# Artificial & Computational Intelligence

## AIML CLZG557

## M1 : Introduction

## &

## M2 : Problem Solving Agent using Search


Indumathi V

Guest Faculty,

BITS - WILP

**BITS** Pilani

Pilani Campus

# Course Plan

| | |
|---|---|
| M1 | Introduction to AI |
| M2 | Problem Solving Agent using Search |
| M3 | Game Playing |
| M4 | Knowledge Representation using Logics |
| M5 | Probabilistic Representation and Reasoning |
| M6 | Reasoning over time |
| M7 | Ethics in AI |

# Learning Objective

At the end of this class , students Should be able to:

1. Design problem solving agents
2. Create search tree for given problem
3. Apply uninformed search algorithms to the given problem
4. Compare performance of given algorithms in terms of completeness, optimality, time and space complexity
5. Differentiate for which scenario appropriate uninformed search technique is suitable and justify

# Dimensions of Task Environment

PEAS
S/W | H/W
3

**Sensor Based:**

- Observability : Full Vs Partial

**Action Based:**

- Dependency : Episodic Vs Sequential

**State Based:**

- No.of.State : Discrete Vs Continuous

**Agent Based:**

> Cardinality : Single Vs MultiAgent

**Action & State Based:**

- State Determinism : Deterministic Vs Stochastic | Strategic
- Change in Time : Static Vs Dynamic

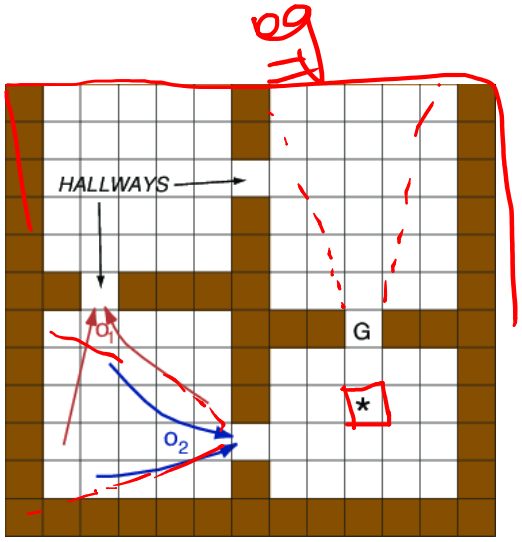
# Task Environment

A rational agent is built to solve a specific task. Each such task would then have a
different environment which we refer to as Task Environment

Based on the applicability of each technique for agent implementation its task environment design is determined by multiple dimension

**Sensor Based:**

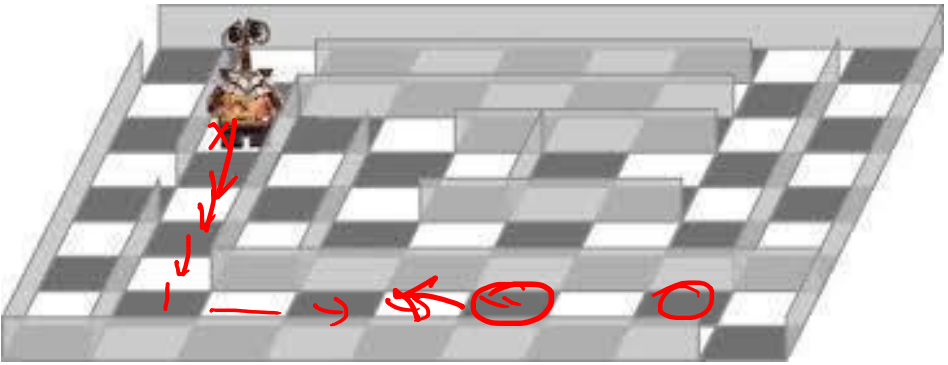
- Observability : Full Vs Partial



# Task Environment

**Action Based:**

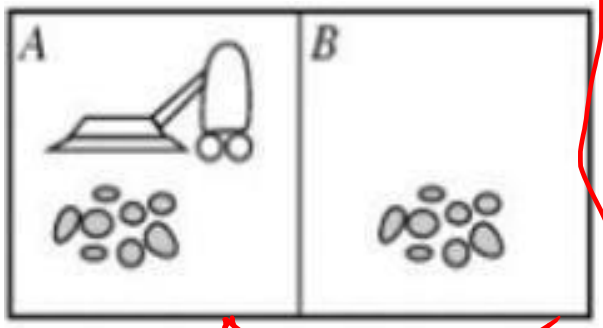
- Dependency : Episodic Vs Sequential

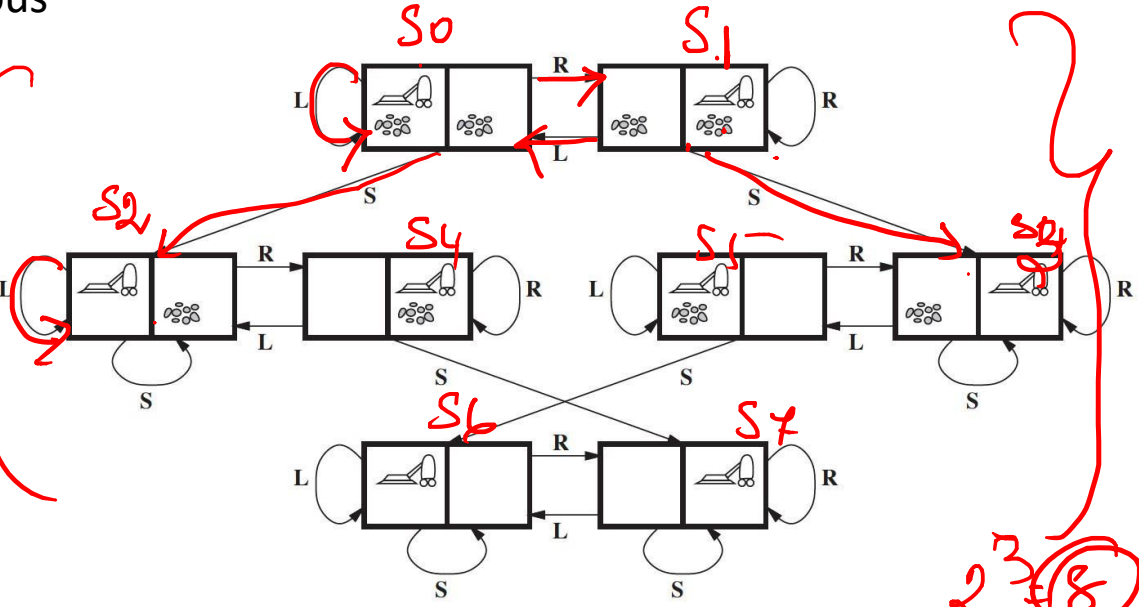


# Task Environment

**State Based:**

- No.of.State : **Discrete** Vs Continuous



# Task Environment

**State Based:**

- No.of.State : Discrete Vs Continuous

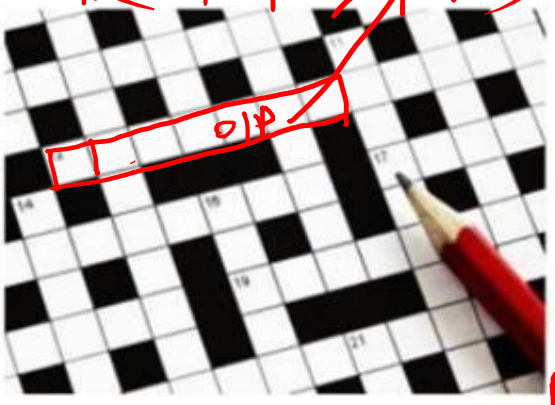
VS.

