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

Batch: C1 Roll No.: 16010122323
Experiment / assignment / tutorial No. 2
Grade: AA / AB / BB / BC / CC / CD /DD
Signature of the Staff In-charge with date

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.
CO3	Represent and formulate the knowledge to solve the problems using various reasoning techniques

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:

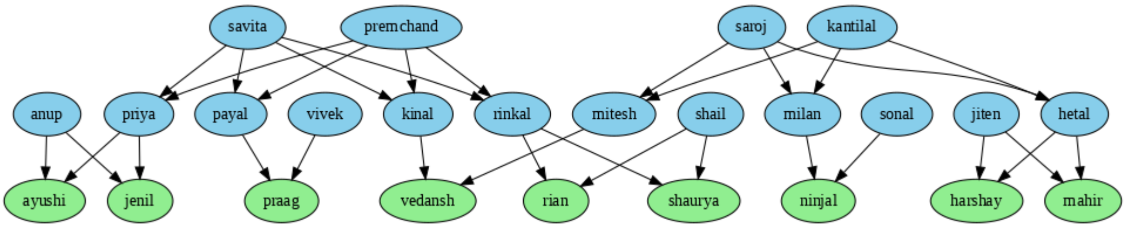
$$F: P * - > A$$

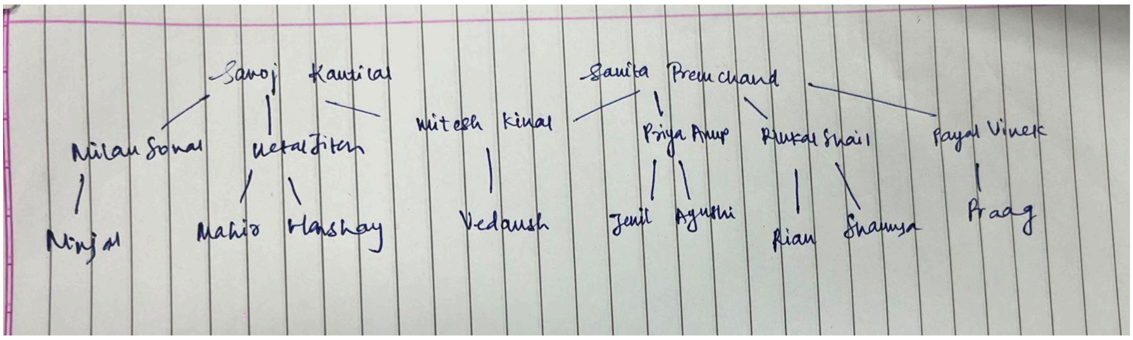
Algorithm for 'Condition-Action Rule Table' Agent function:

```
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
```

