

Experiment No. 6

1. Aim: Implementation of PROLOG Programs

2. Objectives: From this experiment, the student will be able to

- Understand logical programming syntax and semantics
- Design programs in PROLOG language

3. Outcomes: The learner will be able to

- Ability to design models for reasoning with uncertainty as well as the use of unreliable information.

4. Software Required : SWI-Prolog

5. Theory:

Prolog is a general purpose logic programming language associated with artificial intelligence and computational linguistics.

Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a query over these relations. Prolog is well-suited for specific tasks that benefit from rule-based logical queries such as searching databases, voice control systems, and filling templates.

Rules and facts:

Prolog programs describe relations, defined by means of clauses. Pure Prolog is restricted to Horn clauses. There are two types of clauses: facts and rules. A rule is of the form

Head:- Body.

and is read as "Head is true if Body is true". A rule's body consists of calls to predicates, which are called the rule's goals. The built-in predicate, /2 (meaning a 2-arity operator with name,) denotes conjunction of goals, and; /2 denotes disjunction. Conjunctions and disjunctions can only appear in the body, not in the head of a rule.

Clauses with empty bodies are called facts. An example of a fact is:

cat (tom).

which is equivalent to the rule:

cat(tom) :- true.

The built-in predicate true/0 is always true.

Given the above fact, one can ask:

is tom a cat?

?- cat(tom).

Yes

what things are cats?

?- cat(X).

X = tom

Clauses with bodies are called rules. An example of a rule is:

animal(X) :- cat(X).

If we add that rule and ask what things are animals?

?- animal(X).

X = tom

Prolog must be able to handle arithmetic in order to be a useful general purpose programming language. However, arithmetic does not fit nicely into the logical scheme of things. That is, the concept of evaluating an arithmetic expression is in contrast to the straight pattern matching we have seen so far. For this reason, Prolog provides the built-in predicate 'is' that evaluates arithmetic expressions. Its syntax calls for the use of operators.

X is <arithmetic expression>

The variable X is set to the value of the arithmetic expression. On backtracking it is unassigned.

The variable X is set to the value of the arithmetic expression. On backtracking it is unassigned.

The arithmetic expression looks like an arithmetic expression in any other programming language.

Here is how to use Prolog as a calculator.

?- X is 2 + 2.

X = 4

?- X is 3 * 4 + 2.

X = 14

Parentheses clarify precedence.

?- X is 3 * (4 + 2).

X = 18

?- X is (8 / 4) / 2.

X = 1

In addition to 'is,' Prolog provides a number of operators that compare two numbers. These include 'greater than', 'less than', 'greater or equal than', and 'less or equal than.' They behave more logically, and succeed or fail according to whether the comparison is true or false. Notice

the order of the symbols in the greater or equal than and less than or equal operators. They are specifically constructed not to look like an arrow, so that the use arrow symbols in programs is without confusion.

