Artificial Intelligence
Lab sheet No:1

Introduction to Prolog

Prolog is a very important tool in programming artificial intelligence applications and in the development of expert systems. By allowing the programmer to model the logical relationships among objects and processes, complex problems are inherently easier to solve, and the resulting program is easier to maintain through its lifecycle. With Visual Prolog, applications such as customized knowledge bases, expert systems, natural language interfaces, and smart information management systems are easy to develop. Prolog is what is known as a *declarative* language. This means that given the necessary facts and rules, Prolog will use deductive reasoning to solve the programming problems. This is in contrast to traditional computer languages, such as C, BASIC and Pascal, which are procedural languages. We can also use prolog as any other programming languages in a procedural manner.

So prolog can be viewed as a tool to solve problems in the field of artificial intelligence or it can be very well used a general programming language.

Prolog enforces a different problem solving paradigm complementary to traditional programming languages so it is believed that a student of computer should learn programming in prolog.

Data types in prolog

- Atoms and numbers
- Variables
- Structures

Atoms and numbers

Atoms can be constructed in three different ways

1. strings of letters, digits, and the underscore character '_' starting with a lower case letter.
   for example:
   > man, ram, comp_students, pc_ct_059.

2. strings of special characters
   for example:
   > <------->
   > :::::::::::
   Care should be taken not to use the character combination that may have some built in meaning.

3. strings of characters enclosed in quotes
   for example
   > 'Ram'
   > 'Bird'

Numbers used in prolog are integers and real numbers.

Variables

Variables are strings of letters digits and underscore that start with an underscore or an upper-case letter. The scope of a variable is one clause only. So the same variable used in different clauses mean different thing.

For example:

X, Ram, _weight etc.

Note here that Ram is a variable unlike the earlier use 'Ram' where it was a constant, an atom.

An underscore '_' also known as anonymous variable is used in clauses when a variable need not be inferred to more than once.

Structures

Structures are objects that have different components. The components can be atoms or yet some other structures. A functor is used to construct a structure as follows.

family(father, mother, children)

Here family is a structure that has father, mother and the children as its elements. The father and mother may be atoms while the children may be yet another structure or a list of atoms. List is a special built in structure in prolog.

A **list** is a built in structure in prolog. It can be thought of as a sequence of elements ordered linearly however it is internally represented as a binary tree.

For example:

[ram,shyam,hari,sita]

The representation is as follows
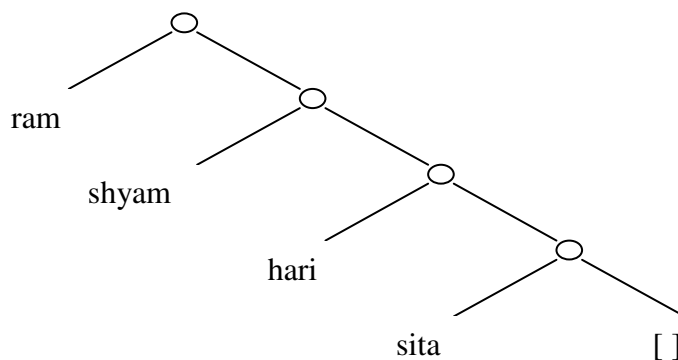
The list as such can be broken down into two parts, the **HEAD** and the **TAIL**. The head is the first element of the list and the tail is the remaining list. The above list can be broken down as

    [H| T]

Where H= ram

and   T= [shyam, hari,sita]

List is one of the most useful structure in prolog.

Writing programs in prologs

All prolog programs start from the goal. It then uses the facts and clauses to break down the goal to sub-goals and tries to prove the subgoals. A clause is said to have succeeded if there is a combination of facts and clause(s) that holds true.

Prolog has built in backtracking mechanism i.e. whenever there are multiple paths, it chooses one tries to prove it, and comes back to the other choices whether the first one succeeds or fails.

Program 1: A typical prolog program

```
PREDICATES
bigger(integer,integer,integer)

CLAUSES
bigger(X,Y,Z):-
    X>Y,Z=X.
bigger(X,Y,Z):-
    X<Y,Z=Y.

GOAL
 bigger(5,7,X).
```

In program 1 we have defined a predicate that gives us the larger of the two given integers. The predicate bigger is defined in the PREDICATES section and the relation is defined in the CLAUSES section. It is customary to put together the predicate definition and clause definition for each predicate but it is not necessary. However all the clauses for a predicate have to be written together.

The predicates section defines what types of relations or facts we are going to use. It is somewhat like declaring functions in programming languages like C. The clauses section holds the relation and facts and can be compared to function definition.

Program 2: A program combining facts and rules**.**

```
PREDICATES
husband(STRING,STRING)
father(STRING,STRING)
mother(STRING,STRING)
son(STRING,STRING)
```

```
CLAUSES
mother("Kaushalya","Ram").
mother("Kaikai","Bharat").
mother("Sumitra","Laxman").
mother("Sumitra","Satrughan").

husband("Dasarath","Kaushalya").
husband("Dasarath","Kaikai").
husband("Dasarath","Sumitra").

son(A,C):-mother(C,A).
son(A,C):-husband(C,B),mother(B,A).

father(A,B):-husband(A,C),mother(C,B).

Goal
son(X,"Kaikai").
```

Here the predicate mother is used to assert facts. There are four facts in that section. Similarly the predicate husband is used to assert more facts. The predicates son and father are used to define rules. We could have defined rules even with mother and husband but we bet that conciseness for program clarity at this stage.
The goal can be checked with
son("Ram",X).
father(X,"Ram").

Program 3: A program to find the length of a list

```
 DOMAINS
int_list=integer*

PREDICATES
length(int_list,integer)


CLAUSES

length([],0).
length([H|T],L):-
    length(T,L1),
    L=L1+1.

GOAL
length([1,2,5,2,1,6,7],X).
```

Program 4 A program to read integers into a list and display them.
```
DOMAINS
list=integer*
```

```
PREDICATES
start
read_a_list(list)
insert(integer,list,list)
display(list)

CLAUSES
start:-
    write("enter the numbers"),
    nl,
    write("enter 0 to stop"),nl,
    read_a_list([]).

read_a_list(Y):-
    readint(X),
    insert(X,Y,Z),
    read_a_list(Z).

insert(0,Y,_):-
    write("these were the elements you inserted"),
    nl,
    write("["),
    display(Y).

insert(X,Y,[X|Y]).

display([ ]):-
    write(" ]"),nl.
display([H|T]):-
    write(H),write("\t"),display(T).

GOAL
start.
```

Assignments

1) Write a program to find the hcf of two numbers.
   Hint: use a predicate like hcf(no1,no2,result).
2) Write a program of your choice. Give some facts and use some rules to make a few deductions.
3) Write a program to add the content of an integer list and display it.
4) Write a program to find the length of a list.
5) Write a program to append two lists.
6) Write a program which takes a list of integers and displays only 1s and 2s. (If the input is [1,2,4,5,2,4,5,1,1] the solution list should be [1,2,2,1,1].)
7) Write a program to delete a given element from the list.