# Projects

1. Function composition
2. Symbolic Matrix Multiplication
3. Symbolic Matrix Inversion
4. Decision Tree creation
5. Solving Covering Problem
6. Solving Satisfiability Problem
7. Creating a Decision Diagram for POS expression
8. Semantics Database using graspe functions with very simple analogy reasoning
9. Society of ants - turtle-like robots
10. Robot arm handling rings on three sticks.

# Project #1

## Function composition

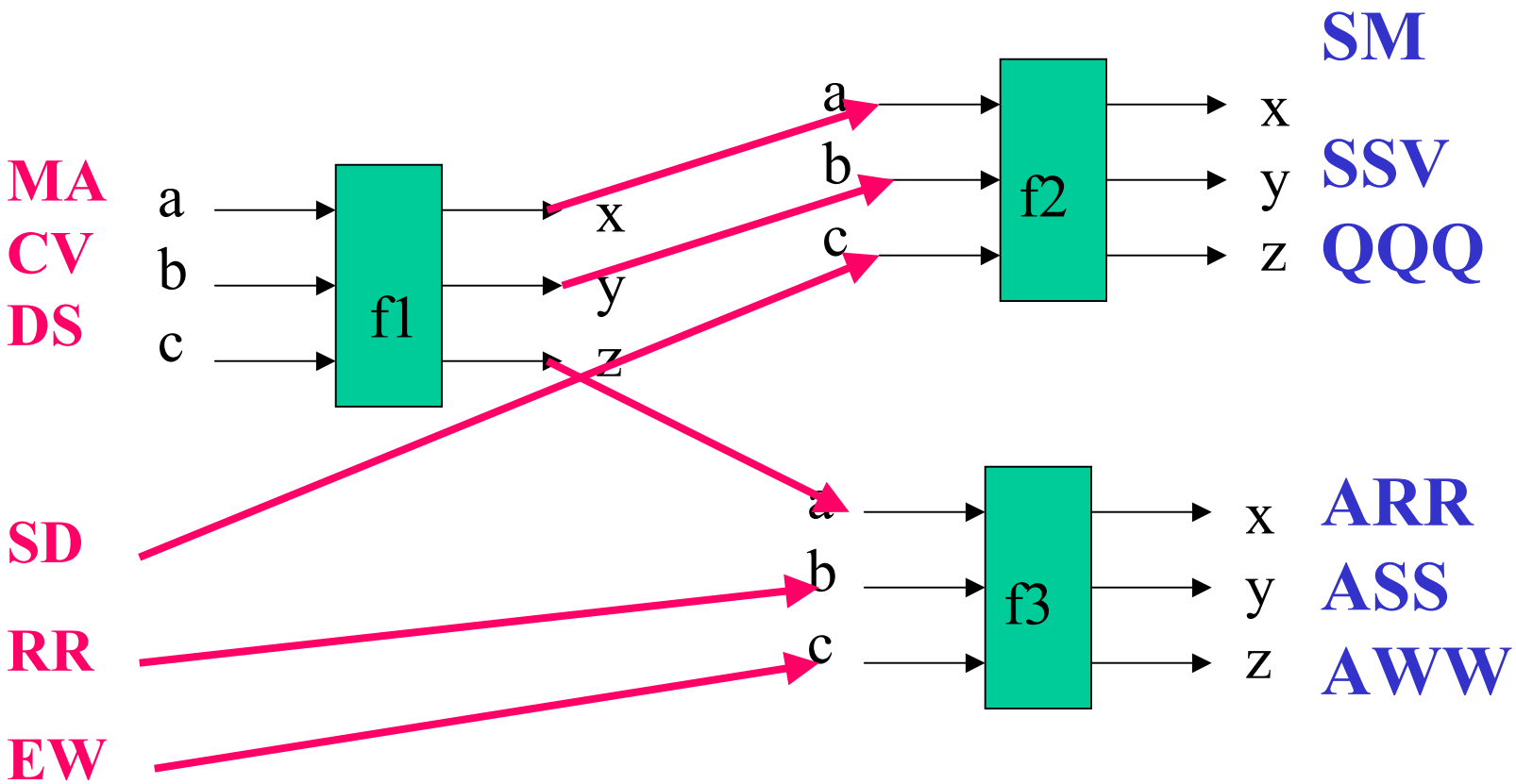
# Function composition 1

- Multi-input, multi-output function can be composed from logic elements, comparator and arithmetic elements. The function can be a mixture of binary and multi-valued signals:
  - Logic gates are NOT, OR, AND, EXOR and MUX
  - Arithmetic gates are +, * and -
  - Comparison gates are > and =
  - the number of logic values is arbitrary, but declared for every signal
- Such function is described as the following Lisp data structure

```
(f1 (type MOD0)) (f2 (type MOD0)) (f3 (type MOD0))
(type MOD0 (inputs a b c) (outputs x y z))
    (MOD0   (x = (or (and b c) (not a)))
            (y = (mux a  b (or c (not b))))
            (z = (a > (= b (not c)))))
```

- Create a Lisp program that:
  - composes such functions
  - displays them in form of truth tables

# Function composition 2




( ( (x f1) = (a f2)) ( (y f1) = (b f2)) ( (z f1) = (a f3))

( (SD IN) = (c f2)) ( (RR IN) = (b f3)) ((EW IN) = (c f3))

((x f3) = (SM OU)) ((y f3) = (SSV OU)) ….

# Function composition 3

- Ideas:

- Use function EVAL or similar to evaluate (simulate) the circuit

- Do loop through all possible values using a binary counter with as many bits as input variables

- Print in a nice clear format

- There should be two forms of function description for prettyprinting:

    - truth tables

    - expressions

- Use prettyprint function from Lisp system, do not write it.

- I recommend to use function substitute or similar

- This project can be extended to a general-purpose design and analysis tool reversible logic and thus publishable. There are no any computer tools like this.

# Project #2

**Symbolic Matrix multiplication**

# Symbolic Matrix multiplication

- Given is a n*n matrix A whose entries are symbolic expressions that use variables and operators +,-,* and /.
- Given is a n*n matrix B whose entries are symbolic expressions that use variables and operators +,-,* and /.
- Write a program that will find matrix C=A*B
