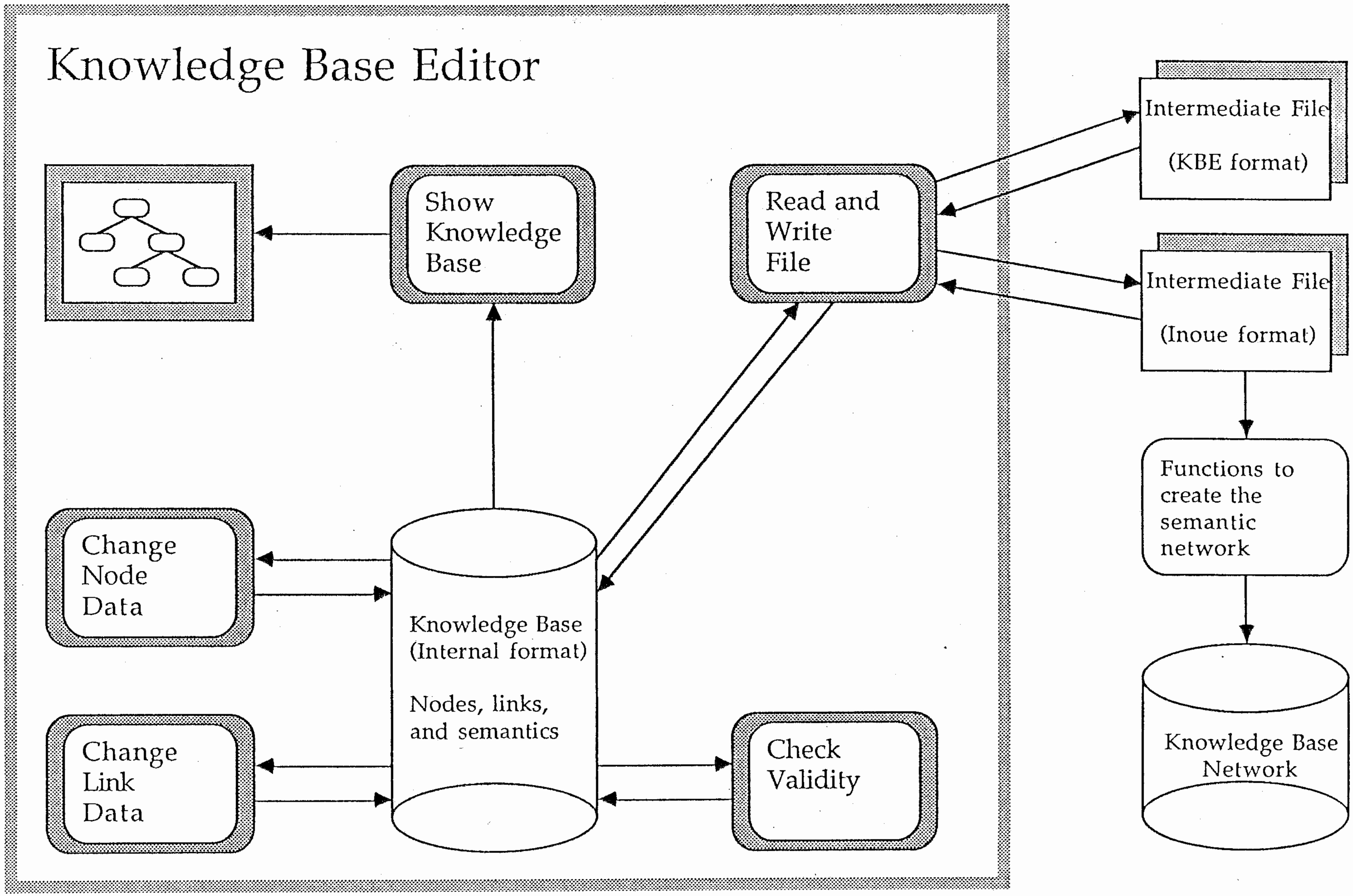
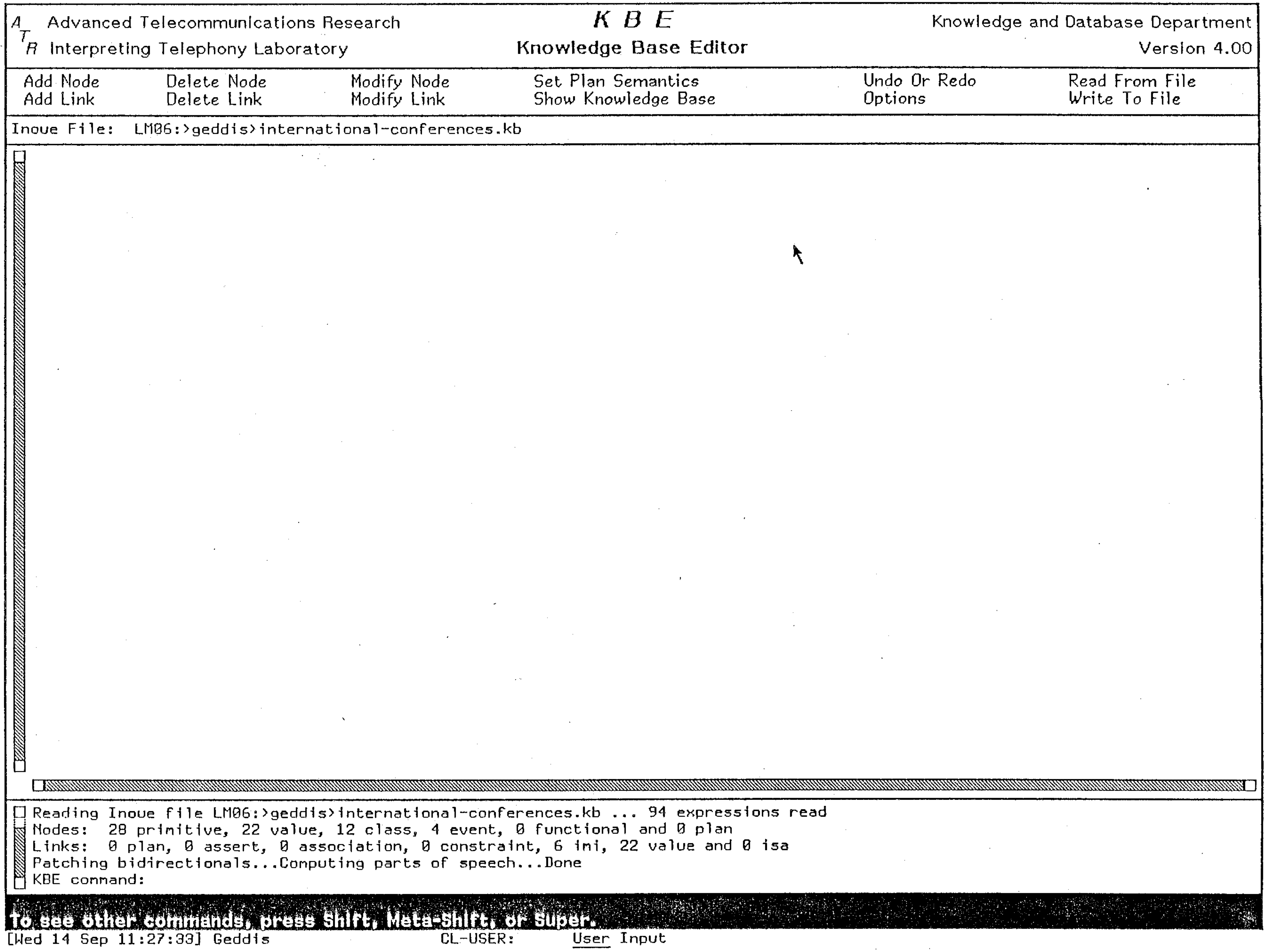

Knowledge Base Editor
Show Knowledge Base
Read and Write File
Intermediate File (KBE format)
Intermediate File (Inoue format)
Functions to create the semantic network
Change Node Data
Knowledge Base (Internal format)
Nodes, links, and semantics
Change Link Data
Check Validity
Knowledge Base Network

Figure 4: Knowledge Base Editor diagram



```
Inoue File:  LM06:>geddis>international-conferences.kb
```

```
Reading Inoue file LM06:>geddis>international-conferences.kb ... 94 expressions read
Nodes:  28 primitive, 22 value, 12 class, 4 event, 0 functional and 0 plan
Links:  0 plan, 0 assert, 0 association, 0 constraint, 6 ini, 22 value and 0 isa
Patching bidirectionals...Conputing parts of speech...Done
KBE connand:
```

```
To see other commands, press Shift, Meta-Shift, or Super.
[Wed 14 Sep 11:27:33] Geddis                CL-USER:      User Input
```

Figure 5: KBE screen – Initial

The next pair are used less frequently. *Undo Or Redo* reverses the effects of any command that modifies the knowledge base. It maintains a complete stack of all commands since the knowledge base was last written or read, and offers to undo (or to undo an undo command, which is called redo) the most recent one. The *Options* command brings up a menu of seldom-used other commands. The rightmost pair of commands handles file input and output, in either KBE or Inoue formats.

Many times the program will ask for a node. There are many ways of specifying the particular node, and the input routine is exactly the same in all cases. The name of the node may be typed directly, terminated by a RETURN. A unique prefix may be typed, and then the COMPLETE key will fill out the rest of the name[2]. At any time, typing the HELP key will print the possible completions. More useful perhaps is that pressing the right mouse button will generate a menu of the possible completions[3], so that the choice you want may simply be selected from a menu. An example of using the menu is shown in Figure 6. Typically this is used after typing a short prefix, for example the special typographic symbol at the beginning of many node names. Generating a list for all nodes in the knowledge base can take a long time for large files. The last, and probably best, method is by clicking the left mouse button on any node that has been already displayed in a drawing, which is the same as typing the name of that node.

When adding a new node or modifying an old one, the program will give you the ability to enter the name of the node. With a Japanese system it is possible to enter the name in either kana or kanji. Unfortunately, the Symbolics menu routines seem to be incompatible with typing Japanese directly, but the names can still be entered. Type the name in romaji first, and then convert to Japanese writing by using either s-m-H for hiragana, s-m-K for katakana, or s-m-J for kanji[4].

