THE INFERENCE MACHINE LABORATORY: GRAPHIC TOOLS FOR KNOWLEDGE MANAGEMENT

J.W. Lewis, Ph.D.

Artificial Intelligence Department
Martin Marietta Laboratories
1450 South Rolling Road
Baltimore, MD 21227

ABSTRACT

The Inference Machine Laboratory is a collection of experiments in applying graphic interfaces to various types of knowledge bases. Each experiment involves a canonical representation, invertible transformations into multiple representations, and multiple directly manipulable views of those representations. Initial experiments include RULE*CALC (simple production rules), HAPStation (OPS5-like production rules), RFIX (diagnostics), and TIMLS (frames in PROLOG).

KEYWORDS: expert systems, intelligent interfaces, knowledge acquisition, direct manipulation

INTRODUCTION

The major barrier to successful expert systems development continues to be the acquisition, review, restructuring, and long-term maintenance of large knowledge bases involving complex relationships [1]. Meaningful military and industrial expert systems are expected to require as many as 10,000 "rules" [2], domain coverage of better than 95 %, and an error rate of less than 0.1 %. To achieve these performance figures -- perhaps one to two orders of magnitude beyond the current state of the art -- the next generation of knowledge management tools must enable each individual involved in designing, building, using, and maintaining knowledge bases to view, understand, and manipulate their contents in an intuitive manner.

For simple interactive systems, adequate tools and techniques are already available. The direct manipulation of icons has already lead to successful icon-based interfaces for commercial systems such as the XEROX Star and the Apple MacIntosh [3]. Research activities, such as those in the MIT Media Graphics Laboratory, have shown impressive capabilities for text and video interfaces [4]. More recently, tools such as UNITS [5] and GEN-X [6] have provided effective interfaces to knowledge bases in expert systems.

THE INFERENCE MACHINE LABORATORY

The Inference Machine Laboratory (IML) addresses these performance goals for expert systems by providing each knowledge-base user with a tailored set of directly manipulable views of the knowledge base and by maintaining consistency among the various views. Over the last two years, a family of successively more complex knowledge management systems has been constructed. The early systems have involved simple production rule knowledge bases, and the later systems are based on predicate calculus representations.

Physically, the laboratory consists of two VAX computers, several LISP machines, a color monitor, a color video projector, several mice, a foot-mouse, a DECTALK voice generator, and a Polhemus 3-D graphics pointing device. The half-dozen individuals interacting with the knowledge base are grouped around a small table in front of the projection screen. They are able to select views, make queries against the knowledge base, and modify the knowledge base using the interactive graphics devices.

All of the software systems in the laboratory fit into a common framework (Fig. 1), which supports various kinds of graphic input/output, logic-based knowledge representations, and natural language input/output. In the IML, each system user works with a particular set of windows on the knowledge base. These windows are defined by

- Virtual cameras - to generate shaded images, schematics, tables, graphs, trees, and other diagrams
- Views - to specify the particular image generated by defining the location of the user in the knowledge base,
- Filters - to determine the granularity, or level of detail, in a particular view.

Once a view is presented, the user can modify the knowledge base by pointing at particular elements of the view and indicating changes.

RULE*CALC

The simplest and earliest project in the laboratory is RULE*CALC, a VISI-CALC-style development environment for EMYCIN-class production rules with uncertainty [7]. The rules are laid out in a spreadsheet-like tableau.

