Artificial Intelligence

Unit-8 (Prolog)

Artificial Intelligence 01CE0702

Department of Computer Engineering

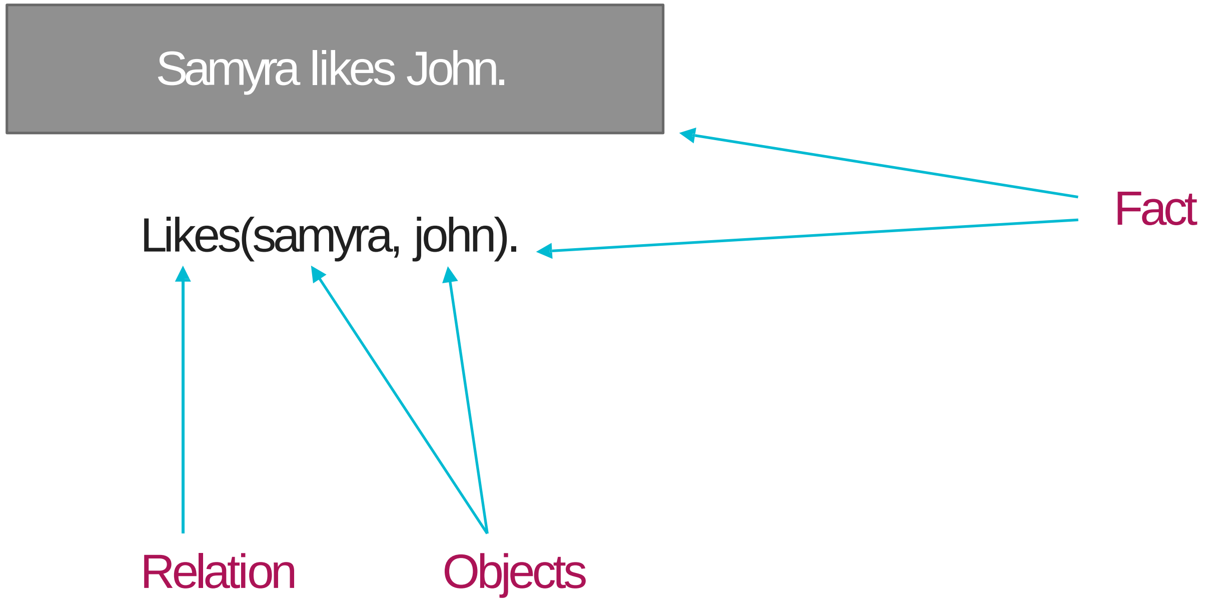
Shilpa
Singhal

Outline

Introduction

- Prolog stands for Programing in logic.
- Prolog is a declarative programming language unlike most common programming languages.
- In a declarative language the programmer specifies a goal to be achieved and then the Prolog system works out how to achieve it.
- Rather than describing how to compute a solution, a program consists of a data base of facts and logical relationships (rules).
- The user asks a question to obtain a solution of a problem.
- When asked a question, the run time system searches through the data base of facts and rules to determine (by logical deduction) the answer.
- Prolog is a declarative language, which means that a program consists of data based on the facts and rules, i.e., relationships among these facts.
- Prolog is used some areas like natural language processing, artificial intelligence, expert systems, automated reasoning, etc.

Facts, Objects, Relations


Samyra likes John.

Likes(samyra, john).

Facts, Objects, Relations

- Facts can be describes as a symbolic Relationships
- Facts are statements about what is true about a problem, instead of instructions how to accomplish the solution.
- The Prolog system uses the facts to work out how to accomplish the solution by searching through the space of possible solutions.
- Example:-
  - ⤷ Bob is a student.
  - ⤷ student(bob).
- This expression is called a clause.
- An object is the name of element of certain type.
- A Relation is a name that defines the way in which a collection of objects or variables belong together.

Predicate

- A relation identifier is referred to as a predicate
- Example:-
  - Car is blue
  - Is(car, blue)
- Predicate express a relationship.
- The element within the parenthesis are the arguments of the predicate, which may be objects or variable.
- The word before the parenthesis is the name of relation.