- This program can be useful next quarter in the class to solve the kinematics problems for a robot arm.

# Project #3

**Symbolic Matrix inversion**

# Symbolic Matrix Inversion

- Given is a 3*3 matrix A whose entries are symbolic expressions that use variables and operators +,-,* and /.
- Write a program that will find inverse matrix $A^{-1}$ such that $A^{-1} * A = 1$, where 1 is a unitary matrix
- This program can be useful next quarter in the class to solve the inverse kinematics problems for various robots.

# Project #4

## Decision Tree Creation

# Decision Tree Creation

•Given is a table of objects and properties. A cross means that object Oi has a property Pj

| Object \ Property | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| O1 | X | | X | | | |
| O2 | | X | X | | | |
| O3 | | | X | X | X | |
| O4 | X | | X | | X | X |
| O5 | | X | | X | | X |

•You ask sequentially questions if an object has some property Pi, i=1…6

•Your goal is to find the which object are you dealing with

•This way your every sequence of questions creates a branch in a decision tree

•The variables along every path can have arbitrary order (free tree)

•Create the tree that has the minimum number of nodes, or the quasi-minimum number of nodes and that separates all objects that are distinguishable (different rows) to separate leaf nodes of the tree.

•Of course, your program should be applicable to arbitrary table

# Explanation of recursion

| Object \ Property | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| O1 | X | | X | | | |
| O2 | | X | X | | | |
| O3 | | | X | X | X | |
| O4 | X | | X | | X | X |
| O5 | | X | | X | | X |

Has property P2?

Objects O2,O5

yes

Objects O1,O3,O4

no

| | P3 | P4 | P6 |
|---|---|---|---|
| O1 | X | | |
| O5 | | X | X |

Has property P3?

yes → **O1**

no → **O2**

Leaf nodes separate objects

# Is it useful?

- This is one of the most famous AI applications in recognition, control, and problem solving.
- May be useful for voice recognition, situation recognition of face recognition for our mobile robots guards in FAB building. Create for test the table of features with objects as humans from ECE department and properties as they features - has glasses? Has a red nose? Is he bald? Etc.
- If you are ambitious you can solve this problem with multivalued variables.
- If you are very ambitious, you can assume continuous variables.