$X > Y$

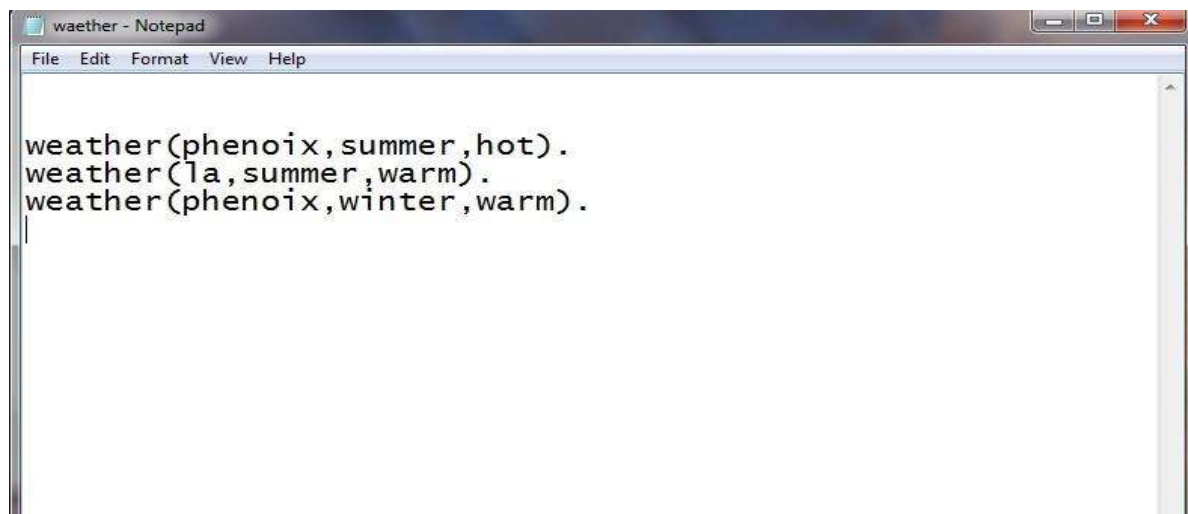
$X < Y$

$X \geq Y$

$X \leq Y$

6. Procedure/ Program:

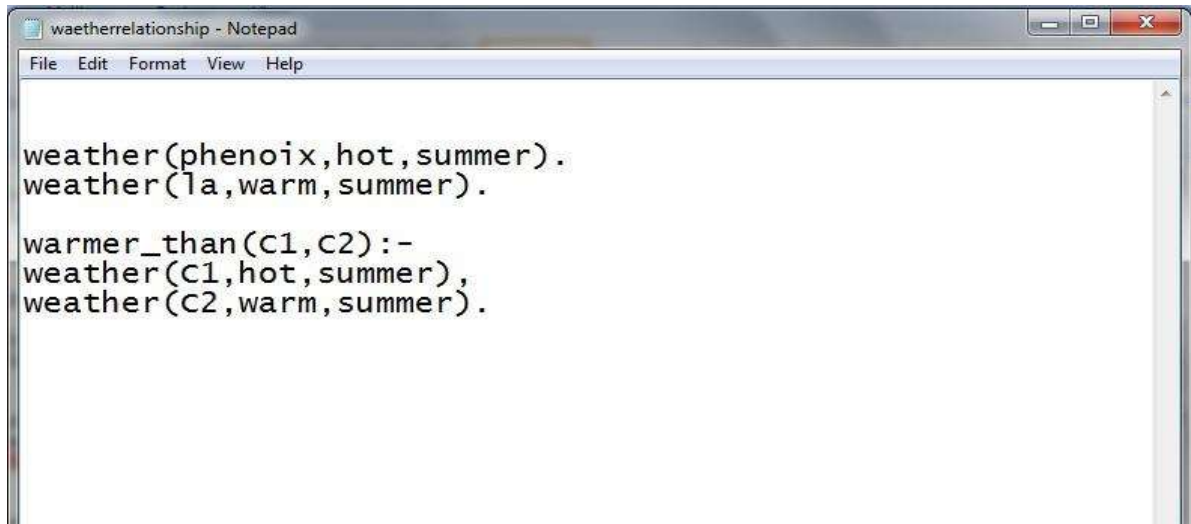
1 (A). Sample program to demonstrate Rules and facts.



```
File Edit Format View Help

weather(phenoix,summer,hot).
weather(la,summer,warm).
weather(phenoix,winter,warm).
```

1.(B) Sample program to demonstrate the relationship in prolog.