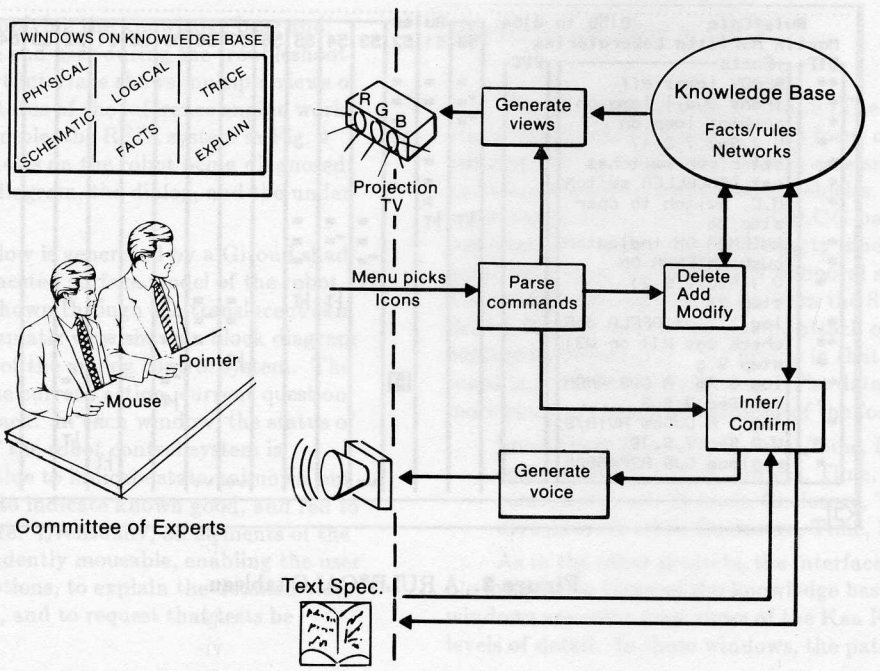

Figure 1. The Inference Machine Laboratory framework

(See Fig. 2 for a few of the rules in a 300-rule system for FAA radar trouble-shooting.) Each row in the tableau corresponds to a fact in the rules, and each column represents one rule. Rules are defined by entering a symbol in the appropriate column, which indicates how that fact is involved in the particular rule. A blank indicates that the fact is not involved in the rule at all; an = or ~= symbol indicates that the fact is a positive or negative clause in the lefthand side of the rule, and an :-T or :-F symbol indicates that the fact is asserted or negated when that particular rule fires. The user modifies the table by pointing to the appropriate entry with a mouse and hitting function keys for setting symbols in the table, cutting/pasting rules (rows), cutting/pasting facts (columns), and looking at different windows on the set of rules.

Each fact defines a simple object which could have one or more of three associated action procedures (methods): on-demand (triggered when a value is requested), on-true (triggered when the fact is asserted true), and on-false (triggered when the fact is asserted false). If the system is run from a terminal, the methods type out messages on the screen and elicit responses from the keyboard. If the system is run by calling in on a telephone, the methods generate a voice over the phone and elicit responses from the telephone keypad.

When the system executes, the state of the inference engine and the resulting dialog can be displayed in a pair of windows in the rule-debugging screen. One window shows the interactive dialog and the other displays the rule stack along with the facts involved in those rules. Altogether, in RULE*CALC there are five fixed views: the rule-edit tableau, fact-edit tableau, method definition, interactive dialog, and rule-debugging.

HAPStation

The second system is HAPStation, a somewhat more complex inference engine with a simpler interactive interface. HAPStation runs on a SYMBOLICS LISP workstation in COMMON LISP (Fig. 3). HAPS is a forward-chaining, production-rule-based language similar in style to OPS4, GRAPES, and OPS83 [8,9]. Like other forward-chaining production rule systems, HAPS is composed of two memories -- a working memory (WM) and a production rule memory (PM) -- with an accompanying interpreter. The working memory elements are composed of a sequence of terms in parentheses, e.g., "(this is a working memory element)." The production rules are composed of a lefthand side (also called the LHS, antecedent, or IF-part) and a righthand side (also called the RHS, consequent, or THEN-part). The LHS of each rule is composed of a sequence of positive and negative clauses, which are in turn composed of the patterns to be matched against the WM. The RHS of each rule is composed of a sequence of action terms involving making and removing working elements, computing expressions, and input/output.

The interpreter cycles through a recognize-act cycle in which it first searches for all working memory elements that match the clauses in the LHS of the rules. The resulting set of rules is called the conflict set (CS), and the rule that fires is selected from the CS by a programmable conflict-resolution strategy. When the rule fires, the terms in its RHS are executed in sequence. This cycle continues indefinitely until the conflict set is empty or a rule executes a HALT action.

The interactive interface is composed of a set of windows on the memories in the production rule interpreter.