# Project #5

## Covering Problem

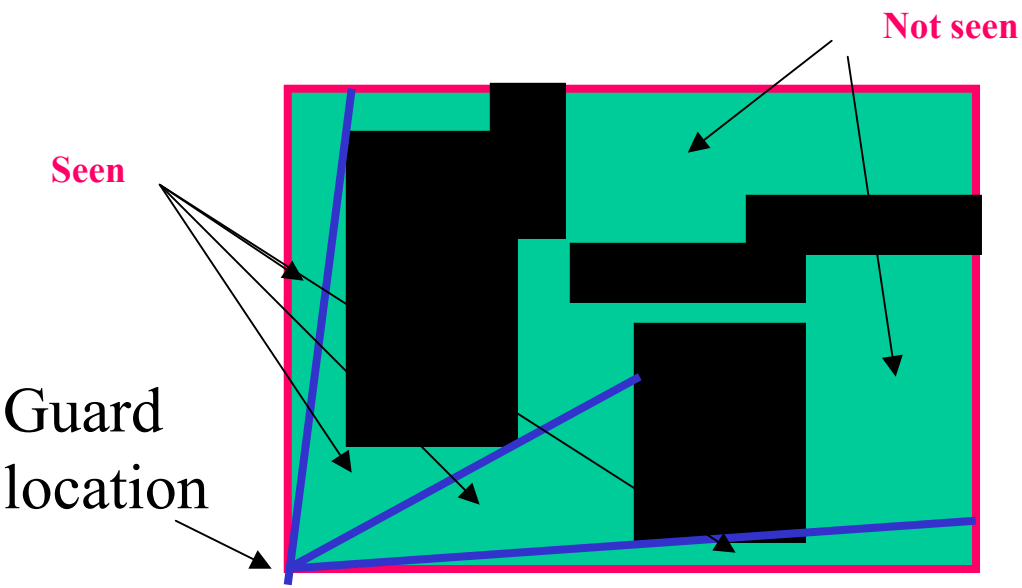
# Covering Problem

| Object \ Property | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| O1 | X | | X | | | |
| O2 | | X | X | | | |
| O3 | | | X | X | X | |
| O4 | X | | X | | X | X |
| O5 | | X | | X | | X |

- Given is the Object/Property table.
- Select the smallest set of objects that each property is covered by at least one object. For instance objects O4 and O5 cover all properties.
- Find all solutions with minimum number of objects
- Of course, your solution should be for arbitrary table

# Is it useful?

- This is one of the most famous AI applications in recognition, control, and problem solving.
- May be useful for object recognition, guard problem, self-test of a robot and many more
- If you are ambitious you can solve the guard problem by reducing it to covering  problem.




Each guard controls the area that he sees.

We need as many guards as necessary to control every green area

How to find the minimum number of guards?

