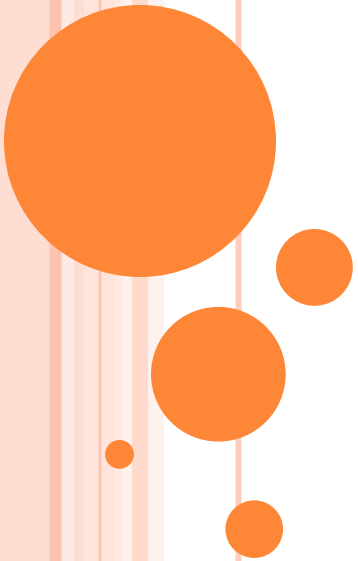
Introduction To Prolog

- Brian Hutchinson



Present by -

W.J.A.I.U. Jayaweera- 100227D

G.K.M.C Sajeewa- 100470N

M.M.D.T.K Wijewardane- 100612E

Outline

- Introduction
- Language Features
- More Features
- Behind the scenes
- Language Classifications
- Examples

Introduction

- Prolog is the most popular language of the logic programing languages.
- It is goal based language, it has automatic backtracking and uses recursion.
- Prolog stands for programmation en logique", or programming in logic."

Introduction [ctd...]

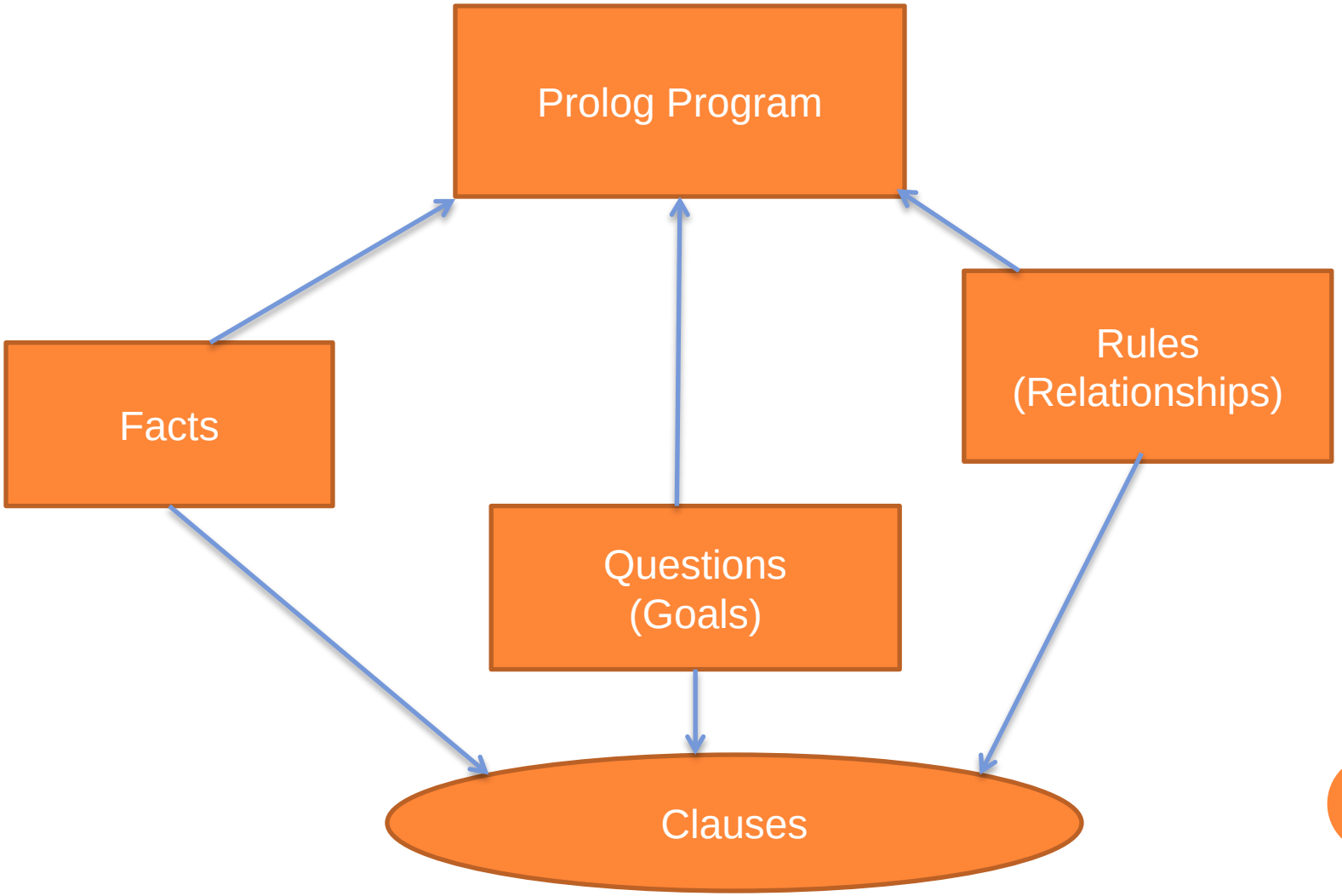
- Invented by Alain Colmerauer and Philippe Roussel of the Groupe d'Intelligence Articielle (GIA) in early 1970s.
- The earliest Prolog interpreter was written in Algol in1972 by Roussel.
- The official ISO standard for Prolog was published in 1996.

Current Implementations

- There are several popular Prolog interpreters available. They include:

- GNU Prolog (http://pauillac.inria.fr/vdiaz/gnu-prolog/)
- SWI-Prolog (http://www.swi-prolog.org/)
- Visual Prolog (http://www.visual-prolog.com/vip6/Download/Default.htm)
- BProlog (http://www.cad.mse.kyutech.ac.jp/people/zhou/bprolog.html)

Language Features



Language Features[ctd...]

- Facts-Something that is declared to always be true.

  Ex-father(bob,george)

  mother(alice,jeffrey)

- Rules- Something which is true only if all conditions (sub goals) on the right are true

  Ex-

  parent(X,Y) :- father(X,Y).

  parent(X,Y) :- mother(X,Y).

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

Language Features[ctd...]

- Questions- Something which asks from the system

  Ex- ?-ancestor(X,cindy),sibling(X,jeffrey)

- Clauses- Something which has a head and a body.

  - Fact = head
  - Rules = head + body
  - Question=body

Language Features [ctd...]

Ex-

- uncle(X,Y) :- sibling(X,Z), parent(Z,Y).

  (head) (body)

- ?- ancestor(X,cindy),sibling(X,jeffrey)

  (body)

- father(john,mary)

  (head)

Language Features[ctd...]

- Terms - term can be number, constant (Albert) or variable(X).
- Compound Terms- data structure, convenient way of grouping relevant data together

  Ex- fullname(joe,brown)

  student(fullname(joe,brown), 25)

Full name , Student are called as functors.

More Features

- Order- order matters in prolog, order is top to bottom.

  Ex- dog(rover).

  dog(duke).

  We enter the following question:

  ?- dog(X).

  Answer : X = rover

- Order is important for the efficiency and correctness of the program.

More Features[ctd...]

- Backtracking-
  - prolog stores data in tree.
  - It uses depth first search to find answers.
  - In its attempt to prove a goal, Prolog sometimes goes down a dead-end, at which point it needs to **backtrack.**

More Features[ctd...]

Ex- Sample Code

Facts- green(poisonivy).
green(broccoli).
edible(icecream).
edible(broccoli).