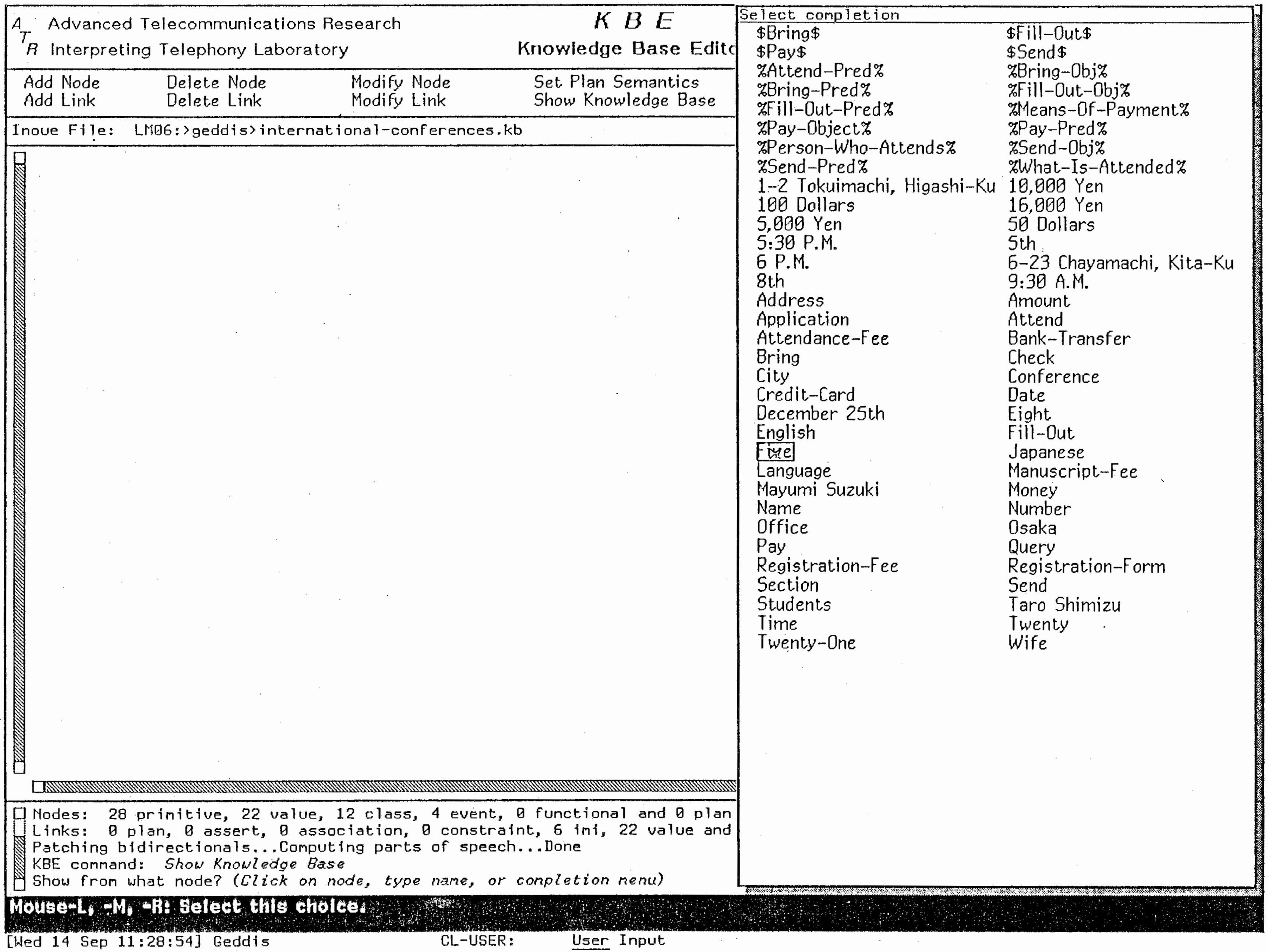
A useful convention is to have all nodes of the same type use a special punctuation character at the beginning and end of their names. This makes it easier to identify the types of the nodes. The current conventions are that names have the following forms: Plan @...@, Functional &...&, Event $...$, Class %...%, and nothing for either Primitive or Value nodes.

The KBE program understands two external file formats, named KBE format and Inoue format. The KBE format is based around nodes, and the connections each node has to other nodes. The Inoue format is used in other programs and is based around node types. The Inoue format for these special uses, but the KBE format is what the program manipulates internally.

[2] If the prefix is not unique, COMPLETE will add characters until the first difference is found between the possible completions. This is the normal behavior for completion on the Symbolics.

[3] Note that the Symbolics system only offers this option when the mouse is not pointing to something "interesting". A safe place to put it is usually the lower right of the screen, in the Lisp listener window pane. Check the top line of the mouse status pane (the black rectangle at the bottom of the screen) to be sure; it should say that mouse-R generates a completion menu.

[4] As usual, the "s-m" prefix means to type the last character while holding down the super and meta keys. These are the standard keyboard conversion sequences, but they can be altered. Type FUNCTION-J to see the current settings.



```
A
 T  Advanced Telecommunications Research          K B E
  R  Interpreting Telephony Laboratory            Knowledge Base Editc

 Add Node        Delete Node        Modify Node        Set Plan Semantics
 Add Link        Delete Link        Modify Link        Show Knowledge Base

Inoue File:   LM06:>geddis>international-conferences.kb
```

```
Select completion
 $Bring$                          $Fill-Out$
 $Pay$                            $Send$
 %Attend-Pred%                    %Bring-Obj%
 %Bring-Pred%                     %Fill-Out-Obj%
 %Fill-Out-Pred%                  %Means-Of-Payment%
 %Pay-Object%                     %Pay-Pred%
 %Person-Who-Attends%             %Send-Obj%
 %Send-Pred%                      %What-Is-Attended%
 1-2 Tokuimachi, Higashi-Ku       10,000 Yen
 100 Dollars                      16,000 Yen
 5,000 Yen                        50 Dollars
 5:30 P.M.                        5th
 6 P.M.                           6-23 Chayamachi, Kita-Ku
 8th                              9:30 A.M.
 Address                          Amount
 Application                      Attend
 Attendance-Fee                   Bank-Transfer
 Bring                            Check
 City                             Conference
 Credit-Card                      Date
 December 25th                    Eight
 English                          Fill-Out
 Fine                             Japanese
 Language                         Manuscript-Fee
 Mayumi Suzuki                    Money
 Name                             Number
 Office                           Osaka
 Pay                              Query
 Registration-Fee                 Registration-Form
 Section                          Send
 Students                         Taro Shimizu
 Time                             Twenty
 Twenty-One                       Wife
```

```
Nodes:  28 primitive, 22 value, 12 class, 4 event, 0 functional and 0 plan
Links:  0 plan, 0 assert, 0 association, 0 constraint, 6 ini, 22 value and
Patching bidirectionals...Computing parts of speech...Done
KBE command:  Show Knowledge Base
Show from what node? (Click on node, type name, or completion menu)
Mouse-L, -M, -R: Select this choice.
[Wed 14 Sep 11:28:54] Geddis                 CL-USER:        User Input
```

Figure 6: KBE screen – Node menu

Change Node Data

The three commands on the left side of the top command line have to do with altering node data. *Add Node* initializes the data for a new node, and then puts up a window for you to enter the particular values for that node. *Delete Node* removes a node from the knowledge base, as well as all of the links connecting that node to other nodes. *Modify Node* lets you edit the data for an already existing node, using the same menu as the *Add Node* command. An example of *Modify Node* is shown in Figure 7.

The keyboard shortcut for *Add Node* is "A", for *Delete Node* is "D", and for *Modify Node* is "M".

Change Link Data

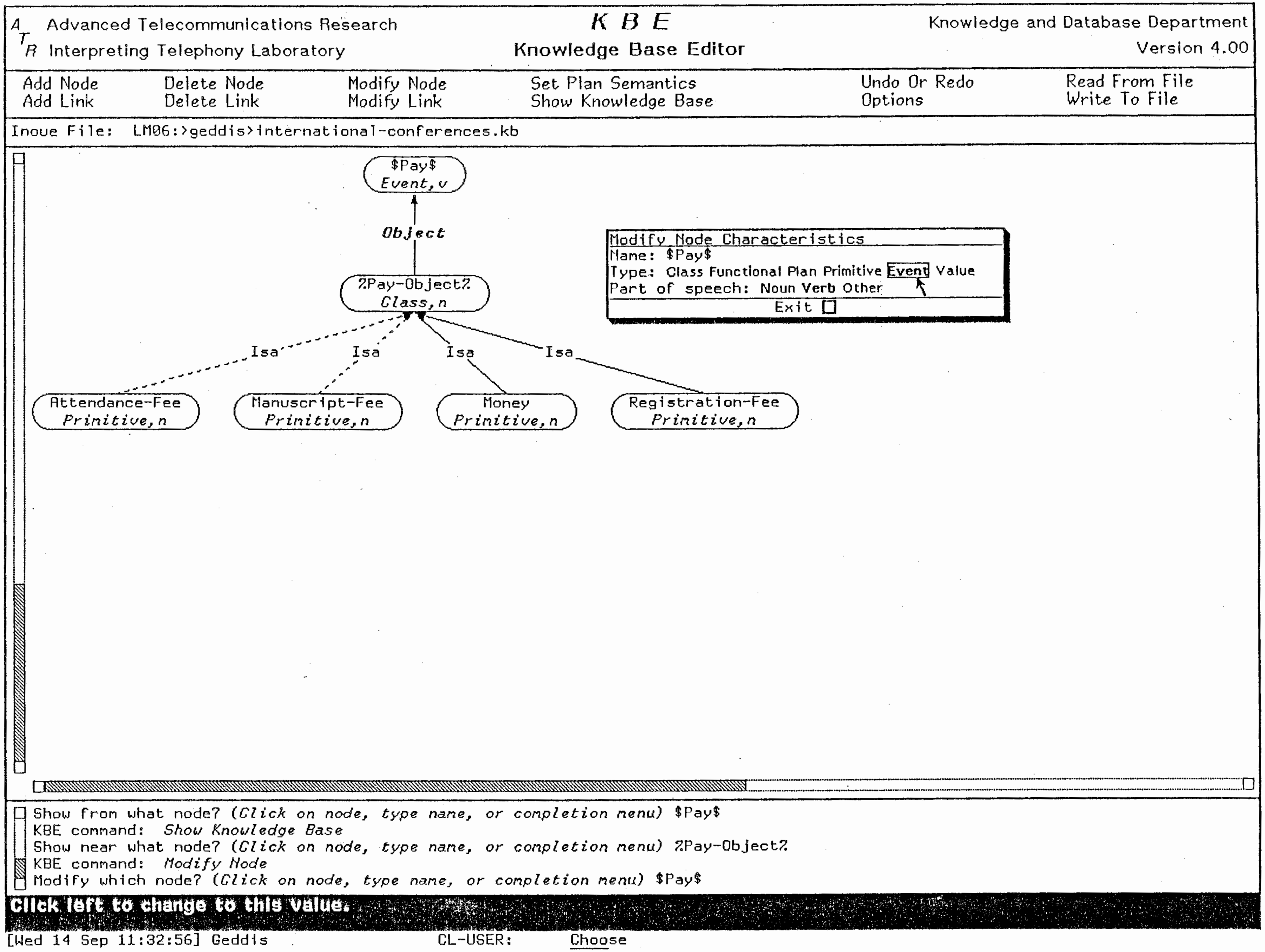
The three commands on the left side of the bottom command line alter link data. *Add Link* creates a link between two nodes, allowing you to specify the type of the link. It understands about node types as well, so only the link types that are valid between the two selected nodes are offered for you to choose from. *Delete Link* removes a link between two nodes. *Modify Link* uses the same window as *Add Link*, but with the values initialized to the data for the selected link. And example of *Modify Link* is shown in Figure 8.

