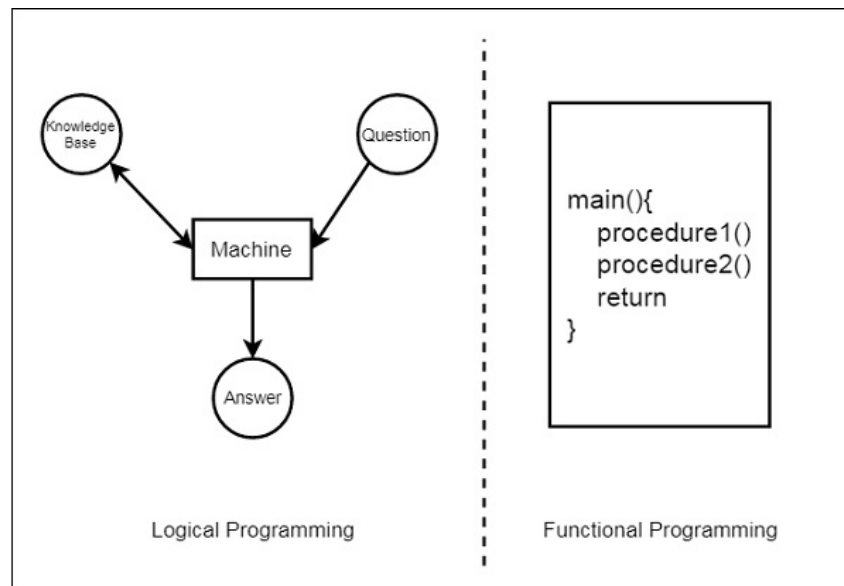


Introduction:

Prolog as the name itself suggests, is the short form of LOGical PROgramming. Logic Programming is one of the Computer Programming Paradigm, in which the program statements express the facts and rules about different problems within a system of formal logic.



From this illustration, we can see that in Functional Programming, we have to define the procedures, and the rules of how the procedures work. These procedures work step by step to solve one specific problem based on the algorithm. On the other hand, for Logic Programming, we will provide a knowledge base. Using this knowledge base, the machine can find answers to the given questions, which is totally different from functional programming.

In functional programming, we have to mention how one problem can be solved, but in logic programming we have to specify for which problem we actually want the solution. Then logic programming automatically finds a suitable solution that will help us solve that specific problem.

What is Prolog?

Prolog or PROgramming in LOGics is a logical and declarative programming language. It is one major example of the fourth generation language that supports the declarative programming paradigm. This is particularly suitable for programs that involve symbolic or non-numeric computation. This is the main reason to use Prolog as the programming language in Artificial Intelligence, where symbol manipulation and inference manipulation are the fundamental tasks.

In Prolog, we need not mention the way how one problem can be solved, we just need to mention what the problem is, so that Prolog automatically solves it. However, in Prolog we are supposed to give clues as the solution method.

Prolog language basically has three different elements –

Facts – The fact is predicate that is true, for example, if we say, “Tom is the son of Jack”, then this is a fact.

Rules – Rules are extensions of facts that contain conditional clauses. To satisfy a rule these conditions should be met. For example, if we define a rule as –

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

This implies that for X to be the grandfather of Y, Z should be a parent of Y and X should be father of Z.

Questions – And to run a prolog program, we need some questions, and those questions can be answered by the given facts and rule

Official Website

This is the official GNU Prolog website where we can see all the necessary details about GNU Prolog, and also get the download link.

<http://www.gprolog.org/>

Direct Download Link

Given below are the direct download links of GNU Prolog for Windows.

<http://www.gprolog.org/#download>

First Program:

After running the GNU prolog, we can write hello world program directly from the console. To do so, we have to write the command as follows –

Input	Output
<pre>write('Hello World').</pre>	<pre>Copyright (C) 1999-2018 Daniel Di ?- write('Hello World'). Hello World yes ?- </pre>

Note – After each line, you have to use one period (.) symbol to show that the line has ended.

Now let us see how to run the Prolog script file (extension is *.pl) into the Prolog console.

Before running *.pl file, we must store the file into the directory where the GNU prolog console is pointing, otherwise just change the directory by the following steps –

Step 1 – From the prolog console, go to File > Change Dir, then click on that menu.

Step 2 – Select the proper folder and press OK.