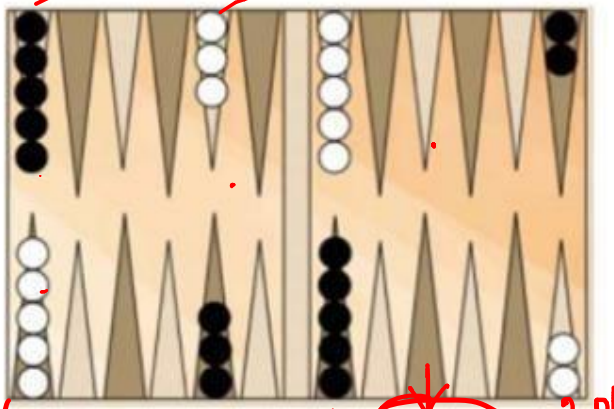
# Task Environment

**Action & State Based:**

- State Determinism : Deterministic Vs Stochastic | Strategic

(If the environment is deterministic except for the actions of other agents, then the environment is strategic)

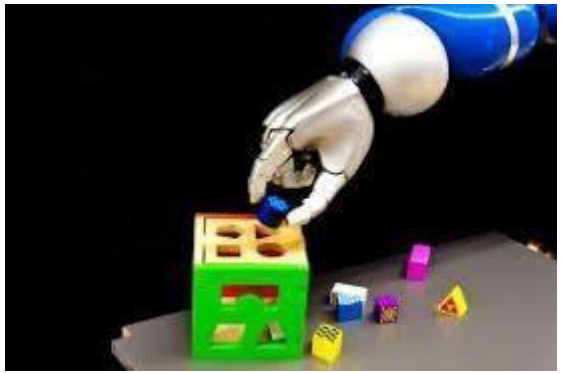


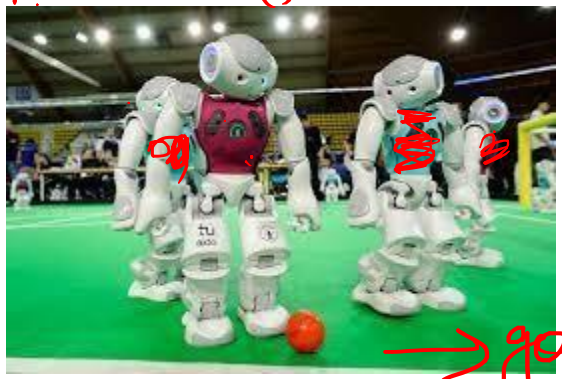


# Task Environment

**Agent Based:**

> Cardinality : Single Vs MultiAgent

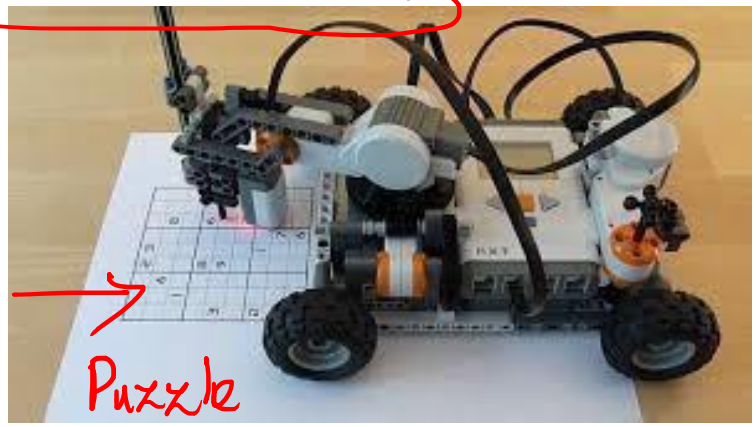




# Task Environment

**Action & State Based:**

- Change in Time : Static Vs Dynamic
- (The environment is semi dynamic if the environment itself does not change with the passage of time but the agent's performance score does)


1st Dynamic

2 hrs → 4st hour → 90%
2nd hour → "

