

Lab#1

Object: Introduction to Prolog and Sample Program

There are two types of programming languages: Procedural and Declarative. Procedural languages are used to perform numerical computations like: Computing area of a circle, finding percentage marks of a student, finding the person with greater age. All these problems involve some sort of numerical computations (formulae). Some of the procedural programming languages are: C/C++, JAVA, C#, BASIC FORTRAN etc. Whereas declarative languages are used to perform non-numerical computations like: who is the mother of John? Finding all the students who like AI? Are Shahid and Saeed brothers? Now all these problems involve non-numeric computations there is no formula to find out who is the mother of John?; you do not have any formula to conclude which students like AI; you do not have formula to tell whether Shahid and Saeed are brothers of each other. These types of problems cannot be solved by traditional procedural languages. Some of the declarative programming languages are: PROLOG (Programming in Logic), LISP (List Processing) and ML (Meta Language).

Prolog:

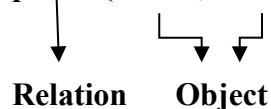
PROLOG stands for **P**rogramming in **L**ogic. Prolog is a high-level declarative programming language for symbolic and non-numeric computation. Prolog is a logic language that is particularly suited to programs that involve symbolic or non-numeric computation. For this reason it is a frequently used language in Artificial Intelligence where manipulation of symbols and inference about them is a common task.

Prolog consists of a series of rules and facts. A program is run by presenting some query and seeing if this can be proved against these known rules and facts. In this tutorial we will attempt to give you a flavor of how all this is achieved and teach you how to write a simple program for yourself.

Simple Facts

PROLOG is specially well suited for solving problems that involve objects and relations between objects. Figure 6.1 shows an example: a family relation. The fact that Tomi's a parent of Bob can be written in Prolog as:

parent("tom", "bob"). Read as: Tom is parent of bob



Here we choose parent as the name of a relation; tom and bob are its arguments. For reasons that will become clear later we write names like tom with an initial lower-case letter. IN Prolog we can make some statements by using facts. Facts either consist of a particular item or a relation between items. For example we can represent the fact that it is sunny by writing the program:

sunny.

We can now ask a query of Prolog by asking

?- sunny.

?- is the Prolog prompt. To this query, Prolog will answer yes. **sunny** is true because (from above) Prolog matches it in its database of facts.

Facts have some simple rules of syntax. Facts should always begin with a lowercase letter and end with a full stop. The facts themselves can consist of any letter or number combination, as well as the underscore _ character. However, names containing the characters -, +, *, /, or other mathematical operators should be avoided.

Basic elements of PROLOG (Facts, Rules and Questions)

Prolog programs are composed of declarative sentences to be asserted. These sentences are called predicates. These predicates are of two types: Facts and Rules.

1.1 Facts

Facts are the declarative sentences which are always true. Facts can represent either relationships between objects or properties of objects. For example:

Facts (Relationship)

Arslan is father of Saima. **father(arslan, saima).**

Ali is friend of Farhan. **friend(ali, farhan).**

Farhan can drive car. **drive(farhan, car).**

Ali likes icecream. **likes(ali, icecream).**

Facts (Properties)

Saima is woman. **woman(saima).**

Farhan is man. **man(farhan).**

Farhan can walk. **walk(farhan).**

Ali is happy. **happy(ali).**

General Syntax of Fact: predicate(arg1, arg2, arg3, . . . , argN).

Examples (I) of Simple Facts:

Here are some simple facts about an imaginary world. /* and */ are comment delimiters

- john_is_little. /* john is little */
- raining. /* it is raining */
- john_Forgot_His_Raincoat. /* john forgot his raincoat */
- fred_lost_his_car_keys. /* fred lost is car keys */

1.2 Rules

Will discuss in lab#02.

1.3 Questions

After writing facts and rules we will ask some questions. Those questions will be evaluated from specified facts and rules. For example:

?- **father(shahid, saima)**. Is Shahid father of Saima?

?- **play(ali, cricket)**. Does Ali play cricket?

?- **friend(ali, farhan)**. Is Ali friend of Farhan?

?- **happy(saima)**. Is Saima happy?

Exercise

1. Covert following sentences in to PROLOG FACTS and specify whether they represent RELATIONSHIP or PROPERTY:

<i>Declarative Sentence</i>	<i>PROLOG FACT</i>	<i>RELATIONSHIP or PROPERTY</i>
<i>AIMS is university</i>		
<i>Faheem likes icecream</i>		
<i>Birds can fly</i>		
<i>Switch is off</i>		
<i>Sara is enemy of saima</i>		

2. Clauses:

We create a file named task.pl.

```

has (jack, apples) .
has (ann, plums) .
has (dan, money) .
fruit (apples) .
fruit (plums) .