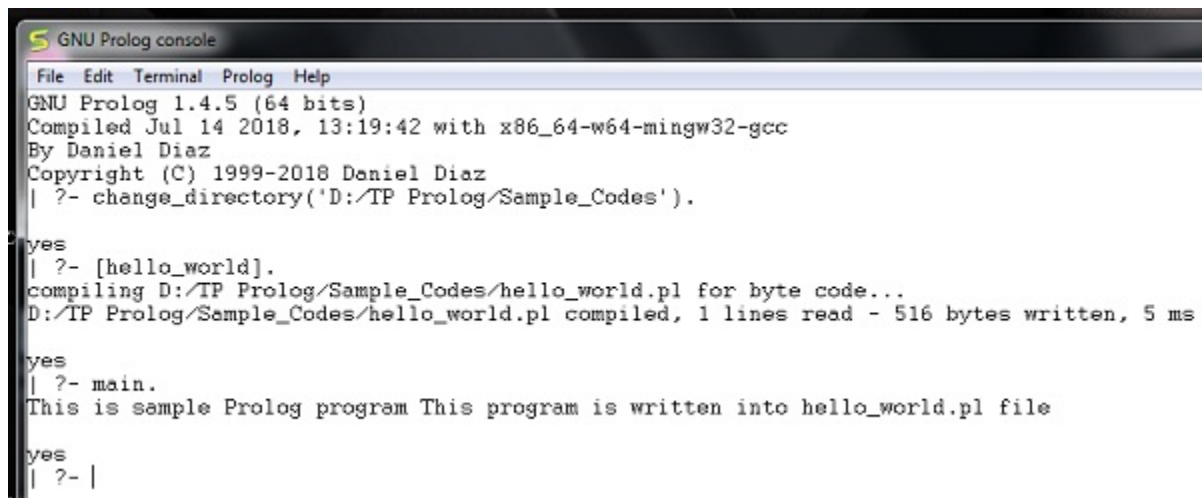
Step 3 – Now create one file (extension is *.pl) and write the code as follows –

```
main :- write('This is sample Prolog program'),  
write(' This program is written into hello_world.pl file').
```

Now let's run the code. To run it, we have to write the file name as follows –

```
[hello_world].
```

The output is as follows –

A screenshot of the GNU Prolog console window. The window has a title bar 'GNU Prolog console' and a menu bar with 'File', 'Edit', 'Terminal', 'Prolog', and 'Help'. The console output shows the version 'GNU Prolog 1.4.5 (64 bits)', compilation details, and the user's commands. The user enters '?- change_directory('D:/TP Prolog/Sample_Codes')', which is successful. Then they enter '?- [hello_world].', which triggers the compilation of 'D:/TP Prolog/Sample_Codes/hello_world.pl'. Finally, they enter '?- main.', which outputs 'This is sample Prolog program This program is written into hello_world.pl file'.

```
GNU Prolog console  
File Edit Terminal Prolog Help  
GNU Prolog 1.4.5 (64 bits)  
Compiled Jul 14 2018, 13:19:42 with x86_64-w64-mingw32-gcc  
By Daniel Diaz  
Copyright (C) 1999-2018 Daniel Diaz  
| ?- change_directory('D:/TP Prolog/Sample_Codes').  
yes  
| ?- [hello_world].  
compiling D:/TP Prolog/Sample_Codes/hello_world.pl for byte code...  
D:/TP Prolog/Sample_Codes/hello_world.pl compiled, 1 lines read - 516 bytes written, 5 ms  
yes  
| ?- main.  
This is sample Prolog program This program is written into hello_world.pl file  
yes  
| ?- |
```

Prolog - Basics

Knowledge Base – This is one of the fundamental parts of Logic Programming. We will see in detail about the Knowledge Base, and how it helps in logic programming.

Facts, Rules and Queries – These are the building blocks of logic programming. We will get some detailed knowledge about facts and rules, and also see some kind of queries that will be used in logic programming.

Here, we will discuss the essential building blocks of logic programming. These building blocks are Facts, Rules and the Queries.

Facts

We can define fact as an explicit relationship between objects, and properties these objects might have. So facts are unconditionally true in nature. Suppose we have some facts as given below –

- Tom is a cat
- Kunal loves to eat Pasta
- Hair is black
- Nawaz loves to play games
- Pratyusha is lazy.

So these are some facts that are unconditionally true. These are actually statements that we have to consider as true.

Following are some guidelines to write facts –
Names of properties/relationships begin with lowercase letters.

The relationship name appears as the first term.

Objects appear as comma-separated arguments within parentheses.

A period "." must end a fact.

Objects also begin with lowercase letters. They also can begin with digits (like 1234), and can be strings of characters enclosed in quotes e.g. color(penink, 'red').

phoneno(agnibha, 1122334455). is also called a predicate or clause.

