REPRESENTATION OF CONTROL KNOWLEDGE IN EXPERT SYSTEMS

Janice S. Aikins

Computer Science Department
Stanford University
Stanford, California 94305

ABSTRACT

This paper presents the results of research done on the representation of control knowledge in rule-based expert systems.* It discusses the problems of representing control knowledge implicitly in object-level inference rules and presents specific examples from a MYCIN-like consultation system called PUFF. As an alternative, the explicit representation of control knowledge in slots of a frame-like data structure is demonstrated in the CENTAUR system. Explicit representation of control knowledge has significant advantages both for the acquisition and modification of domain knowledge and for explanations of how knowledge is used in the expert system.

I INTRODUCTION

This paper emphasizes the importance of representing domain-specific control knowledge *explicitly* and *separately* from other forms of domain knowledge in expert systems. The particular focus of research on this topic has been MYCIN-like consultation systems [6] which represent their domain knowledge in the form of condition-action or production rules. Examples in this paper are taken from the PUFF system [4] which performs consultations in the domain of pulmonary (lung) physiology.

The CENTAUR system was created in response to several knowledge representation and control structure problems in the rule-based systems, among which were the problems caused by the implicit representation of control knowledge. CENTAUR provides a framework for performing tasks using an *hypothesize and match* approach [5] to problem solving. This approach focuses the search for new information around recognized patterns of knowledge in the domain, a strategy that was not represented in the rule-based systems. Knowledge in CENTAUR is represented in the form of frame-like structures, called prototypes, which represent the expected patterns of knowledge, and in production rules, which serve as a stylized form of procedural attachment and are used to infer values or "fill in" slots in the prototype. This knowledge of prototypical situations is used for control of the consultation, for explanation of system performance, and also as a guide for acquiring additional knowledge and for modifying the existing knowledge base.

* This work was supported by the Advanced Research Projects Agency under contract MDA 903-77-C-0322. Computer facilities were provided by the SUMEX-AIM facility at Stanford University under National Institutes of Health grant RR-00785-07. The author is sponsored by the Xerox Corporation under the direction of the Xerox Palo Alto Research Center.

CENTAUR's combination of prototypes and rules results in a knowledge representation that is expressive enough to allow the many kinds of domain knowledge necessary for system performance to be *explicitly* represented. *Control knowledge* for the consultation is represented in slots associated with each prototype, separately from the inference rules. Rules are associated with prototypes as the explicit *contexts* in which the rules are applied. The slots in the prototype specify the *function* of the attached rules, such as to summarize data already given or to refine an interim diagnosis. Other details of the CENTAUR system and a full discussion of the knowledge representation and control structure problems in the rule-based systems can be found in [1].

II THE PUFF SYSTEM

One such rule-based system is the PUFF system which was created using a MYCIN-like framework. PUFF's domain-specific knowledge is represented by a set of approximately 60 production rules. The "IF" part of the production states a set of conditions (the premise clauses) in which the rule is applicable. The action, or "THEN" part of the production, states the appropriate conclusions. The goal in PUFF is to interpret a set of lung function tests performed on a patient, and to produce a diagnosis of pulmonary disease in that patient. Each rule clause is a LISP predicate acting on associative (object-attribute-value) triples in the data base. In PUFF there is a single object, the patient. The attributes (or *clinical parameters*) are the lung function tests and other information about the patient. The PUFF control structure is primarily a goal-directed, backward chaining of the production rules as it attempts to determine a value for a given clinical parameter. written. A complete description of this mechanism is given in [6].

III IMPLICIT CONTROL IN THE RULES

Production rules, in theory, are modular pieces of knowledge, each one capturing some "chunk" of domain-specific expertise. Indeed, one of the advantages of using production rules [3] is that there need be no *direct* interaction of one rule with the others, a characteristic which facilitates adding rules to the knowledge base or modifying existing rules. In practice, however, there are significant interactions among rules. Executing one rule will in turn cause others to be tried when the information needed for the first rule is not already known. Therefore, the *order* of the premise clauses of a rule affects the order in which other rules are executed. Further, in an interactive system such as PUFF, in which the user is asked for information that can not be inferred by rules, the order of the premise clauses also determines the order in which questions are asked.

