76_Adnan Shaikh

Experiment no. 7

Aim: To implement basic code in Prolog.

Requirements: Compatible version of SWI-Prolog.

Theory:

Prolog as the name itself suggests, is the short form of LOGical PROgramming. It is a logical and declarative programming language. Before diving deep into the concepts of Prolog, let us first understand what exactly logical programming is.

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. Here, the rules are written in the form of logical clauses, where head and body are present. For example, H is head and B1, B2, B3 are the elements of the body. Now if we state that “H is true, when B1, B2, B3 all are true”, this is a rule. On the other hand, facts are like the rules, but without any body. So, an example of fact is “H is true”.

Some logic programming languages like Datalog or ASP (Answer Set Programming) are known as purely declarative languages. These languages allow statements about what the program should accomplish. There is no such step-by-step instruction on how to perform the task. However, other languages like Prolog, have declarative and also imperative properties. This may also include procedural statements like “To solve the problem H, perform B1, B2 and B3”.

Prolog or **PRO**gramming in **LOG**ics 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.

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

• phoneno(agnibha, 1122334455). Is also called a predicate or clause. **Syntax**

The syntax for facts is as follows −

relation (object1,object2...).

Example

Following is an example of the above concept −

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?

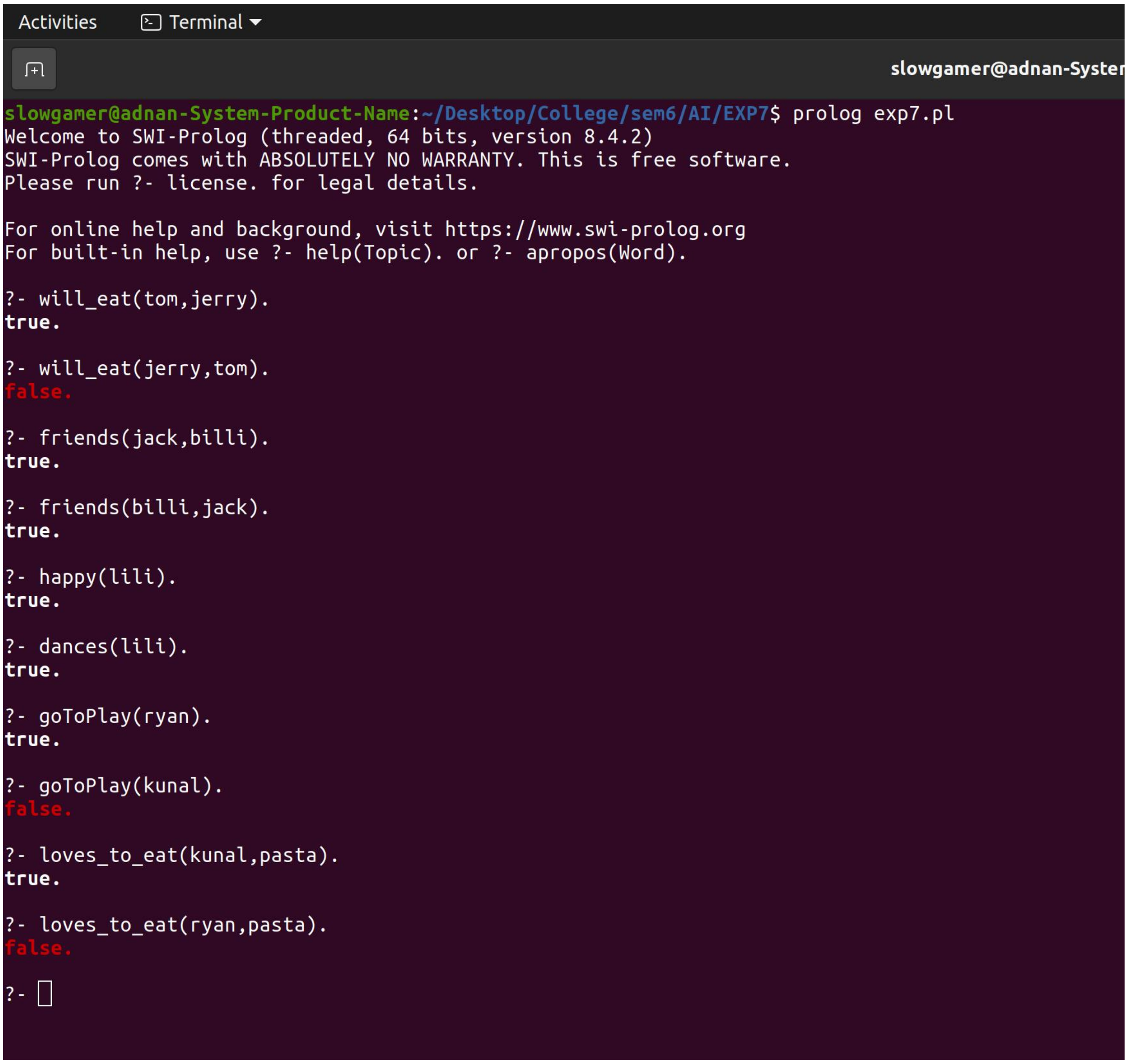
• Will Ryan go to play?

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

Code:

```
mouse(jerry).
cat(tom).
search_for_food(tom).
hungry(X):- search_for_food(X).
will_eat(X,Y):- cat(X),hungry(X),mouse(Y).

play(jack,cricket).
play(billi,cricket).
friends(X,Y):- play(X,Z),play(Y,Z).

eat(kunal,pasta).
loves_to_eat(X,Y):- eat(X,Y).

dances(lili).
happy(X):- dances(X).

is_Closed(school).
free(ryan).
goToPlay(X):- is_Closed(school),free(X).
```

<u>Queries</u>:



```
slowgamer@adnan-System-Product-Name:~/Desktop/College/sem6/AI/EXP7$ prolog exp7.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
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).

?- will_eat(tom,jerry).
true.

?- will_eat(jerry,tom).
false.

?- friends(jack,billi).
true.

?- friends(billi,jack).
true.

?- happy(lili).
true.

?- dances(lili).
true.

?- goToPlay(ryan).
true.

?- goToPlay(kunal).
false.

?- loves_to_eat(kunal,pasta).
true.

?- loves_to_eat(ryan,pasta).
false.

?- 
```

<u>Conclusion</u>: We have successfully implemented basic code of Prolog.