Syntax

The syntax for facts is as follows –

```
relation(object1,object2...).
```

Example

```
cat(tom).
loves_to_eat(kunal,pasta).
of_color(hair,black).
loves_to_play_games(nawaz).
lazy(pratyusha).
```

Rules

We can define rule as an implicit relationship between objects. So facts are conditionally true. So when one associated condition is true, then the predicate is also true. Suppose we have some rules as given below

- Lili is happy if she dances.
- Tom is hungry if he is searching for food.
- Jack and Bili are friends if both of them love to play cricket.
- will go to play if school is closed, and he is free.

So these are some rules that are conditionally true, so when the right hand side is true, then the left hand side is also true.

Here the symbol (:-) will be pronounced as “If”, or “is implied by”. This is also known as neck symbol, the LHS of this symbol is called the Head, and right hand side is called Body. Here we can use comma (,) which is known as conjunction, and we can also use semicolon, that is known as disjunction.

Syntax

```
rule_name(object1, object2, ...) :- fact/rule(object1,
object2, ...)
```

Suppose a clause is like :

P :- Q;R.

This can also be written as

P :- Q.

P :- R.

If one clause is like :

P :- Q,R;S,T,U.

Is understood as

$P :- (Q,R);(S,T,U).$

Or can also be written as:

$P :- Q,R.$

$P :- S,T,U.$

Example

happy(lili) :- dances(lili).

hungry(tom) :- search_for_food(tom).

friends(jack, bili) :- lovesCricket(jack), lovesCricket(bili).

goToPlay(ryan) :- isClosed(school), free(ryan).

Queries

Queries are some questions on the relationships between objects and object properties. So question can be anything, as given below –

- Is tom a cat?
- Does Kunal love to eat pasta?
- Is Lili happy?
- Will Ryan go to play?

So according to these queries, Logic programming language can find the answer and return them.

Prolog - Relations

Relationship is one of the main features that we have to properly mention in Prolog. These relationships can be expressed as facts and rules. After that we will see about the family relationships, how we can express family based relationships in Prolog, and also see the recursive relationships of the family.

We will create the knowledge base by creating facts and rules, and play query on them.

Relations in Prolog

In Prolog programs, it specifies relationship between objects and properties of the objects.

Suppose, there's a statement, "Amit has a bike", then we are actually declaring the ownership relationship between two objects – one is Amit and the other is bike.

If we ask a question, “Does Amit own a bike?”, we are actually trying to find out about one relationship.

There are various kinds of relationships, of which some can be rules as well. A rule can find out about a relationship even if the relationship is not defined explicitly as a fact.

We can define a brother relationship as follows –

Two person are brothers, if,

- They both are male.
- They have the same parent.

Now consider we have the below phrases –

```
parent(sudip, piyus).
```

```
parent(sudip, raj).
```

```
male(piyus).
```

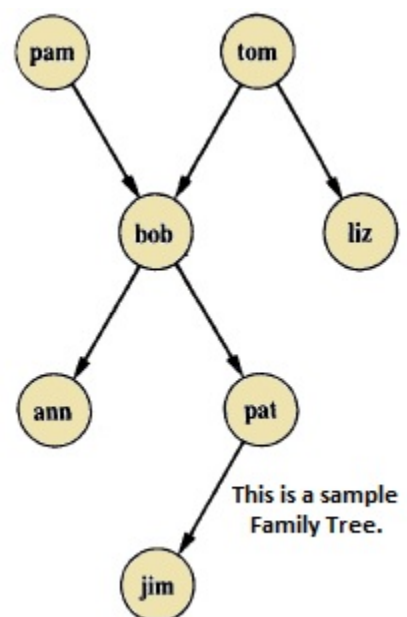
```
male(raj).
```

```
brother(X,Y) :- parent(Z,X), parent(Z,Y),male(X), male(Y)
```

Family Relationship in Prolog

Here we will see the family relationship. This is an example of complex relationship that can be formed using Prolog. We want to make a family tree, and that will be mapped into facts and rules, then we can run some queries on them.

Suppose the family tree is as follows –



Here from this tree, we can understand that there are few relationships. Here bob is a child of pam and tom, and bob also has two children – ann and pat. Bob has one brother liz, whose parent is also tom. So we want to make predicates as follows –

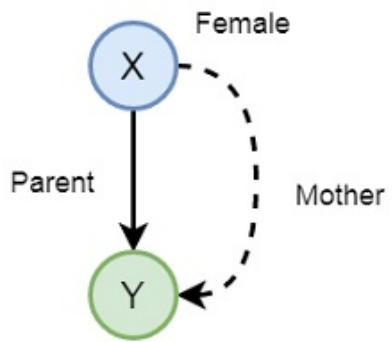
Predicates

```
parent(pam, bob).  
parent(tom, bob).  
parent(tom, liz).  
parent(bob, ann).  
parent(bob, pat).  
parent(pat, jim).  
parent(bob, peter).  
parent(peter, jim).
```

