PRISM: PRototype Inference SysteM

by P. HIRSCH, M. MEIER, S. SNYDER, R. STILLMAN
IBM Corporation
Palo Alto, California

ABSTRACT

A research project at IBM's Palo Alto Scientific Center, called PRISM (PRototype Inference SysteM), is an experimental “shell”—a general purpose system. This system is designed to provide a base for the construction of a variety of specific expert systems. PRISM is in operation within IBM at several sites where it is being used for building applications. A knowledge base builder uses the English-like rules and parameters to construct an application more rapidly than can be done with standard programming techniques.
INTRODUCTION

A research project at IBM's Palo Alto Scientific Center, called PRISM (PRototype Inference System), is an experimental “shell”—a general purpose system. This system is designed to provide a base for the construction of a variety of specific expert systems. The motivation for designing PRISM arose from observing the difficulty with which expert systems are built—often, each system is one of a kind. There is a need for an easy way to build expert systems that can run efficiently and be used in a variety of problem areas.

There are three users of an expert systems shell:

1. The client or end-user of the application
2. The expert
3. The knowledge engineer, who helps the experts encode their knowledge

PRISM incorporates a set of procedures used to design a variety of applications—an empty system into which the knowledge-base builder (an expert or knowledge engineer) inserts his own rules (to define the knowledge base) and choose a reasoning method (the inference engine) that will apply the rules. The result is a specific export system ready for the end-user. PRISM incorporates an editor program that allows a knowledge engineer to create the knowledge base in the form of English-like rules, parameters, and controls.

PRISM also includes a set of basic inference-processing functions. Examples include backward chaining (obtaining a value for a desired goal by working backward to a given premise) and forward chaining (proceeding from given data to make inferences and draw conclusions or carry out actions). From these basic functions, PRISM’s control language permits the building of more complex functions. It is the application of these functions to the rules in the knowledge base that determines the particular characteristics of the expert system. In each system the base set of knowledge-processing functions is the same; what differs is the sequence and focus of their execution.

The PRISM system is now in operation on an IBM System/370 under the VM/CMS operating system. A knowledge base builder uses the English-like rules and parameters to construct an application more rapidly than is possible with standard programming techniques. A knowledge engineer can use either of two standard inference techniques—backward or forward chaining—and can control the inference process through a control language. In addition, a knowledge engineer can organize or group the knowledge base into hierarchical structures, called focus control blocks (FCBs). Each FCB has its own inference engine and its own control steps. PRISM also allows for easy attachment of external procedures to the system to provide or use information from other programs, files, or data bases.

KNOWLEDGE BASE OBJECTS

A PRISM knowledge base is constructed from three types of objects: rules, parameters, and FCBs. Rules contain the logic structure and provide heuristic and definite facts about the application. Parameters are the basic value-holding objects of the system and also provide initial value constraints. FCBs allow a knowledge base builder to control PRISM through a control language and also subdivide the knowledge base into components. All of the objects are entered through an integrated editor that automatically parses and analyzes the information as it is entered. If there is a semantic or syntactic error, immediate feedback is given that allows the knowledge base builder to correct the error at once.

RULES

The production rules are of the “IF...THEN...” form, for example:

1. If income > 30000 then tax rate > .3
2. If Animal produces is ‘gives milk’ then animal class is ‘mammal’
3. If animal class is ‘insect’ or animal eats is ‘nibbles at cotton’ then there is some evidence that animal is ‘boll weevil’
4. If mammal class is ‘ungulate’ and certainty that (animal is ‘zebra’) < .5 then there is certainty .8 evidence that animal is ‘horse’

Note that Rules 2 through 4 are “fuzzy” if rules, FIF rules, which propagate the certainty of the premise to the certainty of the conclusion, whereas Rule 1 is an IF rule and the certainty of the premise is not propagated to the uncertainty of the conclusion. In Rule 1, if the rule premise is true, the conclusion will always be asserted with certainty 1.0. Also note that Rule 4 is a self-referencing rule; that is, the parameter ANIMAL is mentioned both in the premise and in the conclusion.

CERTAINTY

PRISM can maintain heuristic knowledge as well as factual knowledge through the use of uncertain reasoning. A knowledge base builder can create fuzzy rules, such as the FIF rules, through phrases like those shown in rule 3, “there is some evidence,” and in Rule 4, “certainty that (animal is ‘zebra’)
"< .5" and "certainty .8 evidence that." In addition, the user can supply a certainty value along with the answer, for example:

What is the animal class?
- .8 Insect
- Mammal
- Ungulate

In this example, the user determined that the animal class was Insect, with certainty .8.

PARAMETERS

In addition to rules, the knowledge-base builder creates parameters. These parameters can have properties such as constraints, prompt messages, and indications of where information is to be obtained. The parameter constraint could be a list of values or a range of values. For example, the parameter COLOR might be taken from the list ('red', 'white', 'blue') or the parameter INCOME could be constrained to be >0. Parameter values are checked against these constraints during knowledge-base building and consultation. Each parameter has a property that indicates whether the value for that parameter is to come from the processing of rules, from the end-user, from external procedures, or from a default value. In addition, the knowledge-base builder can specify the sequence that the system should use in attempting to obtain the value of the parameter.

For example, the knowledge-base builder might specify that the system should attempt to obtain the parameter value according to the following sourcing sequence:

- External data
- Rule consequent
- User will input from terminal
- Default will be taken

CONTROL

The knowledge engineer may wish to control the dialogue and the processing of the rules in an expert system. The PRISM system permits this through a knowledge object called the focus control block (FCB). The knowledge engineer can partition the knowledge-base hierarchically using FCBs. For example, a system configurator might want the expert system first to focus on the applications, then on the CPU required, and finally on peripheral hardware, such as tape or disk drives. Each component is in a separate control block and they are hierarchically related to one another.

Each FCB can have its own control language. For example a set of control primitives might be as follows:

- Ask initial data;
- Determine goals
- Order rules by least unknown premises first;
- Display results;

In this example, initial data, goals, and results each constitute a group of parameters that the knowledge-base builder has provided. In this example the PRISM system initially would ask for the parameter values that are listed in the group initial data. It then uses a backward-chaining algorithm to determine the goals (another list of parameters) of the system. Upon reaching the goals, the system displays the parameters indicated by the results, which in many cases contain the initial data and the goal parameters.

APPLICATIONS

PRISM is currently being used at several IBM sites. These applications range from diagnostic applications to planning and advisory applications. A sampling of the applications includes a contract advisor for joint research contracts, a system to configure hardware, a hardware diagnostic system, and a system to verify information going into a database.