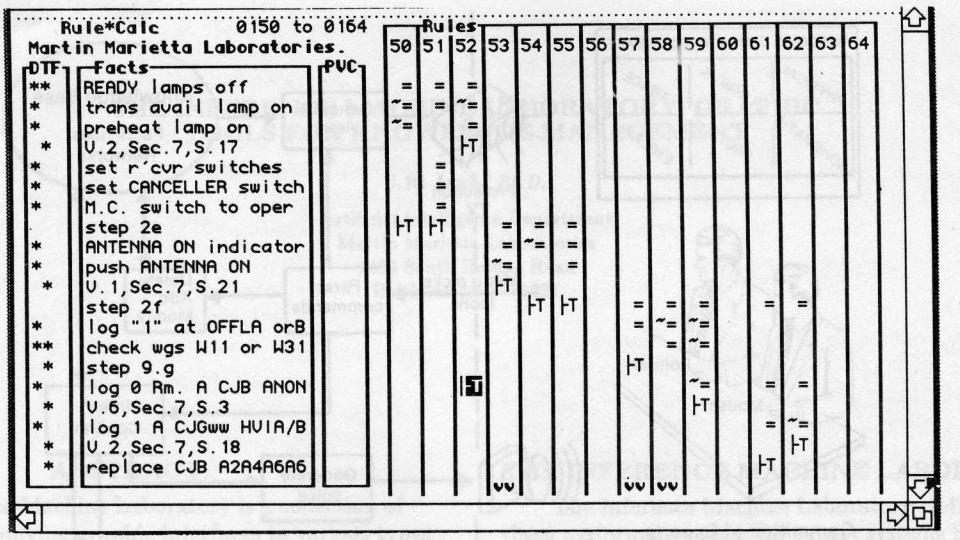

Figure 2. A RULE*CALC tableau

The central window (or HAPS window) contains the normal sequential dialog with the interpreter. The HAPSedit window enables the user to modify the rules through the ZMACS "smart" editor on the LISP machines. Other windows display the WM, the CS, and the possible matches with the LHS of a rule. Many of the elements of the screen are "mousable" so that the user can call up a pop-up menu of actions (run, stop, etc.) and a menu of the rule names against which to apply those actions.

INTEGRATED DIAGNOSTIC SYSTEMS

The third project area is IDS (Integrated Diagnostic Systems). This project is focussed on a specific application: isolating and addressing faults in complex systems. In particular, the project addresses intermittents, multiple faults, consequent faults, design errors, unusual operating modes, and other failures that challenge teams of experts. The IDS has two interfaces, one that serves the domain

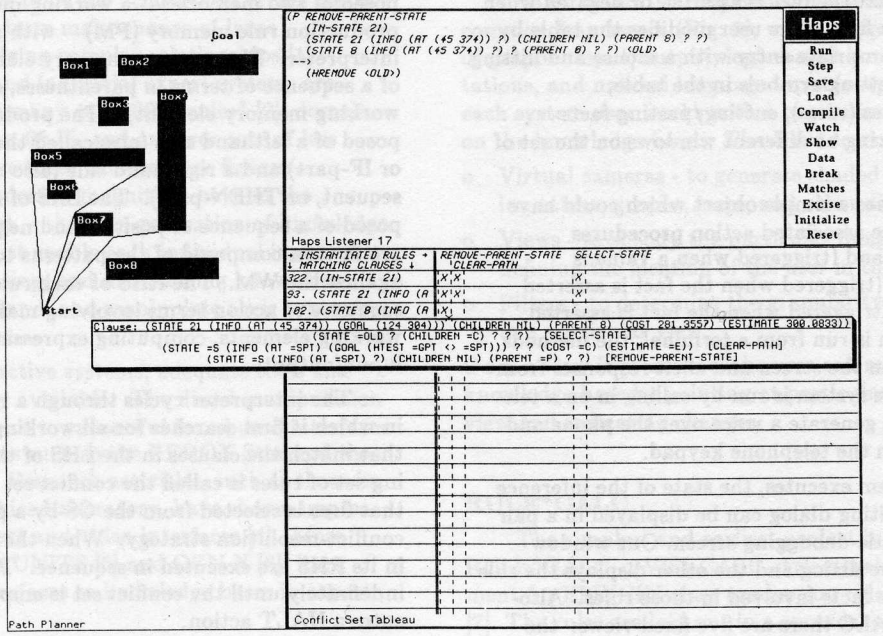

Figure 3. A HAPStation screen

expert in building or applying the knowledge base and another that serves the end user during the troubleshooting process. The expert interface shows multiple views of the device and of the status of the inference engine working against it. For example, the RFIX system in Fig. 4 shows six distinct windows on the robot being diagnosed, the robot's schematic diagram, the dialog, and the underlying inference engine.

The physical window is generated by a Giroud shading algorithm from a faceted surface model of the robot. The drive motors are shown through the translucent skin of the robot. The schematic view shows a block diagram of the LSI11 CPU and of the analog control system. The other windows show the current action, current question, and the current rule stack. In each window, the status of each of the elements of the robot control system is indicated by a color: blue to indicate state unknown but presumed good, green to indicate known good, and red to indicate a known failure. Eventually, all elements of the display will be independently mousable, enabling the user to call up parts descriptions, to explain the detailed state of the indicated object, and to request that tests be applied to the object.