The keyboard shortcut for *Add Link* is "c-A" (control-A), for *Delete Link* is "c-D" (control-D), and for *Modify Link* is "c-M" (control-M).

Set Plan Semantics

Internally, plan semantics are represented as normal semantic links to plan nodes, since logically plan nodes cannot have semantic links. The program, however, enforces a separation between semantic links with plan nodes and other links. This command brings up a menu with three options: *Add Plan Semantics*, *Delete Plan Semantics*, and *Modify Plan Semantics*. These are identical in function, and indeed in lisp code, to the add, delete, and modify links commands.

The keyboard shortcut for this command is "P".

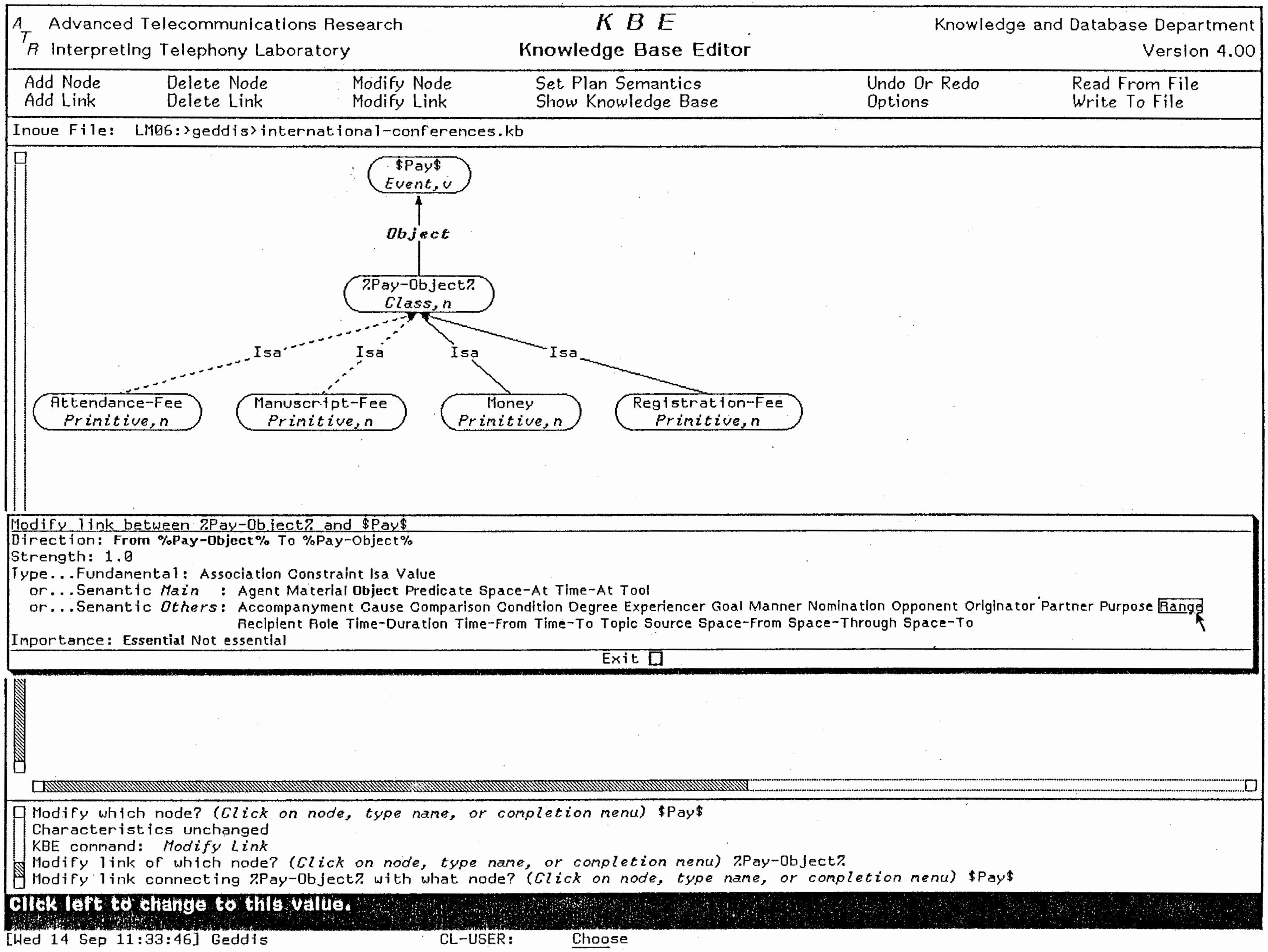


```
Inoue File:  LM06:>geddis>international-conferences.kb
```

Modify Node Characteristics
Name: Pay
Type: Class Functional Plan Primitive [✓ Event] Value
Part of speech: Noun Verb Other

```
Show from what node? (Click on node, type name, or completion menu) $Pay$
KBE command:  Show Knowledge Base
Show near what node? (Click on node, type name, or completion menu) %Pay-Object%
KBE command:  Modify Node
Modify which node? (Click on node, type name, or completion menu) $Pay$
Click left to change to this value.
[Wed 14 Sep 11:32:56] Geddis            CL-USER:
```

Figure 7: KBE screen – Modify Node



```
Inoue File:  LM06:>geddis>international-conferences.kb
```

```
Modify link between %Pay-Object% and $Pay$
Direction: From %Pay-Object% To %Pay-Object%
Strength: 1.0
Type...Fundamental: Association Constraint Isa Value
  or...Semantic Main   : Agent Material Object Predicate Space-At Time-At Tool
  or...Semantic Others: Accompanyment Cause Comparison Condition Degree Experiencer Goal Manner Nomination Opponent Originator Partner Purpose Range
                        Recipient Role Time-Duration Time-From Time-To Topic Source Space-From Space-Through Space-To
Importance: Essential Not essential
```

```
Modify which node? (Click on node, type name, or completion menu) $Pay$
Characteristics unchanged
KBE command:  Modify Link
Modify link of which node? (Click on node, type name, or completion menu) %Pay-Object%
Modify link connecting %Pay-Object% with what node? (Click on node, type name, or completion menu) $Pay$
Click left to change to this value.
[Wed 14 Sep 11:33:46] Geddis            CL-USER:
```

Figure 8: KBE screen – Modify Link

Show Knowledge Base

This command brings up a menu with four possible options, each representing a different scope of viewing the knowledge base. In each case, the procedure takes the nodes it is to display and runs a complicated algorithm to position the nodes in a way that would look most natural. The algorithm itself was reverse-engineered from the Symbolics *Format-Graph-From-Root* function[5]. It works perfectly on mathematical tree structures, but knowledge bases tend to be directed acyclic graphics (DAGs), so sometimes links overwrite nodes.

Nodes are printed on two lines surrounded by an oval. The top line shows the node name, centered. The bottom line (in italics) shows the node type, and if the part-of-speech is defined, either an "n" for noun or a "v" for verb. Links are drawn from the top of the bottom node's oval to the bottom of the top node's oval, except for non-hierarchical links (association and constraint), which connect the center of both nodes together. Link types are centered both vertically and horizontally on the link, which unfortunately tends to put them all in the same place so they overwrite each other.

Solid lines are drawn for links with a strength greater than 0.5, and dashed lines for strengths less than or equal to 0.5. Association and constraint links are drawn with grey (really very finely dashed) lines. In the link labels, normal text is used for fundamental link types, ***bold-italic*** text for essential semantic links, *italic* text for non-essential semantic links, and small-font abbreviations for association and constraint links (Assoc and Const, respectively).

To make overlapping graphics look prettier, the program erases a bounding rectangle before writing the link labels. (There is also code to do this before drawing the node ovals, but there seems to be a problem in the Symbolics system code. In scrolling dynamic windows such as KBE's display pane, objects that are drawn only have their positions remembered, not the order in which they were drawn. The Symbolics documentation says that to preserve the order of a graphics drawing, you should enclose the drawing in a *dw:with-output-as-presentation* form. I did that, but it doesn't seem to solve the problem. So now scrolling back and forth messes up the link labels, but they usually aren't so important anyway. If a solution to this is found, the commented-out erasing before drawing the nodes should be restored in the code.)

The first option in the menu, *Everything*, draws all nodes in the knowledge base. The second option, *From A Node*, starts at a selected node and expands its subtree

[5] Unfortunately that source code was unavailable at the time, so a tedious trial-and-error coding process had to take place to discover the algorithm. The Show Knowledge Base algorithm is actually a subset of the Symbolics one, with many of the latter's arguments instantiated, but later significant additions were added for the labelling of links.

down a chosen number of link levels (or expands without a limit on the number of links, if desired). An example of drawing from a node is shown in Figure 9. The third option, *Near A Node*, draws everything that is exactly one link away from the selected node, with parent nodes on the first level, the selected node (and nodes connected only by association or constraint links) on the middle level, and child nodes on the third level. An example of drawing near a node is shown in Figure 10, where it is the middle node that was chosen. The last option, *Plan Semantics*, shows the semantics of a plan node as through they were semantic links (which, internally, they are).

The keyboard shortcut for this command is "S".

Undo Or Redo

Every time the knowledge base is modified, the program stores a form that can reverse the modification on a stack. This command allows you to go back through that stack, popping off and executing undo forms that slowly return the knowledge base to its original form.

If you accidentally undo a modification that you really wanted, the redo option keeps a similar stack of everything that was undone, so you can restore the modifications.

Both stacks are cleared when the knowledge base is either read from a file or written to a file. In addition, if you undo a series of changes and then perform another modification (except for another undo or redo), then the redo stack is cleared. This is because the program cannot guarantee to be able to redo a command except in the original order it was done[6].