Rules

- Rules are used when you want to say that a fact depends on a group of facts.
- Rule consist of a head and a body connected by the symbol **:-** (IF)
- **Syntax of rule**
  - <head> :-<body>
  - Read ':-' as 'if'.
  - *likes(john,X) :-likes(X,cricket).*
  - "John likes X if X likes cricket".
  - Rules always end with '.'

Structure of Prolog Program

Domains

// defining Objects, variable

//Person1,Person2 = Symbol

Predicates

// defining a Relation and Function

//likes(Person1,Person2)

clauses

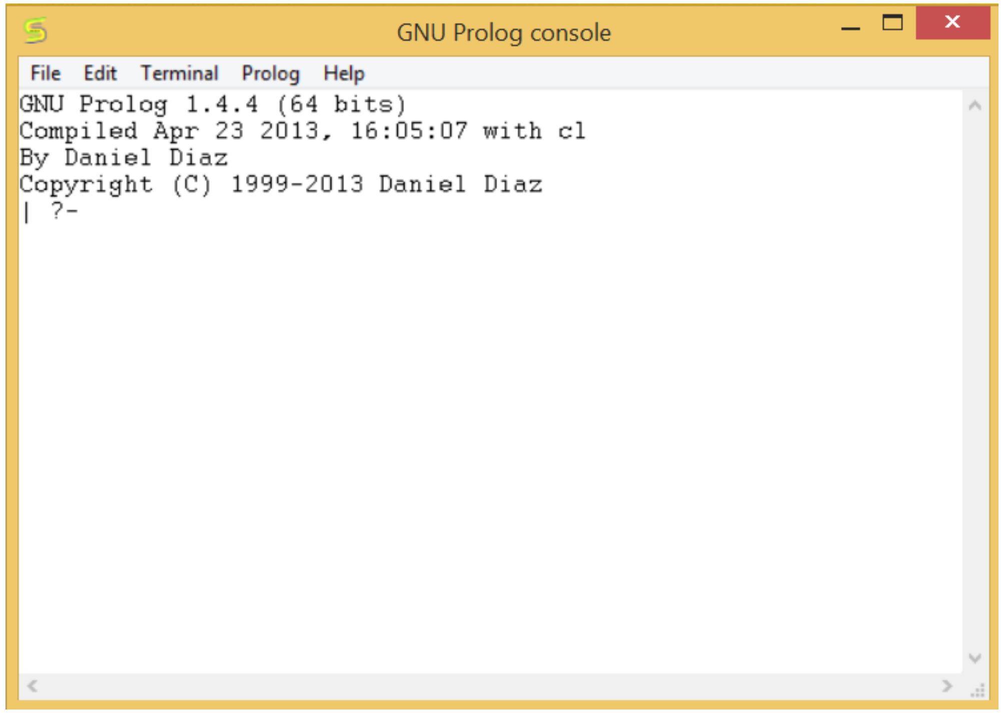
// defining Facts,Rules

// likes(aditya, naina).

//likes(bob, hezal).

Installing GNU prolog Complier

- This is a prolog interpreter working under the GNU licence. Nice things about it are: Windows/UNIX support, comprehensive, free.
  - Download GUN prolog form http://www.gprolog.org/
  - Double click setup file to install
  - After installation complete, run GUN Prolog



```
GNU Prolog 1.4.4 (64 bits)
Compiled Apr 23 2013, 16:05:07 with cl
By Daniel Diaz
Copyright (C) 1999-2013 Daniel Diaz
| ?-
```

How to load and run ?

Load

- ?-[prolog file Name].

Run

- ?-Predicate(arguments).

