KIST COLLEGE OF MANAGEMENT

(Affiliated to Tribhuvan University)

Project Report

Of

"Artificial Intelligence"

Under Supervision of

Mr. Tej Bahadur Shahi

Submitted By

Anjan Pudasaini (7210/16)

Submitted To

TRIBHUVAN UNIVERSITY

Faculty of management

Kirtipur, Kathmandu, Nepal

March, 2021

Lab sheet : Introduction to Prolog

Introduction

Prolog stands for PROgramming in LOGic –an idea that emerged in the early 1970s to use logic as a programming language. Prolog is centered around a smallest of basic mechanisms, including pattern matching, tree-based data structuring and automatic back tracking.

Logic programming is a type of declarative programming. Different from *procedural programming languages*, such as C/C++ and Java, the focus of declarative programming is to describe a situation (a set of knowledge), not to describe a solution (a sequence of instructions). Especially, logic programming uses a language that is similar to logic, writes a program as a list of facts and rules, and treats the execution of a program as an inference process, from given facts and rules to the desired goal.

Many believe that every student of computer science should learn something about prolog at some point because prolog enforces a different problem solving paradigm complementary to other programming language.

Prolog is a programming language for symbolic, non-numeric computation. It is specially well suited for solving problems that involve objects and relations between objects.

How to Get Prolog

The version of *Prolog* that we will use is called *SWI-Prolog*, developed at the Swedish Institute of Computer Science. There are different dialects of Prologs; SWI Prolog is a most popular dialect which runs on both Linux and Windows operating system. To down load the prolog and supporting documentations follow the follow link.

http://www.swi-prolog.org

SWI-Prolog's home page has lots of information about SWI-Prolog, a download area, and documentation.

Syntax and Meanings: SWI prolog Features and Format

1. Representing Facts and Rules

A Prolog program consists of a sequence of facts and rules i.e. Every thing in *Prolog* is defined in terms of two constructs: the *fact* and the *rule*.

Facts

A fact states that a certain tuple of values satisfies a predicate *unconditionally*. a fact is a predicate name followed by an argument list and a dot sign, ".". It represents a relation among the arguments, or a property of a (single) argument.

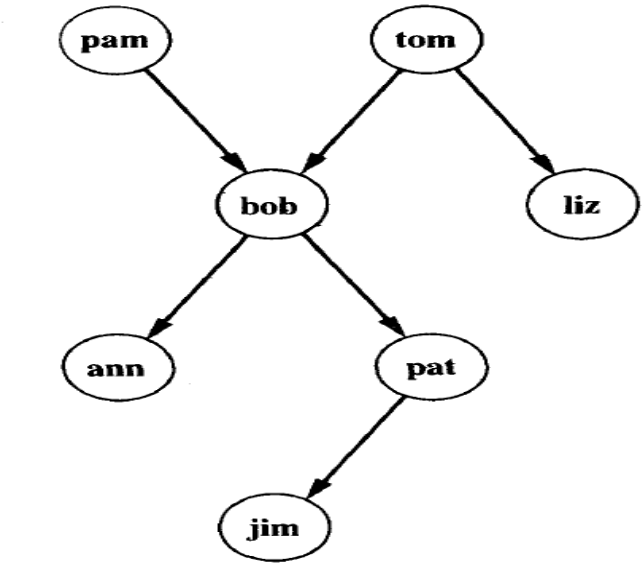
For example, "George is funny" can be represented as a Prolog fact:

funny (george).

"George likes swimming" can be represented as: likes(george, swimming).

Similarly, The fact that Tom is a parent of Bob can be written in Prolog as:

parent(tom, bob). Here we choose parent as the name of a relation; tom and bob are its arguments.



Family Tree

The whole tree can be represented in prolog by:
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

Queries

?- parent(bob, pat).

Prolog will answer:

Yes.

?- parent(liz, pat).

No.

?- parent(X, liz).