This means of controlling question order by placing premise clauses in a specific order is, in fact, exploited by experts who recognize the need for ordering the questions that are asked, but have only this implicit and indirect mechanism for achieving their goal. In this case, the production rule framework itself becomes a programming language where the rules have multiple functions; some rules represent independent chunks of expertise with premise clauses specified in an arbitrary order, while others serve a controlling function with premise clauses that cannot be permuted without altering the behavior of the system.

An example of implicit control knowledge is illustrated by the PUFF rule in Figure 1 below. This rule invokes other rules in an attempt to determine whether there is Obstructive Airways Disease (Clause One), and if so, to determine the subtype (Clause Two) and findings associated with the disease (Clause Three). If Clause One were inadvertently placed after either Clause Two or Three, the system's questions of the user would probe for more detailed information about Obstructive Airways Disease without having confirmed that the disease is present. For example, by reordering the clauses in RULE002, PUFF might begin its consultation by asking about the patient's smoking history, one of the findings associated with Obstructive Airways Disease, and a question that would be inappropriate in a patient without a smoking-related disease. However, this rule contains no explicit indication that the order of the clauses is critical. The problem with implicit representation of control knowledge becomes apparent in working with the knowledge base, either to modify the knowledge or to explain its use in the system.

```
RULE002
-------

If:  1) An attempt has been made to deduce the
        degree of obstructive airways disease
        of the patient,
     2) An attempt has been made to deduce the
        subtype of obstructive airways disease,
        and
     3) An attempt has been made to deduce the
        findings about the diagnosis of
        obstructive airways disease
Then: It is definite (1.0) that there is an
        interpretation of potential obstructive
        airways disease
```

FIGURE 1. PUFF Rule--Implicit Control

Modifying rules is a normal part of system development. Clauses often must be added to or removed from rules in response to perceived errors or omissions in the performance of the system. However, removing or modifying the clauses of a controlling rule can alter the system's behavior in unexpected ways, since the implicit control knowledge also will be altered. Therefore, modifications can be safely done only by persons intimately familiar with the knowledge base. This factor not only limits the set of people who can make modifications, and of course precludes the success of automatic knowledge acquisition systems in which each rule is considered individually, but it also limits the size of the knowledge base, as even the best of knowledge engineers can retain familiarity with only a limited number of rules at a time.

A system's explanations of its own performance also suffer when information critical to performance is not represented explicitly. The rule-based systems studied generate explanations of why questions are being asked using direct translations of those rules which were being used when the question was asked. (See [2] for details.) There is no distinction made between rules that control a line of reasoning, as opposed to rules that infer a piece of information. However, users of the system should be able to ask both kinds of questions in order to obtain justifications of the system's reasoning process as well as justifications of its inference rules. The uniform representation of control and inference knowledge in rule-based systems further confuses the user by mixing the two kinds of explanations.

IV CONTROL KNOWLEDGE IN CENTAUR

Control knowledge about the process of pursuing an hypothesis in CENTAUR is represented in slots associated with each prototype, separate from the inference knowledge which will actually confirm or deny the hypothesis represented as production rules. Each slot specifies one or more LISP clauses, or *control tasks*, that are executed at specific points during the consultation as defined by a top-level prototype representing the "typical" consultation (the *CONSULTATION Prototype*).

For the pulmonary function domain, prototypes correspond to specific pulmonary diseases. During a CENTAUR consultation, initial case data suggest one or more disease prototypes as likely matches. Control knowledge in these prototypes then guides the consultation by specifying what information should be sought next. Expected data values in each prototype enable CENTAUR to pinpoint inconsistent or erroneous information during the consultation. Final conclusions are presented in terms of the prototypical situations determined to be present in the case, and any inconsistencies are noted.

Thus the system developer can specify "what to do" in a given prototype context as an important part of the knowledge about the domain that is distinct from the inferential knowledge used in the consultation. These control tasks are specified as LISP functions, and the system developer can define any new functions as they are required. For example, Figure 2 shows CENTAUR's representation of the control knowledge in the PUFF rule shown in Figure 1. The control knowledge is represented in two of the control slots associated with the Obstructive Airways Disease (OAD) prototype. They specify that when OAD is confirmed (the **If-Confirmed** Slot), the next tasks are to deduce a degree and a subtype for OAD, and, at a later stage in the consultation (when the prototype **ACTION** slots are executed), to deduce and print findings associated with OAD.