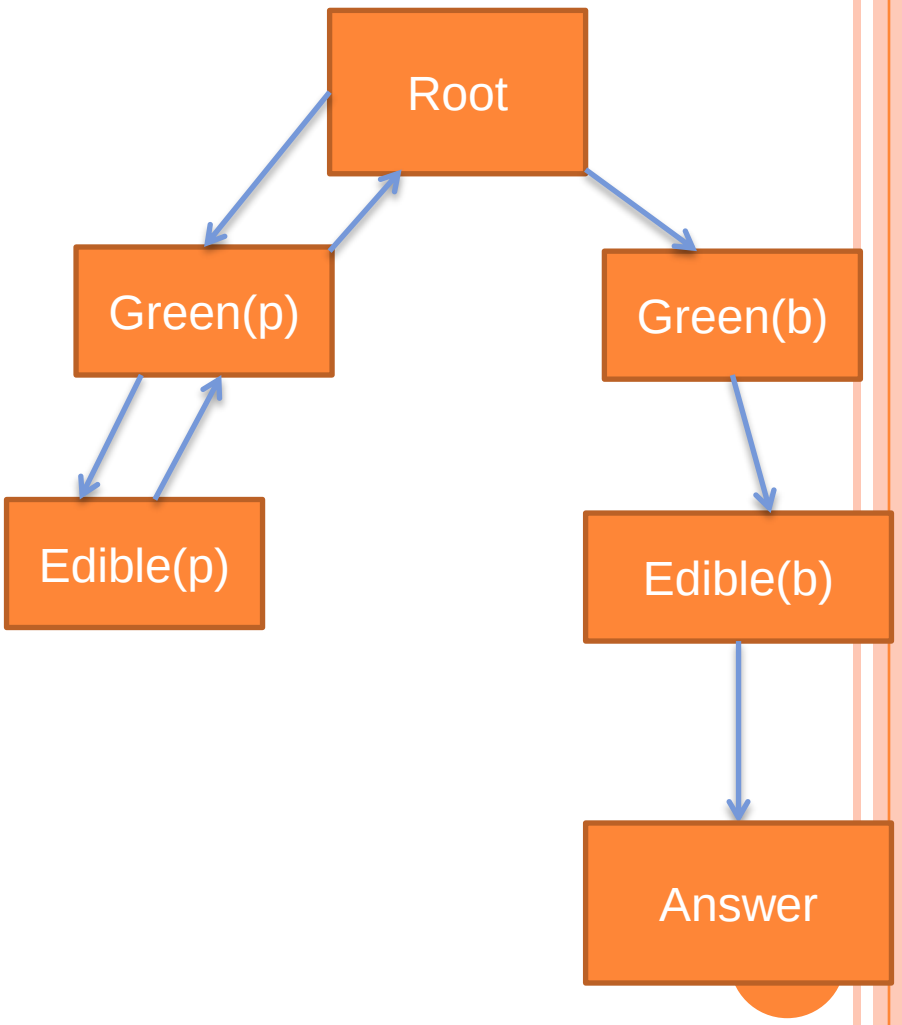
Question- ?- green(X),edible(X).

Answer- X = broccoli

More Features[ctd...]

Execution trace-

?- green(X),edible(X).
1 1 Call: green(poisoniyy) ?
1 1 Exit: green(poisonivy) ?
2 1 Call: edible(poisonivy) ?
2 1 Fail: edible(poisonivy) ?
1 1 Redo:green(poisonivy)
1 1 Call: green(broccoli) ?
1 1 Exit: green(broccoli) ?
2 1 Call: edible(broccoli) ?
2 1 Exit: edible(broccoli) ?



More Features[ctd...]

- Cuts- The cut is denoted by exclamation mark "!" and disallows prolog from backtracking past that point.
- Ex-

With Out Cut

```
grade(G,a) :- G > 90.
grade(G,b) :- G > 80.
grade(G,c) :- G > 70.
grade(G,d) :- G > 60.
grade(_,f).
```

With Cut

```
grade2(G,a) :- G > 90, !.
grade2(G,b) :- G > 80, !.
grade2(G,c) :- G > 70, !.
grade2(G,d) :- G > 60, !.
grade2(_,f).
```

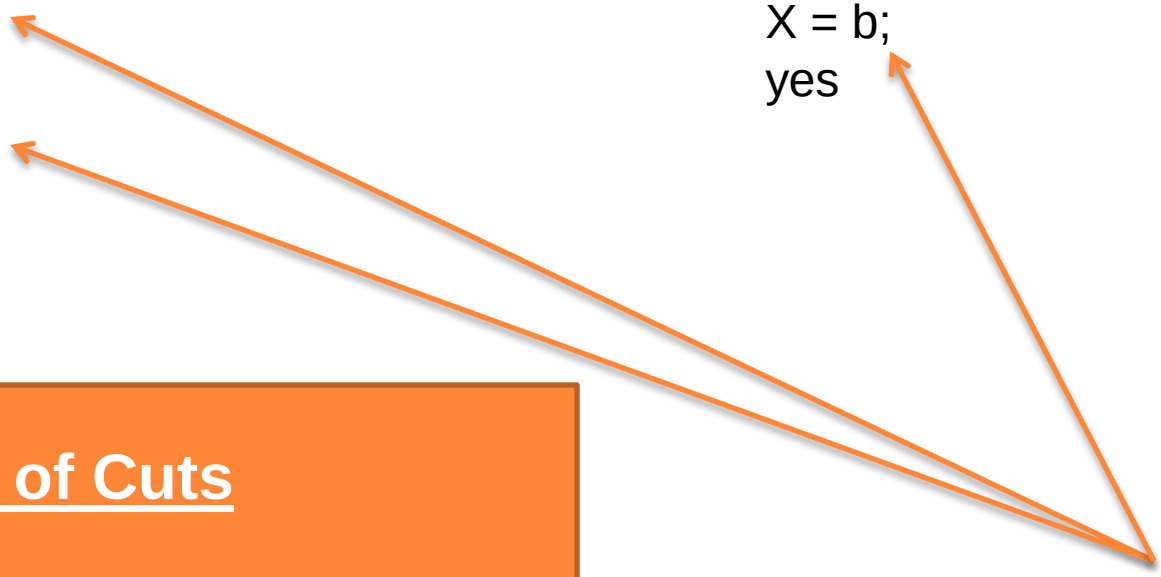
More Features[ctd...]

Forced Backtrack With Grade

?- grade(83,X)
X = b;
X = c;
X = d;
X = f
yes

Forced Backtrack With Grade2

?- grade2(83,X)
X = b;
yes



Types of Cuts

Red Cuts
Green Cuts

Forced back tracking with semi colon.

More Features[ctd...]

Recursion- like functional languages prolog uses recursion.

Ex-

```
ancestor(X,Z) :- parent(X,Z).
```

(Base Case)

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

(Recursive Rule)

More Features [ctd...]

- Lists- important data structure in prolog and are processed recursively.

Ex-

[a, b, a, a, c, d]

[]

[the, dog]

List= Head + Tail
List= [Head | Tail]
Ex-
List- [a,b,a] can be represented as
[a | [b,a]] or
[a, b | [a]] or
[a, b, a | []]

More Features [ctd...]

There are several other features in prolog

- Negation
- Assertion and Retract

Behind the Scenes

Resolution and Unification

- Resolution
  - The resolution principle,
    - If C1 and C2 are Horn clauses and the head of C1 matches one of the terms in the body of C2 then we can replace the term in C2 with the body of C1.
  - Example :
    - takes(Saman, cs123).
    - classmates(X, Y) :- takes(X, Z), takes(Y, Z).
    - classmates(Saman, Y) :- takes(Y, cs123).

Resolution and Unification [ctd…]

- Unification
  - Prolog associates variables and values using a process known as unification
  - Variables that receive a value are said to be instantiated
  - Example :
    - The pattern-matching process used to associate variable X with Saman and Z with cs123 is known as unification.

Resolution and Unification [ctd…]

The unification rules for Prolog :

- A constant unifies only with itself.
- Two structures unify if and only if they have the same functor and the same number of arguments, and the corresponding arguments unify recursively.
  - Example :
    - ?- f(a, b) = f(a, b, c).

