A Pictorial Method of Knowledge Representation

Edward N. Schwartz
Manhattanville College
Department of Mathematics and Computer Science
Purchase, New York 10577

ABSTRACT

Problems in the construction of a rule based expert system include deducing the heuristics of the expert, converting these heuristics into a taxonomy and rule base, and ordering the rules so the system performs efficiently and correctly. We will describe a pictorial method of knowledge representation that we call a "k-tree" and a method of structuring a rule base using what we call "question lattices" that we have found greatly facilitates these tasks.

While constructing an expert system to do academic advising, several colleagues and I found that a major problem is the representation of knowledge gleaned from the expert. A number of researchers suggest that a good representation is if-then rules. This did not work for us. As the number of rules, and the number of antecedents in rules, grew, the if-then form became intractable. We were unable to maintain an overview when looking at these detailed rules. We needed some method that, while carrying the information of if-then rules, would group rules by common antecedents, allow us to get an overview of the system, and allow us to have confidence that our collection of rules was complete.

The form of knowledge representation we settled upon is a pictorial tree that we call a "k-tree" (knowledge-tree). Each branch of a "k-tree" corresponds to a rule. The leaf represents the consequent and the parent nodes represent the antecedents. The "k-tree" became the method of choice for recording the responses of the expert. It becomes easy to see, by looking at the "k-tree" for a set of heuristics, whether all possible values of any property have been accounted for. Moreover, there is a feedback loop. By looking at the "k-trees" generated during an interview we can find any properties not fully explored. We can then return to the expert and ask questions to complete our understanding.

Eventually, software must be built. The taxonomy is easy to construct once the "k-trees" have been written. Every antecedent that appears must be represented in the taxonomy. (There are certain exceptions to this statement. In our system, some antecedents are PROLOG goals. It is not necessary to include these as properties.) The "k-trees" give good guidance for the shape of the taxonomy. The construction of the rule base proceeds directly from the "k-trees". Leaves are turned into consequents and branches become antecedents in rules. By grouping the "k-trees" according to the types of advice that they help generate, rules were automatically grouped in a logical way.

2 Davis, R. Buchanan, B, Shortliffe, E. "Production Rules as a Representation for a Knowledge-Based Consultation Program" pp. 15-45 Artificial Intelligence v 30 n 2 Feb. 1976
3 Preru, D. "Knowledge Acquisition in the Development of a Large Expert System" pp. 43-51 AI Magazine v 8 n 2 Summer 1987
Often, entire trees repeat with only slight changes. When this occurs one can dispense with all but one of the trees by introducing a new property that summarizes the differences among the "k-trees" and some rules to evaluate that property. If the heuristics are written in if-then format, these contractions are difficult to spot. Great savings can be achieved if many rules with the same consequent are replaced by one rule, appropriately placed, with the repeated consequent. Contractions of this type are also easy to spot by looking at the "k-trees".

An important consideration is how to order the rules so the questions asked by the system are posed in the appropriate order. We found that we could summarize interactions pictorially and use these summaries to help to order the rules, and the antecedents within individual rules, so questions were asked in the order that we wished. We call these pictorial representations "question lattices". The "question lattice" is formed by writing down the properties about which questions need to be asked. We try to determine what a general interaction with the system will look like. Whenever there are several possible branches for an interaction we create a "question lattice" for those branches. This lattice is then inserted at the appropriate place in the more general lattice. The lattices allow us to determine the ordering of the antecedents so questions are asked in the correct order. Additionally, they provide a method for doing a structured walk-through of the rule base. They also provide another test of the completeness of the rule base.

CONCLUSIONS

The advantages of the "k-tree" and "question lattice" notation include:

- The notation is a compact way to represent every rule.
- The process of constructing a tree is a systematic one. That process, by its very nature, leads to a complete system.
- The notation makes it easy to trace and search for a given antecedent or consequent.
- The "k-trees" make selection of mop-up rules a simple task.
- The "k-trees" point out when contractions of the rules are likely to be useful.
- The "question lattices" allow us to order the rules so that the system will ask questions in the order that we desire.
- The "question lattices" provide us with another check on the completeness of the rules.
- The "k-trees" and "question lattices" provide detailed system documentation.
- The notation represents a step in the direction of developing an algorithmic approach to knowledge engineering.

FUTURE RESEARCH

In order to demonstrate that the "k-tree" notation is implementation independent we intend to port our complete system to different expert system shells from the "k-tree" representation. We will investigate whether "k-trees" can provide us with a complexity measure similar to complexity measures of procedural programs. We intend to apply the techniques described here to several different types of expert systems.

---