If-Confirmed Slot:
Deduce the Degree of OAD
Deduce the Subtype of OAD

Action Slot:
Deduce any Findings associated with OAD
Print the Findings associated with OAD

FIGURE 2. OAD Prototype Control Slots

Prototypes not only represent the domain-specific knowledge of a particular application, but also represent domain-independent knowledge about the operation of the CENTAUR system. At the highest level in CENTAUR, the Consultation Prototype lists the various stages of the consultation (e.g., entering initial information, suggesting likely prototypes, filling in prototypes) in its control slots. The advantages of explicit representation of control knowledge thus extend to control of the consultation process itself.

V ADVANTAGES OF THE CENTAUR APPROACH

The association of control knowledge with individual prototypes allows control to be specific to the prototype being explored. Thus domain experts can specify a different set of control tasks for each prototypical situation. In the pulmonary domain, for example, the expert proceeds in a different way if he has confirmed OAD rather than some other disease in the patient. Further, because this control knowledge is separate from the inference rules, the expert does not have to anticipate and correct incidental interactions between control and inference knowledge.

Representing the entire consultation process itself as a prototype has additional advantages. First, the system designer's conception of the consultation process is clearly defined for all system users. Second, representing each stage of the consultation as a separate control task allows stages to be added or removed from the consultation process. For example, the *Refinement Stage*, which uses additional expertise to improve upon an interim conclusion, was omitted during early stages of system development for the pulmonary function problem. "Filling in" a consultation prototype with user-specified options, such as a choice of *strategy* for choosing the current best prototype (for example, confirmation, elimination, or fixed-order), results in a control structure that can be tailored to the desires of each individual user.

The organization of knowledge into prototypical situations allows the user to more easily identify the affected set of knowledge when changes to the knowledge base are desired. Points at which specific control knowledge is used during the consultation are clearly defined, with the result that it is easier to predict the effects of any control modifications that may be made.

Explicit representation of control knowledge also facilitates explanations about that knowledge. In addition to the HOW and WHY keywords available in MYCIN, a new keyword, CONTROL, has been defined so that a user of the system can inquire about the control task motivating the current line of reasoning. For example, if the user types "CONTROL" in response to a system question about the patient's smoking history, the system would respond, ***The current control task is to determine the findings associated with OAD.***

VI SUMMARY

This paper has discussed the importance of representing control knowledge explicitly, particularly as it affects knowledge acquisition and explanation in a knowledge-based system. The representation of control knowledge as slots in a prototype in the CENTAUR system demonstrates one feasible approach. Augmenting the rule representation to include rules that function exclusively as control rules might be another. The critical lesson learned from working with the rule-based systems is that the system's representation structures must be expressive enough to represent control knowledge explicitly, so that it will not be inaccessible to the system and to the knowledge engineer.

ACKNOWLEDGMENTS

Many thanks to Doug Aikins, Avron Barr, Jim Bennett, Bruce Buchanan, and Bill Clancey for their helpful advice and comments on earlier versions of this paper.

REFERENCES

[1] Aikins, J. *Prototypes and Production Rules: A Knowledge Representation for Computer Consultations.* (Forthcoming Ph. D. Thesis), Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1980.

[2] Davis R. *Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases.* STAN-CS-76-552, Stanford University, July 1976.

[3] Davis R., and King, J. An Overview of Production Systems. In E. W. Elcock and D. Michie (Eds.), *Machine Intelligence 8.* New York: Wiley & Sons, 1977. Pp. 300-332.

[4] Kunz, J., Fallat, R., McClung, D., Osborn, J., Votteri, B., Nii, H., Aikins, J., Fagan, L., and Feigenbaum, E. *A Physiological Rule Based System for Interpreting Pulmonary Function Test Results.* HPP-78-19 (Working Paper), Heuristic Programming Project, Dept. of Computer Science, Stanford University, December 1978.

[5] Newell, A. Artificial Intelligence and the Concept of Mind. In R. Schank and K. Colby (Eds.), *Computer Models of Thought and Language.* San Francisco: W. H. Freeman and Company,1973. Pp. 1-60.

[6] Shortliffe, E. H. *MYCIN: A Rule-based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection.* Ph. D. dissertation in Medical Information Sciences, Stanford University, 1974. (Also, *Computer-Based Medical Consultations: MYCIN.* New York: American-Elsevier, 1976.