      No
- A variable unifies with anything.

Depth First Search

Prolog uses a depth-first search strategy when traversing the tree.

- Advantage :
  - Less memory usage
- Disadvantage :
  - Prolog may descend down an endless branch (Black holes)

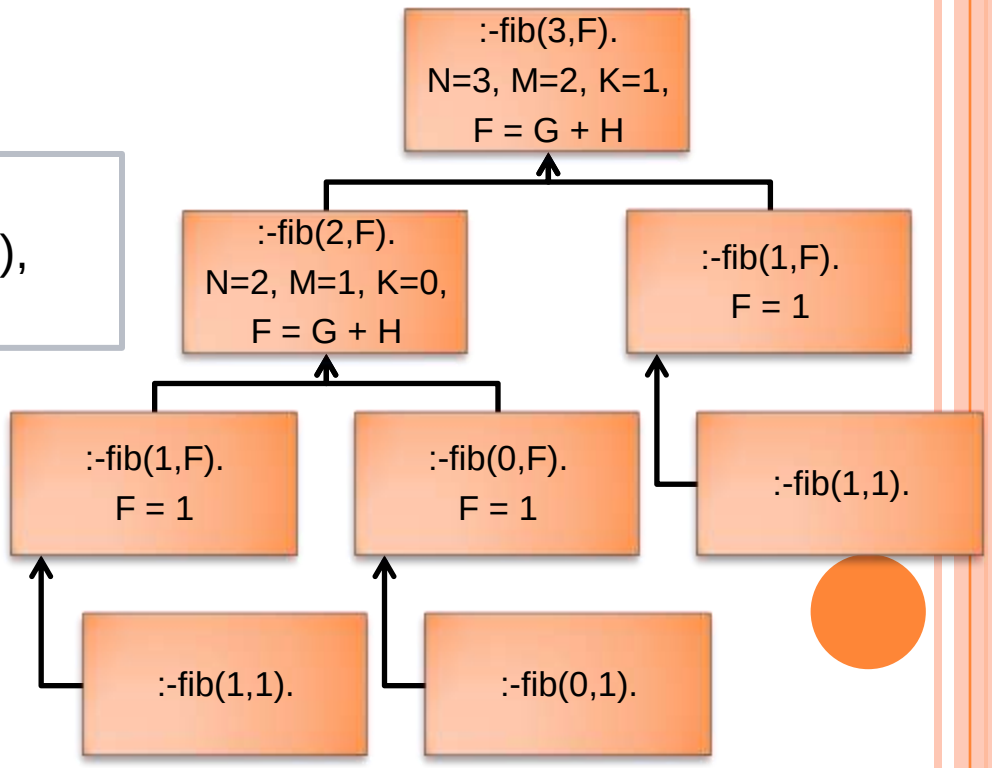
Top Down versus Bottom Up

- Top Down Control
  - we start from the goal and attempt to reach the hypotheses
- Bottom Up Control
  - we start with the hypotheses and attempt to reach the goal

Bottom Up Control

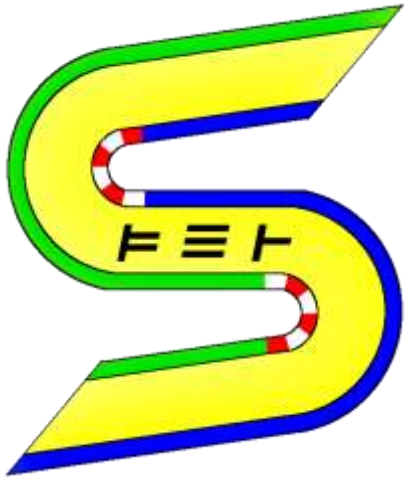
- Example :
  - Method for calculating a fibonacci number,
    - Starting with the bases cases and accumulating until it reaches the goal.

```
fib(0,1). fib(1,1).
fib(N,F) :- N=M+1, M=K+1, fib(M,G),
            fib(K,H), F=G+H, N>1.
```



Compiling Prolog Code

- A program exists called wamcc, based on the Warren Abstract Machine.
- C code may then be compiled to produce stand alone executable.



GNU Prolog logo

Strengths of Prolog [ctd…]

- Simplicity of Prolog
- Ability to model certain problems very concisely and elegantly
- Capability of solving problems that it wasn't originally designed to solve
- Ability to manipulate symbolic data

Strengths of Prolog [ctd…]

Prolog's strength to manipulate symbolic data:

"There are well-known examples of symbolic computation whose implementation in other standard languages took tens of pages of indigestible code. When the same algorithms were implemented in Prolog, the result was a crystal-clear program easily fitting on one page."

- Bratko

Weaknesses of Prolog

- Lack of structure
- Issues with readability of Prolog code
  - serious implications for the readability and declarativeness of code due to cut predicate (!).
- Inefficiency of Prolog code
- Logical imperfectness

Language Classification

Hierarchy of Languages

According to Programming Language Pragmatics by Michael L. Scott,

- Declarative
  - functional
    - Lisp/Scheme
    - ML
    - Haskell
  - dataflow
    - Id
    - Val
  - logic
    - Prolog
    - VisiCalc

Hierarchy of Languages [ctd…]

- Imperative
  - von Neumann
    - Fortran
    - Pascal
    - Basic
    - C
  - Object-Oriented
    - Smalltalk
    - Eiel
    - C++
    - Java

Scope

- Scope of a variable
  - in scope within a clause
- Scope of the head of each clause
  - in global scope
- Scope of constants
  - in global scope

- Example
  1. hates(X,Y) :- friends(X,Z),hates(Z,Y).
  2. loves(X,Y) :- hates(X,Z),hates(Y,Z).

Type Systems

- Prolog is dynamically typed.
- At runtime, if a built-in predicate is given the wrong type of argument an error will occur.
- For example:

1. ?- X is 1+1.
   X = 2
   Yes

2. ?- X is 1+dog.
   uncaught exception: error(type_error(evaluable,dog/0),(is)/2)

Type Systems [ctd…]

- A term can be quarried to find out its type with Prolog in the following way:

  - ?- number(5).

    yes
  - ?- number(dog).

    no
  - ?- var(X).

    yes
  - ?- var(5).

    no

Bindings

- Variable Cells
  - A variable is stored in a single heap cell.
  - This cell contains a tag and a store address of the that which it is bound to.

3	REF	5

  - Unbound variables,
    - by convention, point at their own cell.

2	REF	2

Bindings [ctd…]

- Structure Cells
  - A non-variable term is stored in a structure cell.
  - For a structure $f(t_1:::t_n)$, there will be n + 2 cells in the heap.