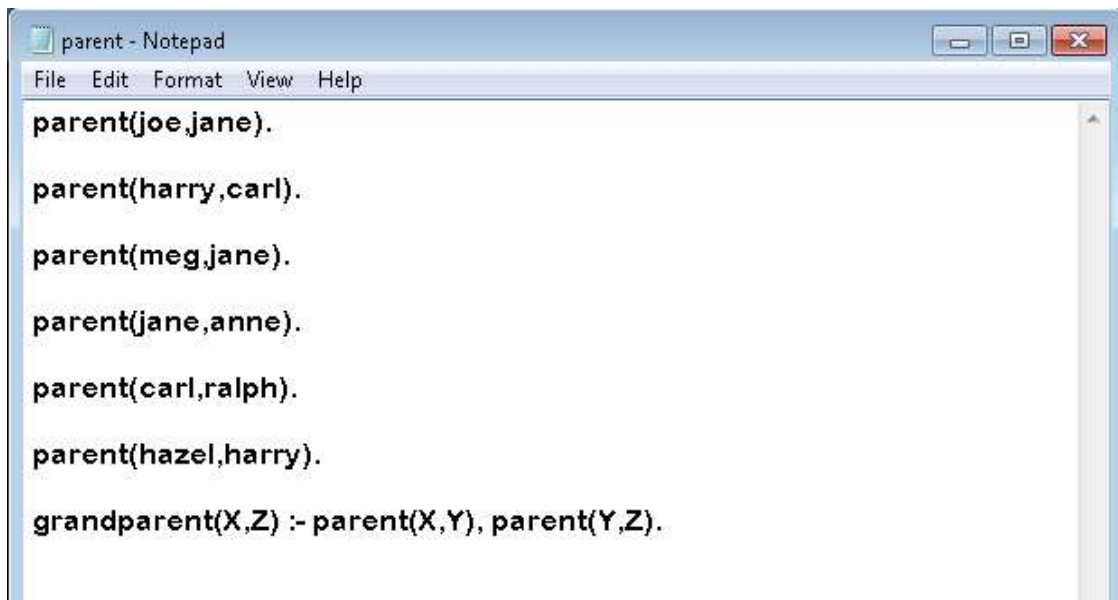


```
waetherrelationship - Notepad
File Edit Format View Help

weather(phenoix,hot,summer).
weather(la,warm,summer).

warmer_than(C1,C2):-
weather(C1,hot,summer),
weather(C2,warm,summer).
```

1.(C) Sample program to demonstrate the relationship in prolog.



```
parent - Notepad
File Edit Format View Help

parent(joe,jane).
parent(harry,carl).
parent(meg,jane).
parent(jane,anne).
parent(carl,ralph).
parent(hazel,harry).

grandparent(X,Z) :- parent(X,Y), parent(Y,Z).
```

7. Results:

1 (A). Sample program to demonstrate Rules and facts.



```
SWI-Prolog -- c:/Users/User/Desktop/waether.pl
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- weather(City,summer,hot).
City = phoenix.

2 ?- weather(City,_,warm).
City = la.

3 ?- weather(City,winter,warm).
City = phoenix.

4 ?- weather(la,winter,warm).
false.

5 ?-
```

1.(B) Sample program to demonstrate the relationship in prolog.

```
SWI-Prolog -- c:/Users/User/Desktop/waetherrelationship.pl
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- write(C1),write(' is warmer than '),write(C2).
_G1367 is warmer than _G1371
true.

2 ?- warmer_than(C1,C2):- weather(C1,hot,summer),weather(C2,warm,summer),write(C1),write('
is warmer than '),write(C2).
ERROR: Undefined procedure: (:-)/2
ERROR: Rules must be loaded from a file
ERROR: See FAQ at http://www.swi-prolog.org/FAQ/ToplevelMode.txt
3 ?- weather(C1,hot,summer),weather(C2,warm,summer),write(C1),write(' is warmer than '),wr
ite(C2).
phoenix is warmer than la
C1 = phoenix.
C2 = la.

4 ?-
```

1.(C) Sample program to demonstrate the relationship in prolog.



```
SWI-Prolog -- c:/Users/USER/Desktop/parent.pl
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 7.2.2)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- parent(carl,Y).
Y = ralph.

2 ?- parent(hazel,X).
X = harry.

3 ?- parent(joe,X),parent(meg,X).
X = jane.

4 ?- parent(Y,harry).
Y = hazel.

5 ?- █
```

8. Conclusion:

Demonstration and implementation of arithmetic operations, rules and facts, relationship is done using SWI-Prolog software. Semantics and syntax of prolog language is well understood. Logical programming concepts required to execute artificial intelligence problem is well understood

9. Viva Questions:

- Note how SWI-Prolog is better than turbo prolog.
- What are rules and facts?
- Explain syntax and semantics.

10. References:

2. Ivan Bratko "PROLOG Programming for Artificial Intelligence", Pearson Education, Third Edition.
3. Elaine Rich and Kevin Knight "Artificial Intelligence "Third Edition
4. Davis E.Goldberg, "Genetic Algorithms: Search, Optimization and Machine Learning", Addison Wesley, N.Y., 1989.
- 5.Han Kamber, "Data Mining Concepts and Techniques", Morgann Kaufmann Publishers