A KNOWLEDGE REPRESENTATION TECHNIQUE FOR SYSTEMS DEALING WITH HARDWARE CONFIGURATION*

Jeff Pierick
ROLM Corporation
4900 Old Ironsides Dr.
Santa Clara, CA 95054

Massachusetts Institute of Technology
Laboratory for Computer Science
77 Massachusetts Ave.
Cambridge, MA 02139

ABSTRACT

A representation language combining the attributes of both rule-based systems and frame-based systems is discussed within the context of developing systems for computer hardware configuration. It is believed that the combination of these two common approaches to knowledge representation provides many advantages over the strict use of either of the two approaches alone.

I INTRODUCTION

The domain which shall be considered in this paper is that of order processing for computer hardware. For years, this task was handled almost exclusively by teams of experts who would meticulously review each customer's order. The expert would check the customer's order by mentally reviewing a set of rules, learned over time. For the most part, the task was tedious and time consuming. However, this all changed with the advent of XCON (McDermott, 1980).

The first commercially successful knowledge-based system to effectively deal with this domain was the XCON system, developed by John McDermott and his colleagues for use by the Digital Equipment Corporation. Since the development of this system, many similar systems have been developed for use by other companies. One such system is the BEACON system (Freeman, 1985), developed for use at Burroughs.

These two systems use two very different approaches toward representing the knowledge necessary for their domain. The XCON system uses simple production rules to represent its knowledge. The BEACON system uses a semantic network augmented with the addition of simple constraints. However, I suggest that neither of these two techniques adequately handles the complexities that arise in this domain.

This paper will discuss a representation language that is tailored to the representational needs of the domain by combining the benefits of production rules and the benefits of semantic networks. While this language is demonstrated within the domain of computer hardware configuration, it should be remembered that it may be equally useful in other domains as well.

II PRODUCTION RULES AS A KNOWLEDGE REPRESENTATION FORMALISM

The typical approach to knowledge representation in knowledge-based systems is the use of production rules (Barr and Feigenbaum, 1981). This tendency is so prevalent, that the term rule-based systems is used almost synonymously with knowledge-based systems. Certainly, the majority of the commercial expert system building tools available today make extensive use of this knowledge representation technique.

There are many reasons that the use of production rules is so desirable. The domain knowledge is represented explicitly within the rules. The knowledge is encoded in such a way that a casual observer can easily understand the intent of the knowledge. This is made possible by the fact that the rules seem natural and the fact that control structures are not freely mixed with the domain knowledge. That is, the information in the rule is declarative rather than procedural. This all leads to the fact that production rules can be used to explicitly represent the important domain knowledge.

Another advantage of production rules is the fact that the knowledge is represented in a uniform manner. This makes for efficient handling of the knowledge. A very simple, generic inference engine can be built to parse the knowledge base. Furthermore, a uniform representation language makes it easy to translate the knowledge into a stylized form of English for use in describing the reasoning of

*The work described in this paper is based in part on research done at ROLM Corporation, through the coordination of the MIT VI-A Internship Program, in partial fulfillment of my SM Thesis at the Massachusetts Institute of Technology.

the system.

Furthermore, production rules have a nice way of dealing with empirical knowledge; in fact, it could be said that all of the knowledge in these systems is simply empirical. This is very useful for domains for which a model cannot easily be formulated. A good example of such a domain is the diagnosis of infectious bacterial diseases, the domain within which the MYCIN (Shortliffe, 1984) system worked.

Within MYCIN's domain, the expert could rarely be certain about the knowledge he was giving the system. Certainty factors were used in an attempt to deal with this problem of reasoning with uncertainty. Moreover, it would have been very difficult for the expert to give MYCIN a reliable model of the domain. The expert could only offer empirical knowledge about the domain, knowledge that he had gathered through past experiences. Thus, a rule-based knowledge representation language was well suited to this domain.

Lastly, production rules tend to be very modular (Davis et al., 1977). Each production rule represents a distinct chunk of knowledge. Each production rule is relatively independent of the other rules in the system. Thus, for a moderately sized knowledge base, the knowledge engineer should be able to modify an existing rule or add an additional rule to the system without having to worry about adversely affecting any of the other rules in the system.*

However, herein lies one of the major limitations of using production rules alone as a knowledge representation technique within the domain of computer hardware configuration. This domain has a great deal of structure which is not properly represented in the form of production rules. A lot of the structure of the domain is lost when it is compiled in the form of production rules. It is unable to exploit the structure of the domain, but rather it can only use the empirical rules which it is given.