X= tom

Data types in Prolog

- Atoms and numbers
- Variables
- Structures

Atoms and numbers

Atoms can be constructed in three different ways strings of letters, digits, and the underscore character '_' starting with a lower case letter.

for example:
man, ram, comp_students, pc_ct_059.
strings of special characters
for example:
<------->
::::::::::::
Care should be taken not to use the character combination that may have some built in meaning.
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 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.

Rules

The syntax for rule is

Head:-body.

A rule can be seen as a "conditional fact". "Susie likes swimming if George likes swimming" is represented as a Prolog rule:

likes(susie, swimming) :- likes(george, swimming).

A rule can have multiple (co-existing) conditions, separated by ",". For example, "Mary likes whatever Susie and George like" is represented as:
likes(mary, X) :- likes(susie, X), likes(george, X).

Note: Prolog symbol meaning

,	and
;	or
:-	only if
not	not

Comment in prolog

Comment line in prolog start with % sign.

Prolog Convention

Prolog uses the convention that a constant is an identifier starting with a lower-case letter, while a variable is an identifier starting with an upper-case letter. A constant stands for a specific entity, and different constants stand for different entities. On the other hand, a variable can stand for any entity, and different variables can stand for the same entity. The predicate name must be a constant, while each argument can either be a constant or a variable.

Every statement in Prologs ends with dot. (Thing to be noticed)

Using the interpreter

To start the *Prolog* interpreter from the command line you enter the command prolog. After a bit of initialization you will see appear the *SWI-Prolog* prompt:

?-

This is the interpreter prompt where you can issue the prolog command and queries.

SWI Prolog comes with inbuilt text editor called emacs. This editor can be opened from the prompt using the command

? – emacs.

After a second, editor opens where you can type prolog source program.

Step in Program Execution in SWI prolog

1.open the prolog interpretor

2. Issue command “emacs.”

3.write the program source code.

4.save the file in prolog directory with the extension .pl

5.consult the file from interpreter prompt.

6. ask the query.

7.to exit the command prompt: press ctrl+D or type halt.

A simple Prolog Program

"Write a program that finds the greatest number between two numbers".

1. Here is the sample code

```
bigger(X,Y,Z):-
    X>Y,Z=X.                % here we have used third variable Z to store the result.
bigger(X,Y,Z):-
    X<Y,Z=Y.
```

2. Save this code into the file named "bigger.pl" in emac editor.

3. Consult the file "bigger.pl" with the command

    ?-consult('bigger.pl'). or

    ?-[bigger]

The extension is optional because prolog assume the .pl extension by default.

4. Pose the query as

    bigger(5,7,X)

Should give the answer as X=7.

Basics commands to use prolog interpreter

To exit from Prolog either input CTRL/D or the query:

?- halt.

To look at the loaded program/database, input

?- listing.

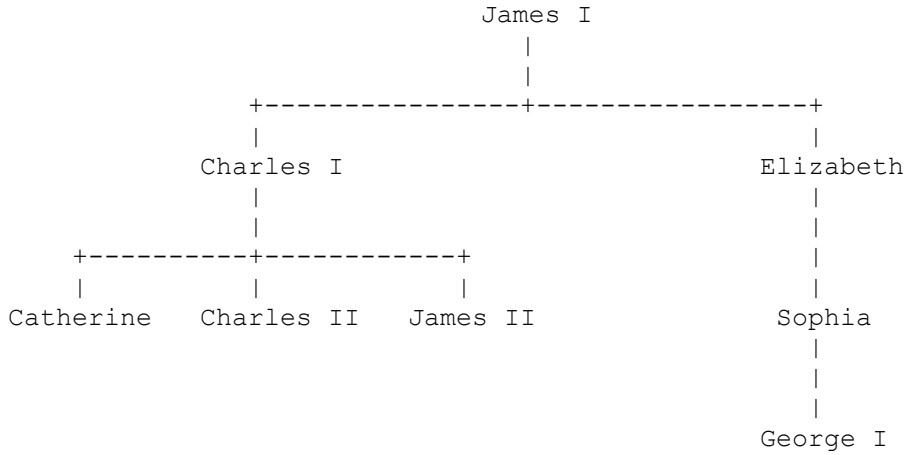
Lab Assignments:

