Paradigms of Programming

The Rule Based Paradigm

Example languages: CLIPS, OPS5, pretty much any Expert System

# Key Features

* State represented by facts.
* Rules evaluated in iterations.
* Each rule evaluates the set of facts and, if the rules conditions are met, alters the facts.
* Changing a fact ends this iteration and starts evaluating the rules from the start again.
* Runtime ends when and iteration evaluates all of the rules and none trigger a change of the facts.

# Description

Rule based programs are composed of a set of facts and a set of rules.

To begin, the system is seeded with facts, like the following:

(customer "Fred" (credit-limit 5000))

(customer "Sally" (credit-limit 1500))

(invoice "Fred" (amount 2000))

(invoice "Fred" (amount 1000))

Then the set of rules are defined. Each rule is a series of criteria which finally ends in an action. When all of the criteria are met by the set of facts, the action is executed. Actions can add, revoke or edit existing rules.

For example, here is a rule (called rule1) which has some criteria (before the =>) and an action (after the =>) to run when all the criteria are true:

// If there is an invoice with an amount greater than or equal to 2000

// change the corresponding customer's credit limit to 2000

(rule1

(invoice ?name (amount >=2000))

=> (customer ?name (credit-limit 2000)

)

Once the seed facts and rules are defined, the program can be run. This means that each of the rules is evaluated in turn. When a rule has its criteria met, its action is run and changes the facts. Then the whole process begins again. This continues until a complete iteration triggers no rules.

# A bit more about it...

The power of the rule paradigm is that it allows work to be done without specifying an explicit flow of a control. The rules are all equal and they may match in any order. This turns design into the process of creating and articulating a set of rules which unambiguously reach particular results.

One of the primary concerns in rule based system design is the creation of rules which will fire only in particular situations. Designers must think carefully about the context of application for many of their rules and treat the system as a large state machine in which there are a rich enough set of transitions to reach an appropriate result.

There are many rule based programming languages (OPS5, CLIPS, etc), however in non-rule based languages, rules are often modelled as data structures subject to a matching operation.

In this exercise, you may assume that all the rule engine runs iteratively on a set of facts until the set of facts is unchanged or a rule tells the engine to abort.