Moreover, compiling the structure of the domain in the form of rules may actually make the system difficult to modify and maintain. This is pointed out in (Koton, 1985) where she states, "One piece of knowledge may be contained in any number of rules in the system, so in order to change that knowledge it must be changed in every rule that uses it." As the size of the knowledge base increases, it becomes even more difficult to determine all of the interrelationships between the rules (Freeman, 1985).

III FRAMES AS A KNOWLEDGE REPRESENTATION FORMALISM

The BEACON system uses a semantic network system to represent its domain knowledge. This paper will consider frame-based systems instead of semantic networks; however, the points discussed are pertinent to both formalisms.

Frames can be used to build a model of the system to be configured. A hardware component is represented as a single frame within the model. If the component is modified, the effect of the modification is isolated to the single frame which represents the component. Thus, the frame-based approach exhibits a form of modularity not inherently found in the rule-based approach. It displays a form of object oriented modularity.

Using a rule-based approach, knowledge about a component may be located in several different rules. When the knowledge of that component must be modified, each relevant rule must be found and changed as appropriate. Using a frame-based approach, all of the information about a component, including its relationship to other components, is located within a single frame; thus, since the knowledge about a component is centralized, the knowledge base is much easier to modify and maintain.

Moreover, the relationships between different components in the system can be represented naturally using a frame-based approach to knowledge representation. For example, if a system has four distinct components, then the frame representing the system would have a slot listing the four components. If two systems within the domain are simply variations of an encompassing system, then there would be a slot in each of the two representative frames representing this fact. Thus, the relationships between different components in the domain can be represented explicitly using frames.

This suggests another advantage of using a frame-based approach to knowledge representation within the domain of hardware configuration. Frame-based systems typically have some form of inheritance between frames (Brachman, 1983). This is provided for through the IS-A or AKO (standing for A Kind Of) slot alluded to in the previous example. Inheritance results in an efficient means of hierarchical knowledge representation which makes the knowledge easier to modify and maintain.

Since knowledge can be inherited by one frame from another, knowledge does not have to be duplicated within the knowledge base. The knowledge is located at the most logical position within the network. It is not spread throughout the knowledge base.

Thus, using a model-based approach to knowledge representation can overcome many of the problems which a rule-based approach introduces. However, just as it has been shown that the exclusive use of the rule-based approach to knowledge representation within this domain may not be appropriate, it can also be shown that the exclusive use of the frame-based approach to knowledge representation may also prove to be a hinderance.

*As the size of the knowledge base increases, this may not necessarily hold.

While a model can represent the structures and the legal combinations of structures within the domain, the user is left to decide which configuration is most appropriate for his needs. If the user is already an expert in the domain, this will be an easy task for him. However, there is no reason to expect that the user of the system is going to be an expert in the domain; in fact, if he were an expert, he probably would not be using the knowledge-based system in the first place. The more likely situation is that the user will know very little about the domain; thus, it is unlikely that he will be able to decide which configuration is appropriate for his needs. A useful knowledge-based system should aid the user in making this decision.

Thus, frames are a good formalism for representing the structure in a domain such as computer hardware configuration. However, it is very difficult to add any form of judgemental reasoning to such a system. The result is that we have a system which will hold a user to a set of outlined constraints, but the user must be smart enough to decide which constraints apply to his situation.

IV MAKING THE TWO WORK AS ONE

If the domain exhibits a great deal of structure, while still requiring a certain amount of judgemental reasoning, (as is the case with hardware configuration) then a solution to this problem presents itself as the combination of these two paradigms. Frames can represent the structure of the domain, while rules can represent the needed judgemental reasoning.

The idea is to hold on to the structure which is naturally provided by the frame-based approach. System components are to be represented as frames with slots, the slots representing the component's relationship to other components in the domain. However, if there is a relationship in the system which is dependent on the user's needs, then production rules are used. That is, if the user has a choice as to whether a particular component is included in his configuration, then the value of that slot would be filled with an if-needed production rule demon. The production rule system would then aid the user during the consultation in deciding whether that component is appropriate for his needs.

This representation language represents the structure of the domain while still being able to deal with empirical knowledge and judgemental reasoning. It represents the natural dependencies within the domain while maintaining a modular style. Knowledge is centralized and logically sectioned. Instead of having a single, large rule base dealing with every facet of the domain, this knowledge is divided into smaller, more manageable, rule bases, each with a very particular purpose.

Thus combining the frame-based approach and the rule-based approach takes advantage of the benefits of both paradigms while overcoming their limitations. The frame-based section divides the knowledge base into logical components, while the rule bases provide the needed judgemental reasoning. Moreover, the important aspects of the domain are made explicit, which is the ultimate goal in the selection of any good representation language (Winston, 1984).

V IMPLEMENTATION DETAILS

As noted in a previous section, the language used in this project incorporates both the structure of a frame-based system and the judgemental reasoning power of a rule-based system. This is accomplished in the following manner.