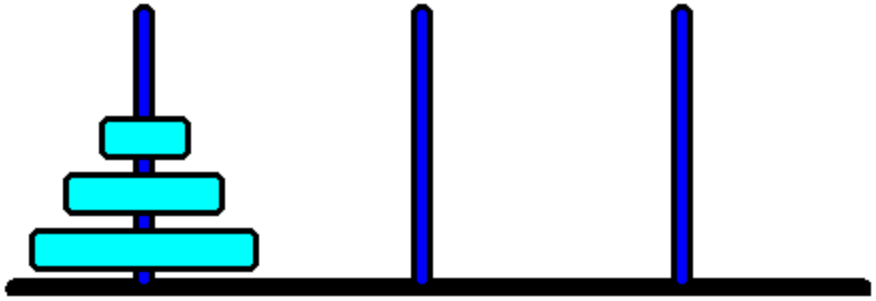
0	STR	1
1	f/n	
2	REF	2
……	……	
……	……	
n + 1	REF	n + 1

Examples

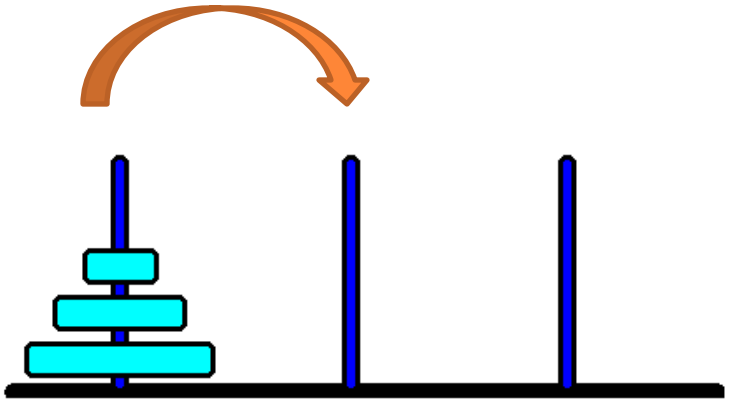
1. Implementing the Towers of Hanoi problem
2. Implementing a non-deterministic finite state automaton

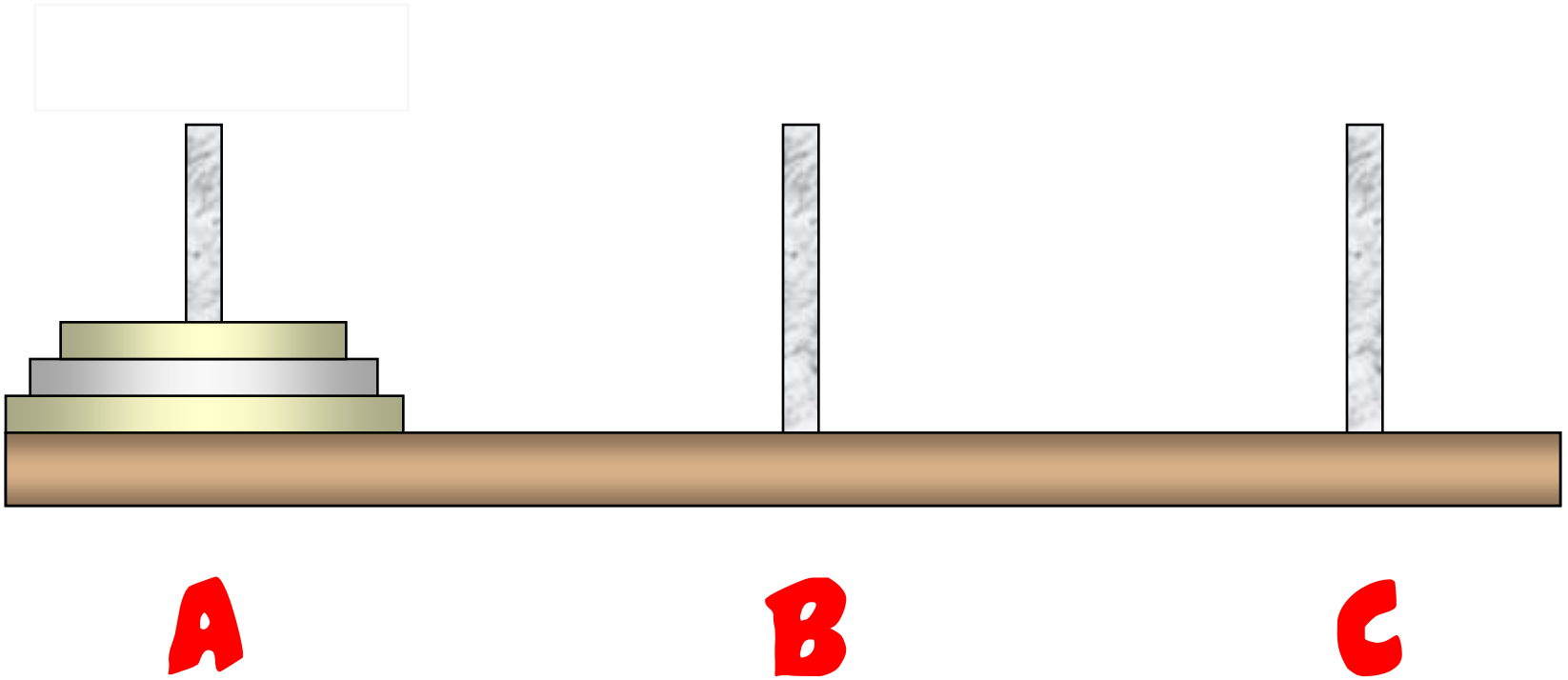
Towers of Hanoi

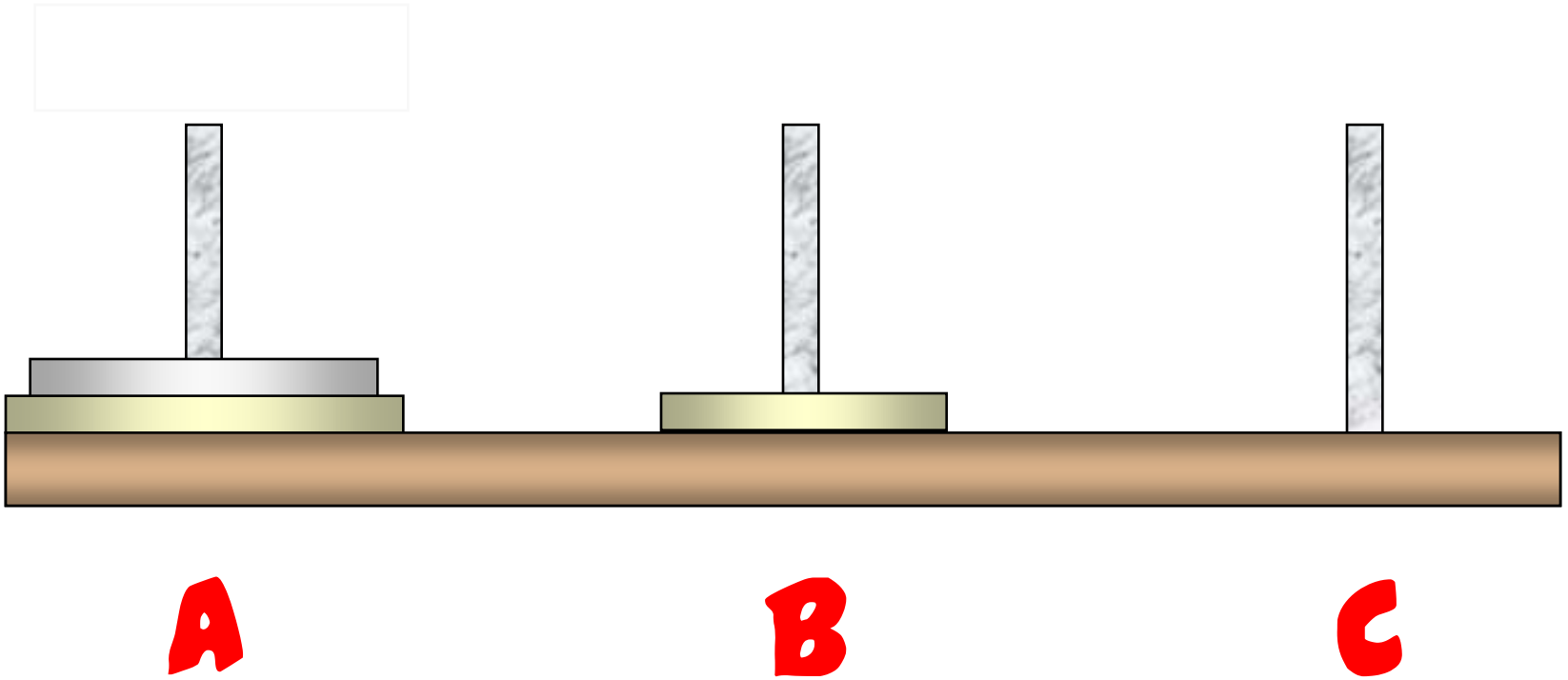
- In this puzzle, we have three pegs and several disks, initially stacked from largest to smallest on the left peg