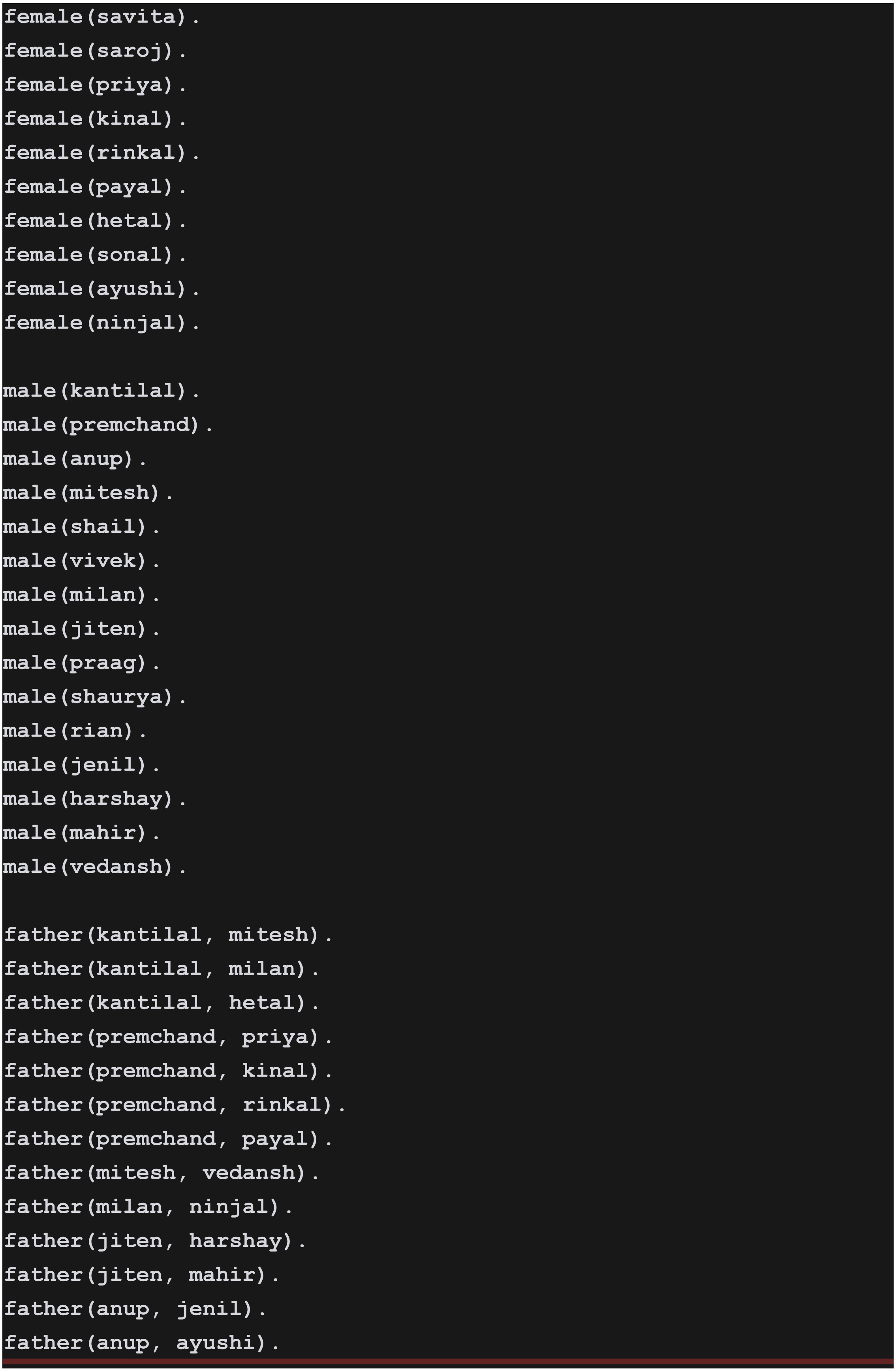
This approach follows a table for lookup of condition-action pairs defining all possible condition-action rules necessary to interact in an environment.