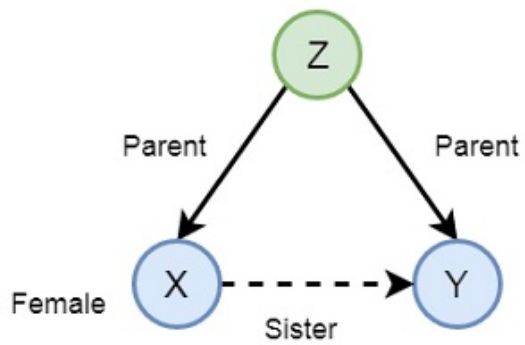
Some facts can be written in two different ways, like sex of family members can be written in either of the forms –

```
female(pam).  
male(tom).  
male(bob).  
female(liz).  
female(pat).  
female(ann).  
male(jim).
```

Now if we want to make mother and sister relationship, then we can write as given below –



Mother Relationship



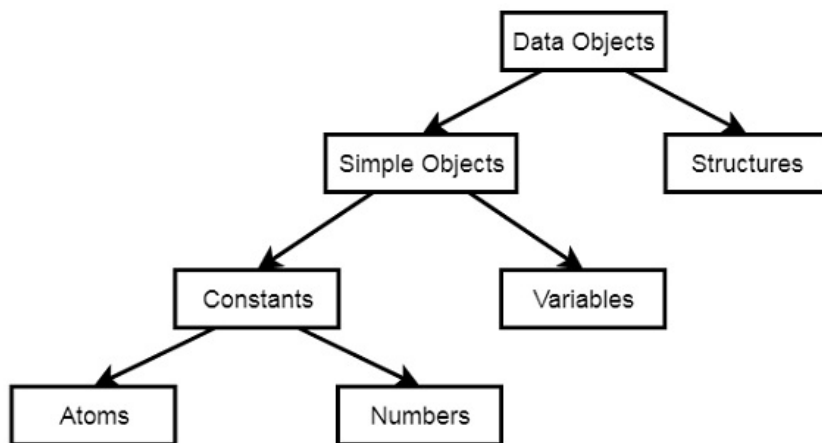
Sister Relationship

In Prolog syntax, we can write -

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

```
sister(X,Y) :- parent(Z,X), parent(Z,Y), female(X), X \== Y.
```

Prolog - Data Objects



Data objects in Prolog

Below are some examples of different kinds of data objects -

Atoms – tom, pat, x100, x_45

Numbers – 100, 1235, 2000.45

Variables – X, Y, Xval, _X

Structures – day(9, jun, 2017), point(10, 25)

Prolog - Operators

Comparison Operators

Comparison operators are used to compare two equations or states. Following are different comparison operators –

Operator	Meaning
$X > Y$	X is greater than Y
$X < Y$	X is less than Y
$X \geq Y$	X is greater than or equal to Y
$X \leq Y$	X is less than or equal to Y
$X == Y$	the X and Y values are equal
$X \neq Y$	the X and Y values are not equal

Example

```
| ?- 1+2==2+1.
```

yes

```
| ?- 1+2=2+1.
```

no

```
| ?- 1+A=B+2.
```

A = 2

B = 1

yes

```
| ?- 5<10.
```

yes

```
| ?- 5>10.
```

no

```
| ?- 10\=100.
```

yes

Arithmetic Operators in Prolog

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
//	Integer Division
mod	Modulus

Program

```

calc :- X is 100 + 200,write('100 + 200 is '),write(X),nl,
        Y is 400 - 150,write('400 - 150 is '),write(Y),nl,
        Z is 10 * 300,write('10 * 300 is '),write(Z),nl,
        A is 100 / 30,write('100 / 30 is '),write(A),nl,
        B is 100 // 30,write('100 // 30 is '),write(B),nl,
        C is 100 ** 2,write('100 ** 2 is '),write(C),nl,
        D is 100 mod 30,write('100 mod 30 is '),write(D),nl.

```

```

| ?- [op_arith].
compiling D:/TP Prolog/Sample_Codes/op_arith.pl for byte code...
D:/TP Prolog/Sample_Codes/op_arith.pl compiled, 6 lines read - 2390 byte

yes
| ?- calc.
100 + 200 is 300
400 - 150 is 250
10 * 300 is 3000
100 / 30 is 3.3333333333333335
100 // 30 is 3
100 ** 2 is 10000.0
100 mod 30 is 10

yes
| ?-

```