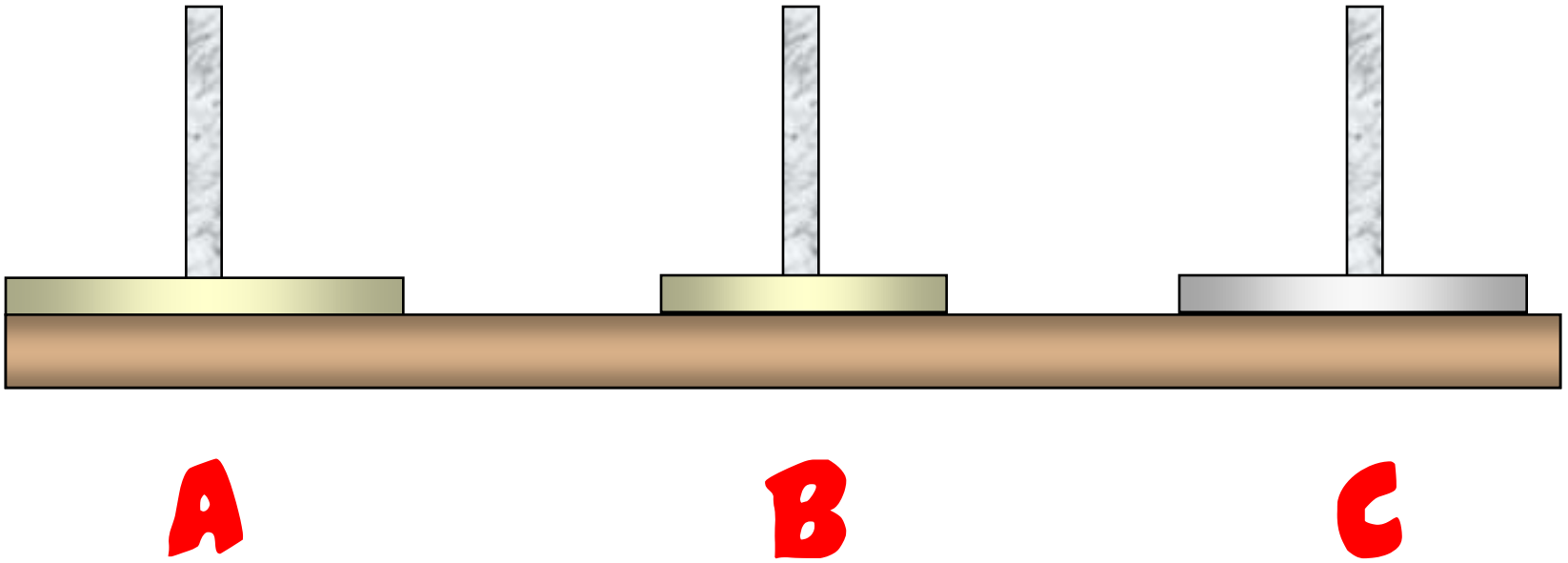


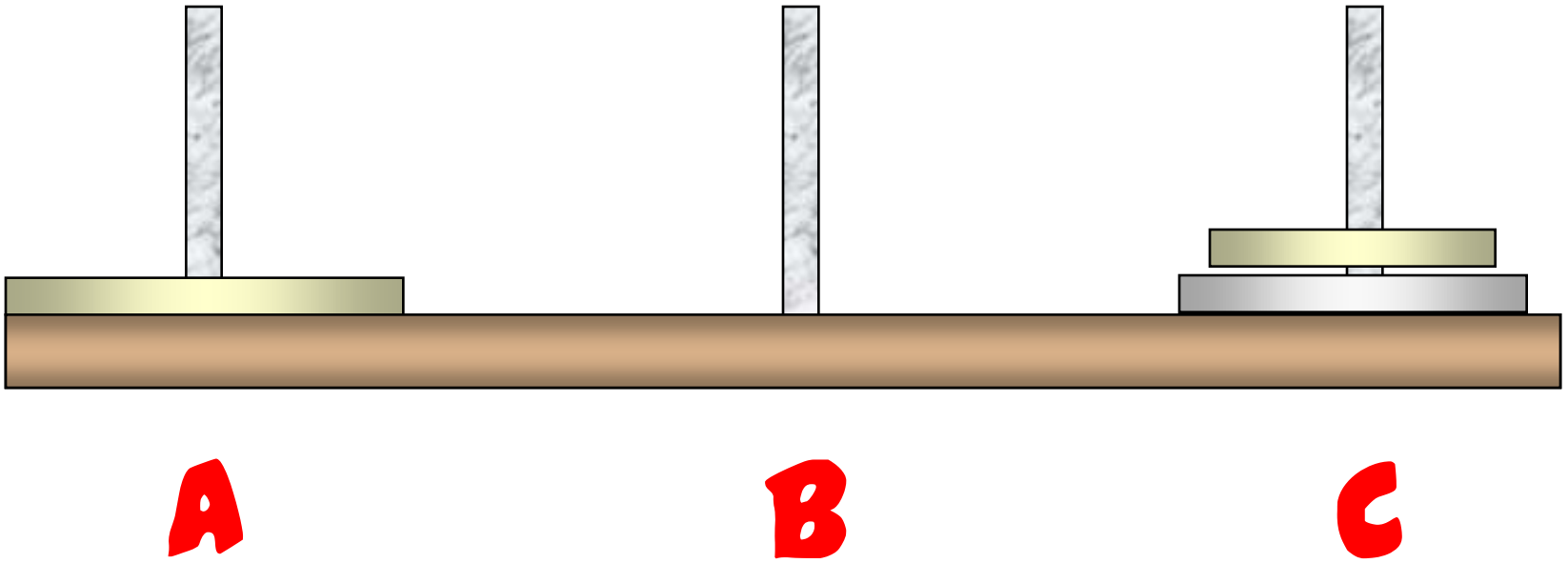
- Our goal is to move the entire tower to the middle peg
- We can only move one disk at a time
- We can use right peg to temporally hold the disks
- We can never place a larger disk on a smaller disk in any peg

