

|  |
| --- |
| **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 Knowledgebase 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:**

**function**SIMPLE-REFLEX-AGENT (percept) **returns** an action

**Static:** *rules,* a set of condition-action rules

*State*🡨INTERPRET-INPUT (percept)

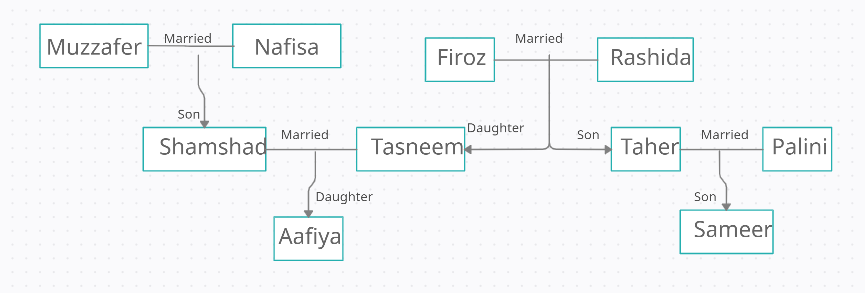
*Rule*🡨*RULE-MATCH (state, rules)*

*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/disease-symptom mapping/ City map with their distances between them:**



**Base Knowledgebase:**

male(firoz).

male(shamshad).

male(taher).

male(sameer).

male(muzzafer).

female(rashida).

female(tasneem).

female(palini).

female(aafiya).

female(nafisa).

parent(firoz,tasneem).

parent(rashida,tasneem).

parent(firoz,taher).

parent(rashida,taher).

parent(shamshad,aafiya).

parent(tasneem,aafiya).

parent(taher,sameer).

parent(palini,sameer).

parent(nafisa,shamshad).

parent(muzzafer,shamshad).

**Rules:**

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

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

hasChild(X):-parent(X,\_).

sibling(X,Y):-parent(Z,X),parent(Z,Y),X\=Y.

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

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

couple(X,Y):- parent(X,Z),parent(Y,Z),X\=Y.

aunt(X,Y) :- female(X),sibling(X,Z),parent(Z,Y).

uncle(X,Y) :- male(X),sibling(X,Z),parent(Z,Y).

cousin(X,Y):-parent(A,X),parent(B,Y),sibling(A,B).

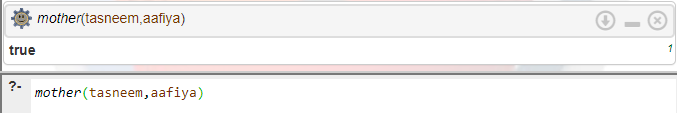
sisterinlaw(X,Y):-female(X),sibling(X,Z),couple(Y,Z).

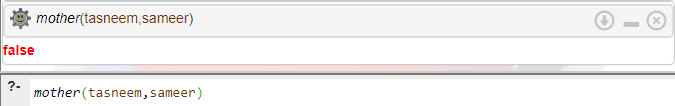
brotherinlaw(X,Y):-male(X),sibling(X,Z),couple(Y,Z).

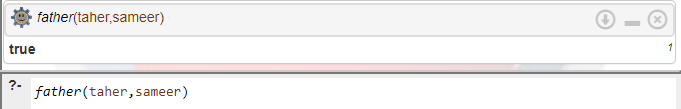
motherinlaw(X,Y):-couple(Z,Y),female(X),parent(X,Z).

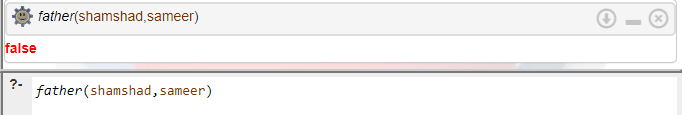
fatherinlaw(X,Y):-couple(Y,Z),male(X),parent(X,Z).

