Introduction to Prolog

What is Prolog

Prolog stands for programming in logic. In the logic programming paradigm, prolog language is most widely available. Prolog is a declarative language, which means that a program consists of data based on the facts and rules (Logical relationship) rather than computing how to find a solution. A logical relationship describes the relationships which hold for the given application. To obtain the solution, the user asks a question rather than running a program. When a user asks a question, then to determine the answer, the run time system searches through the database of facts and rules.

The first Prolog was '**Marseille Prolog**', which is based on work by Colmerauer. The major example of fourth-generation programming language was prolog. It supports the declarative programming paradigm.

In 1981, a Japanese computer Project of 5th generation was announced. After that, it was adopted Prolog as a development language. In this tutorial, the program was written in the 'Standard' Edinburgh Prolog. Prologs of PrologII family are the other kind of prologs which are descendants of Marseille Prolog.

Prolog features are 'Logical variable', which means that they behave like uniform data structure, a backtracking strategy to search for proofs, a pattern-matching facility, mathematical variable, and input and out are interchangeable.

To deduce the answer, there will be more than one way. In such case, the run time system will be asked to find another solution. To generate another solution, use the backtracking strategy. Prolog is a weakly typed language with static scope rules and dynamic type checking.

Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it.

Prolog is used in some areas like database, natural language processing, artificial intelligence, but it is useless in some areas like a numerical algorithm or instance graphics.

In artificial intelligence applications, prolog is used. The artificial intelligence applications can be automated reasoning systems, natural language interfaces, and expert systems. The expert system consists of an interface engine and a database of facts. The prolog's run time system provides the service of an interface engine.

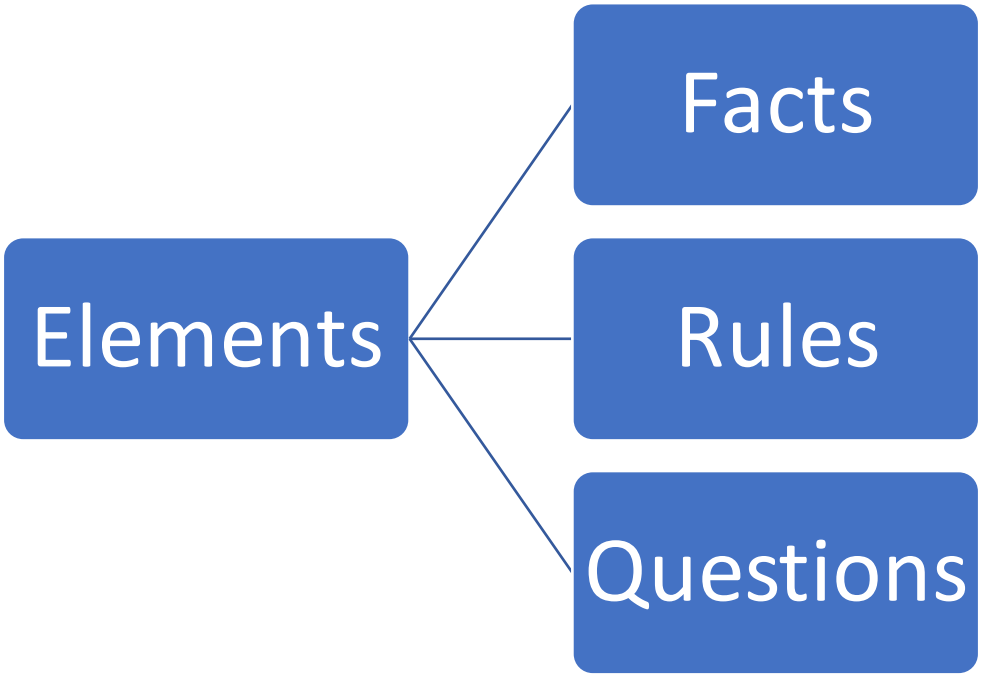
A basic logic programming environment has no literal values. An identifier with upper case letters and other identifiers denote variables. Identifiers that start with lower-case letters denote data values. The basic Prolog elements are typeless. The most implementations of prolog have been enhanced to include integer value, characters, and operations. The Mechanism of prolog describes the tuples and lists.

Functional programming language and prolog have some similarities like Hugs. A logic program is used to consist of relation definition. A functional programming language is used to consist of a sequence of function definitions. Both the logical programming and functional programming rely heavily on recursive definitions.

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 extinctions 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 rules.

Applications of Prolog

Prolog is used in various domains. It plays a vital role in automation system. Following are some other important fields where Prolog is used −

- Intelligent Database Retrieval
- Natural Language Understanding
- Specification Language
- Machine Learning
- Robot Planning
- Automation System
- Problem Solving

Getting Started

SWI-Prolog is installed as 'swipl'. SWI-Prolog is normally operated as an interactive application simply by starting the program:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
```

- **Defining rules**

In SWI Prolog we can define rules and facts according to our convenience of whatever we want.

For example:

```
% Fathers Side
male(varun).
male(karan).
male(yogesh).
male(mahendra).
male(prashant).
male(sanjay).
male(yash).
male(arvind).

% Fathers Side
female(amita).
female(jyoti).
female(deepal).
female(taruni).
female(archana).
female(arpita).
female(hemlata).

% Fathers Side
parent_of(mahendra,varun).
parent_of(amita,varun).
parent_of(jyoti,karan).
parent_of(jyoti,deepal).
parent_of(yogesh,karan).
parent_of(yogesh,deepal).
parent_of(prashant,yash).
parent_of(prashant,arpita).
parent_of(taruni,yash).
parent_of(taruni,arpita).
parent_of(arvind,mahendra).
parent_of(arvind,yogesh).
parent_of(arvind,prashant).
parent_of(arvind,sanjay).
parent_of(hemlata,yogesh).
parent_of(hemlata,mahendra).
parent_of(hemlata,sanjay).
parent_of(hemlata,prashant).

% Fathers Side
married(yogesh,jyoti).
married(mahendra,amita).
married(prashant,taruni).
married(sanjay,archana).
married(arvind,hemlata).
```

```
%self roles rule
father(X, Y) :- male(X), parent_of(X, Y).
mother(X,Y) :- female(X), parent_of(X,Y).

brother(X, Y) :- male(X), father(Z, Y), father(Z, X), X \= Y.
brother(X, Y) :- male(X), mother(Z, Y), mother(Z, X), X \= Y.

sister(X, Y) :- female(X), father(Z, Y), father(Z, X), X \= Y.
sister(X, Y) :- female(X), mother(Z, Y), mother(Z, X), X \= Y.

uncle_of(X,Y) :- parent_of(Z,Y), brother(Z,X).

grandparent(X, Y) :- parent_of(X, Z), parent_of(Z, Y).

grandmother(X, Y) :- parent_of(X, Z), parent_of(Z, Y),female(X).
grandfather(X, Y) :- parent_of(X, Z), parent_of(Z, Y),male(X).

aunt_of(X,Y) :- female(X), father(Z,Y), brother(Z,W), married(W,X).
```

And to verify any code using the codes then:

```
?- parent_of(X,varun).
X = mahendra ;
X = amita.

?- uncle_of(X,deepal).
X = mahendra ;
X = prashant ;
X = sanjay ;
X = mahendra ;
X = sanjay ;
X = prashant.

?- grandparent(X,varun).
X = arvind ;
X = hemlata .

?- aunt_of(X,karan).
X = amita ;
X = amita ;
X = taruni ;
X = taruni ;
X = archana ;
X = archana ;
false.

?-
```