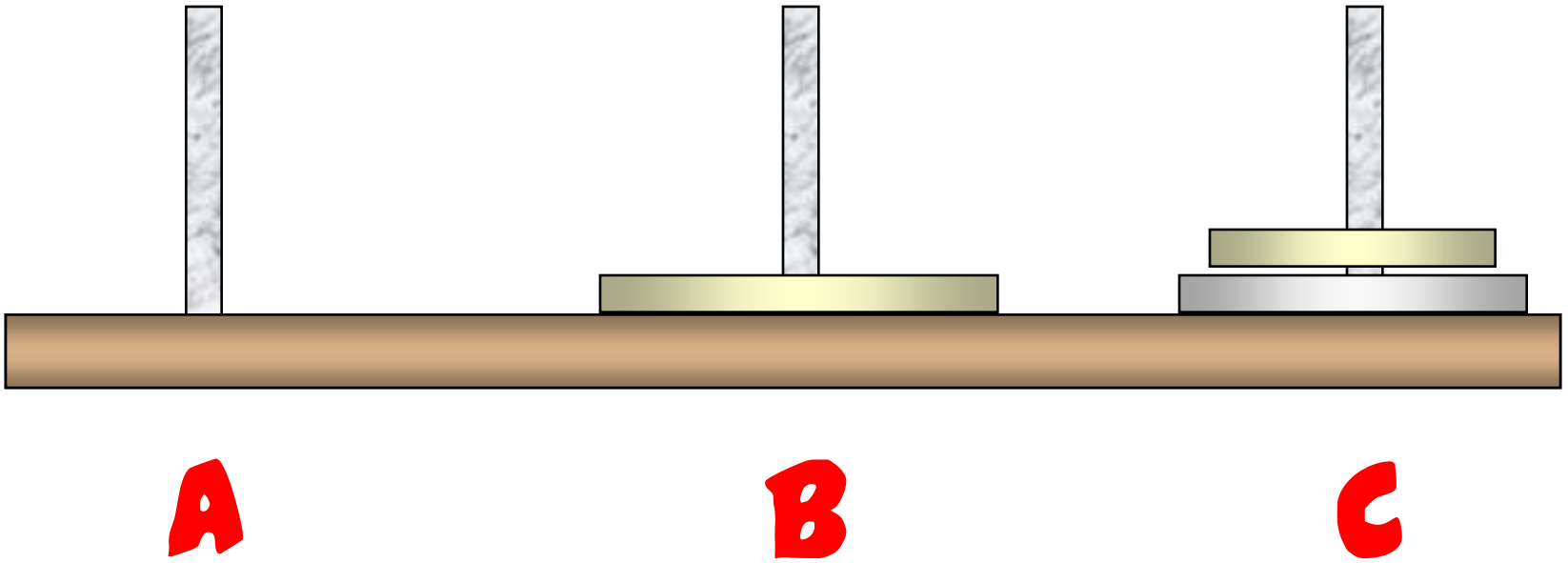


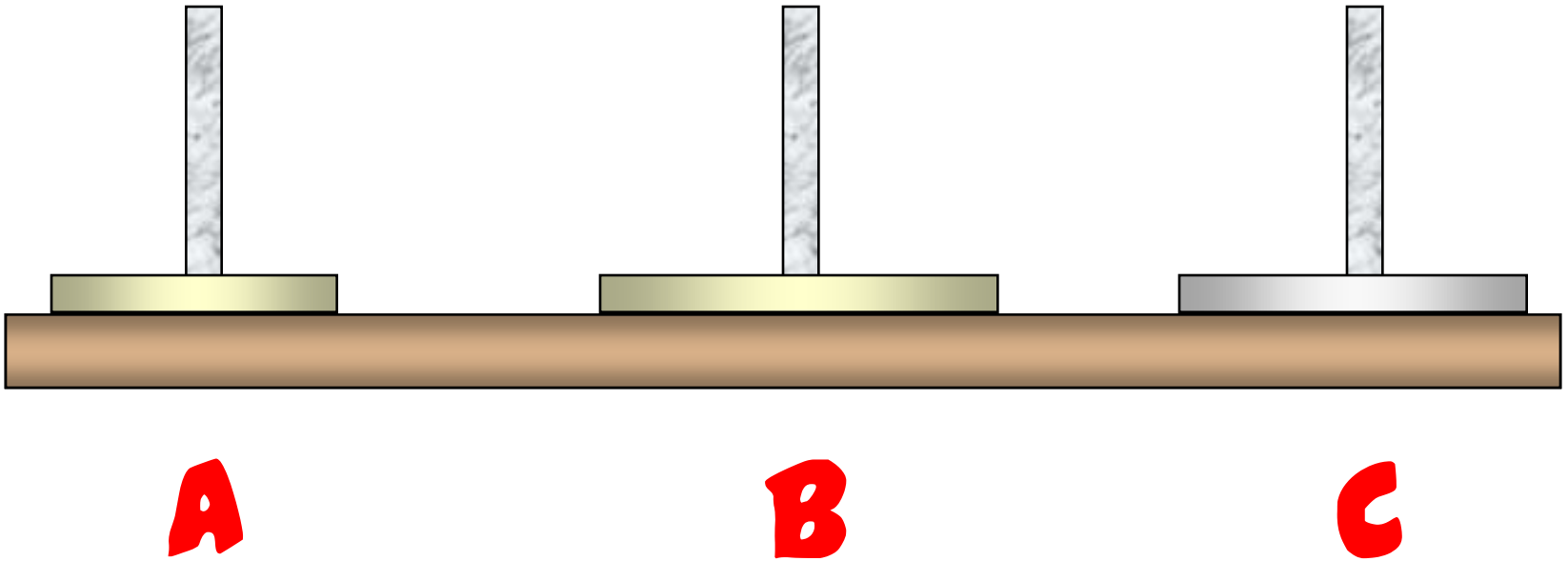

A
B
C

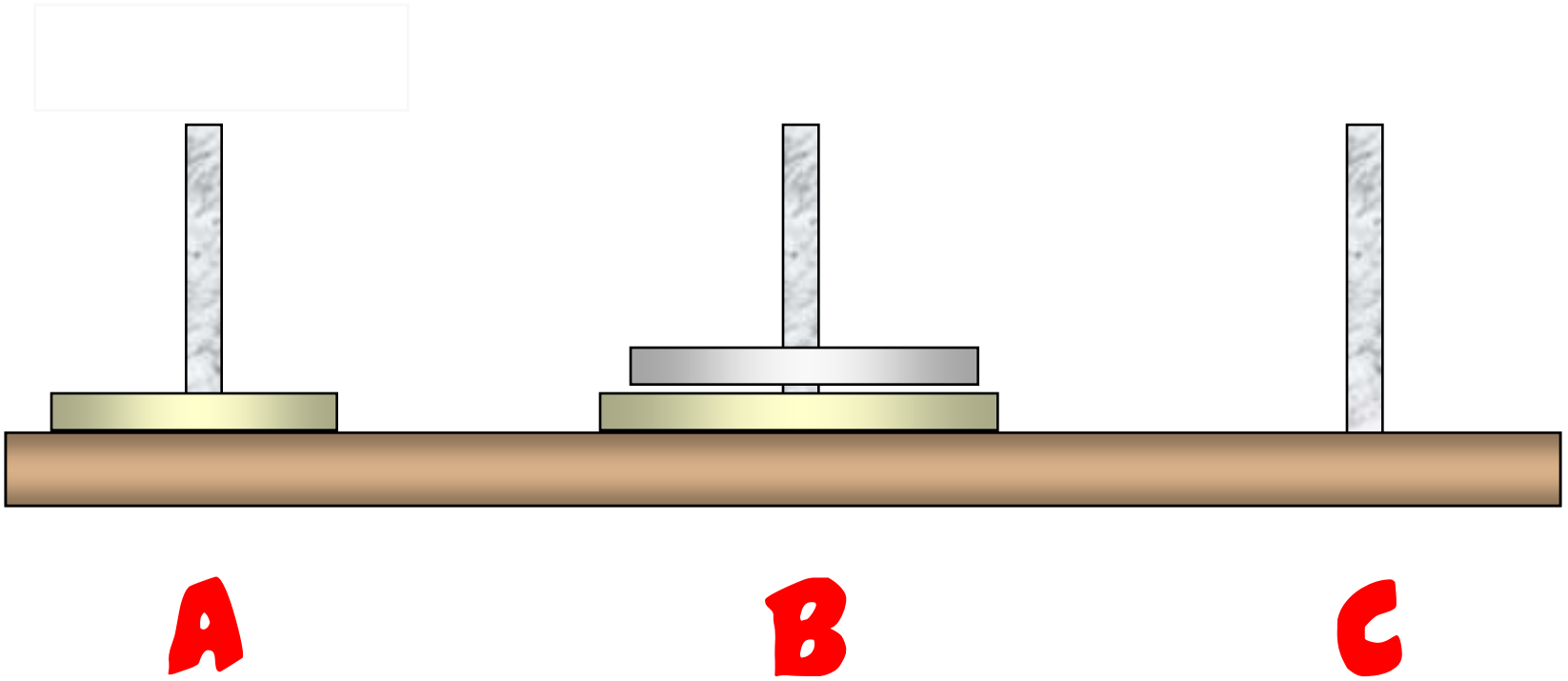

A
B
C

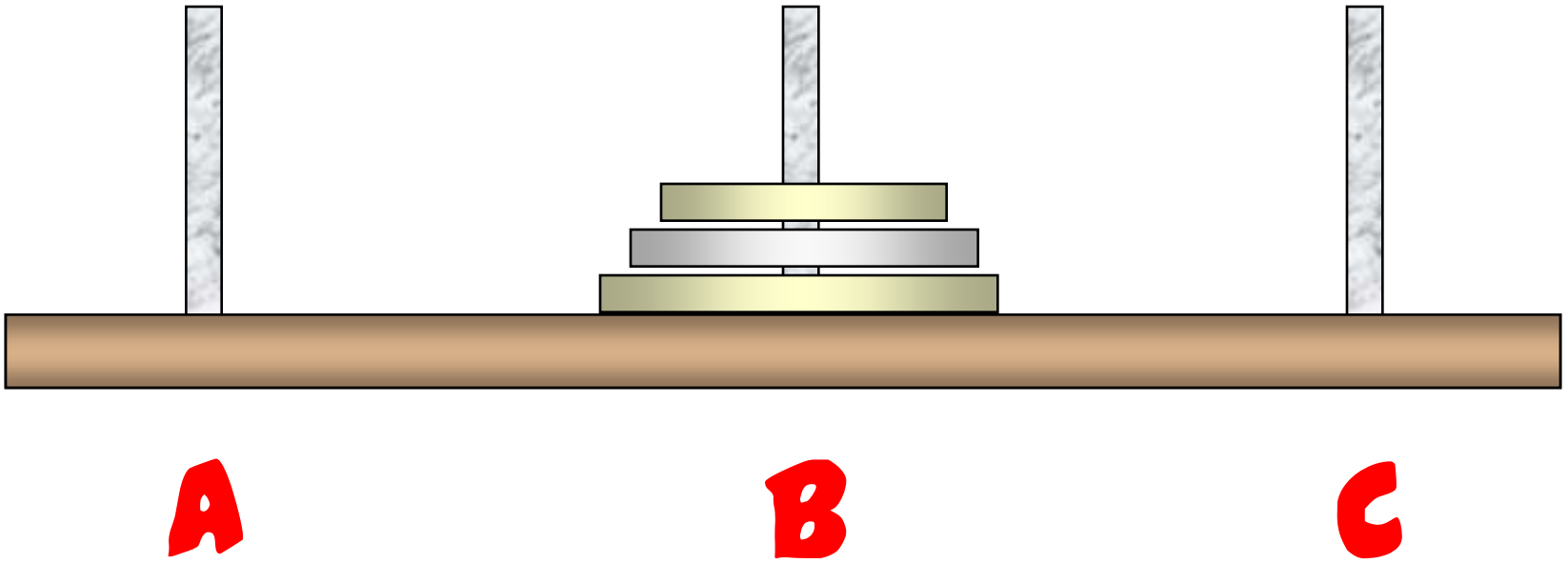

A
B
C


A
B
C


A
B
C


A
B
C


A
B
C


A
B
C

Recursive solution

- Move N - 1 discs from stack 1 to 3 with the help of stack 2.
- Move the N^{th} disc from 1 to 2
- Move N - 1 discs from stack 3 to 2 with the help of stack 1

Implementation in Prolog

```
hanoi(N)  :-   dohanoi(N, 1, 2, 3).
dohanoi(0, _ , _ , _ )     :-   !.
dohanoi(N, A, B, C)     :-   N_1 is N-1,
                                dohanoi(N_1, A, C, B),