# Project #6

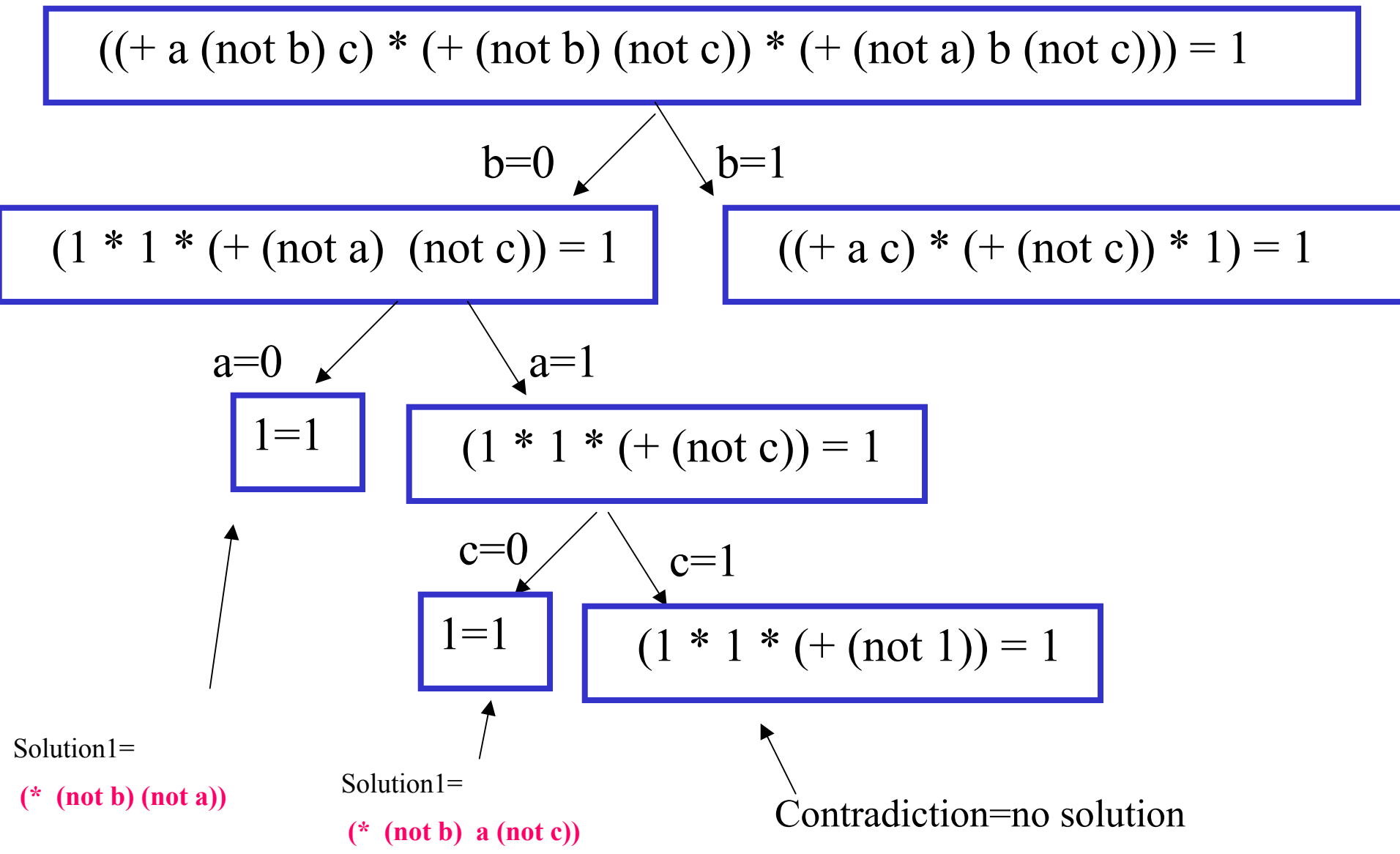
## Satisfiability Problem

# Satisfiability Problem

- Given is a Boolean Formula in the form of Product of Sums (POS) of Binary variables and their negations
- For instance

  ((+ a (not b) c) * (+ (not b) (not c)) * …. Etc
- Every variable can be negated and not negated at the same time in various sums
- The number of literals (variable or negated variable) in the sum is arbitrary
- The number of sums is arbitrary
- Find all product of literals that satisfy the POS form or prove that there is no solution.
- ***Remember about trees and recursion***.

# Satisfiability Problem



# Satisfiability Problem

- It is important to select good branching variables,
- the tree is free (arbitrary order in every branch) or ordered, your choice
- use recursion
- Even with random or arbitrary choice of variable orders, you can find a good solution
- This is a very important practically problem.
- In theory, every NP-complete problem can be reduced to it.
- People in Bell Labs are designing special computer to solve this problem. Do you have an idea how to do this?