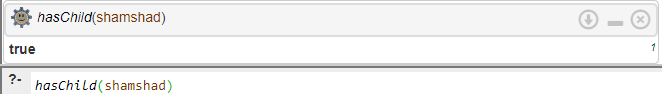
**Some Sample queries and Outputs:**

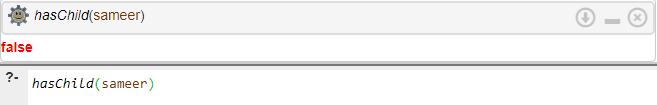
****

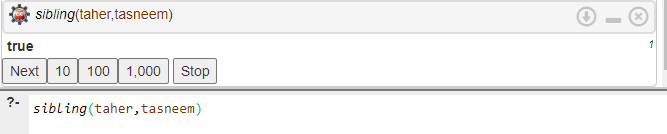
****

****

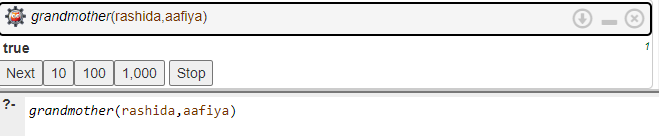
****

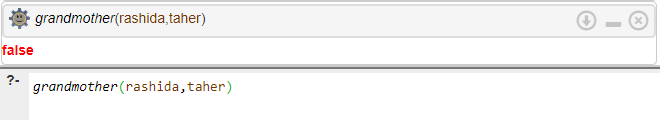
****

****

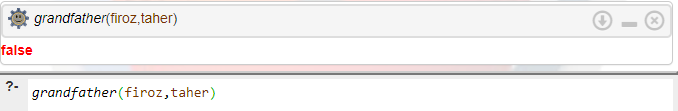
****

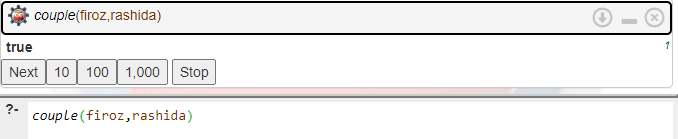
****

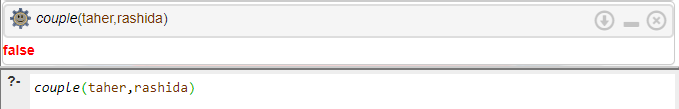
****

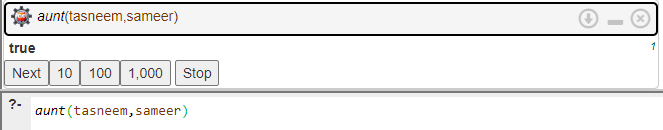
****

****

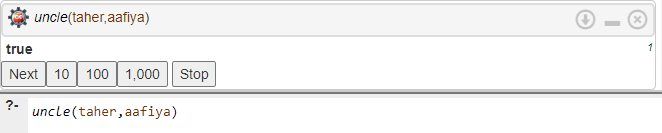
****

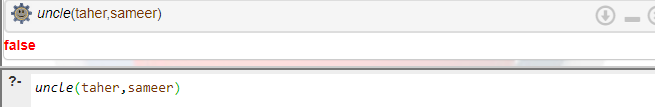
****

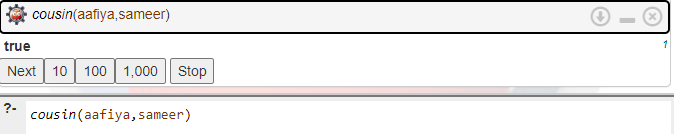
****

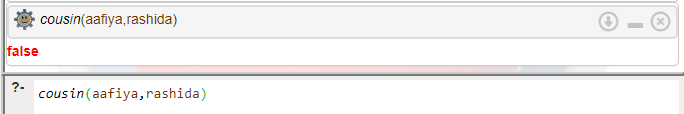
****

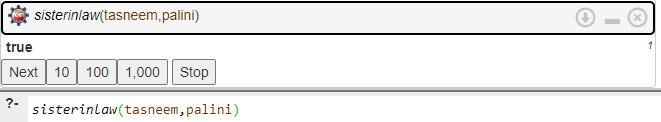
****

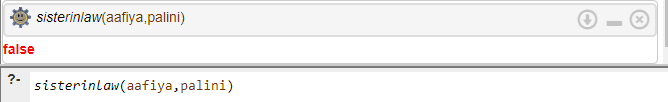
****

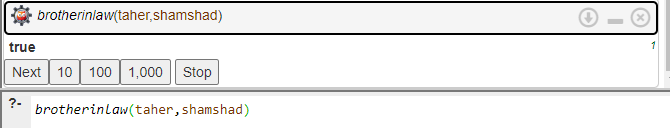
****

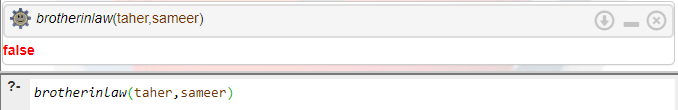
****

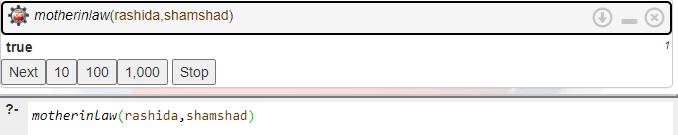
****

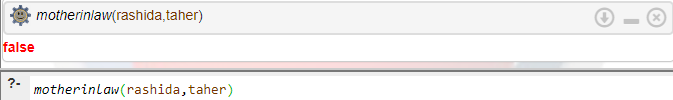
****

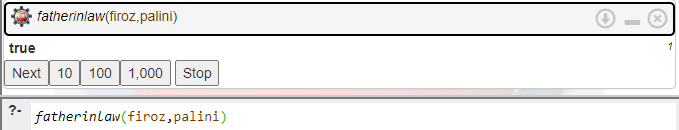
****

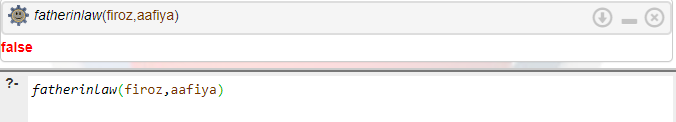
****

****

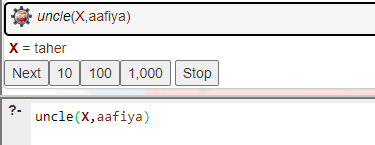
****

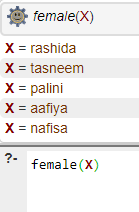
****

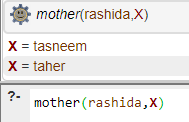
****

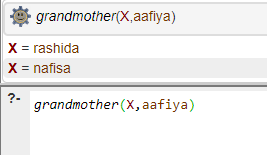
****

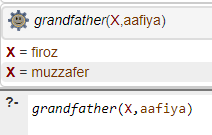
**Free Variables:**

****

****

****

****

****

**Team Members:**

**1.**

**2.**

**Post Lab Objective Questions**

1. **The PROLOG suit is based on**
   1. Interpreter
   2. Compiler
   3. None of the above

**Answer: a] Interpreter**

1. **State true of false**

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

**Answer: True**

1. **State true of false**

In PROLOG program the variable declaration is a compulsory part.

**Answer: True**

**Post Lab Subjective Questions**

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

**Predicate:**

* A predicate is a name given to a fact or a rule. It is defined in the ‘Predicate’ section.
* By defining a predicate we specify which information needs to be known for the property denoted by the predicate to be true.
* Syntactically: cousin(name,name)