                                moveit(A, B),
                                dohanoi(N_1, C, B, A).
moveit(F, T)   :-     write([move, F, -->, T]),  nl.
```

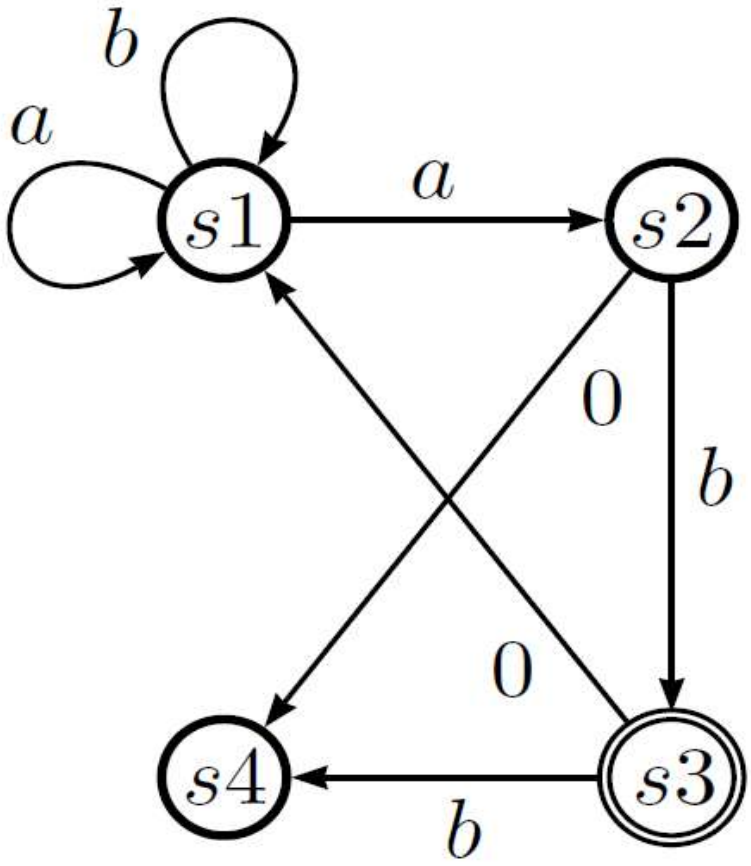
Output when n=3

```
?- hanoi(3).
[move,1,-->,2]
[move,1,-->,3]
[move,2,-->,3]
[move,1,-->,2]
[move,3,-->,1]
[move,3,-->,2]
[move,1,-->,2]
yes
```

Output when n=4

```
?- hanoi(4).
[move,1,-->,3]
[move,1,-->,2]
[move,3,-->,2]
[move,1,-->,3]
[move,2,-->,1]
[move,2,-->,3]
[move,1,-->,3]
[move,1,-->,2]
[move,3,-->,2]
[move,3,-->,1]
[move,2,-->,1]
[move,3,-->,2]
[move,1,-->,3]
[move,1,-->,2]
[move,3,-->,2]
yes
```