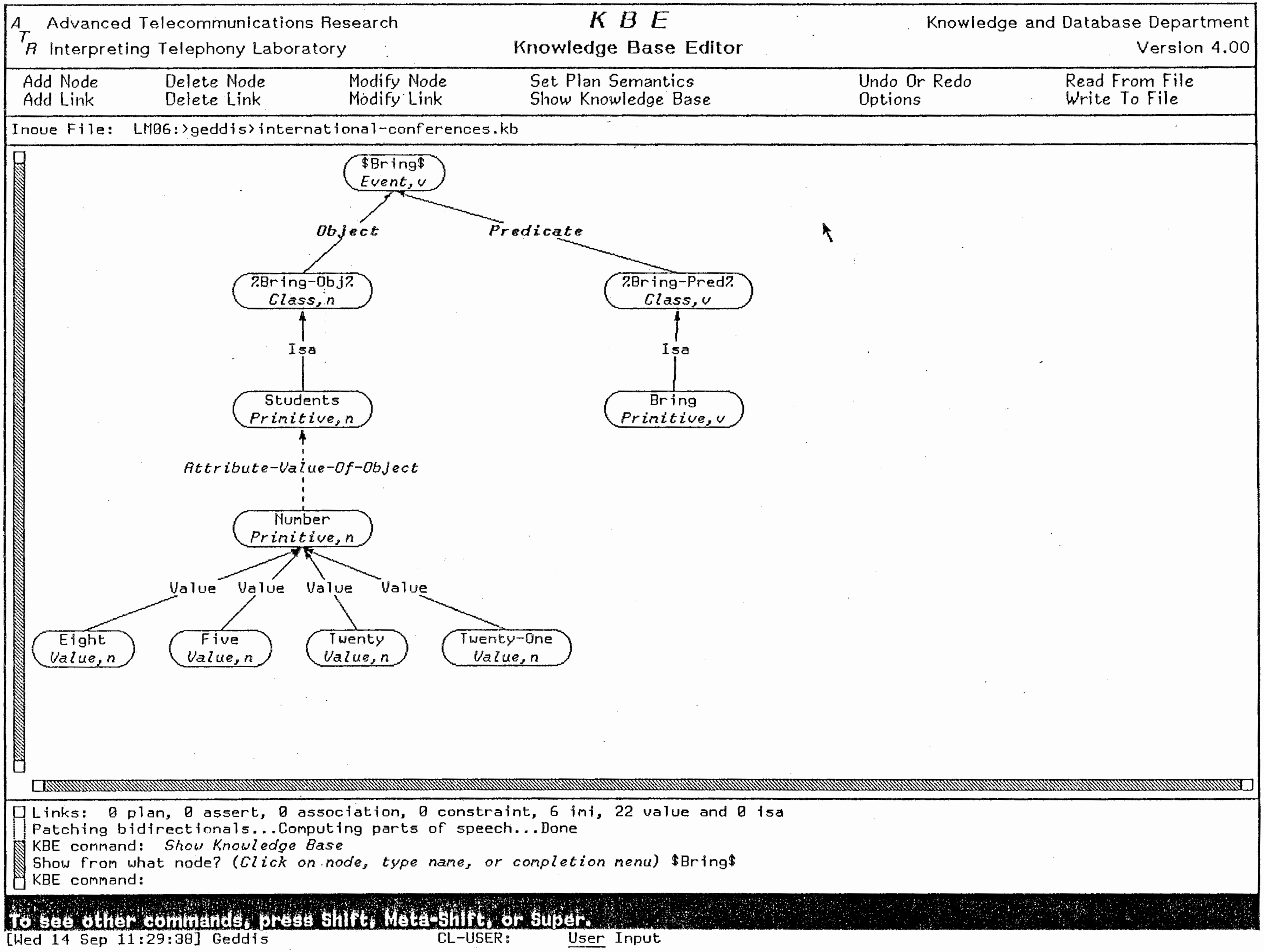
The keyboard shortcut for this command is "U".

Options

The command brings up a menu of other, less frequently used, commands. They are not in a major command row on the screen, because the additional space would make less area available for displaying the knowledge base.

The keyboard shortcut for this command is "O".

[6] For example, consider undoing a modification where you rename a node. You might then delete that node. It now becomes impossible to redo the renaming.

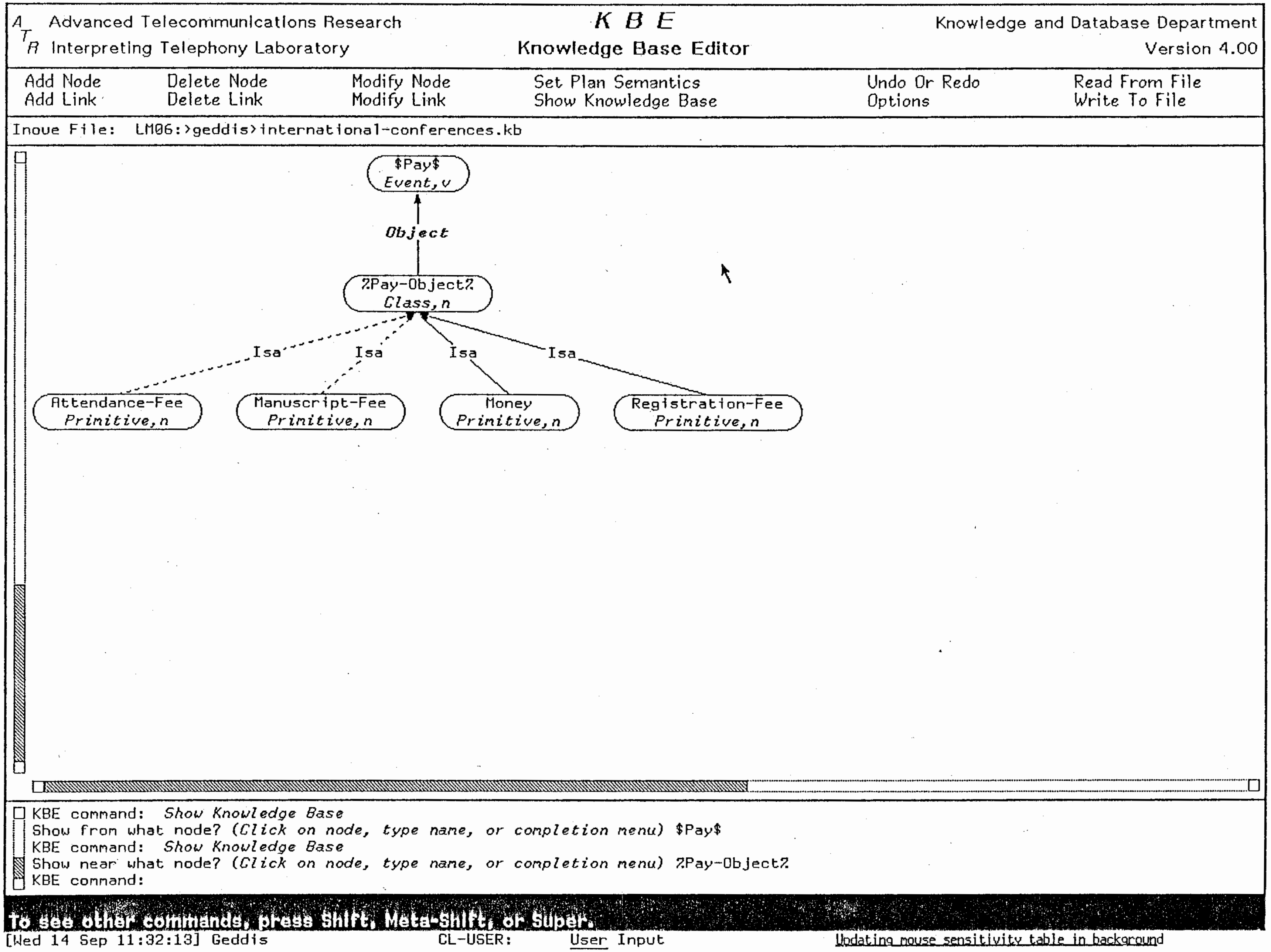


```
Inoue File:  LM06:>geddis>international-conferences.kb
```

```
Links:  0 plan, 0 assert, 0 association, 0 constraint, 6 ini, 22 value and 0 isa
Patching bidirectionals...Computing parts of speech...Done
KBE command:  Show Knowledge Base
Show from what node? (Click on node, type name, or completion menu) $Bring$
KBE command:
```

```
To see other commands, press Shift, Meta-Shift, or Super.
[Wed 14 Sep 11:29:38] Geddis                    CL-USER:        User Input
```

Figure 9: KBE screen – Show from $Bring$



```
Inoue File:  LM06:>geddis>international-conferences.kb
```

```
KBE command:  Show Knowledge Base
Show from what node? (Click on node, type name, or completion menu) $Pay$
KBE command:  Show Knowledge Base
Show near what node? (Click on node, type name, or completion menu) %Pay-Object%
KBE command:
```

To see other commands, press Shift, Meta-Shift, or Super.

```
[Wed 14 Sep 11:32:13] Geddis          CL-USER:     User Input          Updating mouse sensitivity table in background
```

Figure 10: KBE screen – Show near %Pay-Object%

Online KBE Documentation

This option brings up a scrollable window giving a short summary of this manual. It answers some of the most frequently asked questions, and provides a reminder for the display coding scheme used when the knowledge base is drawn.

The window is shown in Figure 11.

Set User Parameters

It was originally intended that many aspects of the program would be customizable by the user, and this option would set the preferences for a particular user. As it turns out, the only option that was ever needed was setting the file type of the output of the program, as either a KBE file or an Inoue file.