1. Prolog program that illustrates family relation.

Example 1:

```
likes(mary,food).

likes(mary,wine).

likes(john,wine).

likes(john,mary).


The following queries yield the specified answers.


 | ?- likes(mary,food).

 yes.

 | ?- likes(john,wine).

 yes.

 | ?- likes(john,food).

 no.
```

Example 2:



Here are the resultant clauses:

```
male(james1).
male(charles1).
male(charles2).
male(james2).
male(george1).

female(catherine).
female(elizabeth).
female(sophia).

parent(charles1, james1).
parent(elizabeth, james1).
parent(charles2, charles1).
parent(catherine, charles1).
parent(james2, charles1).
parent(sophia, elizabeth).
parent(george1, sophia).
```

Here is how you would formulate the following queries:

Was George I the parent of Charles I?

```
Query: parent(charles1, george1).
```

Who was Charles I's parent?

```
Query: parent(charles1,X).
```

Who were the children of Charles I?

```
Query: parent(X,charles1).
```

Now try expressing the following rules

M is the mother of X if she is a parent of X and is female

F is the father of X if he is a parent of X and is male

X is a sibling of Y if they both have the same parent.

Furthermore add rules defining:

"sister", "brother",

"aunt", "uncle",

"grandparent", "cousin"

2.Prolog program that stores information about your family, and will answer queries about relationships in Artificial Intelligence.

Objective : To stores information about my family, and will answer queries about relationships.

Software and Tools : GNU-Prolog Console, Notepad++.

Code :

```
% Prolog program that store information about my family

% Alamgir, CSE, JUST

% Facts

father(mosiur, sharmin).

father(mosiur, rawshanara).

father(mosiur, alamgir).

father(mosiur, rahim).

father(auncle, liza).

father(auncle, robin).

father(robin, abid).

father(robin, snigdho).

father(sumon, apu).

mother(morium, sharmin).

mother(morium, alamgir).

mother(morium, rahim).
```

```
mother(aunty, liza).

mother(aunty, robin).

mother(sharmin, abid).

mother(sharmin, snigdho).

mother(rawshanara, apu).

% Rules-------

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

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

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

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

mama(X, Y) :- mother(X, Z), father(Z, Y).

huswife(X, Y) :- father(X, Z), mother(Y, Z).

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

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

Input :

```
parent(X, Y).
```

Output :

```
X = mosiur

Y = sharmin ? ;

X = mosiur

Y = rawshanara ? ;

X = mosiur

Y = alamgir ? ;

X = mosiur

Y = rahim ? ;
```

3.Write a Prolog program for addition of two numbers in Artificial Intelligence.

Objective : To find the summation of two numbers.

Software and Tools : GNU-Prolog Console, Notepad++.

Pseudo code :

Read X , Y

Set S to **X**+**Y**

Set M to **X***Y**

Write S,M

Code :

```
% Prolog program for addition and multplication--------
% Alamgir, CSE, JUST
go:- write('Enter first number : '),read(X),nl,
write('Enter second number : '),read(Y),nl,
addmul(X,Y).
addmul(X,Y):-
S is X+Y,
M is X*Y,
write('Addition of the two number is : '),write(S),nl,
write('Multiplication of the two number is : '),write(M).
```

Input:

```
go.
Enter first number : 20.
Enter second number : 10.
```

Output :

```
Addition of the two number is : 30
Multiplication of the two number is : 200
```

4.Write a prolog program to print a Fibonacci series in Artificial Intelligence.

Objective : To calculate the factorial of a number using recursion.

Software and Tools : GNU-Prolog Console, Notepad++.

Pseudo code :

Read N

Repeat function fact and multiple N by N-1

Set result into F

Write F

Code :

```
% Alamgir, CSE, JUST
fact(0,1).
fact(N,F):-
(
N>0 ->
(
N1 is N-1,
fact(N1,F1),
F is N*F1
);
write('N should be greater than 0.')
).
```

Input : fact(8,R).

Output : R = 40320