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

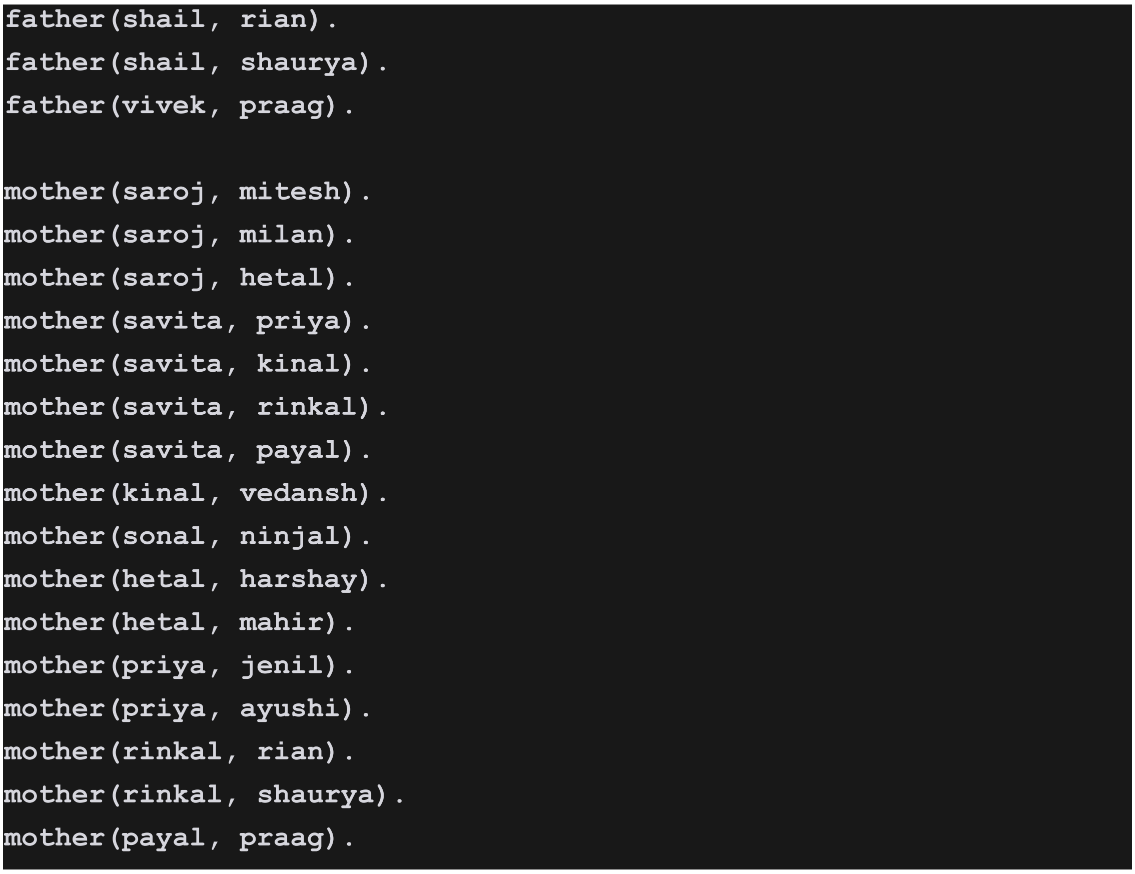



Saroj Kantilal
Sarita Premchand
Milan Sonal
Hetal Jiten
Mitesh Kinal
Priya Anup
Rupal Shail
Payal Vivek
Ninjal
Mahir
Akshay
Vedansh
Jenil
Ayushi
Rian
Shanya
Praag

Base Knowledgebase:

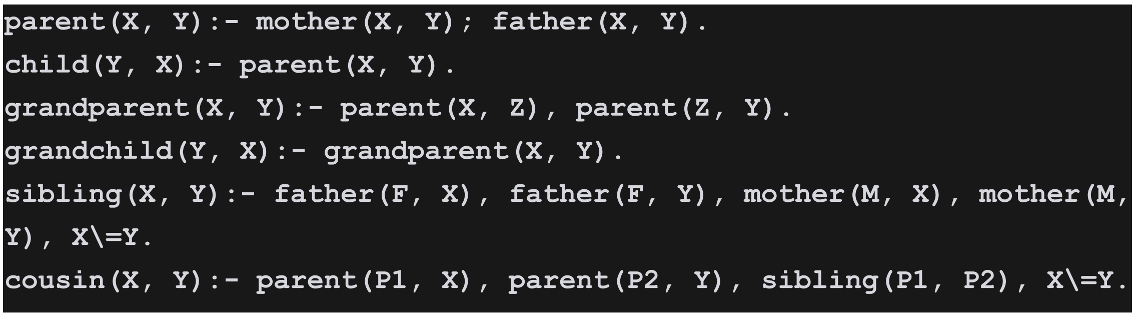


```
female(savita).
female(saroj).
female(priya).
female(kinal).
female(rinkal).
female(payal).
female(hetal).
female(sonal).
female(ayushi).
female(ninjal).

male(kantilal).
male(premchand).
male(anup).
male(mitesh).
male(shail).
male(vivek).
male(milan).
male(jiten).
male(praag).
male(shaurya).
male(rian).
male(jenil).
male(harshay).
male(mahir).
male(vedansh).

father(kantilal, mitesh).
father(kantilal, milan).
father(kantilal, hetal).
father(premchand, priya).
father(premchand, kinal).
father(premchand, rinkal).
father(premchand, payal).
father(mitesh, vedansh).
father(milan, ninjal).
father(jiten, harshay).
father(jiten, mahir).
father(anup, jenil).
father(anup, ayushi).
```



