CLIPS: An Expert System Tool for Delivery and Training

Gary Riley, Chris Culbert, Robert T. Savely, and Frank Lopez
NASA/Johnson Space Center
Artificial Intelligence Section - FM7
Houston, TX 77058

Abstract

The 'C' Language Integrated Production System (CLIPS) is a forward chaining rule-based language developed by the Artificial Intelligence Section at NASA/Johnson Space Center. This paper examines the requirements necessary in an expert system tool which is to be used for development, delivery, and training. Because of its high portability, low cost, and ease of integration with external systems, CLIPS has great potential as an expert system tool for delivery and training. In addition, its representation flexibility, debugging aids, and performance, along with its other strengths make it a viable alternative for expert system development.

Introduction

Expert system technology is a major subset of Artificial Intelligence and has been aggressively pursued by researchers since the early 1970's. In the last few years, both government and commercial application developers have given expert systems considerable attention as well. An entire industry has grown to support the development of expert system tools and applications, with a wide variety of both hardware and software products now available. The availability of expert system tools has greatly reduced the effort and cost involved in developing an expert system.

Despite all this, expert systems have generally failed to make a major impact in application environments. This failure has stemmed from tool vendor's overemphasis on expert system development environments to the detriment of options for delivery of expert systems and training in expert system technology. Viable delivery options are necessary to field expert systems. Training options in expert system technology are necessary for the widest possible dissemination of this technology.

The 'C' Language Integrated Production System (CLIPS) is a forward chaining rule-based production system developed by the Artificial Intelligence Section at NASA/Johnson Space Center. Version 1.0 of CLIPS, developed in the spring of 1985 in a little over two months, accomplished two major goals. The first of these goals was to gain useful insight and knowledge about the construction of expert system tools and to lay the groundwork for future versions. The second of these goals was to address the delivery problems of integrating and embedding expert systems into conventional environments. Version 1.0 successfully demonstrated the feasibility of continuing the project.

Subsequent development of CLIPS greatly improved its portability, performance, and functionality. A reference manual[2] and training guide[5] were written. The first release of CLIPS, version 3.0, was in July of 1986. The latest version of CLIPS, version

4.1, was completed in September of 1987. CLIPS is currently available through COSMIC (see appendix).

The development of CLIPS, though not a significant advance in expert system representation capability, is a significant advance in the concept of providing a low cost tool that can be used for training, development, and delivery. The use of CLIPS in these areas will be examined in this paper with special emphasis on CLIPS's capabilities as a delivery and training tool.

Delivery

CLIPS addresses several issues key to the use of an expert system tool for delivery. Among these issues are: the ability to run on conventional hardware; the ability to run on a wide variety of hardware platforms; the ability to be integrated with and embedded within conventional software; low-cost delivery options; the ability to separate the development environment from the delivery environment (i.e. run-time modules); the ability to run efficiently (both speed and memory), and migration paths from development to delivery environments.

One major requirement for a delivery tool is the ability to run on conventional hardware. Current state-of-the-art expert system software tools are almost all based in LISP and run only on specialized LISP hardware, such as the Symbolics or TI Explorer, or on the new generation of workstations such as the SUN or Apollo. While these workstations do provide a conventional platform for delivery, investment in conventional hardware often precludes adding additional or specialized hardware to support expert systems. The question to ask when considering delivery is not "Is there a conventional machine that supports the expert system tool I want to use?", but rather "Does the conventional machine I have support the expert system tool I want to use?"

Portability of the expert system tool code insures the ability to deliver on a wide range of hardware from microcomputers to minicomputers to mainframes. Because CLIPS is written in C and special care was taken to preserve portability, CLIPS is able to provide expert system technology on a wide variety of conventional computers. CLIPS has been hosted on over a dozen brands of computer systems ranging from microcomputers to mainframes without code changes. To maintain portability, CLIPS utilizes the concept of a portable kernel. The kernel represents a section of code which utilizes no machine dependent features. To provide machine dependent features, such as windowed interfaces or graphics editors, CLIPS provides fully documented software hooks which allow machine dependent features to be integrated with the kernel.

The ability to integrate with and embed within existing code is an important feature for a delivery tool. Integration guarantees that an expert system does not have to be relegated to performing tasks better left to conventional procedural languages. It also allows existing conventional code to be utilized. The capability to be embedded allows an expert system to be called as a subroutine (representing perhaps only one small part of a much larger program). Many tools view themselves as the "master" program and only permit control to be passed to other programs through them. CLIPS allows integration with C programs as well as integration with other languages such as FORTRAN and ADA. In addition, many functions are provided which allow CLIPS to be

manipulated externally. Because the source code is available, CLIPS can be modified or tailored to meet a specific user's needs.

Applications should be delivered as economically as possible. Many tools require the entire development environment to run an application. This necessitates buying a new copy of the tool for every delivered application. Some tools provide the capability to generate run-time modules. These run-time modules are basically equivalent to the executable modules generated by compilers for procedural languages. Run-time modules allow the unneeded functionality and information associated with the development environment to be stripped away from the delivery environment. This is a desirable characteristic, but for many tools, each copy of a run-time module must be purchased.

CLIPS effectively addresses the problems of low cost delivery. The cost for CLIPS source code is $200. This initial cost provides unlimited copies of CLIPS for delivery, development, and training. In addition, CLIPS also provides the capability to generate run-time modules.