# Project #7

## Creating a Decision Diagram for POS expression

- Given is a Boolean Formula in the form of Product of Sums (POS) of Binary variables and their negations
- For instance

  ((+ a (not b) c) * (+ (not b) (not c)) * …. Etc
- Every variable can be negated and not negated at the same time in various sums
- The number of literals (variable or negated variable) in the sum is arbitrary
- The number of sums is arbitrary
- Find the tree for given order of variables (not free, ordered)
- Combine recursively every two nodes that represent the same functions until no more such nodes exist.
- What you get is the famous Binary Decision Diagram for your POS
- Write a program that uses above method to create the Binary Decision Diagram for given order of expansion variables. (do not look for them, just assume sorted list of variables)
  - ***Observe the similarity with the previous problem***
  - ***Use it.***
  - ***Remember about trees and recursion***.

# Project #8

**Semantics Database using graspe functions with very simple analogy reasoning**

# Semantics Database using graspe functions with very simple analogy reasoning

- Given are three figures A, B, C and three figures D1,D2 and D3.
- You have to solve the intelligence test (reasoning by analogy)
- A is to B as C is to what ? You should select only from the set D1, D2 and D3
- **Example**

Figure A

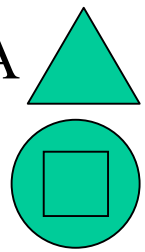


Figure B

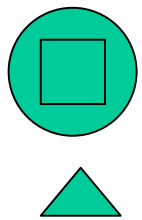


Figure C

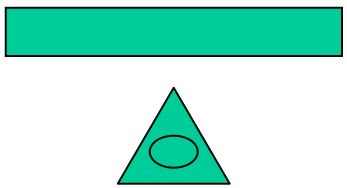


Figure D1

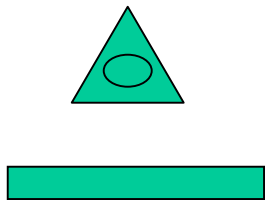


Figure D2

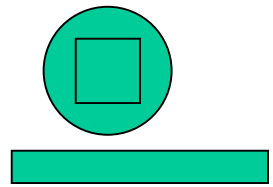


Figure D3

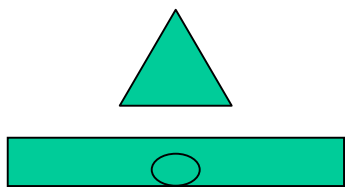
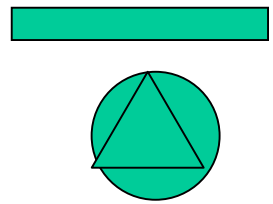


Figure D4



## Semantics Database using graspe functions with very simple analogy reasoning

- **Represent each figure as a semantic network.**
- **You can use functions GRASPE from my book to process the semantic network or define your own functions for this task.**

- **HINTS**
  - **I suggest to describe the transformation from A to B**
  - **Then apply the same transformation to C and compare the TRANSFORMED (C) with D1, D2 and D3.**
  - **There are other ways to solve this problem.**
- **Observation (just to give you more motivation)**
  - **We can use this method for robot reasoning by analogy**

# Project #9

**Society of ants - turtle-like robots**

# Society of ants - turtle-like robots

- Solve the homework 2 in its full formulation, it means with several “turtles” and with moving obstacles.
There are also piles of food
The “turtles” are now called “ants”.
In addition to what was in homework 2, ants try to find “food” and store it in their “homes”.
If two ants are close to a food, they collaborate to bring food to the store.
Single ant cannot bring the food. It has to find another ant.
Show the stages of the ants collaboration on your screen.