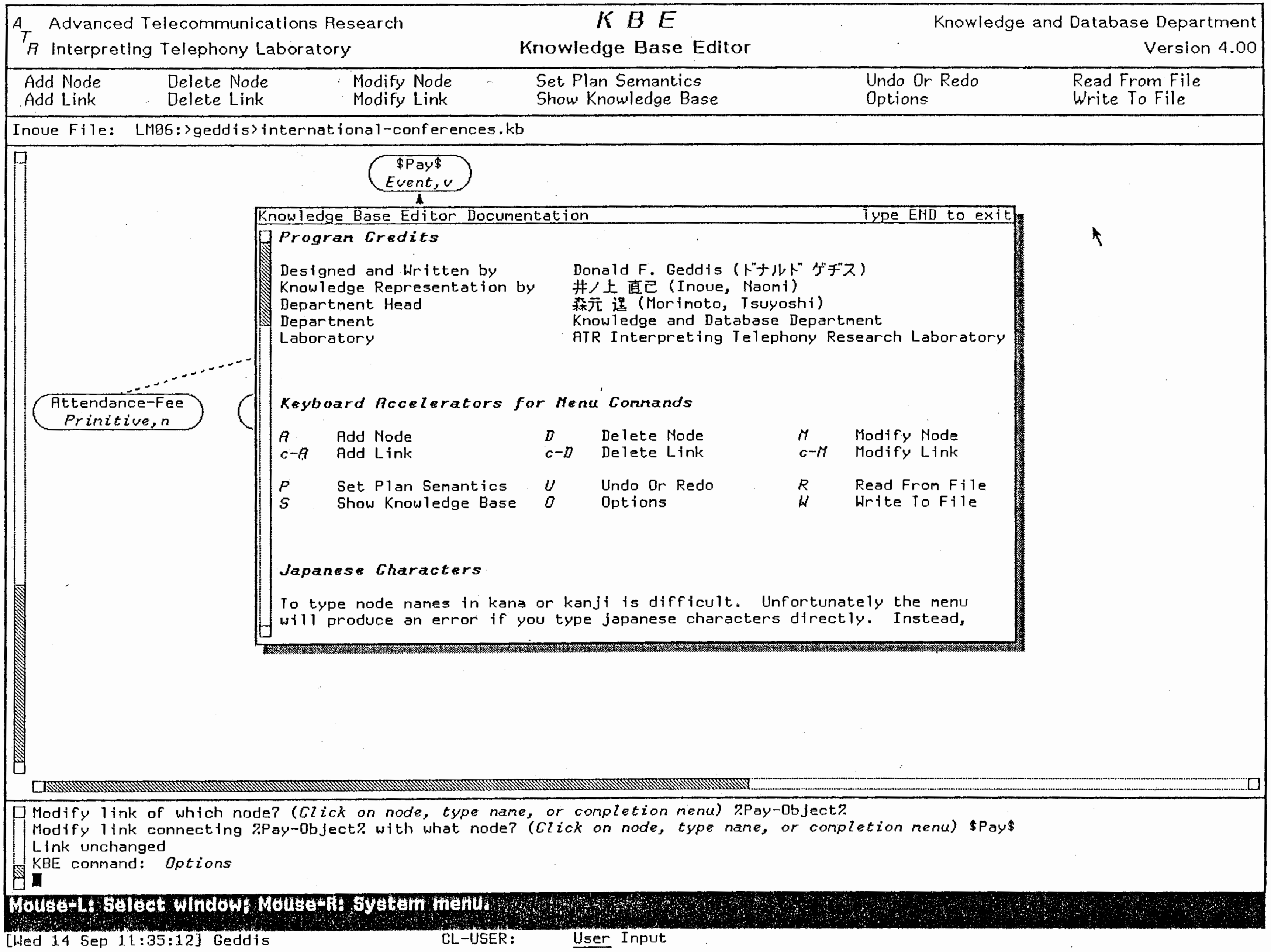
KBE format is best in normal use, because it shows the connections between any particular node and every other node to which it is linked. The Inoue format is necessary when using the resulting file to create a network knowledge base, which is then used to predict words in that are likely to be near other words.

Clear Display History

Every time the *Show Knowledge Base* command is executed, the display area of the screen is cleared and the new drawing is shown. The display pane is a dynamic window, though, and so it may be scrolled to view areas that are currently off the screen, for drawings that are too big to show all on the screen.

Similarly, the old drawings of previous show commands are saved in the window pane above the current drawing, and may be viewed by scrolling backwards through the pane. After long use of the program, however, there may be many previous drawings that are no longer useful, either because the knowledge base has since been modified, a new file is being edited, or the drawings are no longer of interest. In this case, this clear option erases the contents of the drawing window[7]. Especially after long editing sessions, this can also recover some memory in the machine.

[7] By sending the window pane a ":Clear-History" message.



```
Inoue File:  LM06:>geddis>international-conferences.kb
```

```
Knowledge Base Editor Documentation                                   Type END to exit
```

Program Credits

Designed and Written by	Donald F. Geddis (ドナルド ゲデス)
Knowledge Representation by	井ノ上 直己 (Inoue, Naomi)
Department Head	森元 逞 (Morimoto, Tsuyoshi)
Department	Knowledge and Database Department
Laboratory	ATR Interpreting Telephony Research Laboratory

Keyboard Accelerators for Menu Commands

A	Add Node	*D*	Delete Node	*M*	Modify Node
c-A	Add Link	*c-D*	Delete Link	*c-M*	Modify Link
P	Set Plan Semantics	*U*	Undo Or Redo	*R*	Read From File
S	Show Knowledge Base	*O*	Options	*W*	Write To File

Japanese Characters

To type node names in kana or kanji is difficult. Unfortunately the menu will produce an error if you type japanese characters directly. Instead,

```
Modify link of which node? (Click on node, type name, or completion menu) ?Pay-Object?
Modify link connecting ?Pay-Object? with what node? (Click on node, type name, or completion menu) $Pay$
Link unchanged
KBE command:  Options
```

```
Mouse-L: Select window; Mouse-R: System menu.
[Wed 14 Sep 11:35:12] Geddis                    CL-USER:        User Input
```

Figure 11: KBE screen – Online Documentation

Edit Knowledge Base Comments

To keep track of the purpose of various versions of knowledge bases, KBE allows comment text to be kept with each knowledge base. When this option is selected, a standalone ZMACS editor window appears. Any text may be typed here, and it will be saved when the file is written. The window has all of the functionality of a normal ZMACS editor, except for file input and output. To leave the editor window, type the END key.

This window is shown in Figure 12.

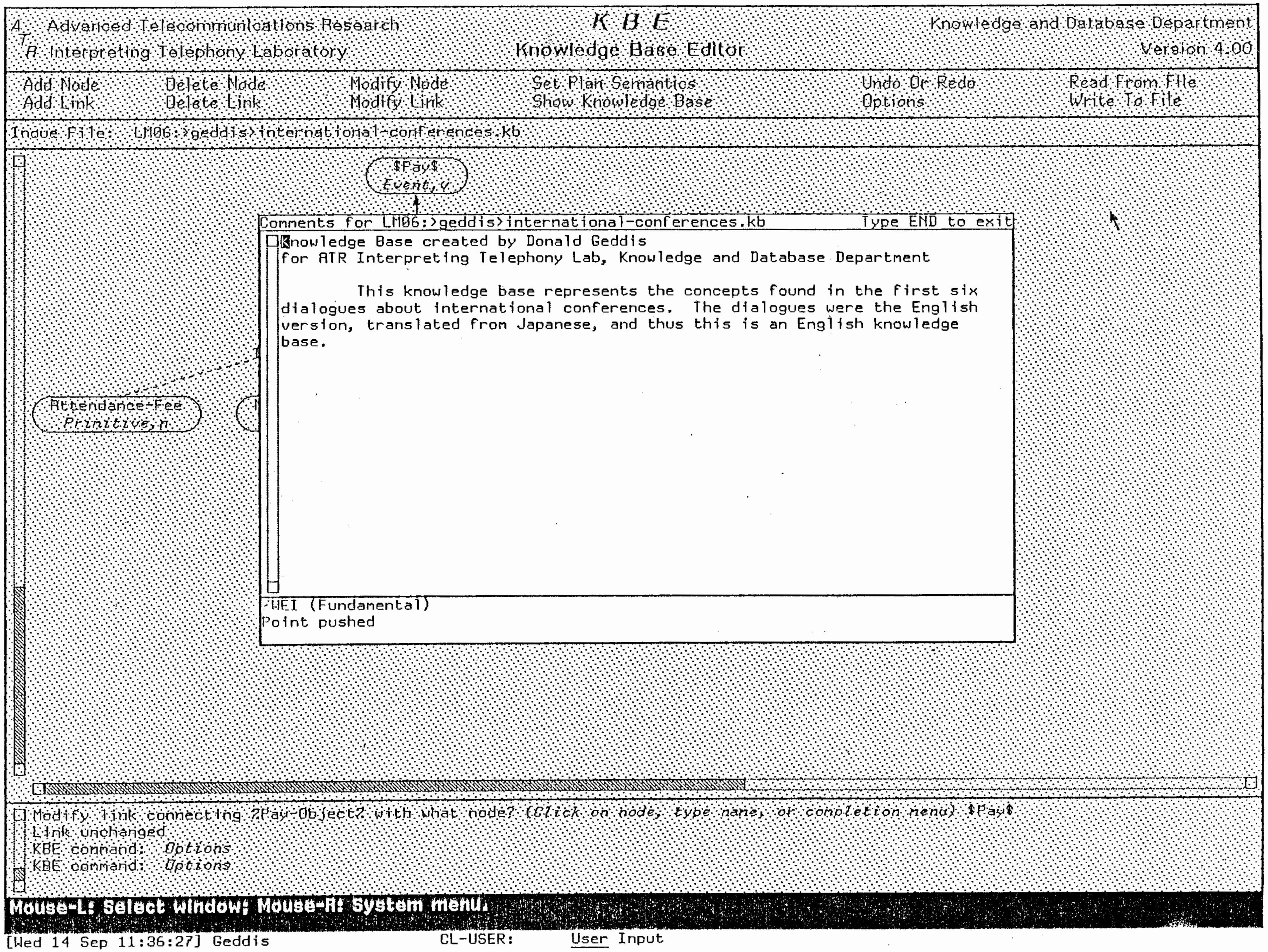
Check Validity of Knowledge Base

There are many global conventions about how a knowledge base ought to be structured. The KBE program enforces these constraints in two ways. The most common is simply by restricting the possible choices in a menu, for example in adding a link between two nodes. There, only the allowed link types are offered as possibilities.

The second checking mechanism, though, is for more difficult or computationally intensive operations. For example, although it is not currently implemented, this option would be the place to look for cycles in the knowledge base graph, which are not only incorrect but also might cause other parts of the program (namely the *Show Knowledge Base* drawing routine) to fail to work properly.

Currently, the Check Validity procedure performs eight tests. Four are for the form of node names: plan nodes must be @...@, functional nodes must be &...&, event nodes must be $...$, and class nodes must be %...%. There is also an imposed restriction on the connection possibilities between types of nodes: event nodes should only be connected to plan or class nodes, and value nodes should only be connected to class or primitive nodes. The last two tests relate to plan semantics: the semantic nodes should reside in one of the subtrees connected to the plan node by an instantiation link, and the direct link of the primitive node to its parent in that subtree should be of the same semantic type as the plan semantic's type.

