**Experiment 1: Write simple programs in Prolog for facts, rules, and queries.**

**Aim:**

To write simple programs in **Prolog** using **facts**, **rules**, and **queries** to represent and infer knowledge based on logical reasoning.

**Description:**

**Prolog** (Programming in Logic) is a declarative logic programming language used for artificial intelligence and computational linguistics.

The main components of a Prolog program are:

1. **Facts**: Define known information.

father(john, david).

1. **Rules**: Define relationships based on facts and other rules.

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

1. **Queries**: Ask questions to infer answers based on facts and rules.

?- grandfather(john, david).

**Facts**

In Prolog, a fact is a simple declarative statement that asserts a relationship between entities. Facts are the fundamental building blocks of a Prolog program and are always assumed to be true.

Syntax of Facts:

* A fact consists of a predicate (a name) followed by arguments enclosed in parentheses, and terminated by a full stop (period).
* The predicate name should start with a lowercase letter.
* Arguments can be constants (atoms or numbers) or variables.
* Facts are essentially ground clauses (clauses without variables), and they represent the simplest form of a Prolog clause.

Example:

parent(john, mary).

male(john).  
female(mary).  
likes(john, pizza).

In these examples:

* parent(john, mary) states that john is the parent of mary.
* male(john) states that john is male.
* female(mary) states that mary is female.
* likes(john, pizza) states that john likes pizza.

These facts define relationships and properties within a knowledge base that Prolog can use to answer queries and deduce new information.

**Rules**

In Prolog, a rule defines a relationship between facts or other rules. It has a head (conclusion) and a body (conditions), separated by :- (meaning "if"). If the conditions in the body are true, then the head is considered true. Rules are used to infer new information from existing facts and rules within a Prolog program.

Code

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

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

This example defines a parent as either a father or a mother. parent(X, Y) is the head, and father(X, Y) or mother(X, Y) are the conditions in the body. The comma (,) acts as a logical AND, meaning both conditions must be true for the rule to be satisfied.

Here's a breakdown:

* **Head:**

The left-hand side of the rule, representing the conclusion. In the example, it's parent(X, Y).

* **Body:**

The right-hand side of the rule, containing conditions that must be met for the head to be true. In the example, it's father(X, Y).

* :-:

This symbol, known as the "neck symbol", separates the head and the body of the rule, and is read as "if".

* .:

A period (full stop) at the end of the rule signifies the end of the clause.

Rules can also contain variables (like X and Y in the example), which allow for generalization. A rule with variables will be true for any values that satisfy the conditions.

For example, the rule friends(X, Y) :- likes(X, Y), likes(Y, X). states that X and Y are friends if X likes Y and Y likes X. This rule uses the comma (,) to represent the logical AND, indicating that both conditions must be true.

Furthermore, multiple rules can exist for the same predicate (e.g., parent/2 in the first example). The Prolog engine will try each rule until one succeeds or all have been tried.

**Queries**

In Prolog, queries are used to ask questions about the knowledge base (facts and rules). They are written after the prompt ?- and terminated with a full stop (.). Queries can involve variables, which Prolog attempts to unify with known facts and rules to find solutions.

Basic Query Syntax:

* ?-: This is the prompt symbol, indicating a query.
* .: The full stop (period) marks the end of the query.
* **Variables**: Variables in Prolog start with an uppercase letter or an underscore

(e.g., X, Name, \_variable).

* **Predicates**: Queries use predicates (like facts and rules) to express relationships.

Example:

Let's say you have a knowledge base with the following facts:

parent(john, mary).

parent(jane, peter).parent(peter, sue).

Here are some example queries:

**Simple Fact Query:**

?- parent(john, mary).

This query asks if the fact "parent(john, mary)" is true. The answer would be yes.

**Query with a Variable:**

?- parent(john, X).

This query asks, "Who is a child of john?". Prolog will attempt to find a value for X that makes the parent relation true. The answer would be X = mary.

**Query with Multiple Goals (and):**

?- parent(X, Y), parent(Y, sue).

This query asks, "Who is a parent of someone who is a parent of sue?". Prolog will find the grandparent-grandchild relationship where the grandchild is sue. The answer would be X = john, Y = peter.

**Query with Negation (not):**

?- not(parent(jane, john)).

This query asks if it's not true that jane is a parent of john. The answer would be yes.

**Explanation of Concepts:**

| **Component** | **Description** | **Example** |
| --- | --- | --- |
| **Fact** | A statement that is always true | father(john, mary). |
| **Rule** | A logical implication (if-then) | parent(X, Y) :- father(X, Y). |
| **Query** | A question to retrieve info | ?- parent(john, Y). |

**Sample Input (Prolog Program)**

% Facts

father(john, david).

father(john, mary).

father(david, kevin).

mother(susan, david).

mother(susan, mary).

mother(mary, alice).

% Rules

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

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

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

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

**Sample Queries and Output**

**Query 1**:

?- father(john, mary).

**Output**:

true.

**Query 2**:

?- grandfather(john, kevin).

**Output**:

true.

**Query 3**:

?- parent(susan, mary).

**Output**:

true.

**Query 4**:

?- parent(X, david).

**Output**:

X = john ;

X = susan.

**Execution Process:**

1. Write the Prolog code with facts and rules in a file named family.pl.
2. Open a Prolog interpreter like **SWI-Prolog**.
3. Load the program:

?- [family].

1. Enter your queries at the prompt.
2. Prolog will match queries with facts and infer answers using rules.

**Simple Prolog Program: Family Relationships**

% File: family.pl

% Facts

father(john, david).

father(john, mary).

father(david, kevin).

mother(susan, david).

mother(susan, mary).

mother(mary, alice).

% Rules

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

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

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

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

**How to Run in SWI-Prolog:**

1. Save the program as family.pl.
2. Open SWI-Prolog.
3. Load the program:

prolog

CopyEdit

?- [family].

1. Run queries like:

prolog

CopyEdit

?- parent(john, mary).

?- grandfather(john, kevin).

?- parent(X, alice).