```

```

?- [task].                /* loads the file */
?- listing(fruit).        /* lists the clauses */
?- listing(has).
?- has(jack,X).           /* what has Jack? */
?- has(jack,_).           /* does Jack have something? */
?- has(X,apples),has(Y,plums). /* who has apples and who has plums? */
?- has(X,apples),has(X,plums). /* does someone have apples and plums? */
*/
?- has(dan,X),fruit(X).   /* has Dan fruits? */

```

Task: Execute Above Code

The screenshot displays the SWI-Prolog IDE interface. The top window, titled 'youza.pl (2017-CS-208)', shows the Prolog code being executed. The bottom window, titled 'SWI-Prolog (AMD64, Multi-threaded, version 8.0.2) (2017-CS-208)', shows the execution results. The code defines facts for 'has' and 'fruit' predicates. The execution results show that Jack has apples, Ann has plums, and Dan has money. The query 'has(jack,X)' returns 'X = apples'. The query 'has(jack,_)' returns 'true'. The query 'has(X,apples),has(Y,plums)' returns 'X = jack, Y = ann'. The query 'has(X,apples),has(X,plums)' returns 'false'. The query 'has(dan,X),fruit(X)' returns 'false'.

```

youza.pl (2017-CS-208)
File Edit Browse Compile Prolog Pce Help
youza.pl
has(jack,apples).
has(ann,plums).
has(dan,money).
fruit(apples).
fruit(plums).

SWI-Prolog (AMD64, Multi-threaded, version 8.0.2) (2017-CS-208)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- [youza].
true.

?- listing(fruit).
fruit(apples).
fruit(plums).

true.

?- listing(has).
has(jack, apples).
has(ann, plums).
has(dan, money).

true.

?- has(jack,X).
X = apples.

?- has(jack,_).
true.

?- has(X,apples),has(Y,plums).
X = jack,
Y = ann.

?- has(X,apples),has(X,plums).
false.

?- has(dan,X),fruit(X).
false.

```

3. Make prolog facts from following sentences and answer the questions by typing queries on prolog.

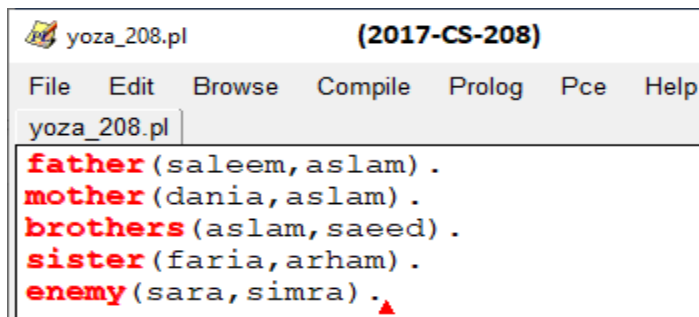
Saleem is father of aslam

Dania is mother of aslam

Aslam and saeed are brothers

Faria is sister of arham

Sara is enemy of simra



```
yoza_208.pl (2017-CS-208)
File Edit Browse Compile Prolog Pce Help
yoza_208.pl
father(saleem, aslam) .
mother(dania, aslam) .
brothers(aslam, saeed) .
sister(faria, arham) .
enemy(sara, simra) .
```

Questions

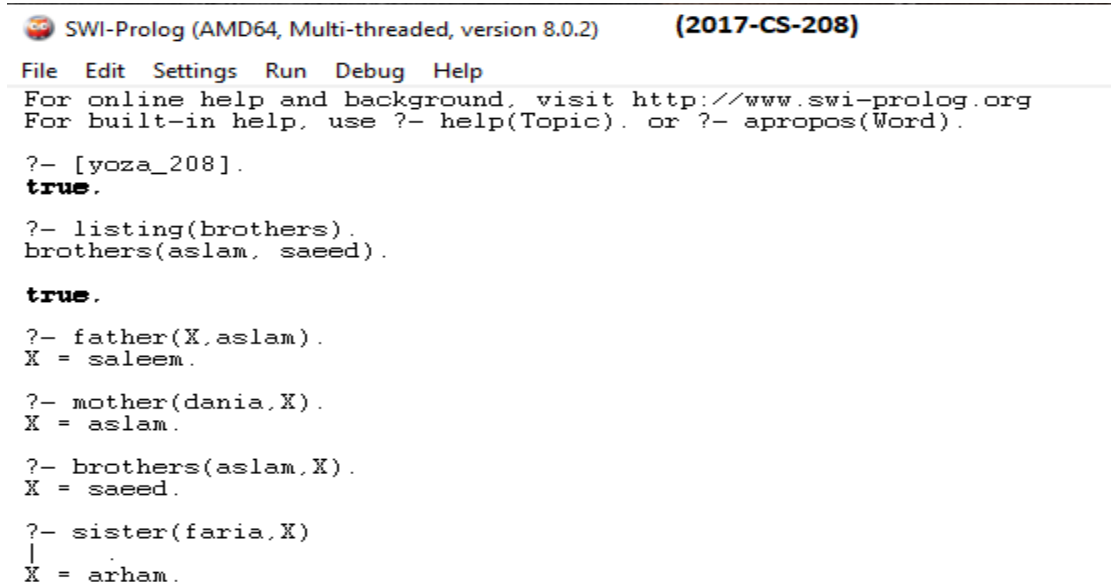
List brother.

Who is father of aslam?

Dania is mother of whom?

Who is brother of aslam?

Faria is sister of whom?



```
SWI-Prolog (AMD64, Multi-threaded, version 8.0.2) (2017-CS-208)
File Edit Settings Run Debug Help
For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [yoza_208].
true.

?- listing(brothers).
brothers(aslam, saeed).

true.

?- father(X, aslam).
X = saleem.

?- mother(dania, X).
X = aslam.

?- brothers(aslam, X).
X = saeed.

?- sister(faria, X)
|
X = arham.
```