Batch: B1	Roll No.: 16010121045
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:

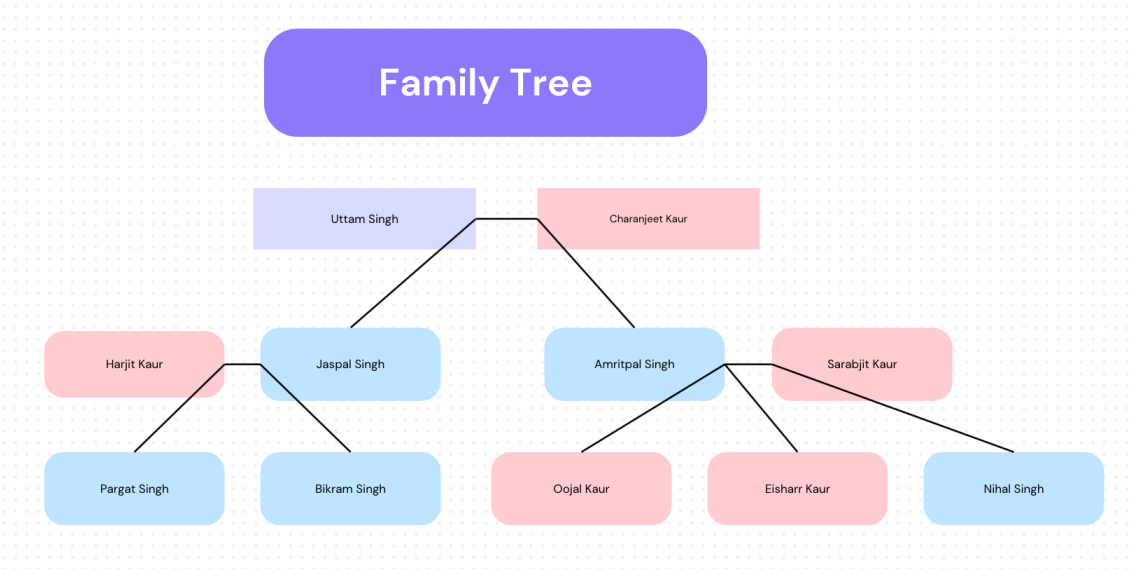
$$F: P * - > A$$

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



Base Knowledgebase:

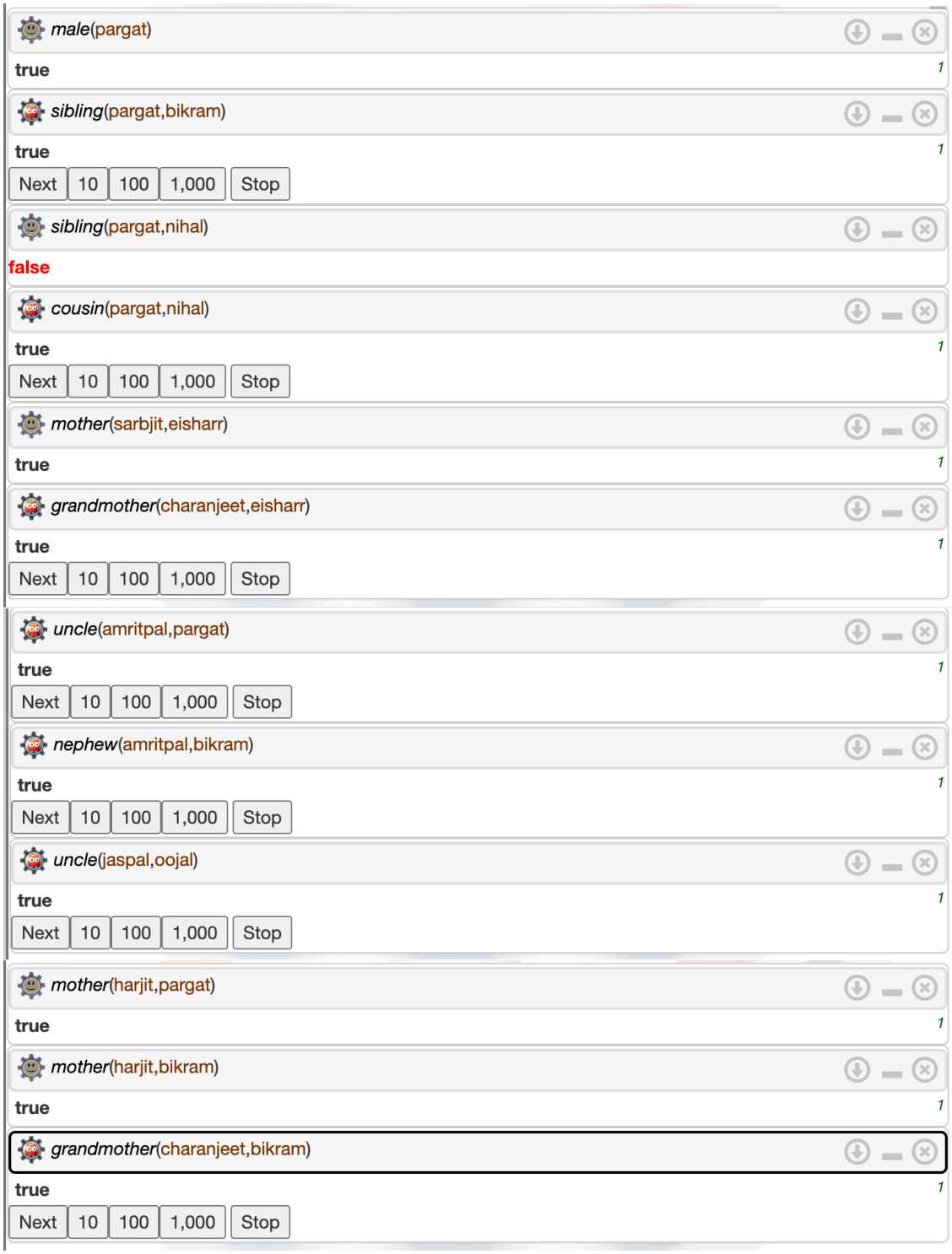
```
male(jaspal).
male(pargat).
male(bikram).
male(uttam).
male(amritpal).
male(nihal).
female(charanjeet).
female(harjit).
female(sarbjit).
female(oojal).
female(eisharr).

parent(uttam,jaspal).
parent(charanjeet,jaspal).
parent(uttam,amritpal).
parent(charanjeet,amritpal).
parent(jaspal, pargat).
parent(jaspal, bikram).
parent(harjit, pargat).
parent(harjit, bikram).
parent(amritpal, nihal).
parent(amritpal, eisharr).
parent(amritpal, oojal).
parent(sarbjit, nihal).
parent(sarbjit,eisharr).
parent(sarbjit,oojal).
```

