Rule-Based Programming

Definitions

A rule-based programming environment consists of

- A *data base* (which we'll leave undefined for now);
- A set of *rules*: condition-action pairs. The condition evaluates some aspect of the data base. The action(s) typically modify some aspect of the data base.
- A *monitor* that determines which rules can be applied (i.e. which rules have conditions that evaluate to true), and that chooses an applicable rule and applies it.

What happens in a rule-based environment

The following sequence of events is repeated by the monitor:

1. Decide what rules are applicable.
2. Choose a rule to apply. When there are more than one applicable rule, there are several possible strategies ("meta-rules"):
   a. choose the highest priority rule;
   b. choose the most specific rule;
   c. choose the rule that refers to the element most recently added to the data base;
   d. choose the most recently applied rule;
   e. choose the rule that has been applicable for the longest time;
   f. choose a rule that has not previously been applied;
   g. choose the first rule in the list;
   h. make a random choice.

   The monitor can also choose *not to choose*, i.e. to explore all the applicable rules in parallel.
3. Apply the rule.

How this is done in Scheme

Here is a function that implements the monitor of a rule-based system. It takes a list of rules and a data base as arguments.

```
(define (monitor rules data-base)
   (let ((rule-set (applicable rules database)))
      (if (not (empty? rule-set))
         (monitor rules
            (apply-rule (choose rule-set) data-base) ) ) ) )
```

The applicable function returns a *list* of applicable rules; it often involves pattern-matching. The choose function applies one of the criteria mentioned above. (Note the difference between this and the way a Scheme cond is executed.) The apply-rule function "executes" the actions in the action-part of the rule.

Applications

Rule-based programming is good for *expert systems*, in which expertise is modelled by rules. The expertise is typically difficult to organize into a sequential set of conditions to be checked. Some categories:

category	*problem addressed*
interpretation	inferring situation descriptions from sensor data
prediction	inferring likely consequences of given situations
diagnosis	inferring system malfunctions from observables
design	configuring objects under constraints
planning	designing actions
monitoring	comparing observations to plan vulnerabilities
debugging	prescribing remedies for malfunctions
repair	executing a plan to administer a prescribed remedy
instruction	diagnosing, debugging, and repairing student behavior
control	interpreting, predicting, repairing, and monitoring system behaviors

Note an advantage of rule-based programming: one can ask *why* a given conclusion was reached.

References

Building Expert Systems, F. Hayes-Roth et al. (editors), Addison-Wesley, 1983.

Handbook of Artificial Intelligence, A. Barr et al. (editors), Addison-Wesley.