Another key feature for a delivery tool is efficiency. CLIPS is based on the Rete algorithm[3] which is an extremely efficient algorithm for pattern matching. CLIPS version 4.1 compares quite favorably to other commercially available expert system tools based on the Rete algorithm. CLIPS performs its own memory management, eliminating the problems associated with garbage collection that plague most LISP based expert system tools.

A good delivery tool should also provide a graceful migration from the development environment to the delivery environment. CLIPS was designed to be as compatible as possible with other forward chaining rule-based languages. Thus the capabilities of CLIPS closely mimic the forward chaining capabilities of tools such as ART[1] and OPS5[4]. Such similarity allows the migration of programs developed using the powerful environments provided by LISP machines and tools such as ART to conventional hardware. This approach allows problems to be prototyped using the most productive environment available and then delivered in the desired environment.

Training

CLIPS addresses several issues key to the use of an expert system tool for training. A good training tool should have knowledge paradigms of the appropriate complexity, be low cost, run on easily accessible hardware, and have easy to use interfaces and debugging tools.

A good training tool should meet the Goldilocks criteria. That is, its knowledge paradigms should be neither too complex nor too simple. Hybrid tools incorporating multiple paradigms such as ART can be extremely difficult to learn to use. In addition, training using hybrid tools can often require additional training on the use of the LISP language and environment. On the other hand, simple tools such as decision tree builders, often lack the necessary depth to teach a reasonable variety of expert system concepts. CLIPS's single paradigm, forward chaining rules, provides a reasonable compromise between complexity and simplicity. It also provides a training path to the more complex hybrid tools.

A low cost tool is necessary for the widespread use of expert systems technology. Groups interested in expert system technology may not be able to spend large sums of money to use tools which utilize some of the more powerful knowledge representation paradigms. Inexpensive tools that provide only limited paradigms may give a false impression of the types of problems that can be solved with expert system technology. CLIPS provides a reasonably powerful representation paradigm at an extremely low cost.

A tool for training should run on easily accessible computers. In other words, whatever computers are available. This means that the tool should be as portable as possible and run on microcomputers (which make very good target machines for training tools). Portability for the training tool is also essential because the development and delivery environments may differ from the training environment. It is not productive to train with a tool that cannot then be used to develop and deliver an expert system.

A training tool should be easy to use. Mouse/menu driven windowed interfaces with graphical knowledge browsers and integrated editors are highly desirable. Because of portability concerns, the standard version of CLIPS uses a command line interface. However, software hooks are available to allow more sophisticated interfaces to be built. Several such training and development interfaces have been developed. Version 4.1 of CLIPS has an integrated editor (designed to run on UNIX, VMS, and MS-DOS machines) which allows quick editing and recompilation of rules. CLIPS has a wide variety of debugging commands to assist in examining and debugging an expert system.

Development

Several features are key to the use of an expert system tool for development. Among these features are: multiple knowledge representation paradigms; sophisticated user interfaces, debugging aids, and browsers; integrated editors, compilers, and other system tools; and the ability to rapidly prototype and iteratively refine an expert system.

Access to multiple representation paradigms is quite useful during the development stage of an expert system since the representation paradigms needed to solve a problem are not always known in advance and can frequently change as the problem solution evolves. CLIPS utilizes only one major paradigm: forward chaining rules. The use of a single paradigm in a tool is a drawback for development, however, expert systems such as DEC's XCON[6] built using OPS5 demonstrate that single paradigm tools can be used to develop significant expert systems.

Most of the interface features described for training are also desirable for development. Sophisticated interfaces provide ease of use in creating, debugging, and examining an expert system. An integrated environment provides quick turn-around time during development which facilitates rapid prototyping and iterative refinement.

Clearly the development environment is the area in which CLIPS compares least favorably to other expert system tools. However, while CLIPS does not provide a development environment as powerful as the major state-of-the-art expert system tools, it does provide the basic features necessary for the development of expert systems.

Conclusion

CLIPS has several characteristics which make it highly advantageous to use as a tool for delivery and training. Among these are its portability, low-cost, and ease of integration with conventional environments. In addition, it provides an environment suitable for the development of expert systems.

References

[1] ART Reference Manual, Inference Corporation, Los Angeles, CA. 1986.

[2] Culbert, C. "CLIPS Reference Manual". NASA Technical Memo FM7(87-220), December, 1986.

[3] Forgy, C. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem". Pages 17-37, Artificial Intelligence 19, (1982).

[4] Forgy, C. OPS5 User's Manual. Department of Computer Science document CMU-CS-81-135, Carnegie-Mellon University, Pittsburgh, PA. 1981.

[5] Giarratano, J. The CLIPS User's Guide. NASA Internal Note 86-FM-25, October, 1986.

[6] Harmon, P., and King, D. Expert Systems: Artificial Intelligence in Business. John Wiley & Sons Inc. New York, NY, 1985.

Appendix

NASA, DoD, and other government agencies may obtain CLIPS by contacting the CLIPS help desk 9:00 AM to 4:00 PM CST weekdays at (713) 280-2233. CLIPS is available outside the government through the COmputer Software Management and Information Center (COSMIC), the distribution point for NASA software. The program number is MSC-21208. Program price is $200.00 and documentation price is $17.00 (as of January 14, 1987). The program price is for the source code. Further information can be obtained from COSMIC:

COSMIC
Software Information Services
The University of Georgia
Computer Services Annex
Athens, Georgia 30602
(404) 542-3265