Apr 1st, 8:00 AM

Knowledge Based Programming at KSC

James H. Tulley

Senior Software Engineer , IPS Engineering and Software Production , Space Station Project Office, Lockheed Space Operations Company , John F. Kennedy Space Center, Florida

Carl I. Delaune

Project Engineer Space Station Project Office NASA, John F. Kennedy Space Center, Florida

KNOWLEDGE BASED PROGRAMMING AT KSC

James H. Tulley, Jr.
Senior Software Engineer
LPS Engineering and Software Production
Lockheed Space Operations Company
John F. Kennedy Space Center, Florida

Carl I. Delaune
Project Engineer
Space Station Project Office
NASA, John F. Kennedy Space Center, Florida

ABSTRACT

In the last five years the knowledge based programming effort at KSC has grown from a few small technology studies to a viable applied research program. Our experience from this research has taught us to appreciate the potential of the discipline. Recent spinoff projects are adding to our understanding and yielding useful products. Our results indicate that knowledge based programming is a powerful tool which can profitably be applied in many engineering problems.

INTRODUCTION

The progress in knowledge based programming at KSC reflects that experienced by the rest of industry. In a few years the emphasis has shifted from research in a poorly understood field (at least it was poorly understood by us!) to development of a set of tools with applications in several different areas. In 1981, when we first started working with knowledge based systems, they were confined almost exclusively to computer science labs in a few universities. Only a few had been used for real applications, and they had not yet been popularized in magazines and on television. In 1986 knowledge based programming is recognized as a useful technique for capturing expertise in non-algorithmic domains and for rapid construction of prototype systems. KSC has several knowledge based programming projects in various stages of completion, and we have achieved encouraging results in nearly all of these projects.

DEFINITIONS

The terms artificial intelligence, knowledge based systems, and expert systems, are often used interchangeably. Although closely related, they have different meanings. The definitions we use (which are commonly but not universally accepted) are as follows:

Artificial Intelligence (AI) - A discipline of computer science, AI is the study of ideas that enable computers to perform intelligently. This definition is a large umbrella that attracts many types of research and includes many topics. AI includes such diverse problems as pattern matching, techniques of search, vision, natural language understanding, and knowledge based systems. Its practitioners include computer scientists, psychologists, linguists, and many others. The glue that binds these people together is the difficulty of the problems that they pursue. Problems that do not appear soluble by traditional programming methods are often bundled with the other "hard" or mysterious AI problems. An ironic side effect is that AI workers tend to lose the fruits of their labor. If AI research leads to better understanding or the solution of a problem, the problem loses its mystery and hence its claim to be AI.

Knowledge based programming has at its core the idea that the domain specific knowledge in a computer program can be kept separate from its control structure. This idea can be implemented in several ways, of which rule based systems is probably the most familiar. In rule based systems knowledge is expressed in rules or predicate calculus assertions; another part of the system controls the execution of these rules. The archetypical rule based system is MYCIN

(Shor76), which diagnoses blood infections and recommends treatments. Another popular knowledge representation method is frames, wherein domain knowledge is stored descriptively, much like records in a database. The power of frames systems is that they also use mechanisms for property inheritance and procedural attachment. A classic frames system is KNOBS (Engl83), written by the MITRE corporation to plan tactical missions for the Air Force. The common characteristic of these techniques is that both keep knowledge about the application domain separate from control of the program. Ideally, the job of a knowledge based programmer should be to express domain specific knowledge, not to develop procedures.

Expert Systems - In an active research discipline, terms change meaning quickly. In 1981 "expert systems" and knowledge based systems were synonymous, both referring to any of the programs or programming techniques that exemplified the field. In 1986, the term expert system is generally used to mean a program which works so well that it appears to duplicate the reasoning process of an expert. Such a program could be designed and coded in many ways, but the most successful have been written using techniques of knowledge based programming.

Thus, to oversimplify, knowledge based programming is a design philosophy (which may be instantiated in several ways), an expert system is a particularly successful knowledge based program, and both are contained within the larger discipline of artificial intelligence.

KNOWLEDGE BASED SYSTEMS AT KSC

The first knowledge based programming project at KSC was the LOX Expert System (LES). This system, used to monitor liquid Oxygen loading of the space shuttle, has been described extensively elsewhere (Scar84, Jami85, Scar86). LES has been interfaced with the Launch Processing System at KSC and has successfully monitored shuttle launches.

We believe that LES makes an important contribution in the field of hardware diagnosis. In earlier systems, mostly rule based, the problem of sensor credibility always caused difficulty and was usually treated as a special case. In the LES paradigm of instrumentation, sensors are treated the same as other components and in fact analysis of sensor failures turns out to be an especially tractable problem. This is not surprising since the original goal of the LES project was to solve sensor problems.

