(All rule (Conclusion rule Ident Organism-1 E. Coli cf)) and then display each rule in that list.

A why question is essentially a request to display the current reasoning chain, or the chain of rule calls extant at that time. The housekeeping necessary to determine this chain is accomplished as a side-effect of evaluating the first condition in a Loglisp rule. As we have seen, if a Loglisp rule is numbered X, the first condition in that rule is (Rule X). The evaluation of this condition results in pushing X onto a global stack that grows downward, called Current_Rules. Thus, the top of the stack always represents the rule being currently considered.

If, during the consideration of Rule X, Rule Y is referenced, we first push a separator symbol "[]" before pushing Y on Current_Rules. To determine the rule that called Y, we "climb" the stack till we find the []. The rule number immediately above it represents the rule that called Y. The rule that called X, as well as the chain of rule calls, may be similarly determined. Details of how the stack is maintained may be found in Narain. 5

Other aspects of Mycin translated. Mycin was translated more completely than might have been implied by the account given here (details may be found in Narain5). Besides the function Same, all other functions occurring in premises of Mycin rules (about 15 in all) were translated. A mechanism for the growth of the context tree at runtime was implemented to simulate that of Mycin. Mycin's method of preventing reasoning loops at runtime was also implemented. All diagnostic rules (about 103), including rules with disjuncts in the premise, were translated into assertions of Loglisp. Mycin's model of inexact reasoning was completely reexpressed in Loglisp. The translated version does runtime-type checking of user responses in exactly the style of Mycin. A simple browser for the knowledge base was also built.

Aspects of Mycin not translated. Because of the unavailability of Mycin algorithms used for implementing its natural-language front end, and because designing such a front end is a significant problem in itself, there is no facility in the translated version for English-like discourse with the system in the style of Mycin. Therapy recommendation in Mycin was done using algorithmic means, not rule-based reasoning. For this reason, and because of the unavailability of the relevant algorithms, the therapy recommendation capability of Mycin was also not translated.

Space and time requirements
Mycin was implemented in Interlisp on a DECSystem-10 running the Tenex operating system. The total space requirement for Interlisp, the compiled Mycin program, the knowledge base, knowledge tables, and working area was about 245K words of memory. The total elapsed interaction time during the consultation recorded in Shortliffe1 was about 20 minutes. No corresponding CPU time seemed to be mentioned.

Loglisp is written in Rutgers-UCI Lisp on a DECSystem-10 running the Tops-10 operating system. The total space requirement for Lisp, Loglisp, the translated knowledge base, Lisp functions, and working area was about 150K words. The total CPU time for the diagnosis portion (including explanations) of the consultation in Shortliffe1 was about 2.5 minutes.

Other approaches for building expert systems
Most other Prolog-based expert systems6,7 use essentially the same approach as that outlined in this article. Data structures are handled using pattern matching, if-then rules are represented using Horn clauses under procedural or declarative interpretation, and the inference engine is built on top of the resident Prolog theorem prover. Since the Prolog theorem prover is already quite powerful, the inference engines built on top of it are very compact.

An elegant implementation of an explanation system in Prolog is given...