THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/01
EXPERIMENT NAME: Realization of commands, syntax of PROLOG

PROGRAM:

- Prolog (*pro*gramming in *log*ic) is one of the classical programming languages developed specifically for applications in AI. It is a *declarative* programming language. Prolog is particularly useful for certain problem-solving tasks in AI, in domains such as search, planning, and knowledge representation
- Programming in Prolog means describing the world. The simplest way of describing the world is by stating *facts*,
  ```
  bigger(elephant, horse).
  ```
- This state the fact that an elephant is bigger than a horse

-
  ```
  bigger(elephant, horse).
  bigger(horse, donkey).
  bigger(donkey, dog).
  bigger(donkey, monkey).
  ```
- After compilation we can ask the Prolog system questions (or *queries* in proper Prolog jargon) about it

-
  ```
  ?- bigger(donkey, dog).
  Yes
  ```
- This query succeeds, because the fact bigger(donkey, dog) has previously been communicated to the Prolog system.

-
  ```
  ?- bigger(monkey, elephant).
  No
  ```
- Query fails as expected

-
  ```
  ?- bigger(elephant, monkey).
  No
  ```
- Animal X is bigger than animal Y either if this has been stated as a fact or if there is an animal Z for which it has been stated as a fact that animal X is bigger than animal Z and it can be shown that animal Z is bigger than animal Y. In Prolog such statements are called *rules*.

-
  ```
  is_bigger(X, Y) :- bigger(X, Y).
  is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).
  ```
- In these rules :- means something like "if" and the comma between the two terms bigger(X, Z) and is_bigger(Z, Y) stands for "and". <u>X, Y, and Z are variables, which in Prolog is indicated by using capital letters.</u>

The Neotia University

- ?- is_bigger(elephant, monkey).
  Yes
- When doing so the two variables get instantiated: X = elephant and Y = monkey. The rule says that in order to prove the *goal* is_bigger(X, Y) Prolog has to prove the two *subgoals* bigger(X, Z) and is_bigger(Z, Y), again with the same variable instantiations. This process is repeated recursively until the facts that make up the chain between elephant and monkey are found and the query finally succeeds.
- ?- is_bigger(X, donkey).
- Which animals are bigger than a donkey
- The Prolog interpreter replies as follows:
  ?- is_bigger(X, donkey).
  X = horse
- In case we want to find out if there are more animals that are bigger than the donkey, we can press the semicolon key, which will cause Prolog to search for alternative solutions to our query. If we do this once, we get the next solution X = elephant: elephants are also bigger than donkeys. Pressing semicolon again will return a No, because there are no more solutions
- ?- is_bigger(X, donkey).
  X = horse ;
  X = elephant ;
  No
- The central data structure in Prolog is that of a term. There are terms of four kinds: *atoms*, *numbers*, *variables*, and *compound terms*. Atoms and numbers are sometimes grouped together and called *atomic terms*.
- Atoms are usually strings made up of lower- and uppercase letters, digits, and the underscore, *starting with a lowercase letter*. These are constants. The following are all valid Prolog atoms:
  elephant, b, abcXYZ, x_123, how_are_you_today
  On top of that also any sequence of arbitrary characters enclosed in single quotes denotes an atom.
  'This is also a Prolog atom.'
  Finally, strings made up solely of special characters like + - * = < > : & are also atoms.
  +, ::, <------>, ***
- All Prolog implementations have an integer type: a sequence of digits, optionally preceded by a - (minus)
- Variables are strings of letters, digits, and the underscore, *starting with a capital letter or an underscore*
  X, Elephant, _4711, X_1_2, MyVariable, _
  The last one is called the *anonymous variable* and is used when the value of a variable is of no particular interest. Multiple occurrences of the anonymous variable in one

The Neelia University

expression are assumed to be distinct, i.e., their values don't necessarily have to be the same.

