AI LAB-1

Objective: Study of Facts, Objects, Predicates and Variables in PROLOG.

Description:

The name PROLOG was taken from the phrase "*programming in logic*." PROLOG is unique in its ability to infer facts and conclusions from other facts. PROLOG is known for its in-built depth-first search engine and for its ease in quick prototyping. Mainly PROLOG is used to build expert systems. Knowledge engineers design and build the system that use PROLOG. The knowledge engineer listens to experts in a particular domain, abstracting the subjective methods used by human mind and defining an objective system of formal reasoning that can be supported by a PROLOG structure.
PROLOG program consists of a collection of knowledge about specific subject. This collection is called a database and database is expressed in facts and rules.

Facts:
A **fact** is just what it appears to be --- a fact.
PROLOG describes facts as symbolic relationships. For example, if the right speaker in your stereo system is not omitting sound, you can express this in English as

The right speaker is dead.

This fact can be expressed in PROLOG as

is(right_speaker,dead).

This factual expression is also called ***clause*** and it must ends with **.**(period)**.**

Objects:
An object is the name of an element of certain type. It represents an entity or a property of an entity in the real world.
In the above example right_speaker and dead both are objects.

- **Types of Objects**:
  There are six types of objects.
  - Char Single char (enclosed between single quotation marks)
  - Integer Integer from -32,768 to32,767
  - Real Floating-point number($1e^{-307}$ to $1e^{308}$)
  - String Character sequence(enclosed between double quotation marks)
  - Symbol Char sequence of letters, numbers and underscores with the first Character a lowercase letter.
  - File Symbolic file name

Predicates:-

A Predicate is a function with a value of true and false. Predicate express a property or relationship.

e.g.

is (right_speaker,dead)

The word before parentheses is the name of the ***relation***. The elements within the parentheses are the **arguments of the predicate**, which **may be objects or variables**.

❖ **Sample Program:**

```
domains
        disease, indication=symbol
predicates
        symptom (disease, indication)
clauses
        symptom(chicken_pox,high_fever).
        symptom(chicken_pox,chills).
        symptom(flu,chills).
        symptom(cold,runny_nose).
        symptom(flu,runny_nose).
```

I/P:

Goal: symptom(cold,runny_nose) ↵

O/P: Turbo PROLOG respond with True and prompt for another goal:

True
Goal:

Code-1

Variable:

A variable must begin with capital letter and may be from 1 to 250 characters long. Except from first character in the name, you may use uppercase or lowercase letters, digits or the underline character.

e.g.

Change the Goal for code1

I/P:

Goal: symptom(Disease,runny_nose)

O/P: Turbo PROLOG will respond

Disease=cold
Disease=flu
2 Solutions
Goal:

- **Types of variables:**
  Bound variable: If the variable has a value at a particular time, it is said to be bound.
  Free variable: If the variable does not have a value at a particular time, it is said to be free. In above example program start with a free variable, Disease.

Anonymous variables:

Sometimes you may wish PROLOG to ignore value of one or two arguments when determining a goal's failure or success. To accomplish this express the argument as an underline.

Change the Goal for code 1.

I/P:

Goal:symptom(_,chills)

O/P:

True

Goal:

Because there is no variable, no binding, if PROLOG can match the relation name and the last argument, the goal succeeds.

You can also use an underscore in a clause to express the fact that the clause is true for all argument values.

likes(jane,_).

Compound Goals:

In Compound goal all the specified condition must succeed for the goal to succeed. Suppose the patient has a high_fever & chills. This could be expressed as this PROLOG goal.

Change the goal for code 1.

I/P: Goal:symptom(Disease, high_fever),symptom(Disease, chills)

O/P: Disease = chicken_pox

1 solution

Goal:

Backtracking:

Backtracking is a process. When a subgoal fails, the PROLOG system traces its steps backwards to the previous goal and tries to *resatisfy* it.

This means that all variables that were bound to a value when that goal was satisfied are now made free again. Then the PROLOG system tries to resatisfy that goal by matching with the *next clause* starting at the clause just after the one that matched the subgoal.

This continues until either the subgoal is satisfied, or until the program database has been exhausted, at which point PROLOG backtracks yet again to the subgoal that was before the current subgoal.

Exercise:

Write a PROLOG program for the following facts.

i. Color of b1 is red
ii. Color of b2 is blue
iii. Color of b3 is yellow
iv. Shape of b1 is square
v. Shape of b2 is circle
vi. Shape of b3 is square
vii. Size of b1 is small
viii. Size of b2 is small
ix. Size of b3 is large

What will be the outcome of each of the following queries?

i. What is the shape of b3?
ii. Which component is having large size and yellow color?

Here are some simple clauses.

```
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
```

The following queries yield the specified answers (Verify it, PLEASE!!!@).

```
| ?- likes(mary,food).
yes.
| ?- likes(john,wine).
yes.
| ?- likes(john,food).
no.
```

How can you answer following questions?

1. John likes anything that Mary likes
2. John likes anyone who likes wine

Here are some simple clauses.

```
has(jack,apples).
has(ann,plums).
has(dan,money).
fruit(apples).
fruit(plums).
```

How can you answer following questions?

1. What Jack has?
2. Does Jack have something?
3. Who has apples and Who has plums?
4. Does someone have apples and plums?
5. Has Dan fruits?

DO YOUR BEST