When Check Validity detects an error of one of the above kinds, there are two possible responses it can have. In the first mode, it simply writes a summary of the error to a previously specified error file. In the second, interactive, mode, it brings up a menu dialogue that describes the error, and then offers to correct the problem. The proposed corrections for node names is renaming the node, while the proposed corrections for the other problems is typically deleting the bad link. You have the option of either performing the correction on the knowledge base or ignoring the error and continuing the check. In any case, all the corrections have the same status as a user modification, so they may all be undone in the reverse order (see *Undo Or Redo*).



```
Input File:  LM06:>geddis>international-conferences.kb

Comments for LM06:>geddis>international-conferences.kb          Type END to exit
Knowledge Base created by Donald Geddis
for ATR Interpreting Telephony Lab, Knowledge and Database Department

        This knowledge base represents the concepts found in the first six
dialogues about international conferences.  The dialogues were the English
version, translated from Japanese, and thus this is an English knowledge
base.

ZWEI (Fundamental)
Point pushed

Modify link connecting ?Pay-Object? with what node? (Click on node, type name, or completion menu) $Pay$
Link unchanged
KBE command:  Options
KBE command:  Options

[Wed 14 Sep 11:36:27] Geddis                CL-USER:      User Input
```

Figure 12: KBE screen – Edit Comments

Read From File

This command initializes the knowledge base and then loads a new one from a file. If the current knowledge base in memory has been modified but not saved, you are asked first if you wish to save it. (If yes, the program branches to the *Write To File* section as a subroutine.) If you specify a file name to read from that doesn't exist, the knowledge base is just initialized as a new file[8].

If the specified filename is actually found, the file is loaded. The structure is assumed to contain some lines of header comments that the program writes out (the name of the file, the date and time written, and the version of KBE that wrote it), followed by a blank line, followed by the knowledge base comments (see the *Edit Knowledge Base Comments* under the *Options* command), followed by a blank line, followed by the knowledge base data itself.

The program can automatically decide whether it is reading an Inoue or a KBE file[9]. If it is a KBE file the reading procedure is simple, because the information in the file exactly matches the data structures of the program. For an Inoue file, the part of speech information (i.e. whether a node is a noun, a verb, or something else), must be computed, because it is not stored in the file itself. In either format there is some limited error checking that can detect if the program doesn't understand an expression in the file.

The keyboard shortcut for this command is "R".

Write To File

This command can write either KBE or Inoue format files. In either case, it first writes a header giving the name of the file, the time and date it was written, the name of the KBE program, and the version number of the program. After that comes a blank line, and then the knowledge base comments (see *Edit Knowledge Base Comments* under the *Options* command). Then a blank line, and then the knowledge base itself.

For KBE files, the first piece of data is the form *(A KBE Knowledge Base)*

[8] This follows the convention of ZMACS.

[9] Because the first Lisp form in a KBE file is "(A KBE Knowledge Base)", so anything else is an Inoue file.

followed by a KBE form for each node in the following format: *(kbe-node :kbe-name* **name** *:kbe-data* **data** *:kbe-links* **links***)*, where the words in **boldface** are replaced by the particular information for the node. Nodes are written in alphabetical order sorted on the names.

Inoue files have sections separated into similar groups, so all the "make-primitive-node"s are together, etc.

See the appendix for an example of both file types.

The keyboard shortcut for this command is "W".

Technical Documentation

This section is only of interest to maintainers of the KBE Lisp code, or those who want to modify the program for some special purpose. It can safely be ignored by most users of the program.

Code

The program is built around a *Define-Program-Framework* shell. The screen is split into a series of horizontal window panes, with the second being the command menu. Commands are defined with a *Define-KBE-Command* macro definition. All of this is standard structure for large programs on the Symbolics, under Genera 7.1.

Typically, in the KBE program, the *Com*-Command routine serves only as a user interface, further specifying the exact nature of the command to be executed (for example, the node to draw from). Almost every command then calls another routine named Command-*Internal* which actually updates the data structures or performs the operation.

The Command-*Internal* routines often share a similar structure themselves. Those that change the knowledge base data structure first update the Undo stack with a form that can reverse the change they are about to make. (This ability to undo any operation is required of all commands that modify the knowledge base.) Then they actually perform the updating, and finally they print a message to the bottom window pane summarizing the change.

The undo stack is a list of Lisp forms that can be evaluated to reverse all modifications to the knowledge base, typically by calling routines named

Command-*Internal*. To return the data structures to the form they were in just after the last file operation, you should be able to execute a *(mapcar #'(lambda (x) (apply (car x) (cdr x))) undo-list)*.

For debugging purposes, I had added some code in the program that prints the values of internal variables at various interesting places. In the program listing, all this code is in UPPERCASE. Most users should never notice its existence. For your own purposes, however, you can enable the code: type SUSPEND to the Lisp listener, and then type *(debug-kbe)*. There are two choices, toggling the debugging printing, and reinitializing global variables. The second is only useful during program development, when program state variables (defined in the *Define-Program-Framework* form) change values. The first causes the program to print variable values at many points, and perhaps most usefully allows printing the knowledge base in a text format. (This is good to check things like cycles in the knowledge base.) To print in a text format, enable debugging with the *debug-kbe* routine, and then restart the program (with RESUME or ABORT). Then choose the *Show Knowledge Base* command, and select the option *Everything*. Unlike before, you now have a further choice, to show everything in text format. This can give a good idea of what the program thinks the knowledge base is like.

Data Structures

The format that KBE uses to store the knowledge base it edits is unfortunately rather inefficient, especially for large knowledge bases. It was developed as a simple and straightforward way of editing small example files, and is really not suited for large data bases. Thus it is often unnecessarily slow and wasteful of memory.

Most of the data structures serve two purposes: they store the information about the knowledge base, and they are used as lists to various user-interface routines. For the second purpose, they are constantly kept sorted alphabetically, and they are all lists of two-element lists, where the first element is a capitalized string version of the name of the item, and the second is the data (itself either an atom or another list). These were originally used in *dw:menu-choose* routines, but now are used in the same way for *dw:complete-from-sequence* or *accept*. This constant sorting is one source of inefficiency; the other is the serial search required because of the list structure: for large data bases a hash table should be used.

The list of all nodes is stored in the state variable *Node-List*. The form is *(("Node-1" node-1) ("Node-2" node-2) ...)*.

Information for a node is stored on the property list of the symbol corresponding to its name. (This is also a poor choice for large data bases: I should have used flavors and messages at the least.) The property *kbe-data* is of the form *(*node-type node-part-of-speech*)*, where node-type is one of the *Node-Types*, and node-part-of-speech is one of the *Parts-Of-Speech*. Links for the node are stored on the property *kbe-links*, and are of the form *((*"Node-1" *(*node-1 link-type-1 from-me-1 strength-1*) ...)*. Link-type can be either one of *Link-Types* (except *Semantic*), or a list of the form *(semantic* sem-type essential-p*)*, where **sem-type** is one of *All-Semantic-Links*, and **essential-p** is either *T* or *Nil* depending on whether this is an essential semantic link or not. **From-me** is either *T* or *Nil*, to give the direction of this link. **Strength** is a real number, usually between 0.0 and 1.0 inclusive.

All links between nodes are stored on the *kbe-links* list of both the from node and the to node, with identical data except for the **from-me** item, which is *T* in one case and *Nil* in the other. This is true even for non-hierarchical links, although in that case it doesn't matter which node has the *T* and which the *Nil*, even though they must be distinct.

Knowledge Base Types

This is the section of the program that is most likely to change over time. The allowed types of nodes and links are stored in state variables of the program. Most could have been initialized in the *Define-Program-Framework* macro, but I wanted a *LET** kind of sequential assignment rather than the *LET* parallel assignment the macro provides. Thus I cheated a little bit and did the initializations in my *kbe-initialize* routine, which gets called when the program is first selected in order to draw the title in the title window pane.

As in the knowledge base data structures above, all these lists are composed of further two-element lists, with the first element being the capitalized string name, and the second element being the corresponding lisp atom. By following the examples it should be easy to add further types of either nodes of links. Keep in mind, however, that the special handling of the constraints between various types is unfortunately spread out all over the program, and so it would be very difficult to create a new class of types, as opposed to simply adding some new instances of existing ones. By the way, these lists are also sorted, but only because I typed them in that way in the program. It makes them look better in menus, so if you add new types please put them in alphabetical order.

Special note: The semantic link types are three-place lists instead of two-place lists, where the third item is the Inoue three-letter[10] abbreviation for the type.

[10] And in very rare cases, four letters: "pred" for predicate, and "rest" for restriction.

Conclusion and Summary

A small English knowledge base was created, in the same way as the Japanese knowledge base is being created in the Department. Using this knowledge base, the association mechanism that was used for the Japanese knowledge base was checked to see whether it could be applied to the English knowledge base. The results confirmed that the association mechanism has no problems with the English knowledge base. However, as my English knowledge base is very small, I think it is necessary to check the association mechanism using a larger knowledge base of English.

To aid in editing and modifying language knowledge bases, in both English and Japanese, a graphical editing program called KBE was designed and written. The primary focus for the program was on ease of use and on a natural and intuitive user interface. In this, the program succeeds and meets its goals. The editing is not only much easier than typing the Lisp expressions directly, but also various kinds of error checking and other convenient functions help prevent mistakes in the knowledge base that otherwise are very difficult to detect. In addition, the complex graphing algorithm gives a significantly clearer view of the relations between concepts.

The program is beginning to be used with very large knowledge bases. Some thought was given to the special problems that large knowledge bases pose to such editing programs. In the internals of the program, the file input and output are not as fast as I would like once files become very big. In addition, the data structure is not as efficient as possible, although this actually is not too important in most uses of the program, because response time is still fast. The best aspect of dealing with large knowledge bases is the different options for drawing parts of the knowledge base. It was found that being able to specify very precisely what nodes were to be drawn was an enormous advantage in making the program useful and natural for the user.

Acknowledgements

The author would like to thank Dr. Akira Kurematsu, President of ATR Interpreting Telephony Research Laboratories, for allowing me to work in his laboratory and for supporting my research.

I am also very grateful to Mr. Tsuyoshi Morimoto, Head of the Knowledge and Database Department, who directed my internship and helped me both in my research and in my enjoyment of Japanese culture. I am especially indebted for the invitation to visit his house for a weekend early during my stay in Japan.

I wish to thank Mr. Naomi Inoue, who patiently explained the lab's knowledge base structures, and who offered many helpful comments during the development of the program.

I also would like to thank all the employees of ATR who greeted me in a very friendly way and who made me feel comfortable and at ease, both inside and outside of work. In particular, I am grateful to: Mr. Hashimoto, for offering travel advice for many weekends; Mr. Hara, for organizing ATR's soccer club and inviting me to participate; and Mr. Matsuda, who allowed me to attend Aikido practice in Osakajokoen's Shudokan Budo-hall.