PROLOG QUERY TABLE

The final project area is TIMLS (The Inference Machine Laboratory System). The focus of TIMLS is building and maintaining situation assessment or planning knowledge bases for autonomous vehicles. The current knowledge base is a complex PROLOG-based frame system that implements simple property inheritance and temporal logic using the method of temporal arguments [10]. The model scenario was drawn from the September 1943 British X-craft midget submarine attack on the German battleship Tirpitz [11]. All events in that attack are captured in a PROLOG data base. The data base includes more than one hundred relations of the form

break_clear (X-craft, Barrier, Time, Flags).
come_out_to (X-craft, Object, Time, Flags).
come_out_from (X-craft, Enclosure, Time, Flags).
dive_into (X-craft, Enclosure, Time, Flags) . . .

As in the other projects, the interface provides multiple interactive views of the knowledge base. Two of the windows are color-map views of the Kaa Fjord at different levels of detail. In those windows, the paths and positions

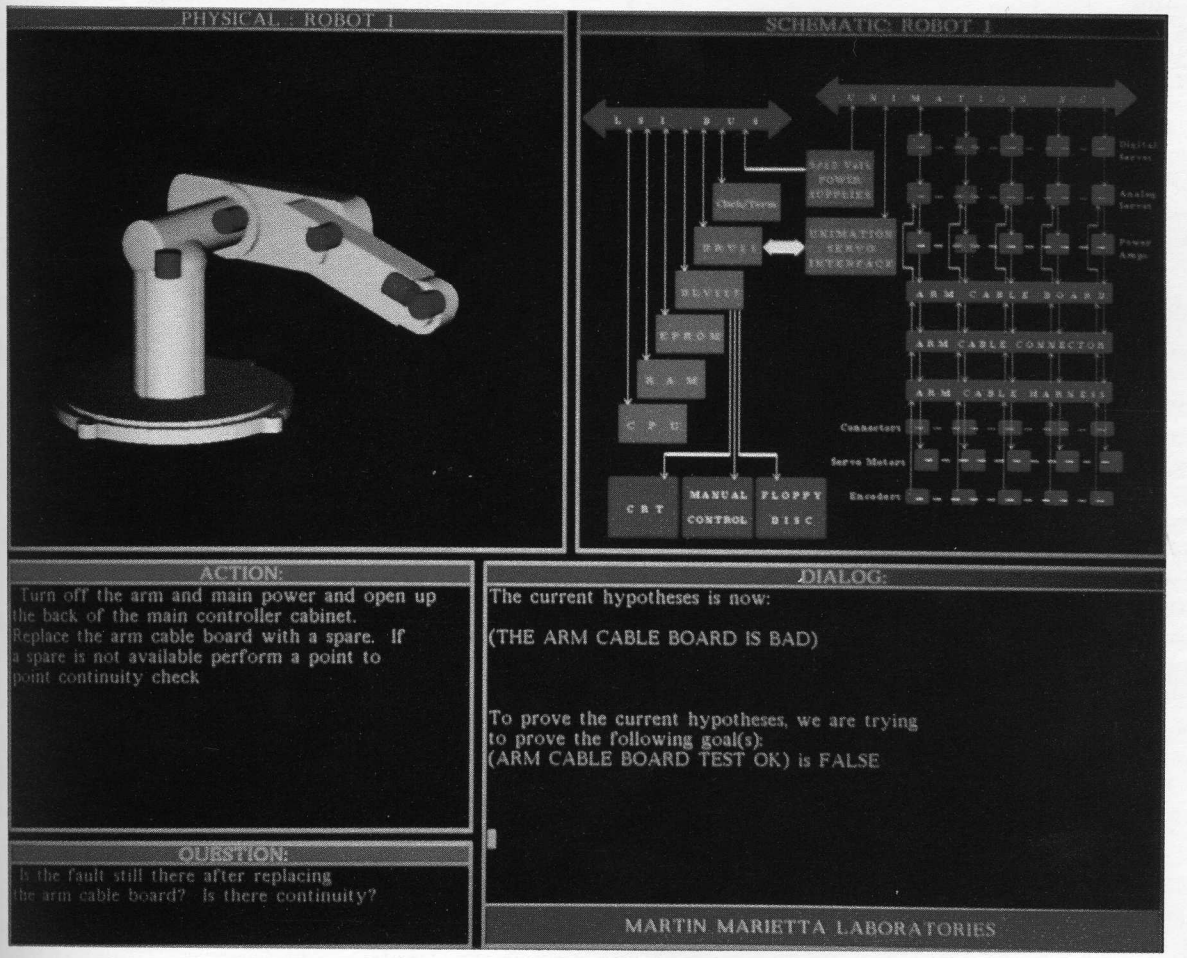

Figure 4. RFIX: The Robot Troubleshooting System

of the German and British ships are shown relative to obstacles such as submarine nets and buoys. The other windows contain the raw PROLOG, an English-language description of the weather generated from case frames, and the PROLOG query table (PQT).

