Introduction to ProLog

“PROgramming in LOGic"

Prolog is a language in which statements are logical expressions.

GNU project's gprolog is an open source implementation of Prolog.
Each implementation of Prolog usually offers features and even syntax that differ from other implementations,

Coding in gprolog:
Technically, you don't create a Prolog program – you create a Prolog database.
The first step is to create the database using any **ascii text editor**.
Traditionally, the database file has a **.pl** file extension,

Prolog Program Components:

- Constants are in lower case
- Variables are in upper case
- Entries always end with a period
- Blank lines are OK
- Comments in Prolog start with a percent (%) sign and stop at the end of the line

In the database are two types of clauses: **Facts** and **Rules**.

FACT

- As the name suggests, a **fact** is a single piece of information.

- Examples,
  blue(sky).
  mammal(rabbit).

- Facts can have multiple arguments. These are called relations.

- Example,
  plays(john,hockey). %this is read as “john plays hockey”

RULES

- Rules are used to generate new information from facts, other rules, and even themselves.

- Common operators in rules:
  logical and (**,**)
  logical or (**;**)
  logical not (**\+**)
  left-side implication (←) (**:-**)
  equality comparison (**=**)
  not equal to (**\=**)

- Rule form
  head :- body.

  Examples,
  grandparent(X,Z) :- parent(X,Y) , parent(Y,Z).
  ancestor(X,Y) :- parent(Z,Y), ancestor(X,Z).

  Note: The parent() clauses on either side are called subgoals.

  The example is the same in this expression in logic
  $\forall x \, \forall y \, \forall z \, ((P(x,y) \wedge P(y,z)) \rightarrow G(x,z))$

Assume the following database (family.pl):
parent(amy,bob).
parent(bob,cathy).
parent(bob,doug).
grandparent(X,Z) :- parent(X,Y) , parent(Y,Z).
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(Z,Y) , ancestor(X,Z).

Providing multiple definitions for a rule is equivalent to OR operation.

The same thing could be expressed concisely using a semicolon (;) for OR:
ancestor(X,Y) :- parent(X,Y); parent(Z,Y) , ancestor(X,Z).

Starting gprolog:
At the shell prompt, type: gprolog and press Enter.

```
GNU Prolog 1.2.16
By Daniel Diaz
Copyright (C) 1999-2002 Daniel Diaz
| ?-
```

The last line is the Prolog prompt – it's ready for you to type a command.

To load your family database into Prolog, use this command at that prompt:
[family].

If Prolog finds that everything is in order, it will say something like this:

```
compiling /***/***/family.pl for byte code...
/***/***/family.pl compiled, 14 lines read - 1204 bytes written, 109 ms

yes
| ?-
```

queries:
| ?- parent(hank,denise).
yes
| ?- parent(denise,hank).
no
| ?- grandparent(irene,frank).
true ? **;**
no
| ?- parent(hank,X).
X = ben ? **;**
X = denise
yes
| ?- **grandparent(hank,X).**
X = carl ? **;**
X = frank ? **;**
X = gary
yes
| ?-

When Prolog prints a response and follows it with a **question** mark, it is asking if you want it to keep searching for more answers. Typically, you do. If so, just press the semicolon, which tells Prolog to keep going.

ANOTHER EXAMPLE (family2.pl):
male(ali).
male(veli).
female(zeynep).
parent(ali,ayse).
parent(ali,ahmet).
parent(zeynep,ayse).
father(X,Y):-parent(X,Y), male(X).
somebodysparent(X):-father(X,Y); mother(X,Y).
hasnochild(X):- \+ parent(X,Y).
brother(X, Y) :- male(X), father(F,X), father(F,Y), mother(M,X), mother(M,Y), \+(X=Y).

In cases like the above, we do not care about the value of Y.
You can use the underscore (_) as an anonymous variable
somebodysparent(X):-father(X,_); mother(X,_).
hasnochild(X):- \+ parent(X,_).

Queries:
male(ali).
yes
| ?-

male(X).
X = ali ?

father(X,Y). → to find all pairs which has a fatherhood relation
X = ali,
Y = ayse;

X = ali,
Y = ahmet.