eg 1 Static

90% attentiveness → 90%
" → 95-

↓ Student Agent score

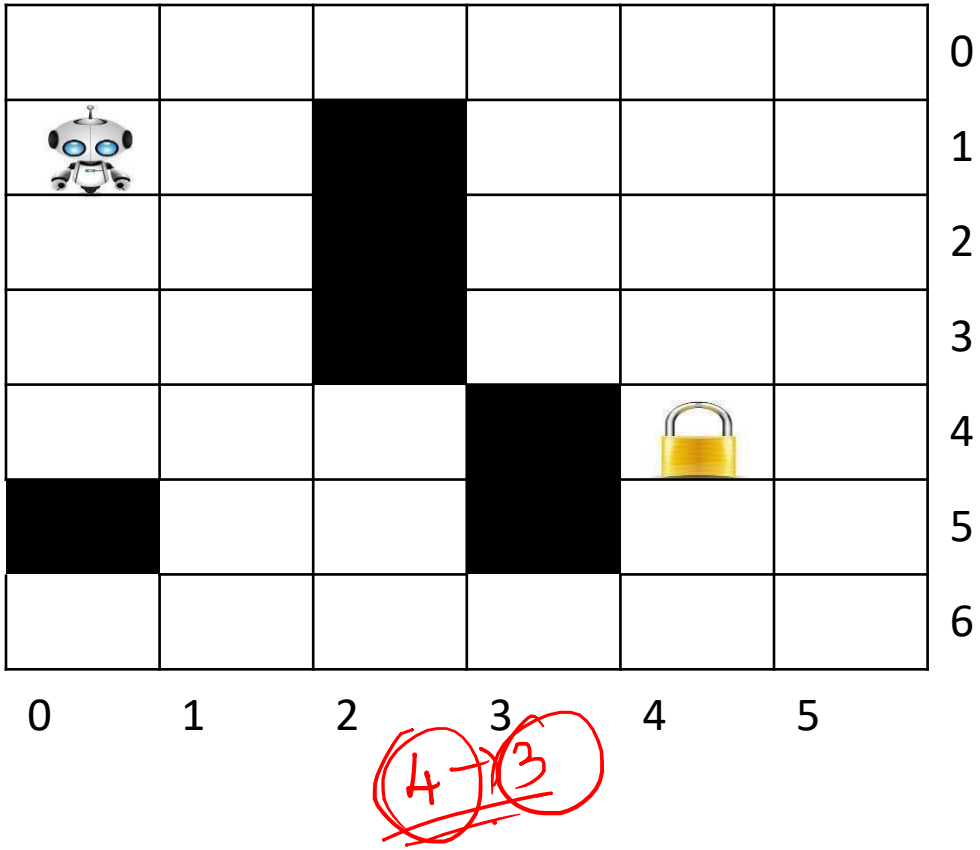
60%

Semi dynamic

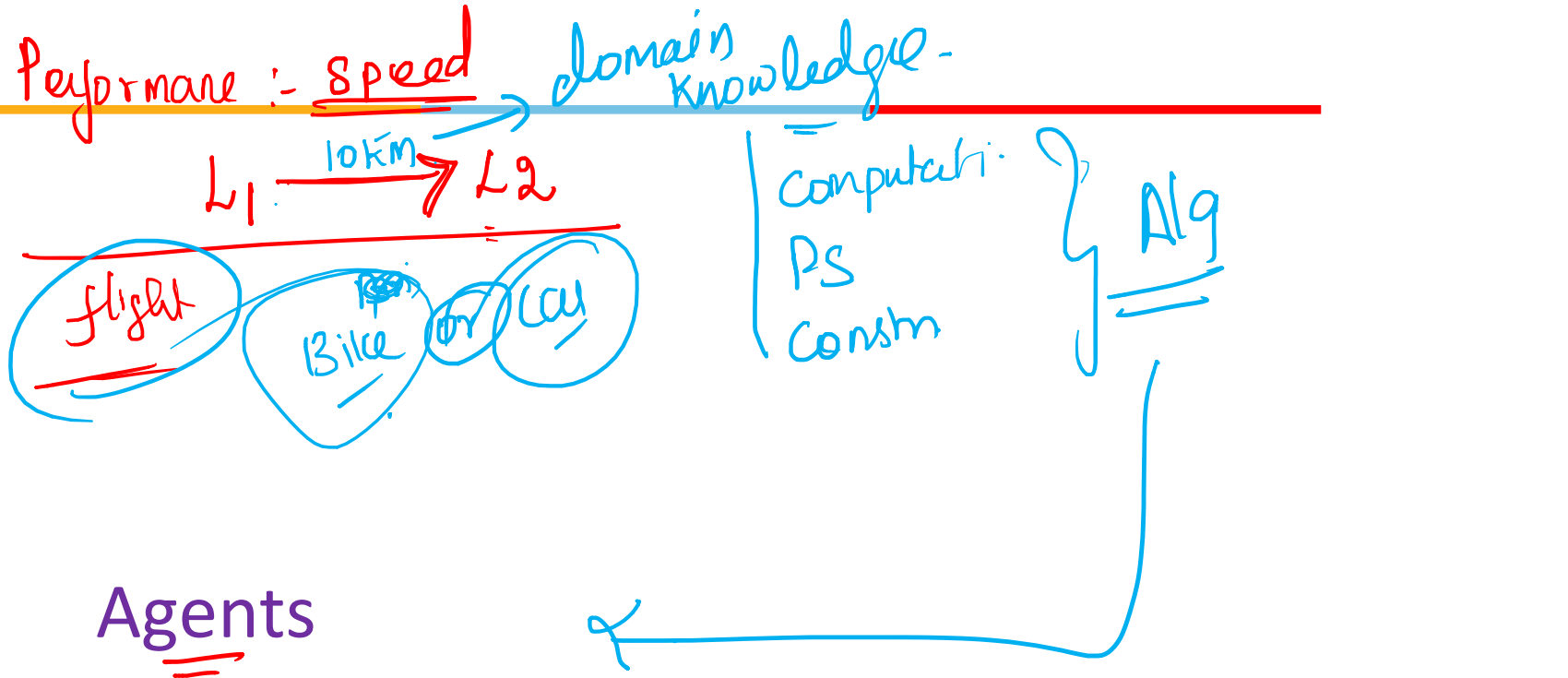
# Task Environment

| Task Environment | Fully vs Partially Observable | Single vs Multi-Agent | Deterministic vs Stochastic | Episodic vs Sequential | Static vs Dynamic | Discrete vs Continuous |
|---|---|---|---|---|---|---|
| Medical diagnosis system | Partially | Single | Stochastic | Sequential | Dynamic | Continuous |
| Satellite Image Analysis System | Fully | Single | Deterministic | Episodic | Static | Continuous |
| Interactive English tutor | Partially | Multi | Stochastic | Sequential | Dynamic | Discrete |

# Path finding Robot - Lab Example




| Agent |
| --- |
| **Observability** |
| F/P : Full |
| **No.of.Agents** |
| Single |
| **No.of.States** |
| Discrete |
| **Determinism** |
| Deterministic |
| **Dynamicity** |
| Static |
| **Output Dependency** |
| Sequential |

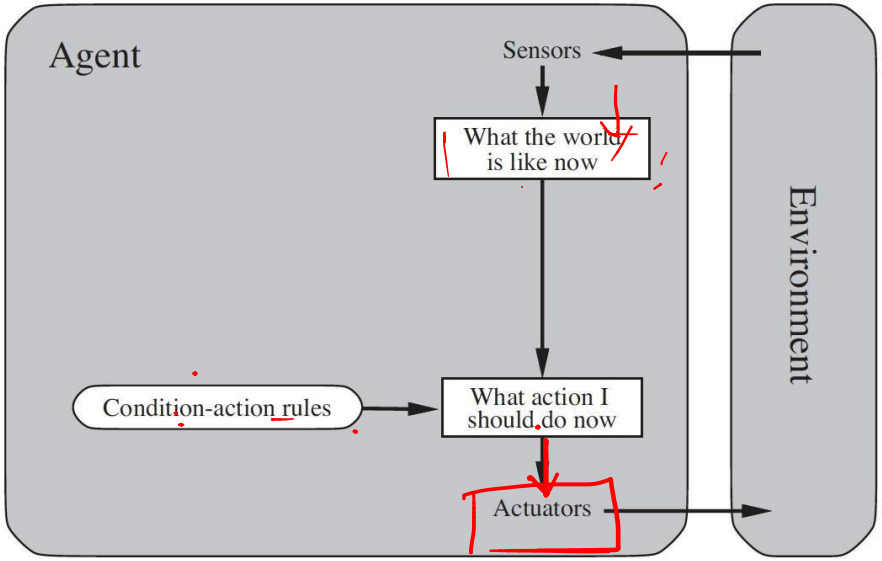



# Agents Architectures

# Agent Architectures

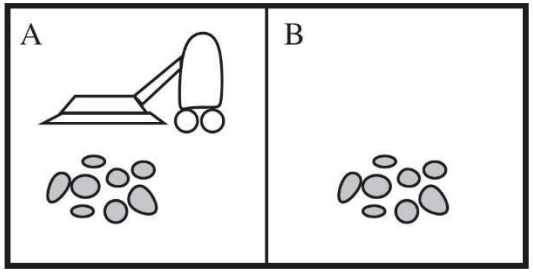
## Reflex Agent

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    persistent: rules, a set of condition–action rules
    state←INTERPRET-INPUT(percept)
    rule←RULE-MATCH(state, rules)
    action ←rule.ACTION
    return action
```




```
function REFLEX-VACUUM-AGENT( [location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