Example

Example1.pl

```
Domains
        Person1,Person2 = Symbol
Predicates
        likes(Person1,Person2)
Clauses
         likes(aditya, naina).
         likes(bob, hezal).
```

Output

```
Goal:likes(bob, hezal).
True
Goal:likes(bob, mery).
False
```

Example2.pl

```
boy(abc).
boy(bob).
boy(johan).
girl(hezal).
girl(xyz).
```

Output

```
Goal:boy(bob).
True
Goal:girl(bob).
False
```

Find Maximum number from two numbers

Max.pl

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

Max.pl

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

Output

```
Goal:max(3,4,MAX).
MAX = 4
Goal:max(5,4,MAX).
MAX = 5
```

Recursion in prolog

- Any function which calls itself is called recursive function.
- In Prolog, recursion appears when a predicate contain a goal that refers to itself.
- This simply means a program calls itself typically until some final point is reached.
- In Prolog and in any language, a recursive definition always has at least two parts.
- A first fact that act like a stopping condition and a rule that call itself simplified.
- At each level the first fact is checked. If the fact is true then the recursion ends. If not the recursion continue.
- A recursive rule must never call itself with the same arguments. If that happens then the program will never end.

Find factorial of given number

fact.pl

```
fact(0,Result) :-
    Result is 1.

fact(N,Result) :-
    N > 0,
    N1 is N-1,
    fact(N1,Result1),
    Result is Result1*N.
```

Output

```
Goal:fact(2,F).
F = 2
Goal:fact(3,F).
F = 6
```

Find the sum of first N natural numbers

Number.pl

```
sum(0,0).
sum(N,R):-
      N > 0,
      N1 is N-1,
      sum(N1,R1),
      R is R1+N.
```

Output

```
Goal:sum(4,SUM).
SUM = 10
```

Print Fibonacci series

Fibonacci.pl

```
fibonacci(1).
fibonacci(N) :-
        N1 = N - 1,
        N1 >= 0,!,
        fibonacci(N1),
        write(F1," ,"),
        F = F1 + N.
```

Output

```
Goal:fibonacci(2).
0 1
Goal:fibonacci(3).
0 1 2
```

Fail

- Fail predicate simply fails the rule.
- A typical use of fail is a negation of a predicate
- when Prolog fails, it tries to backtrack. Thus fail can be viewed as an instruction to force backtracking.

Fail.pl

```
a(X) :- b(X),c(X),fail.
a(X) :- d(X).



b(1).
b(4).
c(1).
c(3).
d(4).
```

Output

```
Goal:a(x).
X = 4
```

Cut

- Sometimes it is desirable to selectively turn off backtracking.
- Cut always succeeds, but cannot be backtracked.
- The cut effectively tells Prolog to freeze all the decisions made so far in this predicate. That is, if required to backtrack, it will automatically fail without trying other alternatives.
- Performance is the main reason to use the cut.
- The Symbol of cut predicate is "!"