Finally, Japan as a country has my appreciation for the tastiness of its food, the significance of its culture, the natural beauty of its terrain and the kindness of its people.

Appendix

English Knowledge Base listing

1. KBE Format
2. Inoue Format

```
;;;   KBE file LM06:>Geddis>international-conferences.kbe-kb written at 9/14/88 12:53:17
;;;   By the Knowledge Base Editor [KBE], Version 4.00

; Knowledge Base created by Donald Geddis
; for ATR Interpreting Telephony Lab, Knowledge and Database Department
;
;       This knowledge base represents the concepts found in the first six
; dialogues about international conferences.  The dialogues were the English
; version, translated from Japanese, and thus this is an English knowledge
; base.

(A KBE Knowledge Base)

(kbe-node
  :kbe-name  $BRING$
  :kbe-data  (EVENT VERB)
  :kbe-links (("%Bring-Obj%" (%BRING-OBJ% (SEMANTIC OBJECT T) NIL 1.0))
 ("%Bring-Pred%" (%BRING-PRED% (SEMANTIC PREDICATE T) NIL 1.0)))
  )

(kbe-node
  :kbe-name  $FILL-OUT$
  :kbe-data  (EVENT VERB)
  :kbe-links (("%Fill-Out-Obj%" (%FILL-OUT-OBJ% (SEMANTIC OBJECT T) NIL 1.0))
 ("%Fill-Out-Pred%" (%FILL-OUT-PRED% (SEMANTIC PREDICATE T) NIL 1.0)))
  )

(kbe-node
  :kbe-name  $PAY$
  :kbe-data  (EVENT VERB)
  :kbe-links (("%Means-Of-Payment%" (%MEANS-OF-PAYMENT% (SEMANTIC TOOL NIL) NIL 0.5))
 ("%Pay-Object%" (%PAY-OBJECT% (SEMANTIC OBJECT T) NIL 1.0))
 ("%Pay-Pred%" (%PAY-PRED% (SEMANTIC PREDICATE T) NIL 1.0)))
  )

(kbe-node
  :kbe-name  $SEND$
  :kbe-data  (EVENT VERB)
  :kbe-links (("%Send-Obj%" (%SEND-OBJ% (SEMANTIC OBJECT T) NIL 1.0))
 ("%Send-Pred%" (%SEND-PRED% (SEMANTIC PREDICATE T) NIL 1.0)))
  )

(kbe-node
  :kbe-name  %ATTEND-PRED%
  :kbe-data  (CLASS NIL)
  :kbe-links (("Attend" (ATTEND ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %BRING-OBJ%
  :kbe-data  (CLASS NOUN)
  :kbe-links (("$Bring$" ($BRING$ (SEMANTIC OBJECT T) T 1.0)) ("Students" (STUDENTS ISA NIL 1.0))
)
  )

(kbe-node
  :kbe-name  %BRING-PRED%
  :kbe-data  (CLASS VERB)
  :kbe-links (("$Bring$" ($BRING$ (SEMANTIC PREDICATE T) T 1.0)) ("Bring" (BRING ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %FILL-OUT-OBJ%
  :kbe-data  (CLASS NOUN)
  :kbe-links (("$Fill-Out$" ($FILL-OUT$ (SEMANTIC OBJECT T) T 1.0))
 ("Application" (APPLICATION ISA NIL 1.0))
 ("Registration-Form" (REGISTRATION-FORM ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %FILL-OUT-PRED%
  :kbe-data  (CLASS VERB)
  :kbe-links (("$Fill-Out$" ($FILL-OUT$ (SEMANTIC PREDICATE T) T 1.0)) ("Fill-Out" (FILL-OUT ISA
NIL 1.0)))
  )
```

```
(kbe-node
  :kbe-name  %MEANS-OF-PAYMENT%
  :kbe-data  (CLASS NOUN)
  :kbe-links (("$Pay$" ($PAY$ (SEMANTIC TOOL NIL) T 0.5)) ("Bank-Transfer" (BANK-TRANSFER ISA NIL
 1.0))
 ("Check" (CHECK ISA NIL 1.0)) ("Credit-Card" (CREDIT-CARD ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %PAY-OBJECT%
  :kbe-data  (CLASS NOUN)
  :kbe-links (("$Pay$" ($PAY$ (SEMANTIC OBJECT T) T 1.0)) ("Attendance-Fee" (ATTENDANCE-FEE ISA N
IL 0.5))
 ("Manuscript-Fee" (MANUSCRIPT-FEE ISA NIL 0.5)) ("Money" (MONEY ISA NIL 1.0))
 ("Registration-Fee" (REGISTRATION-FEE ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %PAY-PRED%
  :kbe-data  (CLASS VERB)
  :kbe-links (("$Pay$" ($PAY$ (SEMANTIC PREDICATE T) T 1.0)) ("Pay" (PAY ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %PERSON-WHO-ATTENDS%
  :kbe-data  (CLASS NIL)
  :kbe-links (("Name" (NAME (SEMANTIC PART NIL) NIL 1.0)) ("Wife" (WIFE ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %SEND-OBJ%
  :kbe-data  (CLASS NOUN)
  :kbe-links (("$Send$" ($SEND$ (SEMANTIC OBJECT T) T 1.0))
 ("Registration-Form" (REGISTRATION-FORM ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %SEND-PRED%
  :kbe-data  (CLASS VERB)
  :kbe-links (("$Send$" ($SEND$ (SEMANTIC PREDICATE T) T 1.0)) ("Send" (SEND ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  %WHAT-IS-ATTENDED%
  :kbe-data  (CLASS NIL)
  :kbe-links (("Conference" (CONFERENCE ISA NIL 1.0)))
  )

(kbe-node
  :kbe-name  |1-2 TOKUIMACHI, HIGASHI-KU|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Section" (SECTION VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |10,000 YEN|
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Amount" (AMOUNT VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |100 DOLLARS|
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Amount" (AMOUNT VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |16,000 YEN|
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Amount" (AMOUNT VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |5,000 YEN|
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Amount" (AMOUNT VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |50 DOLLARS|
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Amount" (AMOUNT VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |5:30 P.M.|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Time" (TIME VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  5TH
  :kbe-data  (VALUE NIL)
  :kbe-links (("Date" (DATE VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |6 P.M.|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Time" (TIME VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |6-23 CHAYAMACHI, KITA-KU|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Section" (SECTION VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  8TH
  :kbe-data  (VALUE NIL)
  :kbe-links (("Date" (DATE VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  |9:30 A.M.|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Time" (TIME VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  ADDRESS
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("City" (CITY (SEMANTIC PART NIL) NIL 1.0)) ("Section" (SECTION (SEMANTIC PART NIL)
 NIL 1.0)))
  )

(kbe-node
  :kbe-name  AMOUNT
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("10,000 Yen" (|10,000 YEN| VALUE NIL 1.0)) ("100 Dollars" (|100 DOLLARS| VALUE NIL
 1.0))
 ("16,000 Yen" (|16,000 YEN| VALUE NIL 1.0)) ("5,000 Yen" (|5,000 YEN| VALUE NIL 1.0))
 ("50 Dollars" (|50 DOLLARS| VALUE NIL 1.0))
 ("Registration-Fee" (REGISTRATION-FEE (SEMANTIC PART NIL) T 1.0)))
  )

(kbe-node
  :kbe-name  APPLICATION
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Fill-Out-Obj%" (%FILL-OUT-OBJ% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  ATTEND
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("%Attend-Pred%" (%ATTEND-PRED% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  ATTENDANCE-FEE
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Pay-Object%" (%PAY-OBJECT% ISA T 0.5)))
  )

(kbe-node
  :kbe-name  BANK-TRANSFER
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Means-Of-Payment%" (%MEANS-OF-PAYMENT% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  BRING
  :kbe-data  (PRIMITIVE VERB)
  :kbe-links (("%Bring-Pred%" (%BRING-PRED% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  CHECK
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Means-Of-Payment%" (%MEANS-OF-PAYMENT% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  CITY
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("Address" (ADDRESS (SEMANTIC PART NIL) T 1.0)) ("Osaka" (OSAKA VALUE NIL 1.0)))
  )

(kbe-node
  :kbe-name  CONFERENCE
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("%What-Is-Attended%" (%WHAT-IS-ATTENDED% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  CREDIT-CARD
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Means-Of-Payment%" (%MEANS-OF-PAYMENT% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  DATE
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("5th" (5TH VALUE NIL 1.0)) ("8th" (8TH VALUE NIL 1.0))
 ("December 25th" (|DECEMBER 25TH| VALUE NIL 1.0)))
  )

(kbe-node
  :kbe-name  |DECEMBER 25TH|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Date" (DATE VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  EIGHT
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Number" (NUMBER VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  ENGLISH
  :kbe-data  (VALUE NIL)
  :kbe-links (("Language" (LANGUAGE VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  FILL-OUT
  :kbe-data  (PRIMITIVE VERB)
  :kbe-links (("%Fill-Out-Pred%" (%FILL-OUT-PRED% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  FIVE
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Number" (NUMBER VALUE T 1.0)))
  )
```

```
(kbe-node
  :kbe-name  JAPANESE
  :kbe-data  (VALUE NIL)
  :kbe-links (("Language" (LANGUAGE VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  LANGUAGE
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("English" (ENGLISH VALUE NIL 1.0)) ("Japanese" (JAPANESE VALUE NIL 1.0)))