SPINOFFS FROM LES

The success of LES has resulted in two spinoff projects: Knowledge Based Automatic Test Equipment (KATE) and Shuttle Connector Analysis Network (SCAN) (Giff86). These projects have two similar objectives. The first is to port the LES code to microcomputers and engineering workstations. One of the constraints on knowledge based systems has been the requirement for a dedicated, single user LISP computer as a host. Preliminary results from both projects indicate that the current generation of microcomputers, supported by full featured LISP compilers, make affordable, accessible AI development and delivery systems. The second common objective of SCAN and KATE is to further our understanding of how knowledge based systems can be used to represent and use engineering knowledge. During the LES development we used schematic diagrams to describe a system's physical connectivity - its structure. It was discovered that this structural information can be recovered from the knowledge base to reproduce the original schematic diagrams (New85). In addition, schematic diagrams, when combined with an engineer's knowledge of component operation and the laws of physics, imply functional knowledge. KATE and SCAN are exploring ways to capture this functional knowledge.

The KATE project is exploring further applications of the knowledge representation scheme developed by LES. If the knowledge base really represents the functionality implicit in schematic diagrams, then a knowledge based system should understand its domain well enough to design, control, and execute procedures on the hardware that it monitors.

Much of the KATE research is based on the discovery that inversion of functional dependencies in the knowledge base can be used to deduce the commands needed to cause a desired system response (Scar86). It appears that this capability can be made independent of system state. KATE currently performs functional inversion to locate all possible command sequences that will produce the desired state. The goal of the project is for KATE to resolve command sequences and then to execute a set of commands that will cause the transition from initial to final system state, constrained by possible failure modes. This would lead to a control system based on specification of desired results rather than procedures built from complex command sequences.

KATE retains all of the LES capabilities of fault detection and diagnosis. We believe that this combination of diagnostic and control capabilities will result in systems with more autonomy and reliability than is possible with the current state of the art.

The primary objective of the SCAN project is to represent structural (physical connectivity) relationships and thereby gain control of the electrical wiring configuration of each orbiter as it passes through the processing flow. Attempts to solve this problem with conventional database methods have not been successful, but knowledge based programming techniques may offer the needed development leverage. The SCAN system uses a knowledge base of frames with an "inference engine" and rule-like functions coded directly in LISP. The intent of this methodology was to mirror the thinking of the system analysts as they trace shuttle wiring, while remaining fast enough to permit rapid analysis of complex situations in a very large knowledge base. SCAN uses the same knowledge base and access functions as LES, and this code was ported from a Symbolics 3600 to an Apollo workstation.

A team of three engineers using knowledge based techniques succeeded in developing a working prototype in three weeks. This prototype system had essentially all of the functional capability needed by the delivery system, including diagnostic logic and an easy to use graphics interface. If successful, SCAN will have evolved from a concept to a production system within six months. It is already providing valuable experience in design, construction, and use of large knowledge bases. Possibly the most important benefit of the system is that the capture of the orbiter connector (and component) structural knowledge will enable more sophisticated diagnostic expert systems to be applied to the shuttle orbiters.

Several other knowledge based programming projects at KSC are reported elsewhere. Without firsthand knowledge of their progress, we thought it inappropriate to discuss them here.

LESSONS LEARNED

Since knowledge based programming techniques were introduced at KSC, we have acquired some understanding of the kinds of problems that they can solve. In learning to recognize candidates for knowledge based programs we have come to appreciate that there are at least two distinct kinds of knowledge with which such programs must deal: factual knowledge and expert knowledge.

Any knowledge based system must contain some factual information, the grist which the inferencing system will mill. This factual information is usually well understood and can be represented explicitly. Rule based systems normally represent facts as predicate calculus assertions, and extensive semantic systems have been constructed for this method (Scha75). The factual information in LES, for example, was derived from schematic diagrams and instantiated in frames. While factual information is usually easier to acquire and represent than expert knowledge, it can still be quite difficult to implement. In real applications the volume of factual data can be overwhelming. A difficulty in making LES operational has been the presence of small but annoying errors in its factual knowledge base. The process of transferring structural knowledge from schematics to frames was mostly manual, so mistakes were inevitable. SCAN faces a much larger problem simply because the volume of its factual knowledge is orders of magnitude greater.

The second type of knowledge describes how, when, and why the factual knowledge can be applied. This "expert knowledge" is more difficult to capture and store. It is often very hard for an expert to describe how he/she arrives at conclusions. (Expert system development is even more risky in applications where there is no expert!) Paradoxically, domains where expertise is poorly organized, expressed, or understood often make good choices for knowledge based systems. This is because the programming paradigms (rules, frames, etc.) are natural vehicles for expressing poorly organized knowledge. If the expert could express his knowledge completely, certainly, or algorithmically, then he could program that knowledge with traditional procedural methods. A pleasant result of knowledge based programming projects is that they almost always produce a better understanding of the expert knowledge with which they deal. When the LES project was conceived, we believed that diagnosis of hardware problems was an area of special expertise that could be mastered by only a few highly trained, experienced people. As the project matured, we learned that successful diagnosis uses only a few simple principles. The experts are people who can apply these principles to the information retrieved from schematic diagrams. The LES problem then became one of representing factual knowledge in a form open to computer analysis - a database problem. Diagnosis had lost much of its mystery.