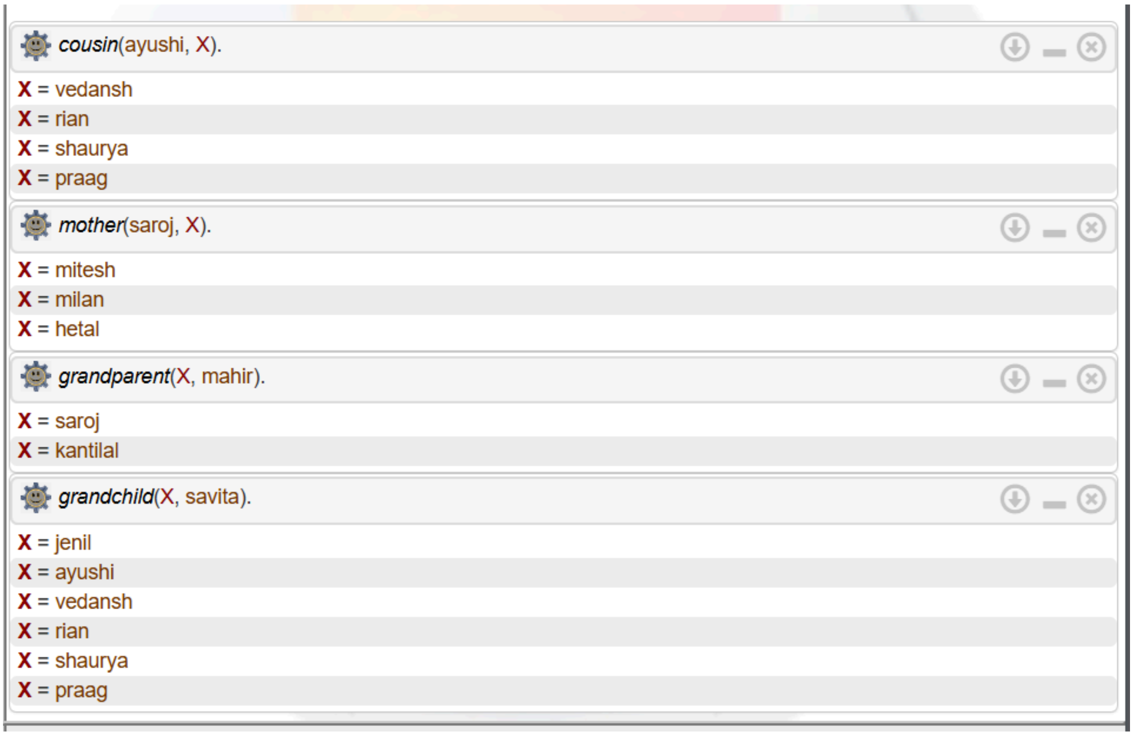
```
father(shail, rian).
father(shail, shaurya).
father(vivek, praag).

mother(saroj, mitesh).
mother(saroj, milan).
mother(saroj, hetal).
mother(savita, priya).
mother(savita, kinal).
mother(savita, rinkal).
mother(savita, payal).
mother(kinal, vedansh).
mother(sonal, ninjal).
mother(hetal, harshay).
mother(hetal, mahir).
mother(priya, jenil).
mother(priya, ayushi).
mother(rinkal, rian).
mother(rinkal, shaurya).
mother(payal, praag).
```