A. The Frame Based System

The frames in the system are implemented in the fashion of FRL frames (Roberts and Goldstein, 1977), with the exception that every datum has a certainty factor associated with it:

```
(frame1 (slot1 (facet1 (datum1 certainty)
                       (datum2 certainty))
               (facet2 (. . .)))
        (slot2 (. . .))
        (. . .))
```

The name of the frame corresponds to the name of a concept or object in the domain. A slot represents an attribute of the object. A facet signals the way in which the data associated with the facet fill the slot. A datum is the actual slot filler. The certainty factor associated with the datum (of which there is always exactly one), represents the strength with which the association between the slot and the slot filler is believed.

In the domain of hardware configuration, the frames represent typical components and typical systems. There is a frame for each distinct component within the domain. There are also frames which represent systems composed of combinations of distinct components and other systems. These frames are linked together through special slots. For example, a system frame has a slot containing the names of the frames representing its constituents.

As with traditional frame-based systems, slots can be filled with explicit values, default values, or demons. The components of a system are represented as a list of explicit values. The purpose of a particular system is given as a default value which may be overridden if the user has a different purpose in mind. The price of a component is given as a demon which would look up the price of the component in a separate, loosely-linked data base.

Inheritance is also an integral part of the representation language. A frame is able to

inherit values for its slots from all of its ancestors. This makes the representation efficient in terms of space requirements. It also makes intuitive sense. For example, it seems natural to say that an IBM PC* has a display because it is a kind of personal computer; we know that all personal computers have displays.

B. Adding Rules to the Frame Based System

The production rules are found in the system in the form of production rule demons. They are identical to their procedural counterparts, except for the fact that they are declarative instead of procedural. There are IF-NEEDED demons, which are activated if a value is needed for a slot which does not have an explicit value, IF-ADDED demons, which are triggered if a new value is added to a slot, IF-REMOVED demons, which are evaluated if a value is removed from a slot, and IF-MODIFIED demons, which are processed if the certainty factor associated with a datum is modified.

A production rule demon is essentially a small rule base containing a number of rules intended to solve a very focused problem. The way in which the demon is evaluated depends on the type of demon it is. If it is an IF-NEEDED demon, the rules are evaluated in a backward chaining manner, in which only those rules which may provide a solution to the current goal are triggered. If the demon is an IF-ADDED, IF-REMOVED, or IF-MODIFIED demon, then it is evaluated in a forward chaining manner, in which every rule is triggered.

The rules themselves are based on attribute, object, value tuples. In this case, the objects are frames, the attributes are slots, and the values are slot fillers. Thus, a typical rule may appear as follows:

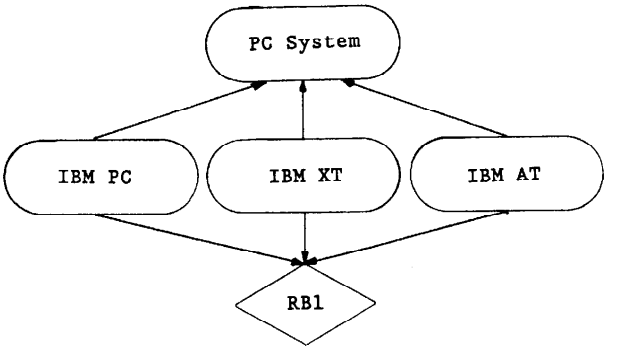
```
(RULE1 ((SAME FRAME1 SLOT1 VALUE DATUM1)
        (KNOWN FRAME2 SLOT2 VALUE))
       ((RETURN FRAME3 SLOT3 VALUE DATUM3 95)))
```

These rules are very similar to the rules that were used in the MYCIN system.

VI HYPOTHETICAL SCENARIO

As an example of the use of the new representation language, I will present a system which would be used by a sales representative to aid in the configuration of an IBM personal computer for a customer. A portion of the knowledge base is shown in Figure 1.

We could imagine that a sales representative, who may have very little computer experience, could sit down with the system and place an order in the following manner. The inference engine would begin with the PC System frame and it would apply Rule Base #1 when the IBM PC frame is activated. Thus, the consultation would proceed as follows:



Rule Base #1 (RB1):

Rule 1.1:
If the customer knows the type of system he wants,
then that is definitely the proper system (1.0).

Rule 1.2:
If speed is important to the customer,
then an IBM AT may be the proper system (0.7).

Rule 1.3:
If expandability is important to the customer,
then an IBM XT may be the proper system (0.7), and
an IBM AT may be the proper system (0.8).

Rule 1.4:
If the customer does not want to spend a lot of money,
then an IBM PC may be the proper system (0.8), and
an IBM XT may be the proper system (0.6).