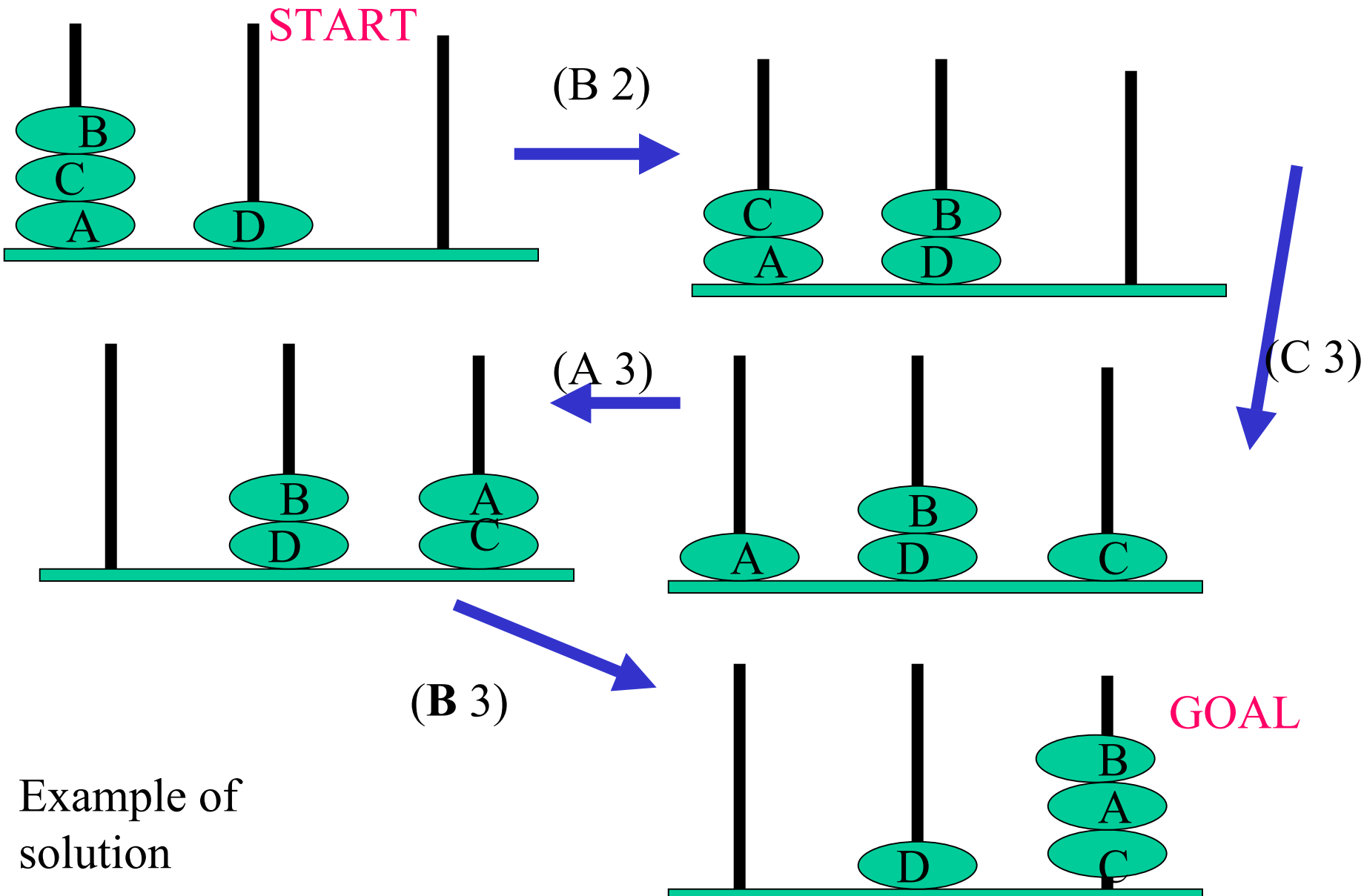
# Project #10

**Tower modifying Robot Arm**

# Tower modifying Robot Arm

- Given are three vertical sticks, 1, 2 and 3, from left to right.
- On each stick you can put one-by-one (like to a stack) an arbitrary number of rings, called A,B,C,…etc.
- Given is an start situation and goal situation.
- Find the sequence of moves to transform the start situation to the goal situation or prove that there is no solution, i.e. no sequence of moves from this start to this goal.
- Each move is described as (RINGi -> Stick j)
- for instance (A 1)(B 2) moves ring A to stick 1 and next ring B to stick 2
- Assume that each stick is long enough to accommodate all rings

# Tower modifying Robot Arm




Example of solution

# Tower modifying Robot Arm

- Remember that you build a building from a fundament and not from the roof, this is a very powerful heuristics
- You can use any type of search, but try to minimize the number of moves