Rules:



```
parent(X, Y):- mother(X, Y); father(X, Y).
child(Y, X):- parent(X, Y).
grandparent(X, Y):- parent(X, Z), parent(Z, Y).
grandchild(Y, X):- grandparent(X, Y).
sibling(X, Y):- father(F, X), father(F, Y), mother(M, X), mother(M,
Y), X\=Y.
cousin(X, Y):- parent(P1, X), parent(P2, Y), sibling(P1, P2), X\=Y.
```

Some Sample queries and Outputs:



```
cousin(ayushi, X).
X = vedansh
X = rian
X = shaurya
X = praag

mother(saroj, X).
X = mitesh
X = milan
X = hetal

grandparent(X, mahir).
X = saroj
X = kantilal

grandchild(X, savita).
X = jenil
X = ayushi
X = vedansh
X = rian
X = shaurya
X = praag
```

.

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 of false**

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

Answer:False

- **A predicate can exist without any facts, especially if it is meant to define relationships or rules that derive facts dynamically through queries.**

3. **State true of false**

    In PROLOG program the variable declaration is a compulsory part.

Answer: False

PROLOG does not require explicit variable declarations. Variables are dynamically identified based on their uppercase starting letter in queries or rules.

Post Lab Subjective Questions

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

Aspect	Fact	Predicate
Definition	A fact is a basic statement that declares a relationship or property to be true.	A predicate is a general definition or rule that can take variables and be used to infer facts.
Purpose	Represents concrete, specific knowledge.	Represents general knowledge and logic used for reasoning.
Syntax	`fact_name(argument1, argument2, ...).`	`predicate_name(Variable1, Variable2, ...) :- condition1, condition2, ... .`
Example	`parent(kantilal, mitesh).`	`child(X, Y) :- parent(Y, X).`

---

2. Differentiate between knowledgebase and rule-based approach:

Aspect	Knowledge Base Approach	Rule-Based Approach
Definition	A collection of factual data or information stored in the form of facts.	A method of reasoning or decision-making based on a set of logical rules.
Structure	Stores static facts about a domain.	Defines dynamic rules to infer new facts or take decisions.
Example	Facts like `male(kantilal).` or `father(kantilal, mitesh).`	Rules like `grandparent(X, Y) :- parent(X, Z), parent(Z, Y).`
Use Case	Focuses on describing the domain.	Focuses on deriving or inferring relationships and solving problems.

---

3. Differentiate between database and knowledgebase:

Aspect	Database	Knowledge Base
Definition	A structured collection of data stored for retrieval and management.	A system that stores facts, rules, and logic for reasoning and problem-solving.
Focus	Focused on data storage and retrieval.	Focused on reasoning and decision-making using stored knowledge.
Structure	Data is organized in tables or structured formats.	Contains facts and rules in logical formats.
Example	SQL database with employee records.	PROLOG program with facts like `employee(john)` and rules like `manager(X, Y)`.
Use Case	Used for CRUD operations (Create, Read, Update, Delete).	Used for problem-solving and artificial intelligence applications.

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

Definition:

A *free variable* in PROLOG is a variable that does not have a specific value assigned to it when it is used in a query or rule. It can take any value that satisfies the constraints of the program.

Example:
prolog
CopyEdit

```
% Facts
father(kantilal, mitesh).
father(kantilal, milan).

% Query with a free variable
?- father(kantilal, X).
```

- Explanation:

- Here, `X` is a free variable. When the query is run, PROLOG tries to find all values of `X` that satisfy the `father(kantilal, X)` condition.

Output:
prolog
CopyEdit

```
X = mitesh ;
X = milan.
```

-

- The free variable allows querying relationships without needing a fixed input, making PROLOG dynamic and flexible for reasoning.