  )

(kbe-node
  :kbe-name  MANUSCRIPT-FEE
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Pay-Object%" (%PAY-OBJECT% ISA T 0.5)))
  )

(kbe-node
  :kbe-name  |MAYUMI SUZUKI|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Name" (NAME VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  MONEY
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Pay-Object%" (%PAY-OBJECT% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  NAME
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("%Person-Who-Attends%" (%PERSON-WHO-ATTENDS% (SEMANTIC PART NIL) T 1.0))
 ("Mayumi Suzuki" (|MAYUMI SUZUKI| VALUE NIL 1.0))
 ("Taro Shimizu" (|TARO SHIMIZU| VALUE NIL 1.0)))
  )

(kbe-node
  :kbe-name  NUMBER
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("Eight" (EIGHT VALUE NIL 1.0)) ("Five" (FIVE VALUE NIL 1.0))
 ("Registration-Form" (REGISTRATION-FORM (SEMANTIC ATTRIBUTE-VALUE-OF-OBJECT NIL) T 0.5))
 ("Students" (STUDENTS (SEMANTIC ATTRIBUTE-VALUE-OF-OBJECT NIL) T 0.5))
 ("Twenty" (TWENTY VALUE NIL 1.0)) ("Twenty-One" (TWENTY-ONE VALUE NIL 1.0)))
  )

(kbe-node
  :kbe-name  OFFICE
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links NIL
  )

(kbe-node
  :kbe-name  OSAKA
  :kbe-data  (VALUE NIL)
  :kbe-links (("City" (CITY VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  PAY
  :kbe-data  (PRIMITIVE VERB)
  :kbe-links (("%Pay-Pred%" (%PAY-PRED% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  QUERY
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links NIL
  )

(kbe-node
  :kbe-name  REGISTRATION-FEE
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Pay-Object%" (%PAY-OBJECT% ISA T 1.0)) ("Amount" (AMOUNT (SEMANTIC PART NIL) NIL
 1.0)))
```

```
  )

(kbe-node
  :kbe-name  REGISTRATION-FORM
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Fill-Out-Obj%" (%FILL-OUT-OBJ% ISA T 1.0)) ("%Send-Obj%" (%SEND-OBJ% ISA T 1.0))
 ("Number" (NUMBER (SEMANTIC ATTRIBUTE-VALUE-OF-OBJECT NIL) NIL 0.5)))
  )

(kbe-node
  :kbe-name  SECTION
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("1-2 Tokuimachi, Higashi-Ku" (|1-2 TOKUIMACHI, HIGASHI-KU| VALUE NIL 1.0))
 ("6-23 Chayamachi, Kita-Ku" (|6-23 CHAYAMACHI, KITA-KU| VALUE NIL 1.0))
 ("Address" (ADDRESS (SEMANTIC PART NIL) T 1.0)))
  )

(kbe-node
  :kbe-name  SEND
  :kbe-data  (PRIMITIVE VERB)
  :kbe-links (("%Send-Pred%" (%SEND-PRED% ISA T 1.0)))
  )

(kbe-node
  :kbe-name  STUDENTS
  :kbe-data  (PRIMITIVE NOUN)
  :kbe-links (("%Bring-Obj%" (%BRING-OBJ% ISA T 1.0))
 ("Number" (NUMBER (SEMANTIC ATTRIBUTE-VALUE-OF-OBJECT NIL) NIL 0.5)))
  )

(kbe-node
  :kbe-name  |TARO SHIMIZU|
  :kbe-data  (VALUE NIL)
  :kbe-links (("Name" (NAME VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  TIME
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("5:30 P.M." (|5:30 P.M.| VALUE NIL 1.0)) ("6 P.M." (|6 P.M.| VALUE NIL 1.0))
 ("9:30 A.M." (|9:30 A.M.| VALUE NIL 1.0)))
  )

(kbe-node
  :kbe-name  TWENTY
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Number" (NUMBER VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  TWENTY-ONE
  :kbe-data  (VALUE NOUN)
  :kbe-links (("Number" (NUMBER VALUE T 1.0)))
  )

(kbe-node
  :kbe-name  WIFE
  :kbe-data  (PRIMITIVE NIL)
  :kbe-links (("%Person-Who-Attends%" (%PERSON-WHO-ATTENDS% ISA T 1.0)))
  )
```

```
;;;   Inoue file LM06:>geddis>international-conferences.kb written at 9/14/88 11:36:41
;;;   By the Knowledge Base Editor [KBE], Version 4.00

; Knowledge Base created by Donald Geddis
; for ATR Interpreting Telephony Lab, Knowledge and Database Department
;
;      This knowledge base represents the concepts found in the first six
; dialogues about international conferences.  The dialogues were the English
; version, translated from Japanese, and thus this is an English knowledge
; base.



;;;
;;;   Primitive nodes
;;;

(make-primitive-node 'ADDRESS)
(make-primitive-node 'AMOUNT)
(make-primitive-node 'APPLICATION)
(make-primitive-node 'ATTEND)
(make-primitive-node 'ATTENDANCE-FEE)
(make-primitive-node 'BANK-TRANSFER)
(make-primitive-node 'BRING)
(make-primitive-node 'CHECK)
(make-primitive-node 'CITY)
(make-primitive-node 'CONFERENCE)
(make-primitive-node 'CREDIT-CARD)
(make-primitive-node 'DATE)
(make-primitive-node 'FILL-OUT)
(make-primitive-node 'LANGUAGE)
(make-primitive-node 'MANUSCRIPT-FEE)
(make-primitive-node 'MONEY)
(make-primitive-node 'NAME)
(make-primitive-node 'NUMBER)
(make-primitive-node 'OFFICE)
(make-primitive-node 'PAY)
(make-primitive-node 'QUERY)
(make-primitive-node 'REGISTRATION-FEE)
(make-primitive-node 'REGISTRATION-FORM)
(make-primitive-node 'SECTION)
(make-primitive-node 'SEND)
(make-primitive-node 'STUDENTS)
(make-primitive-node 'TIME)
(make-primitive-node 'WIFE)



;;;
;;;   Value nodes
;;;

(make-value-node '|1-2 TOKUIMACHI, HIGASHI-KU|)
(make-value-node '|10,000 YEN|)
(make-value-node '|100 DOLLARS|)
(make-value-node '|16,000 YEN|)
(make-value-node '|5,000 YEN|)
(make-value-node '|50 DOLLARS|)
(make-value-node '|5:30 P.M.|)
(make-value-node '5TH)
(make-value-node '|6 P.M.|)
(make-value-node '|6-23 CHAYAMACHI, KITA-KU|)
(make-value-node '8TH)
(make-value-node '|9:30 A.M.|)
(make-value-node '|DECEMBER 25TH|)
(make-value-node 'EIGHT)
(make-value-node 'ENGLISH)
(make-value-node 'FIVE)
(make-value-node 'JAPANESE)
(make-value-node '|MAYUMI SUZUKI|)
(make-value-node 'OSAKA)
(make-value-node '|TARO SHIMIZU|)
(make-value-node 'TWENTY)
(make-value-node 'TWENTY-ONE)
```

```
;;;
;;;   Class nodes
;;;

(make-class-node '%ATTEND-PRED% :hasa-list '((ATTEND 1.0)))
(make-class-node '%BRING-OBJ% :hasa-list '((STUDENTS 1.0)))
(make-class-node '%BRING-PRED% :hasa-list '((BRING 1.0)))
(make-class-node '%FILL-OUT-OBJ% :hasa-list '((APPLICATION 1.0) (REGISTRATION-FORM 1.0)))
(make-class-node '%FILL-OUT-PRED% :hasa-list '((FILL-OUT 1.0)))
(make-class-node '%MEANS-OF-PAYMENT% :hasa-list '((BANK-TRANSFER 1.0) (CHECK 1.0) (CREDIT-CARD 1.
0)))
(make-class-node '%PAY-OBJECT% :hasa-list '((ATTENDANCE-FEE 0.5) (MANUSCRIPT-FEE 0.5) (MONEY 1.0)
 (REGISTRATION-FEE 1.0)))
(make-class-node '%PAY-PRED% :hasa-list '((PAY 1.0)))
(make-class-node '%PERSON-WHO-ATTENDS% :hasa-list '((WIFE 1.0)))
(make-class-node '%SEND-OBJ% :hasa-list '((REGISTRATION-FORM 1.0)))
(make-class-node '%SEND-PRED% :hasa-list '((SEND 1.0)))
(make-class-node '%WHAT-IS-ATTENDED% :hasa-list '((CONFERENCE 1.0)))



;;;
;;;   Event nodes
;;;

(make-event-node '$BRING$ :imi-list '((OBJ %BRING-OBJ% 1.0) (PRED %BRING-PRED% 1.0)) :hissu-kaku
'(OBJ PRED))
(make-event-node '$FILL-OUT$ :imi-list '((OBJ %FILL-OUT-OBJ% 1.0) (PRED %FILL-OUT-PRED% 1.0)) :hi
ssu-kaku '(OBJ PRED))
(make-event-node '$PAY$ :imi-list '((TOO %MEANS-OF-PAYMENT% 0.5) (OBJ %PAY-OBJECT% 1.0) (PRED %PA
Y-PRED% 1.0)) :hissu-kaku '(OBJ PRED))
(make-event-node '$SEND$ :imi-list '((OBJ %SEND-OBJ% 1.0) (PRED %SEND-PRED% 1.0)) :hissu-kaku '(O
BJ PRED))



;;;
;;;   Functional nodes
;;;



;;;
;;;   Plan nodes
;;;



;;;
;;;   Plan links
;;;



;;;
;;;   Assertion links
;;;



;;;
;;;   Association links
;;;



;;;
;;;   Constraint links
;;;
```

```
;;;
;;;   Imi links with primitive nodes
;;;

(make-imi-link-connection 'AMOUNT 'REGISTRATION-FEE 'PAR 1.0)
(make-imi-link-connection 'CITY 'ADDRESS 'PAR 1.0)
(make-imi-link-connection 'NAME '%PERSON-WHO-ATTENDS% 'PAR 1.0)
(make-imi-link-connection 'NUMBER 'REGISTRATION-FORM 'AVO 0.5)
(make-imi-link-connection 'NUMBER 'STUDENTS 'AVO 0.5)
(make-imi-link-connection 'SECTION 'ADDRESS 'PAR 1.0)



;;;
;;;   Value links
;;;

(make-value-link-connection '|1-2 TOKUIMACHI, HIGASHI-KU| 'SECTION)
(make-value-link-connection '|10,000 YEN| 'AMOUNT)
(make-value-link-connection '|100 DOLLARS| 'AMOUNT)
(make-value-link-connection '|16,000 YEN| 'AMOUNT)
(make-value-link-connection '|5,000 YEN| 'AMOUNT)
(make-value-link-connection '|50 DOLLARS| 'AMOUNT)
(make-value-link-connection '|5:30 P.M.| 'TIME)
(make-value-link-connection '5TH 'DATE)
(make-value-link-connection '|6 P.M.| 'TIME)
(make-value-link-connection '|6-23 CHAYAMACHI, KITA-KU| 'SECTION)
(make-value-link-connection '8TH 'DATE)
(make-value-link-connection '|9:30 A.M.| 'TIME)
(make-value-link-connection '|DECEMBER 25TH| 'DATE)
(make-value-link-connection 'EIGHT 'NUMBER)
(make-value-link-connection 'ENGLISH 'LANGUAGE)
(make-value-link-connection 'FIVE 'NUMBER)
(make-value-link-connection 'JAPANESE 'LANGUAGE)
(make-value-link-connection '|MAYUMI SUZUKI| 'NAME)
(make-value-link-connection 'OSAKA 'CITY)
(make-value-link-connection '|TARO SHIMIZU| 'NAME)
(make-value-link-connection 'TWENTY 'NUMBER)
(make-value-link-connection 'TWENTY-ONE 'NUMBER)


;;;
;;;   Isa links
;;;
```