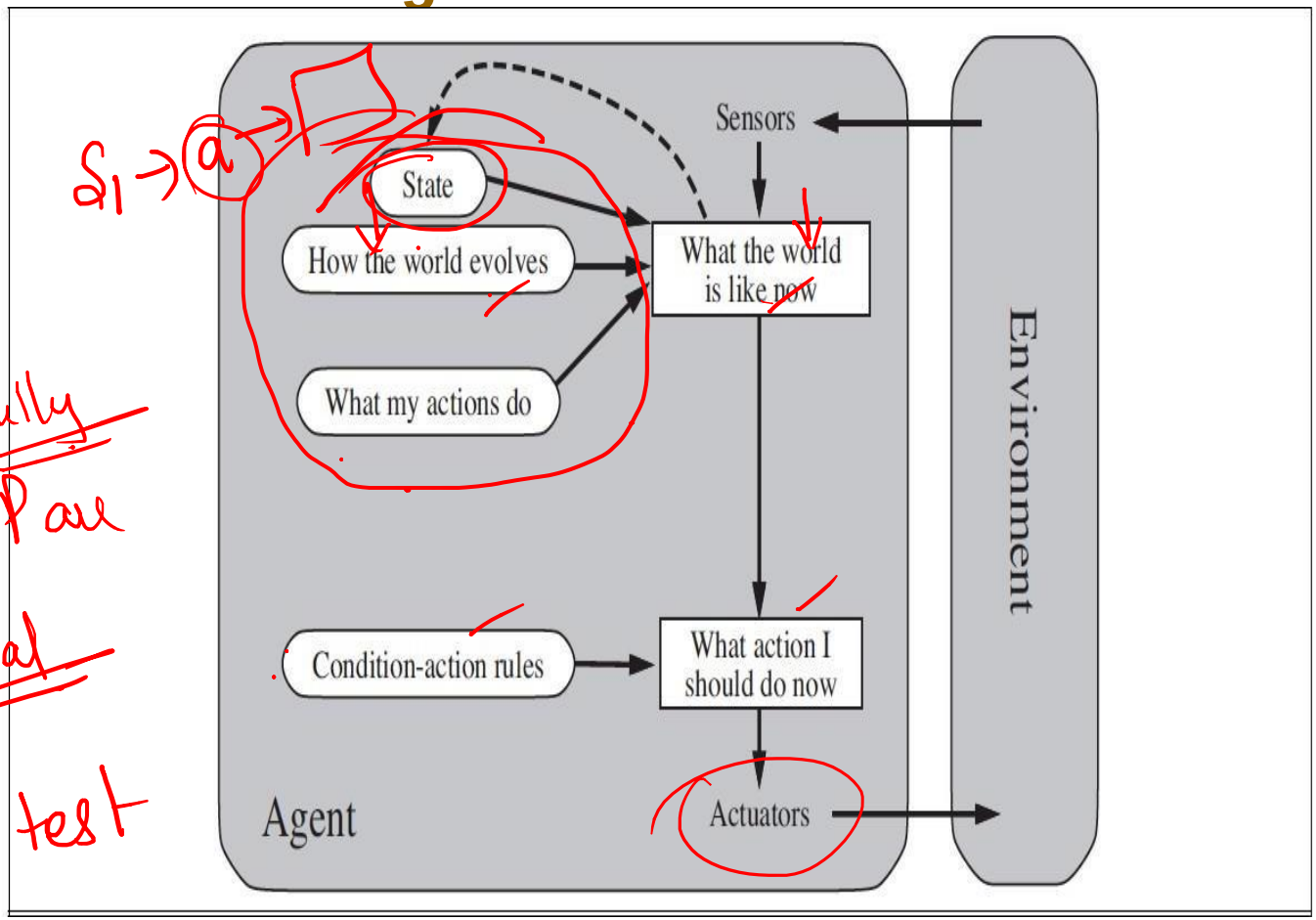
Simple Reflex



# Agent Architectures

## Model based Agent

- Simple Reflex
- Model Based Agents



# Agent Architectures

## Model based Agent

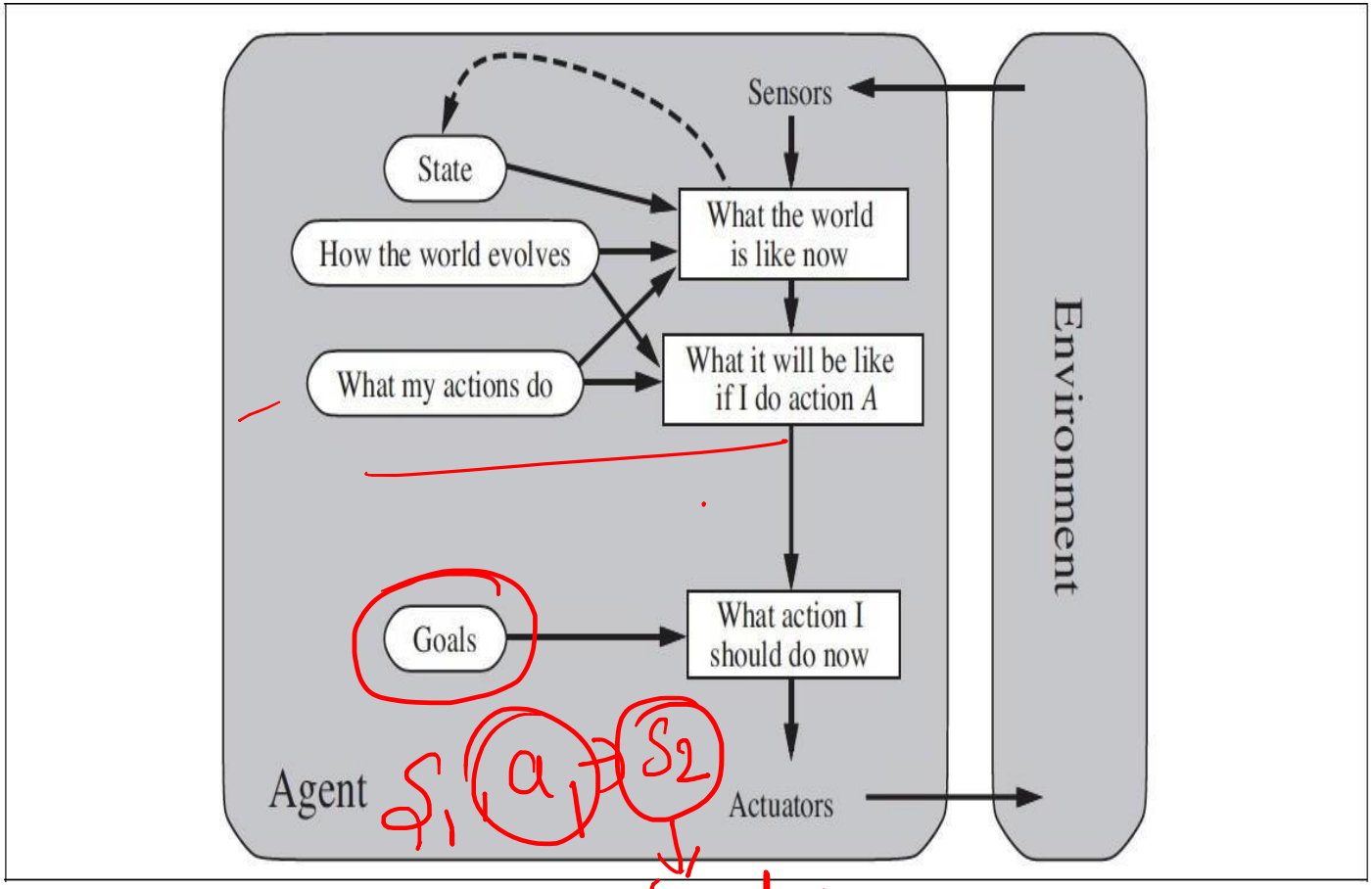
**function** MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action

**persistent:** *state*, the agent's current conception of the world state
*transition model*, a description of how the next state depends on the current state and action
*sensor model* , a description of how the current world state is reflected in the agent's percepts
*rules*, a set of condition–action rules
*action*, the most recent action, initially none

*state* ← UPDATE-STATE(*state, action, percept, transition model, sensor model* )
*rule* ← RULE-MATCH(*state*, *rules*)
*action* ← *rule*.ACTION
**return** *action*