**Fact:**

* A fact on the other hand asserts some property of an object, or the relation between two or more objects.
* Syntactically, one fact for the above given predicate can be written as: cousin(aafiya,sameer).

1. **Differentiate between knowledgebase and Rule base approach.**

|  |  |
| --- | --- |
| Knowledgebase | Rule Base |
| A knowledge based system codifies human knowledge typically in a declarative format and acts upon the knowledge. | A rule based system is typically written using a rule language or as if-then statements. |
| It could be written as algorithmic procedures, or cases, or as ontologies, or even as neural networks. | Rule based systems can also be written in Basic and C |
| The important facet is that human knowledge is codified and acted upon. | Loops until some terminating condition is reached. |

1. **Differentiate between database and knowledgebase.**

|  |  |
| --- | --- |
| **Database** | **Knowledgebase** |
| A database is a collection of information organized for easy access, management, and updating. | A knowledge base is a repository typically used to store answers to questions or solutions to problems enabling rapid search, retrieval, and reuse, either by help desk personnel, or directly by those needing support. |
| Collection of data in the database represents facts. | It consists of information at a higher level of abstraction. |
| Operates on a single object. | Operates on class of objects. |
| Maintained for operational purpose. | Used for planning and data analysis. |
| The data it contains is in a basic form, and thus needs to be further analyzed and/or processed before it can be applied. | The entries in the knowledge base are fully-developed and ready to be applied. |

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

A free variable is a variable that has no limitations. It isn't in jail or tied up in any way. It can represent whatever number it needs to represent. Also, the function or expression depends on the free variables.

For example, the x in the following function is a free variable.f(x) = 3x - 1