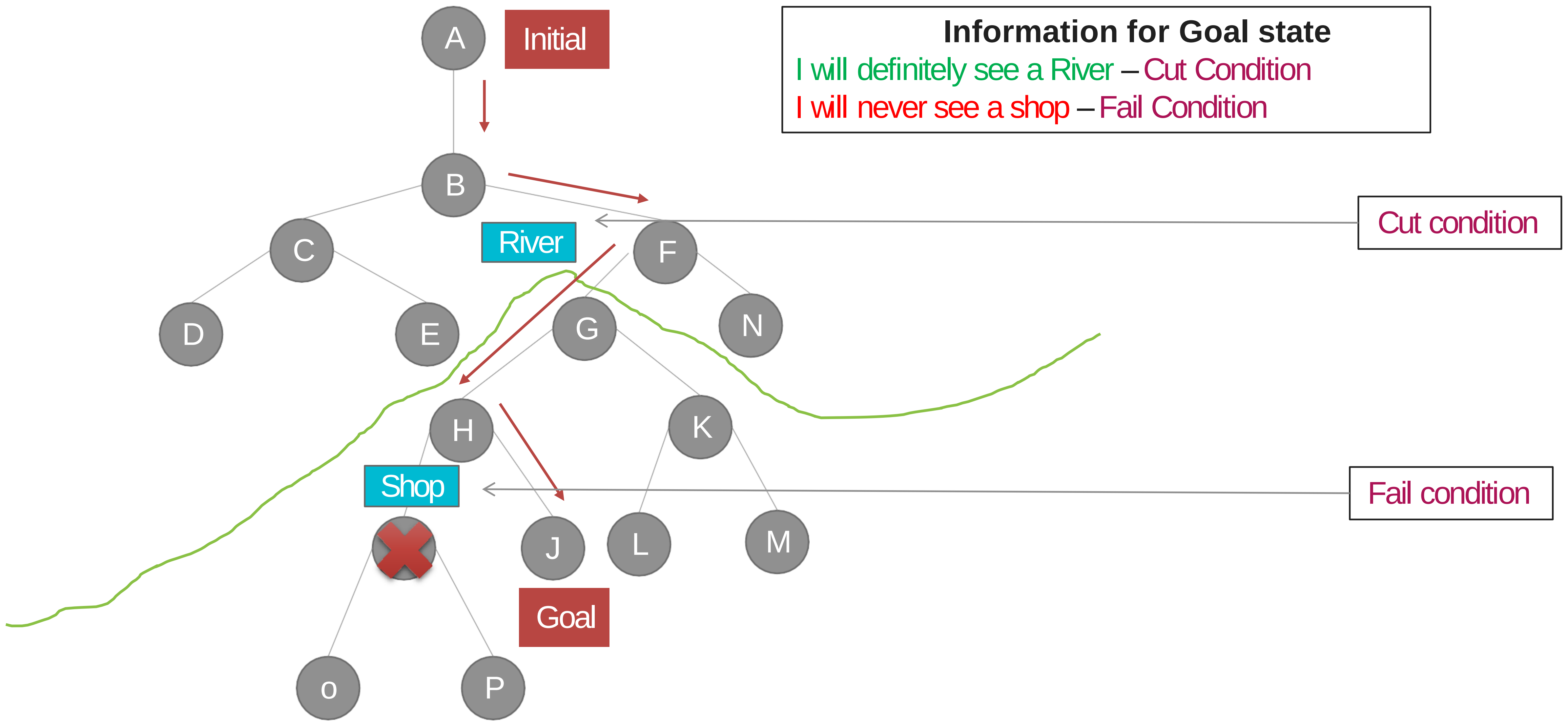
cut.pl

```
a(X) :- b(X),!,c(X).
a(X) :- d(X).



b(1).
b(4).
c(1).
c(3).
d(4).
```

Output

```
Goal:a(x).
X = 1
```

Cut and Fail With Example



Information for Goal state

I will definitely see a River – Cut Condition

I will never see a shop – Fail Condition

List

- it is a finite sequence of elements.
- A list in PROLOG is a structure of the form

  [t1,t2,t3 … tn]

- The elements in list organised in two section head and tail.
- The direct access to only one element call head , while the rest forms the list called the Tail.

  [Head|Tail]

- where Head is a single element, while Tail is list.

Example of List

- To check weather an object X is member of list L or not.

List.pl

```
find(X,[X|TAIL]).
find(X,[_|TAIL]):- find(X,TAIL).
```

Output

```
Goal:find(a,[a,b,c]).
true
Goal:find(d,[a,b,c]).
false
```

- Calculating the number of items of a given list.

List.pl

```
length([],0).
length([_|TAIL],N) :-
                 length(TAIL,N1),N is N1 +1.
```

Output

```
Goal:length([a,b,c],N).
N = 3
Goal:length([],0).
N = 0
```

Example of List

- Find the nth element of a given list.

List.pl

```
match([H|_],0,H).
match([_|T],N,H) :-
    N > 0,
    N1 is N-1,
    match(T,N1,H).
```

Output

```
Goal:match([a,b,c],0,N).
N = a
```

- Merge two List

List.pl

```
 con([],L1,L1).
 con([X|Tail],L2,[X|Tail1]):-
                con(Tail,L2,Tail1).
```

Output

```
Goal:con([a,b,c],[1,2],N).
N = [a,b,c,1,2]
```

Thank You!