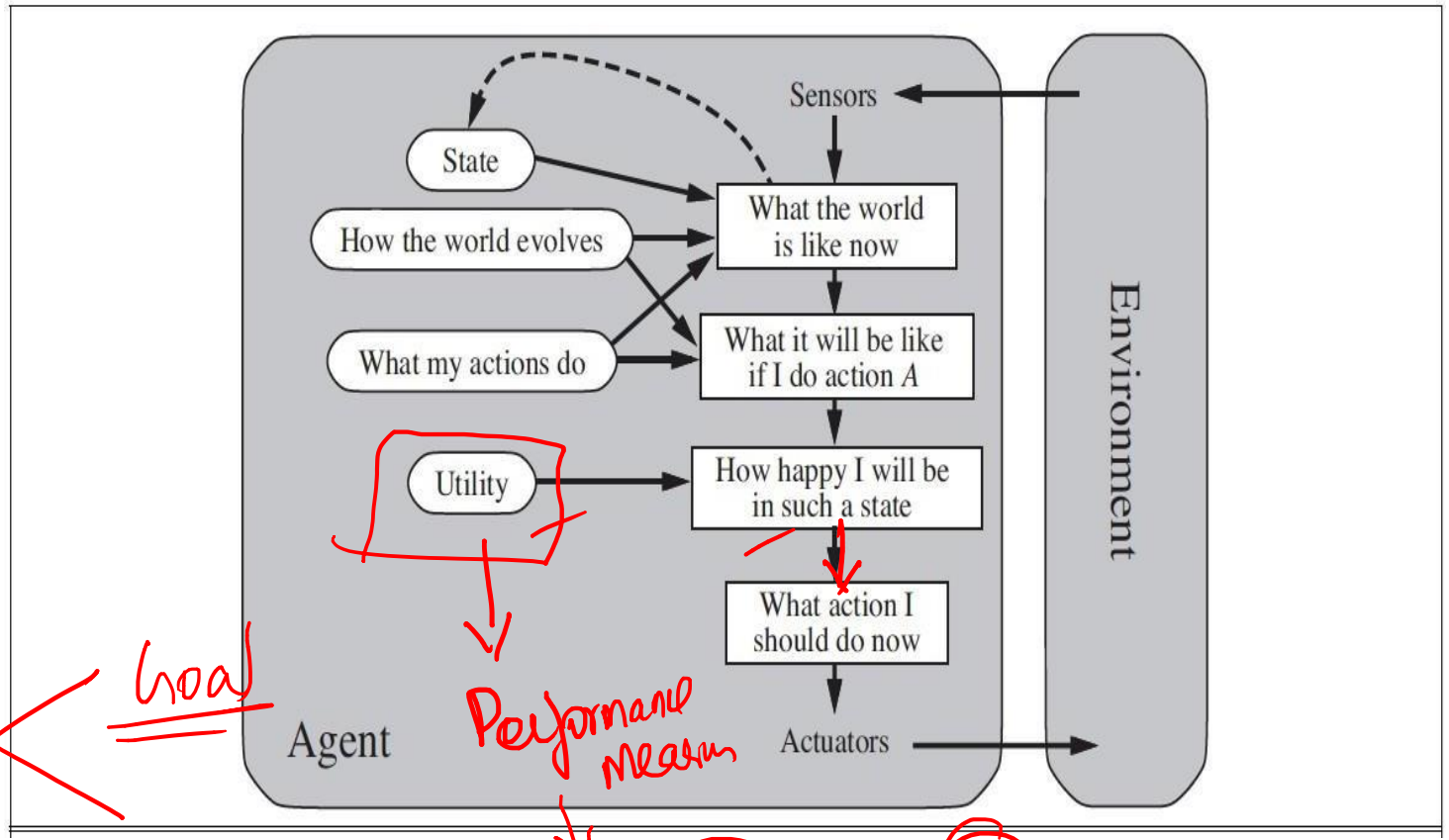
# Agent Architectures

- Simple Reflex Agents
- Model Based Agents
- Goal Based Agents



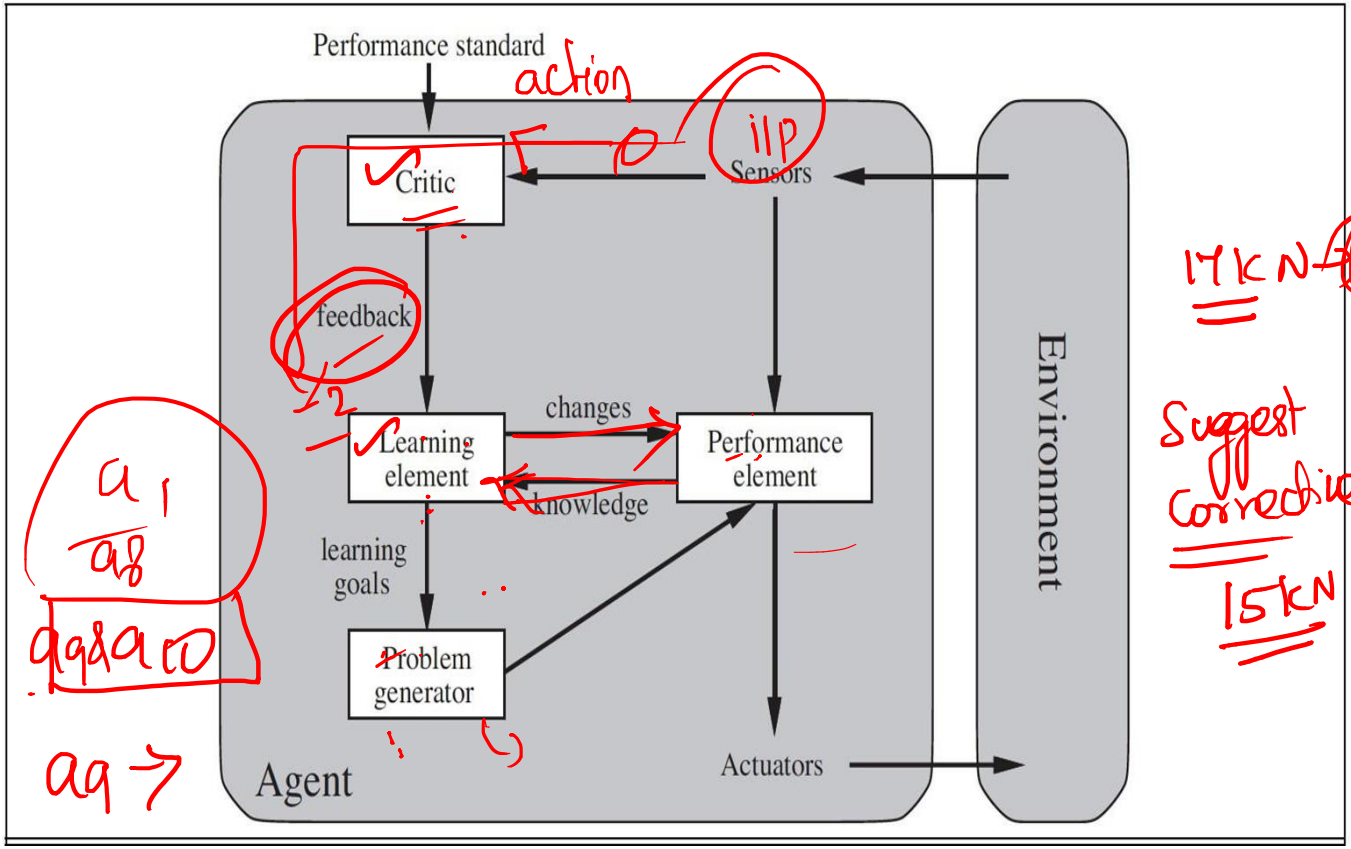
# Agent Architectures

- Simple Reflex Agents
- Model Based Agents
- Goal Based
- Utility Based



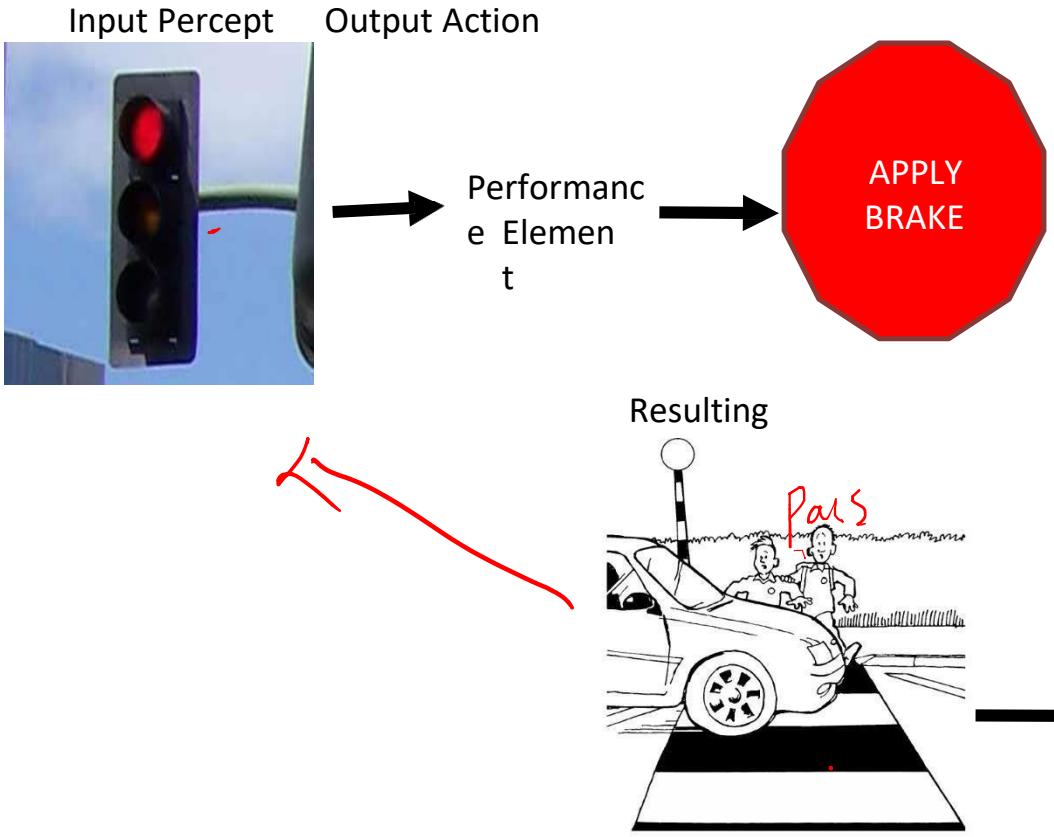
# Agent Architectures

- Simple Reflex Agents
- Model Based Agents
- Goal Based Agents
- Utility Based Agents
- Learning Agents



# Roleof Learning

Agents that improve their performance by learning from their own experiences



# Role of Learning

Input Percept

Possible Actions

Brake