Like RULE*CALC, the PQT follows the VISI-CALC paradigm. It is a close relative of the Query By Example (QBE) System [12] and PROLOG implementations of it [13]. The recently implemented PQT consists of two separate windows (Fig. 5). The upper window shows the current constraints on subsequent queries. Each constraint involves fixed values for one or more of the binary relations defined on the objects in the data base by the frame system. For the example given, the constraints are that the time is 9/22/1943 and that some number of X craft are in the Kaa Fjord.

SITUATION REPORT

Partly cloudy sky today will cover the upper section of Norway and showers will occur. The winds will be from the NW at 15 to 20 knots. The temperature will reach 35 to 40 degrees and visibility will be moderate to good.

PROLOG QUERY TABLE

Constraints...			
TIME	ISA	LOCATION	
X ON 9/22/1943	X-CRAFT	KAA-FJORD	
Query/Response...			
OBJECT	SINK	LOCATION	TIME
X-6	UNDERWATER	UNDER(STARBOARD(BOW(TIRPITZ)))	7:30 ON 9/22/1943
X-7	UNDERWATER	UNDER(BATTLE_ PRACTICE_ TARGET_1)	8:30 ON 9/22/1943

Figure 5. Two of the TIMLS windows

The header of the second window is the specific query on the data base. Each header element is the name of other relations defined by the frame system. The body of the table is the response to the constrained query in the form of a list of n-tuples. The other windows echo that query with path(s) on the map or a new natural-language weather report. Many of the elements in the PQT are mousable so that new relations (columns) and constraints (upper window) can be defined, or old relations and constraints removed.

ACKNOWLEDGMENTS

The key implementor for RULE*CALC has been J. Thorp, and H. Mayerfeld was responsible for the knowledge engineering on the ASR8 radar application. The HAPStation interface was coded by J. Sanborn as part of a collaboration with D. Marshall of Denver Aerospace. The development team for the RFIX interface included J. Mills (integration), C. Phillips (shaded graphics), and T. Burzio (diagrams). The TIMLS work was done by B. Haugh (logic), Y. Sekine (natural language), S. Barash (PQT), and B. Kobler (graphic interface).

REFERENCES

[1] E. A. Feigenbaum, *Knowledge Engineering for the 1980's.* Technical report, Computer Science Department, Stanford University, 1982.

[2] Defense Advanced Research Projects Agency (DARPA), *Strategic Computing New-Generation Computing Technology: A Strategic Plan for Its Development and Application to Critical Problems in Defense.* Oct. 28, 1983.

[3] B. Shneiderman, "Direct Manipulation: a step beyond programming languages," **IEEE Computer 8** (16), pp. 57-68.

[4] R.L. Currier, "Interactive videodisc learning systems," **High Technology,** Nov. 1983, pp. 51-59.

[5] D.A. Waterman, *A Guide to Building Expert Systems.* Addison-Wesley, 1985, p. 350

[6] *Ibid.,* pp. 342, 359.

[7] J.W. Lewis, J.R. Thorp, and H.M. Mayerfeld, *The Rule*Calc Manual.* Martin Marietta Laboratories report MML TR85-6, February 1985.

[8] R. Sauers, "On the requirements of future expert systems," **IJCAI '83,** pp. 740-743.

[9] D. Marshall and J. Sanborn, *The HAPS Users Manual.* Martin Marietta Laboratories Technical Report, 1985.

[10] J. Allen, "Towards a general theory of action and time," **Artificial Intelligence 24,** pp. 123-154, 1984.

[11] G. Frere-Cock, *The Attacks on the Tirpitz.* Naval Institute Press, 1973.

[12] IBM Corporation, *Query-by-Example Terminal User's Guide.* IBM Form No. SH20-2078.

[13] J.C. Neves, S.O. Anderson, and M.H. Williams, "A PROLOG implementation of query-by-example," in *Proc. 7th International Computing Symposium,* Nuremberg, Germany, pp. 318-332.