- Compound terms are made up of a *functor* (a Prolog atom) and a number of *arguments* (Prolog terms, i.e., atoms, numbers, variables, or other compound terms) enclosed in parentheses and separated by commas. The following are some examples for compound terms:
  ```
  is_bigger(horse, X), f(g(X, _), 7), 'My Functor'(dog)
  ```
  It's important not to put any blank characters between the functor and the opening parentheses, or Prolog won't understand what you're trying to say. In other places, however, spaces can be very helpful for making programs more readable.
- A term that doesn't contain any variables is called a *ground term*.
- All text between /* and */ is taken to be a comment and ignored.
- Facts and rules are called *clauses*. They are used to define *predicates*. For example, in our introductory example we defined the predicate bigger by means of five facts and the predicate is_bigger by means of two rules.
- A fact is a predicate followed by a full stop. The intuitive meaning of a fact is that we define a certain instance of a relation as being true.
  ```
  bigger(whale, _).
  life_is_beautiful.
  ```
- A rule consists of a *head* (a predicate) and a *body* (a sequence of predicates separated by commas). Head and body are separated by the symbol :- and, like every Prolog expression, a rule has to be terminated by a full stop. The intuitive meaning of a rule is that the goal expressed by its head is true, if we (or rather the Prolog system) can show that all of the expressions (subgoals) in the rule's body are true.
  ```
  is_smaller(X, Y) :- is_bigger(Y, X).
  aunt(Aunt, Child) :- sister(Aunt, Parent),parent(Parent, Child).
  ```
- A Prolog program is a sequence of clauses.
- After compilation, a Prolog program is run by submitting queries to the interpreter. A query has the same structure as the body of a rule, i.e., it is a sequence of predicates separated by commas and terminated by a full stop. They can be entered at the Prolog prompt.
- Prolog provides a range of useful built-in predicates. Built-in predicate is not allowed to appear as the principal functor in a fact or the head of a rule.
- Most important built-in predicate is = (equality). Instead of writing expressions such as =(X, Y), we usually write more conveniently X = Y. Such a goal succeeds, if the terms X and Y can be matched.
- Sometimes it can be useful to have predicates that are known to either fail or succeed in any case. The predicates fail and true serve exactly this purpose. Some Prolog systems also provide the predicate false, with exactly the same functionality as fail.

The Neotia University

- Program files can be compiled using the predicate consult. The argument has to be a Prolog atom denoting the program file you want to compile. If the compilation is successful, Prolog will reply with Yes. Otherwise a list of errors will be displayed.

```
?- consult('big-animals.pl').
```

- If besides Prolog's replies to queries you wish your program to have further output you can use the write predicate. The argument can be any valid Prolog term. In the case of a variable its value will get printed to the screen. Execution of the predicate nl causes the system to skip a line.

```
?- write('Hello World!'), nl.
Hello World!
Yes


?- X = elephant, write(X), nl.
elephant
X = elephant
Yes
```

- The built-in predicate **read/1** is provided to input terms. It takes a single argument, which must be a variable. Evaluating it causes the next term to be read from the *current input stream*, which by default is the user's keyboard.
  In the input stream, the term must be followed by a dot ('.') and at least one *white space character*, such as space or newline. The dot and white space characters are read in but are not considered part of the term.
- There are a number of built-in predicates available that can be used to check the type of a given Prolog term.

```
?- atom(elephant).
Yes
?- atom(Elephant).
No
?- X = f(mouse), compound(X).
X = f(mouse)
Yes
```

- Most Prolog systems also provide a help function in the shape of a predicate, usually called help. Applied to a term (like the name of a built-in predicate) the system will display a short description, if available.
- `?- help(atom).`

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/02
EXPERIMENT NAME: Write a program in PROLOG to find factorial of a number

PROGRAM:

```
fact(0,1).
fact(N,F):-
    N>0,
    N1 is N-1,
    fact(N1,F1),
    F is N * F1.
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/03
EXPERIMENT NAME: Write a program in PROLOG to find the n-th Fibonacci term.

PROGRAM:

```
fib(0,0).
fib(1,1).
fib(N,NF):-
    N>1,
    A is N-1,
    B is N-2,
    fib(A,AF),
    fib(B,BF),
    NF is AF+BF.
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/04
EXPERIMENT NAME: Write a program in PROLOG to find GCD of two numbers.

PROGRAM:

```
gcd(U, V, U):-
   V=0.

gcd(U, V, X):-
   not(V=0),
   Y is U mod V,
   gcd(V, Y, X).
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/05
EXPERIMENT NAME: Write a program in PROLOG to find the maximum number in a list.

PROGRAM:

```
maxList([X],X).

maxList([X,Y|Rest],Max) :-
   maxList([Y | Rest],MaxRest),
   max(X,MaxRest,Max).

max(X,Y,X) :- X >= Y.
max(X,Y,Y) :- X < Y.
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/06
EXPERIMENT NAME: Write a program in PROLOG to find the relationship among members of a family.