Change Gear to Lower
Change Gear to Higher
Accelerate
Steer left
Steer right

Random

Selected Action

Change Gear to Lower

unexplored action

# Role of Learning

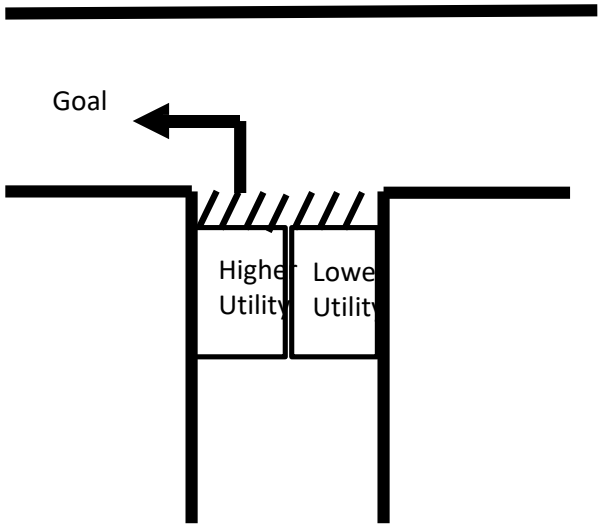
Performance Element – Takes decision on action based on percept

$$f(red\ signal,\ distance) = 15k\ N\ brake$$

$$distance = f'(percept\ sequence)$$

$$f(percepts, distance, raining)$$

- $f(state_0, actionA) = 0.83$,
- $f(state_0, actionB) = 0.45$

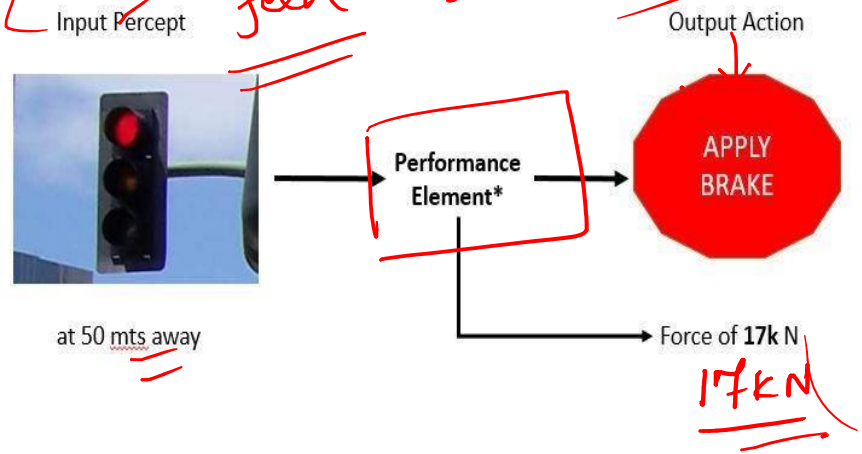


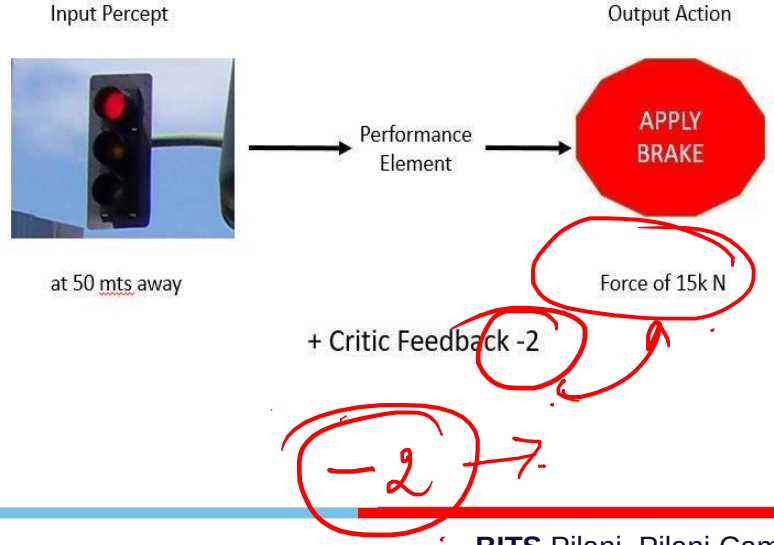
# Roleof Learning

Critic – Provides feedback on the actions taken

Learning :