Rules:

```
father(X, Y) :- male(X), parent(X, Y).
mother(X, Y) :- female(X), parent(X, Y).
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
grandfather(X, Y) :- male(X), grandparent(X, Y).
grandmother(X, Y) :- female(X), grandparent(X, Y).
uncle(X, Y) :- male(X), sibling(X, Z), parent(Z, Y).
aunt(X, Y) :- female(X), sibling(X, Z), parent(Z, Y).
sibling(X, Y) :- parent(Z, X), parent(Z, Y), X \= Y.
cousin(X, Y) :- parent(Z, X), sibling(Z, W), parent(W, Y).
nephew(X, Y) :- male(X), sibling(Z, X), parent(Z, Y).
niece(X, Y) :- female(X), sibling(Z, X), parent(Z, Y).
```

Some Sample queries and Outputs:



.

Post Lab Objective Questions

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

Answer: Interpreter

2. **State true or false**
   There must be at least one fact pertaining to each predicate written in the PROLOG program.

Answer: False

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.

A fact is a straightforward assertion that illustrates the connection between two things. A predicate, followed by a list of constants or variables that correspond to the predicate's arguments, is used to represent a fact.

Think of a parent (nitin,rahi).
Parent is a predicate in this case, while nitin and rahi are constants that stand in for the predicate's arguments. This is a fact, not a rule, as shown by the period ('.') at the end of the line.

A more generic term, predicate, can be used to describe both facts and rules. An object relationship between objects is defined by a predicate, which is utilised in both queries and inferences. For instance: parent(Z, Y): grandparent(X, Y), parent (Y, Z).
Grandparent is the predicate in this case, and the variables X, Y, and Z stand in for the predicate's arguments. This is a rule rather than a fact, as the:- operator denotes. The parent predicate and the relationships between parents and children are used to characterise the relationship between a grandparent X and a grandchild Y.

2. Differentiate between knowledgebase and Rule base approach.

According to the knowledge base method, knowledge is primarily represented as a collection of claims or facts about the outside world. When the objective is to capture and represent the domain, this method is employed in numerous expert systems and knowledge-based systems.

3. Differentiate between database and knowledgebase.

A database is a group of organised data that is stored and accessed online. Databases are used to store and manage huge volumes of structured data, including customer information, sales transactions, and inventory levels. Data insertion, data retrieval, data update, and data deletion are all operations that databases are made to support. They are frequently applied in situations where quick and effective data access is necessary.

On the other hand, a knowledge base is a repository of knowledge that expert systems and artificial intelligence (AI) use to support decision-making and problem-solving. The domain knowledge of a specialist or group of specialists in a given topic is represented by a knowledge base. Unlike a database, a knowledge base can store more complicated information like rules, procedures, and causal links in addition to organised data.

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

In Prolog, a variable that occurs in a query but has no assigned value is referred to as a free variable. To put it another way, a free variable is a variable that is used to indicate an undetermined value.

Take the Prolog query below, for instance: the parent (X, ashwini)

In this case, the name of Ashwini's parent is denoted by the free variable X. Prolog will search its knowledge base when this query is conducted for a fact that matches the predicate parent and the second argument harry. If a matching fact is discovered, it will combine the first argument with the unvalued free variable X.

In this method, the results returned by the search process are stored in free variables, which represent unknowns in Prolog queries. Prolog can express incomplete information in its knowledge base and identify all potential answers to a question by using free variables.