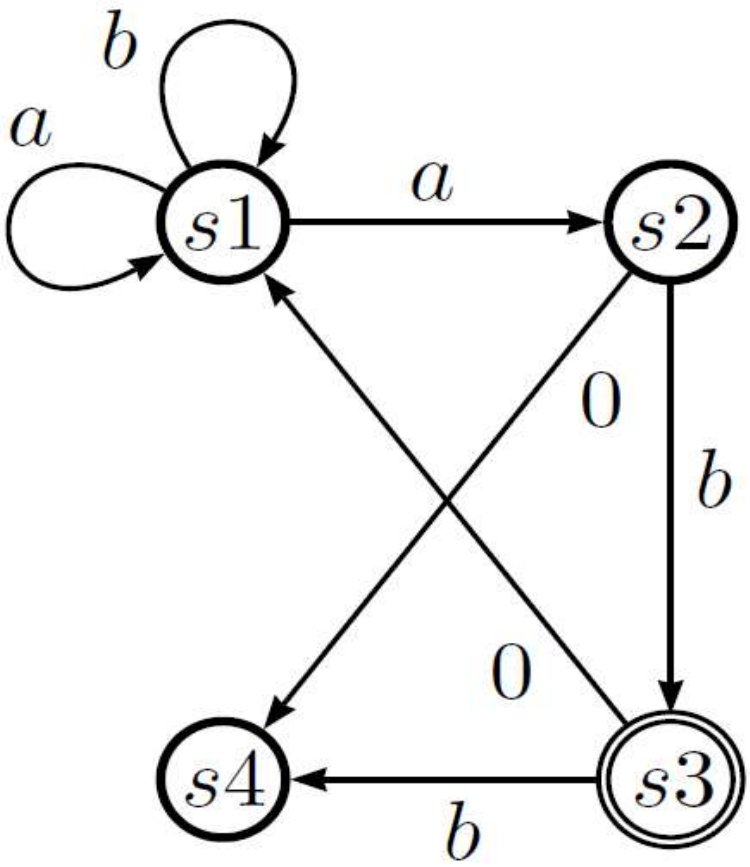
Non-deterministic finite state automaton



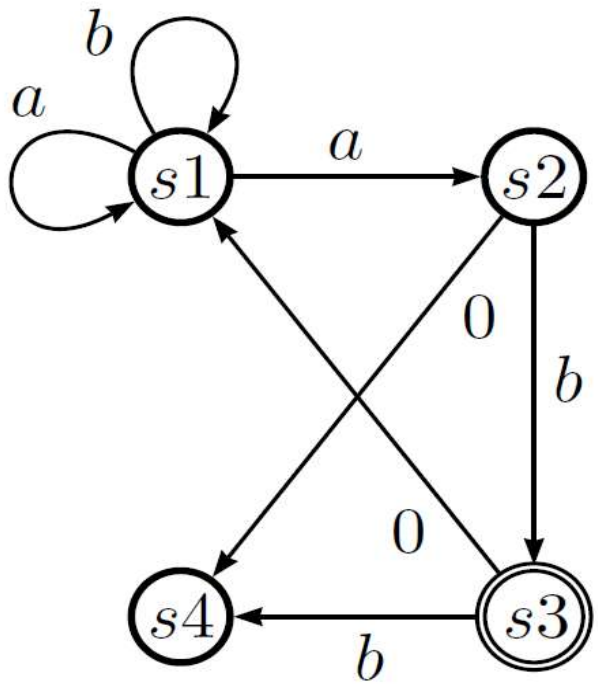
final(s3).

trans(s1,a,s1).
trans(s1,a,s2).
trans(s1,b,s1).
trans(s2,b,s3).
trans(s3,b,s4).

silent(s2,s4).
silent(s3,s1).



```
accepts(State,[])   :-   final(State).
accepts(State, [X|Rest]) :-   trans(State,X,State1),
                                  accepts(State1,Rest).
accepts(State, String)   :-   silent(State,State1),
                                  accepts(State1,String).
```

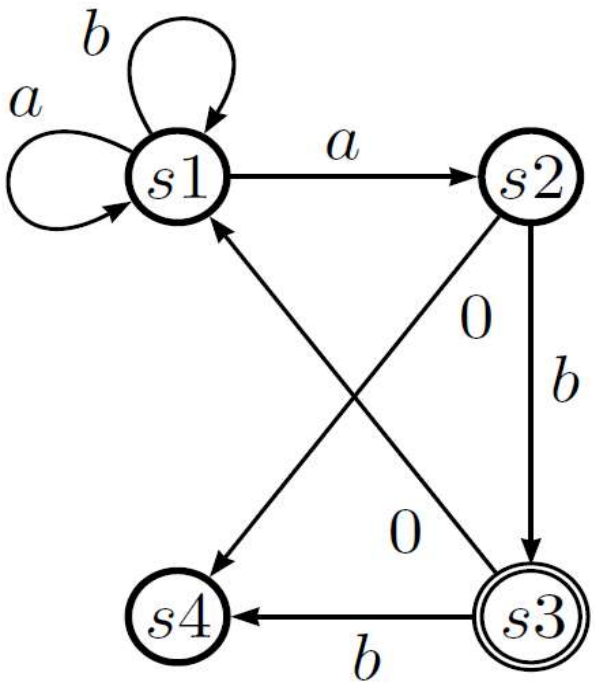


Questions we can ask

1) Whether a given string is accepted by the automaton

?- accepts(s1,[a,a,a,b]).

yes



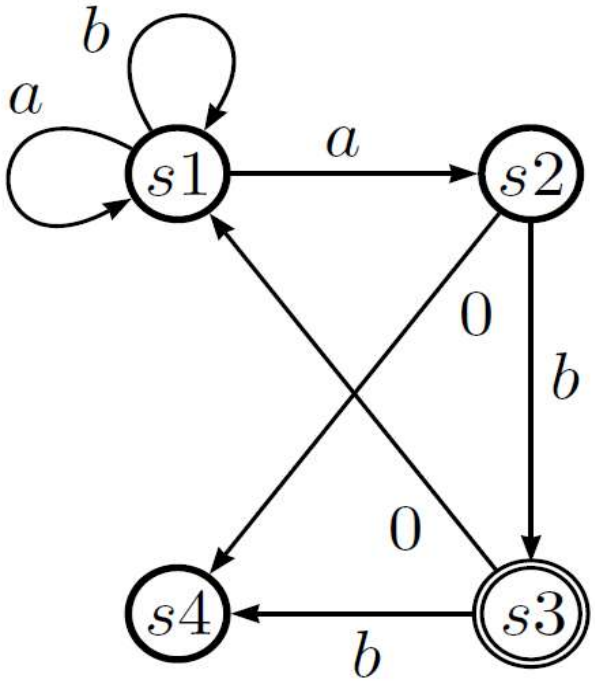
Questions we can ask (cont.)

2) What states can we start in if we accept a certain string?

?- accepts(S,[a,b]).

S = s1;

S = s3



Questions we can ask (cont.)

3) What are all strings of length 3 that can be accepted by the automaton?

?- accepts(s1,[X1,X2,X3]).

X1 = a

X2 = a

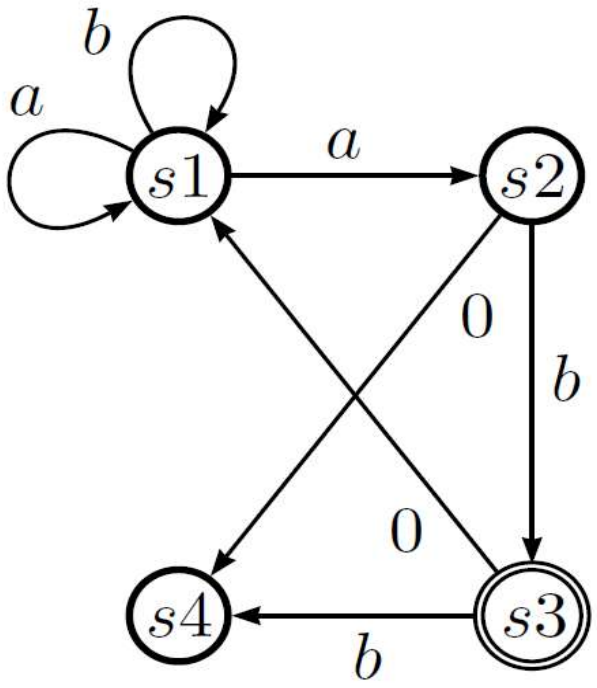
X3 = a;

X1 = b

X2 = a

X3 = b

yes



Conclusion

- Prolog is the most popular language of the logic programing languages.
- It is a declarative logic programming language
- It is goal based language, it has automatic backtracking and uses recursion.

Thank You