Supervised Vs Unsupervised Vs Reinforcement



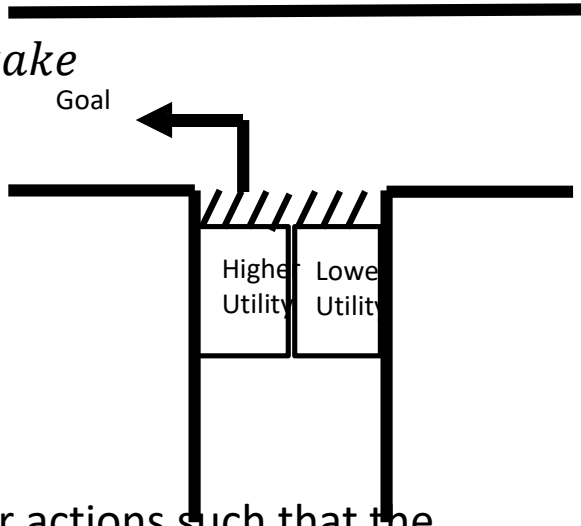
# Role of Learning

Performance Element – Takes decision on action based on percept

$$f(red\ signal,\ distance) = 15k\ N\ brake$$

$$distance = f'(percept\ sequence)$$

$$(f\ percepts, distance, raining)$$




- $(f\ state_0, action A) = 0.83,$
- $(f\ state_0, action B) = 0.45$

Learning Element – Make the performance element select better actions such that the utility function is optimized

Critic – Provides feedback on the actions taken

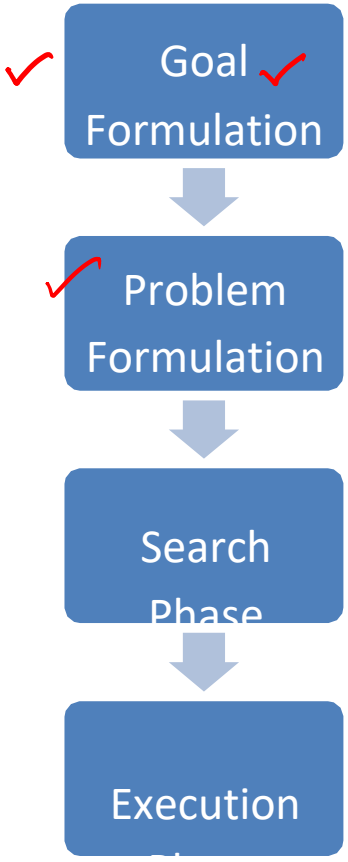
Problem Generator – Make the Performance Element select sub-optimal actions such that you would learn from unseen actions

innovate achieve lead

# Problem Formulation

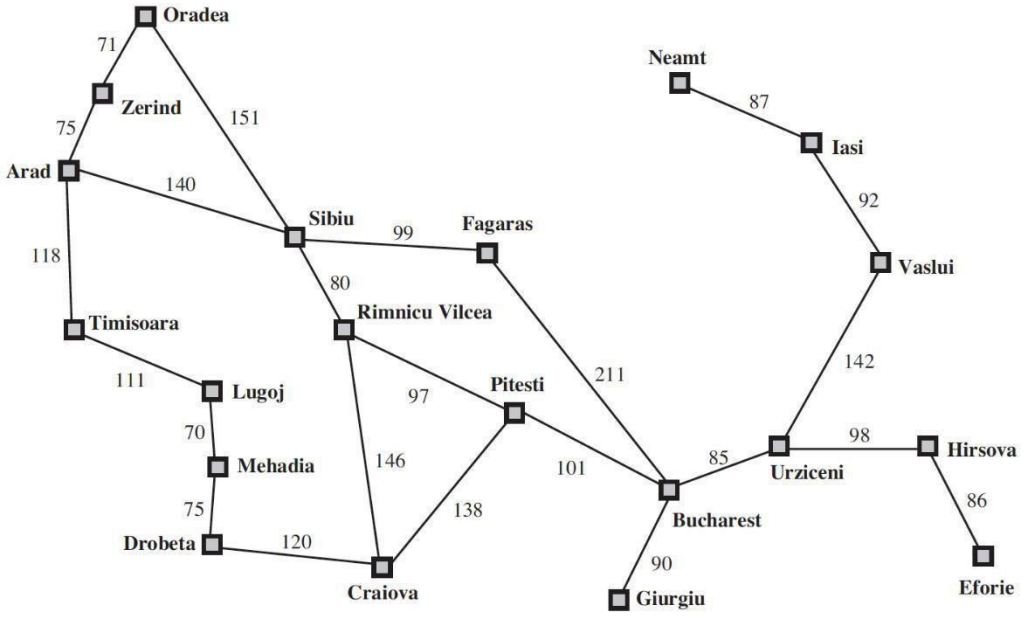
# Problem Solving Agents

Goal based decision making agents finds sequence of actions that leads to the desirable state.

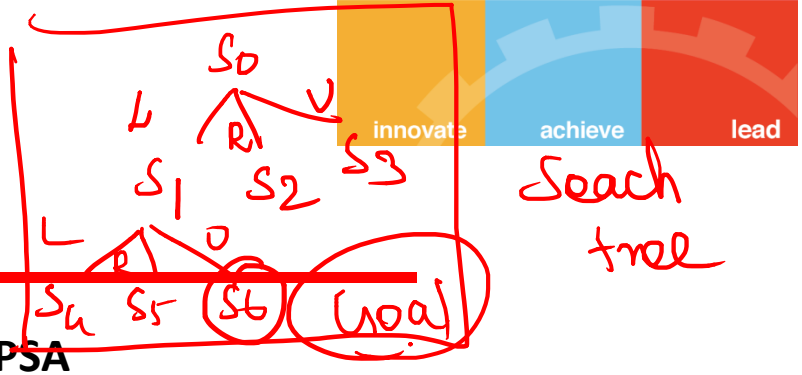



**Phases of Solution Search by PSA**

Optimizes the Objective (Local | Global) Limits the Actions



# Problem Solving Agents

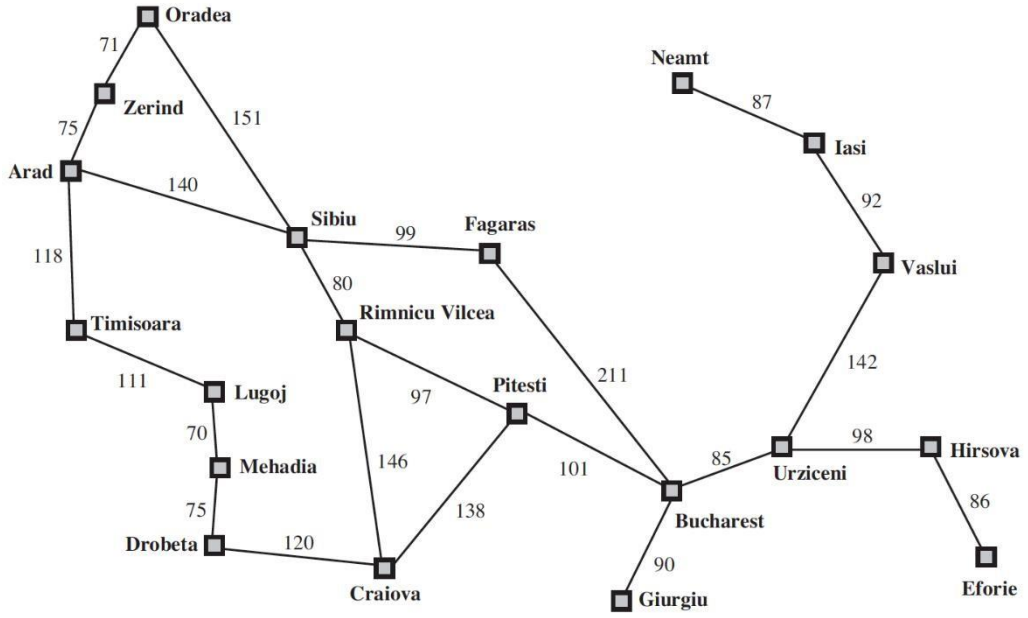



**Phases of Solution Search by PSA**

- Goal Formulatio
- Problem Formulation
- Search Phase
- Execution Phase

State Space Creations [in the path of Goal]