PROGRAM:

```
male(jack).
male(oliver).
male(ali).
male(james).
male(simon).
male(harry).
female(helen).
female(sophie).
female(jess).
female(lily).


parent_of(jack,jess).
parent_of(jack,lily).
parent_of(helen, jess).
parent_of(helen, lily).
parent_of(oliver,james).
parent_of(sophie, james).
parent_of(jess, simon).
parent_of(ali, simon).
parent_of(lily, harry).
parent_of(james, harry).


father_of(X,Y):- male(X),
   parent_of(X,Y).


mother_of(X,Y):- female(X),
   parent_of(X,Y).
grandfather_of(X,Y):- male(X),
   parent_of(X,Z),
   parent_of(Z,Y).
```

The Neotia University

```
grandmother_of(X,Y):- female(X),
   parent_of(X,Z),
   parent_of(Z,Y).


sister_of(X,Y):- %(X,Y or Y,X)%
female(X), father_of(F, Y), father_of(F,X),X \= Y.


sister_of(X,Y):- female(X), mother_of(M, Y), mother_of(M,X),X \= Y.


aunt_of(X,Y):- female(X), parent_of(Z,Y), sister_of(Z,X),!.


brother_of(X,Y):- %(X,Y or Y,X)%male(X), father_of(F, Y), father_of(F,X),X \= Y.


brother_of(X,Y):- male(X),mother_of(M, Y), mother_of(M,X),X \= Y.


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


ancestor_of(X,Y):- parent_of(X,Y).


ancestor_of(X,Y):- parent_of(X,Z), ancestor_of(Z,Y).
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/07
EXPERIMENT NAME: Write a program in PROLOG to add all numbers in a list.

PROGRAM:

```
list_sum([], 0).


list_sum([H|T], TotalSum) :-
   list_sum(T, Sum1),
   TotalSum is H + Sum1.
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/08
EXPERIMENT NAME: Write a program in PROLOG to evaluate m^n.

PROGRAM:

```
pow(M, N, X) :-
        X is M**N.
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/09
EXPERIMENT NAME: Write a program in PROLOG to implement insertion sort.

PROGRAM:

```
insertionSort([], []) :- !.

insertionSort([X|L], S) :- insertionSort(L, S1), insert(X, S1, S).


insert(X, [], [X]) :- !.

insert(X, [X1|L1], [X, X1|L1]) :- X=<X1, !.

insert(X, [X1|L1], [X1|L]) :- insert(X, L1, L).
```

The Neotia University

THE NEOTIA UNIVERSITY
DEPARTMENT OF ROBOTICS ENGINEERING
ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO. : PC-REP 402/10
EXPERIMENT NAME: Write a program in PROLOG to solve the water jug problem.

PROGRAM:

```
waterjug(X,Y):-
   X=0,
   Y=0,
   write("4l jug empty & 3l jug empty"),nl,
   Z=0,
   A=3,
   waterjug(Z,A).

waterjug(X,Y):-
   X=0,
   Y=3,
   write("4l jug empty & 3ljug 3l water"),nl,
   Z=3,
   A=0,
   waterjug(Z,A).

waterjug(X,Y):-
   X=3,
   Y=0,
   write(" 4l jug 3l water & 3l jug empty"),nl,
   Z=3,
   A=3,
   waterjug(Z,A).

waterjug(X,Y):-
   Z=3,
   A=3,
   write("4l jug 3l water& 3l jug 3l water"),nl,
  Z=4,
  A=2,
  waterjug(Z,A).
```

The Neotia University

```
waterjug(X,Y):-
    X=4,
    Y=2,
    write("4l jugh 4l water & 3l jug 2l water"),nl,
   Z=0,
   A=2,
   waterjug(Z,A).

waterjug(X,A):-
    X=0,
    Y=2,
    write("4l jug empty & 3l jug 2l water"),nl,
    Z=2,
   A=0,
   waterjug(Z,A).

waterjug(X,Y):-
    X=2,
    Y=0,
    write("Goal Achieved").
```

The Neotia University