K. J. Somaiya College of Engineering, Mumbai-77

Batch: B2 Roll No.: 110

Experiment / assignment / tutorial No. 2

Title: Implementation of condition-action rules based agent using PROLOG

Objective: Developing a basic level agent program that runs on condition-action rules

Expected Outcome of Experiment:

Course Outcome	After successful completion of the course students should be able to
CO1	Understand the history & various application of AI and choose appropriate agent architecture to solve the given problem.

Books/ Journals/ Websites referred:

1. **https://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html**
2. **http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/pt_framer.html**
3. **http://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/**
4. **"Artificial Intelligence: a Modern Approach" by Russell and Nerving, Pearson education Publications**
5. **"Artificial Intelligence" By Rich and knight, Tata McGraw Hill Publications**
6. **"Prolog: Programming for Artificial Intelligence" by Ivan Bratko, Pearson education Publications**

Pre Lab/ Prior Concepts: Intelligent Agent, Agent Architectures, Rule base Vs Knowledge Based approach

Historical Profile: Agent programs for simple applications need not be very complicated. They can be based on condition-action rules and still they give better results, though not always rational. The family tree program makes use of similar concept.

New Concepts to be learned:

Defining rules, using and programming with PROLOG

A simple agent program can be defined mathematically as an agent function which maps every possible percepts sequence to a possible action the agent can perform or to a coefficient, feedback element, function or constant that affects eventual actions:

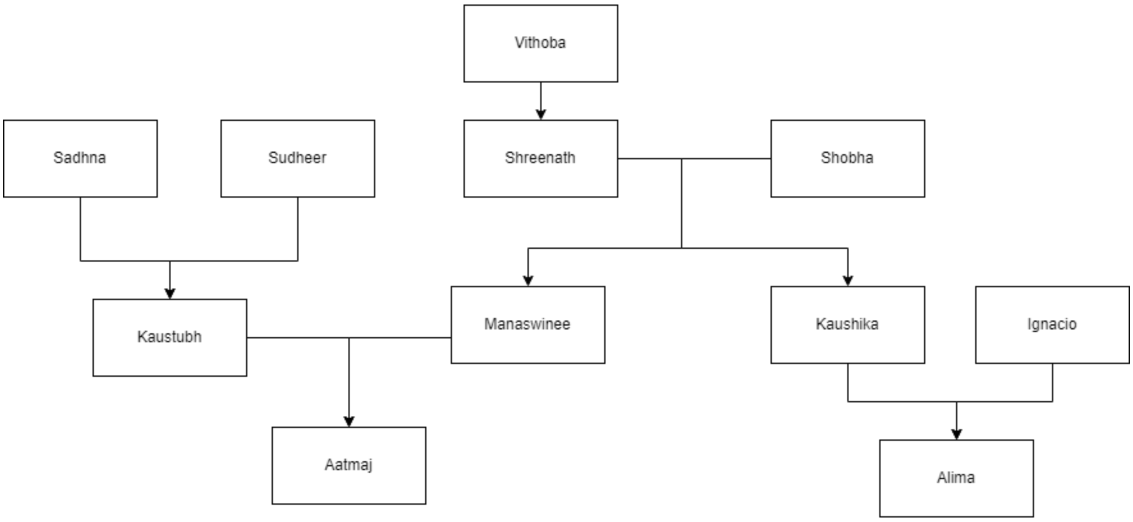
$$F: P * - > A$$

Algorithm for 'Condition-Action Rule Table' Agent function:

```
functionSIMPLE-REFLEX-AGENT (percept) returns an action
Static: rules, a set of condition-action rules
State←INTERPRET-INPUT (percept)
Rule←RULE-MATCH (state, rules)
Action←RULE-ACTION [rule]
Returnaction
```

This approach follows a table for lookup of condition-action pairs defining all possible condition-action rules necessary to interact in an environment.

Example: Family Tree



Base Knowledgebase:

```
male(aatmaj).
male(kaustubh).
male(shreenath).
male(vithoba).
male(sudheer).
male(ignacio).
female(manaswinee).
female(shobha).
female(sadhna).
female(kaushika).
female(alima).
parent(kaustubh,aatmaj).
parent(manaswinee,aatmaj).
parent(sudheer,kaustubh).
parent(sadhna,kaustubh).
parent(shreenath,manaswinee).
parent(shobha,manaswinee).
parent(vithoba,shreenath).
parent(alima,kaushika).
parent(alima,ignacio).
parent(shreenath,kaushika).
parent(shobha,kaushika).
```

Rules:

mother(X,Y):-parent(X,Y),female(X).

father(X,Y):-parent(X,Y),male(X).