Figure 1. Determining the Proper Order Type: This is the portion of the knowledge base which helps the user decide which of the three system types (i.e. PC, XT, or AT) is most appropriate for the customer.

> What type of PC system would the customer like, or would you like me to help you decide which would be appropriate for him (PC, XT, AT, or Assist)?
> >Assist

Since the user has asked for the assistance of the knowledge-based system in determining the proper PC system for the user, the evaluation of the rule base continues:

> Is it critical that the customer's applications run as fast as possible?
> >No
> Does the customer plan on expanding his PC system (e.g. extra memory, a modem, communication interfaces, etc.)?
> >Yes
> Is price an important factor for the customer?
> >Yes

*IBM PC, XT, and AT are trademarks of International Business Machines Corporation.

The evaluation of Rule Base #1 has determined that the most appropriate system for the customer is an IBM XT. Thus, it notes the fact that there should be one IBM XT frame but no IBM PC frame nor IBM AT frame instantiated in the consultation data base. Thus the inference engine halts its evaluation of the IBM PC branch of the knowledge base and begins to evaluate the IBM XT branch.

The rest of the knowledge base would be represented in a similar fashion as that suggested in Figure 1, and the consultation would continue as shown in this example.

VII CONCLUSIONS

The problem of hardware configuration and order processing has proven to be a difficult problem to deal with in any sort of automated way for many reasons. Firstly, the amount of knowledge necessary to make an automated system work effectively in this area is usually quite large. Secondly, it is typically the case that the domain knowledge changes significantly during the life of the product. Thus, no real progress was made in this area until knowledge-based system technology was applied to the problem.

Since the first knowledge-based system was developed for this domain, many other systems have followed. Some of these systems used production rules as a knowledge representation language, while other systems used frames or semantic networks.

It has been argued that the representation language used in these systems may not be appropriate for their domain. Important information about the domain, namely the structure of the target system, is lost when production rules are used. Frame-based systems deal nicely with the structure of the domain, however, they cannot adequately represent the empirical knowledge and judgemental reasoning which is often necessary for a complete system.

For these reasons, a representation language which effectively combines the benefits of a frame-based system and a rule-based system has been proposed. The frames in the language effectively represent the inherent structure in the domain of computer hardware configuration while maintaining a sense of modularity. The rule-bases describe the judgemental reasoning which is involved in the process of hardware configuration and order processing.

These ideas are currently being tested through the development of two prototype systems for hardware configuration. The first system is being developed using a commercial expert system building shell. The knowledge for this system is encoded in the form of production rules. The second system is being developed concurrently using an inference engine based on the new representation language.

It is believed that the second system shall prove to be easier to develop and maintain than the first system. Since the knowledge will be partitioned into logical sections it should be easier to enter the original core of knowledge and easier to maintain the knowledge base thereafter.

ACKNOWLEDGMENTS

I would like to thank Dr. Wing Kai Cheng, of ROLM Corporation, and Prof. Ramesh Patil, of the Massachusetts Institute of Technology, for their comments and suggestions on earlier drafts of this paper and their continued support during my research.

REFERENCES

Barr, A., and Feigenbaum, E. A. (Eds.) "Production Systems". In The Handbook of Artificial Intelligence, Volume 1. Los Altos, CA: William Kaufmann, Inc., 1981, pp. 190-199.

Brachman, Ronald J. "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks". Computer, Volume 16(10), 1983, pp. 30-36.

Davis, R., Buchanan B., and Shortliffe, E. "Production Rules as a Representation for a Knowledge-Based Consultation Program". Artificial Intelligence 8, 1977, pp. 15-45.

Freeman, Michael W. "Case Study of the BEACON Project: The Burroughs Browser/Editor and Automated Configurator". Logic-Based Systems Group SDC, A Burroughs Co. Paoli, PA, 1985.

Koton, Phillis. "Towards a Problem Solving System for Molecular Genetics". Technical Report MIT/LCS/TR-338, Massachusetts Institute of Technology, Cambridge, MA, 1985.

McDermott, John. "R1: An Expert in the Computer Systems Domain". In Proceedings of the First International Joint Conference on Artificial Intelligence, Palo Alto, California, 1980, pp. 269-271.

Roberts, B., and Goldstein, I. "The FRL Manual". MIT AI Memo 409, Massachusetts Institute of Technology, Cambridge, MA, 1977.

Shortliffe, Edward H. "Details of the Consultation System". In Rule-Based Expert Systems, Reading, MA: Addison-Wesley, 1984, pp. 78-132.

Winston, P.H. "Representing Commonsense Knowledge". In Artificial Intelligence, Reading, MA: Addison-Wesley, 1984, pp. 251-289.