Knowledge based methods offer the capability to develop or understand expertise incrementally. The expert, often in collaboration with a "knowledge engineer", can build a working prototype system very quickly. This system will usually be small and incomplete, but it will run. Improvements can then be made by trial and error. This technique runs contrary to the precepts of software engineering, but seems especially appropriate for the uncertain, unstructured domains so frequently encountered in knowledge based systems. It is extremely valuable to have a working model to try in a matter of a few weeks or months. Another characteristic of knowledge based systems is that their information structures (e.g. rules or frames) are very modular, so improvements or additions to the prototype are quick and easy. Popular accounts tend to emphasize the long time required to develop an expert system, but usually ignore this capability for rapid prototyping. This feature may turn out to be one of the most attractive properties of knowledge based programs.

Diagnostic problems tend to make good applications of knowledge based programming techniques. Even well understood diagnostic problems, such as troubleshooting television sets, usually cannot be written algorithmically because television sets have too many failure modes. Even if the analysis is limited to single component failures, there are too many permutations for all of them to be coded explicitly. Knowledge based systems evade this combinatorial explosion by building the search tree at execution time and trying only to explore the most promising branches.

Planning systems are good choices for implementation as knowledge based programs. People seem to plan things by using generate and test procedures, backtracking when necessary and following precedents if possible. All of these strategies are easily represented by knowledge based techniques.

Other attractive candidates are problems that do not require knowledge based techniques but can return large, near term economic payback. Such candidates may offer otherwise unavailable opportunities to capture knowledge domains which can be useful for future knowledge based system development.

There are many tools for knowledge based system development on the market today. They have been implemented on LISP machines, mainframes, engineering workstations, and even personal computers. The best of these tools permits developers to exercise the power of knowledge based systems without having to develop customized inference engines or knowledge representation schemes. This situation is much better than in 1981, when the only way to obtain a knowledge based system was to buy a LISP computer and code a system from scratch. This method was expensive, time consuming, and required one or more LISP programmers (who at that time were scarce). This partially explains why the acceptance of knowledge based programming techniques was so slow. Which way is better? An organization that wants to do research in the field should probably build at least one inference engine - an irreplaceable learning experience. An engineering group whose goal is to solve a specific problem should probably start by using a tool.

The relation between the LISP programming language and artificial intelligence is interesting. There are too many arguments about whether LISP is necessary for a knowledge based system. The answer is no, but it deserves a little discussion. Several good knowledge based programs and tools have been written in languages other than LISP, so clearly LISP is not a prerequisite. At the same time, most of the AI community in the United States programs in LISP. Thus, an expert system developer who chooses another language sacrifices the support he might otherwise expect from the research community. Also, future improvements in knowledge representation and knowledge based systems will probably appear first in LISP. PROLOG is the only strong competitor, especially in Europe and Japan. A more dangerous idea is the notion that a program written in LISP is somehow automatically granted status as an expert system. This is just wrong.

Use of knowledge based programming techniques permits developers to program at a higher level of abstraction. It increases their productivity because it allows them to concentrate on applications rather than on programming details. Knowledge based techniques improve efficiency by encouraging early prototypes and permitting quick and easy modifications. Finally these techniques provide a kind of knowledge representation not previously available. They are natural vehicles for expressing non-algorithmic knowledge. Knowledge based programming is not a panacea for all engineering and software development problems, but we think it will prove to be a valuable tool in future programs at the Kennedy Space Center.

REFERENCES

(Shor76) E. H. Shortliffe, MYCIN: Computer-based Medical Consultations, Elsevier, New York, 1976. Based on a PhD thesis, Stanford University, Stanford, CA, 1974.

(Scha75) Roger C. Schanck, Conceptual Information Processing, North-Holland, Amsterdam (1975).

(Engl83) C. Engleman, J. K. Millen, and E. A. Scarl, KNOBS: An Integrated AI Interactive Planning Architecture, Proceedings of the American Institute of Aeronautics and Astronautics, Compiters in Aerospace IV, Hartford, 1983.

(Scar84) E. A. Scarl, and C. I. Delaune, LOX Expert System, Proceedings, 21st Space Congress, Cocoa Beach, Florida (April, 1984) 2-16.

(Jami85) J. R. Jamieson, E. A. Scarl, and C. I. Delaune, A Knowledge Based Expert System for Propellant System Monitoring at the Kennedy Space Center, Proc. 22nd Space Congress, Cocoa Beach, Florida (April, 1985) 1-9.

(New85) E. New, Schematic Generation for NASA's LOX Expert System, Proceedings of ROBEXS '85, The First Annual Workshop on Robotics and Expert Systems, NASA/Johnson Space Center, (June, 1985) 177-184.

(Scar86) E. A. Scarl, J. R. Jamieson, and C. I. Delaune, Sensor Based Diagnosis Using Knowledge of Structure and Function, Unpublished.

(Giff86) R. D. Giffen, J. H. Tulley, and J. C. Wilkinson, The Shuttle Connector Analysis Network, included in Artificial Intelligence in Lockheed, Lockheed Horizons, Unpublished.