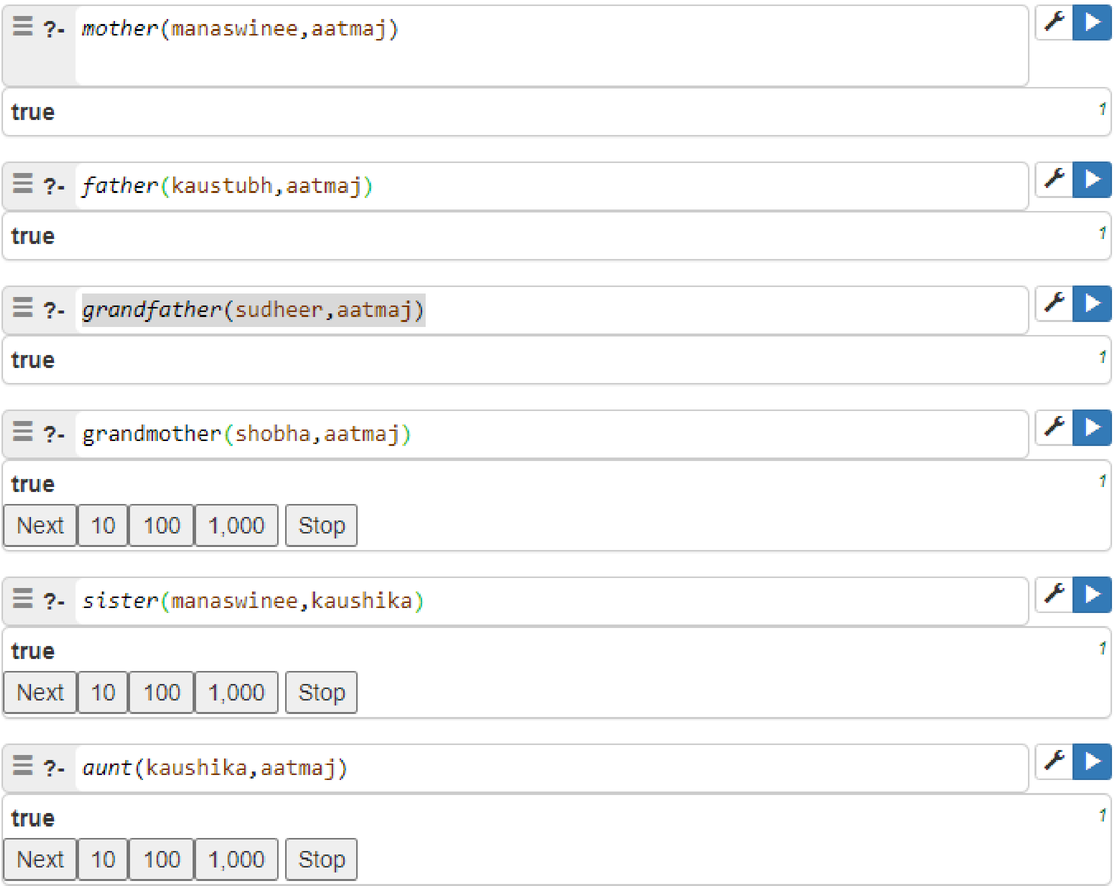
grandfather(X,Y):-father(X,Z),parent(Z,Y).

grandmother(X,Y):-mother(X,Z),parent(Z,Y).

sister(X,Y):-father(Z,Y),father(Z,X).

aunt(X,Y):-sister(X,Z),mother(Z,Y).

Some Sample queries and Outputs:



```
?- mother(manaswinee,aatmaj)
true

?- father(kaustubh,aatmaj)
true

?- grandfather(sudheer,aatmaj)
true

?- grandmother(shobha,aatmaj)
true

?- sister(manaswinee,kaushika)
true

?- aunt(kaushika,aatmaj)
true
```

.

Post Lab Objective Questions

1. **The PROLOG suit is based on**
    - **a.** Interpreter
    - **b.** Compiler
    - c. None of the above

Answer: Compiler

2. **State true or false**

   There must be at least one fact pertaining to each predicate written in the PROLOG program.

Answer: true

3. **State true or false**

   In the PROLOG program the variable declaration is a compulsory part.

Answer: false

Post Lab Subjective Questions

1. **Differentiate between a fact and a predicate with syntax.**

Fact: A fact is a simple statement that asserts a relationship or a property. It represents a piece of information that is considered to be true. In Prolog, facts are often used to define relationships between entities.
Example: father(john, bob). - This fact asserts that John is the father of Bob.

Predicate with Syntax: A predicate is a rule or a relationship that can involve variables. It can be seen as a more general form than a fact. Predicates often include variables and can represent a class of relationships.
Example: parent(X, Y) :- father(X, Y). - This predicate defines a relationship between X and Y based on the father relationship.

2. **Differentiate between knowledge based and Rule base approach.**

Knowledge-Based Approach: In a knowledge-based approach, information is represented as a collection of facts. This approach focuses on the storage and retrieval of information without explicit rules or reasoning.

Rule-Based Approach: In a rule-based approach, in addition to facts, rules are used to represent relationships and logic. These rules provide a way to infer new information from existing facts and relationships.

3. **Differentiate between database and knowledge base.**

Database: A database is a structured collection of data. It is designed for efficient storage, retrieval, and management of data. Databases are generally associated with traditional data management systems.

Knowledge Base: A knowledge base is a collection of information, often in the form of facts and rules, that represents knowledge about a particular domain. Knowledge bases are commonly associated with AI and expert systems.

4. What is a ‘free variable’? Explain with an example.

A free variable is a variable in a logical formula or rule that is not universally or existentially quantified within that formula. It is not bound by any quantifier and can take any value.

Example:
In the Prolog rule ancestor(X, Y) :- parent(X, Y)., the variable Y is a free variable. It is not quantified, allowing it to take any value when the rule is applied. The rule asserts that X is an ancestor of Y if X is a parent of Y, where Y is a free variable.