Lists the Actions



# Problem Solving Agents

- Goal Formulation
- Problem Formulation
- Search Phase
- Execution Phase

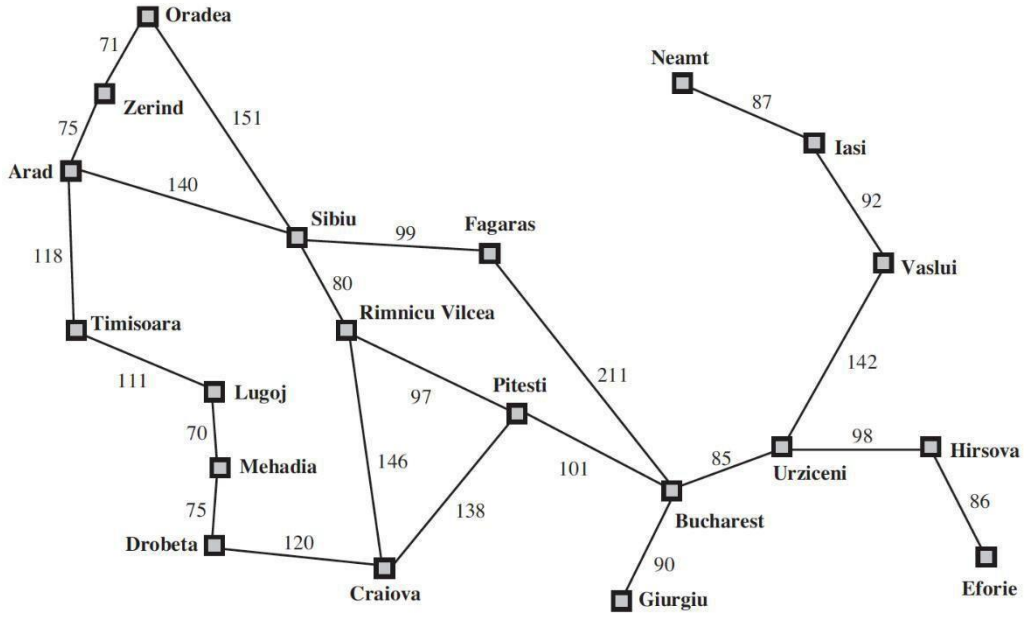
**Phases of Solution Search by PSA**

**Assumptions – Environment :**

**Static**

**Observable Discrete**

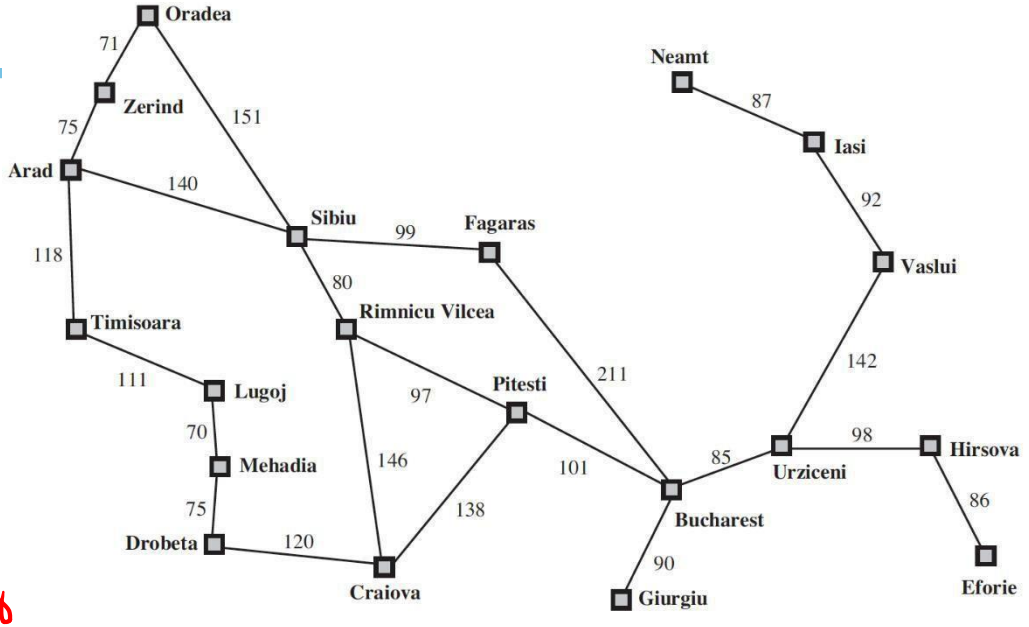
**Deterministic**



# Problem Solving Agents

## Phases of Solution Search

- Goal Formulation
- Problem Formulation
- Search Phase
- Execution Phase




Examine all sequence

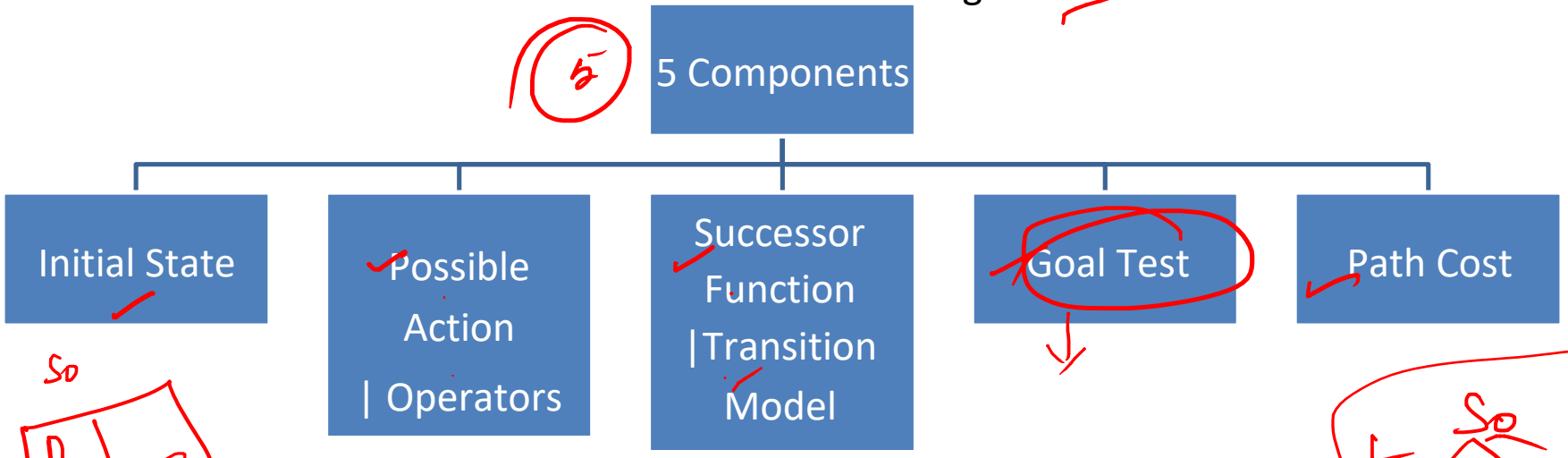
Choose best

Optimal

# Problem Solving Agents – Problem Formulation

Abstraction Representation

Decide what actions under states to take to achieve a goal

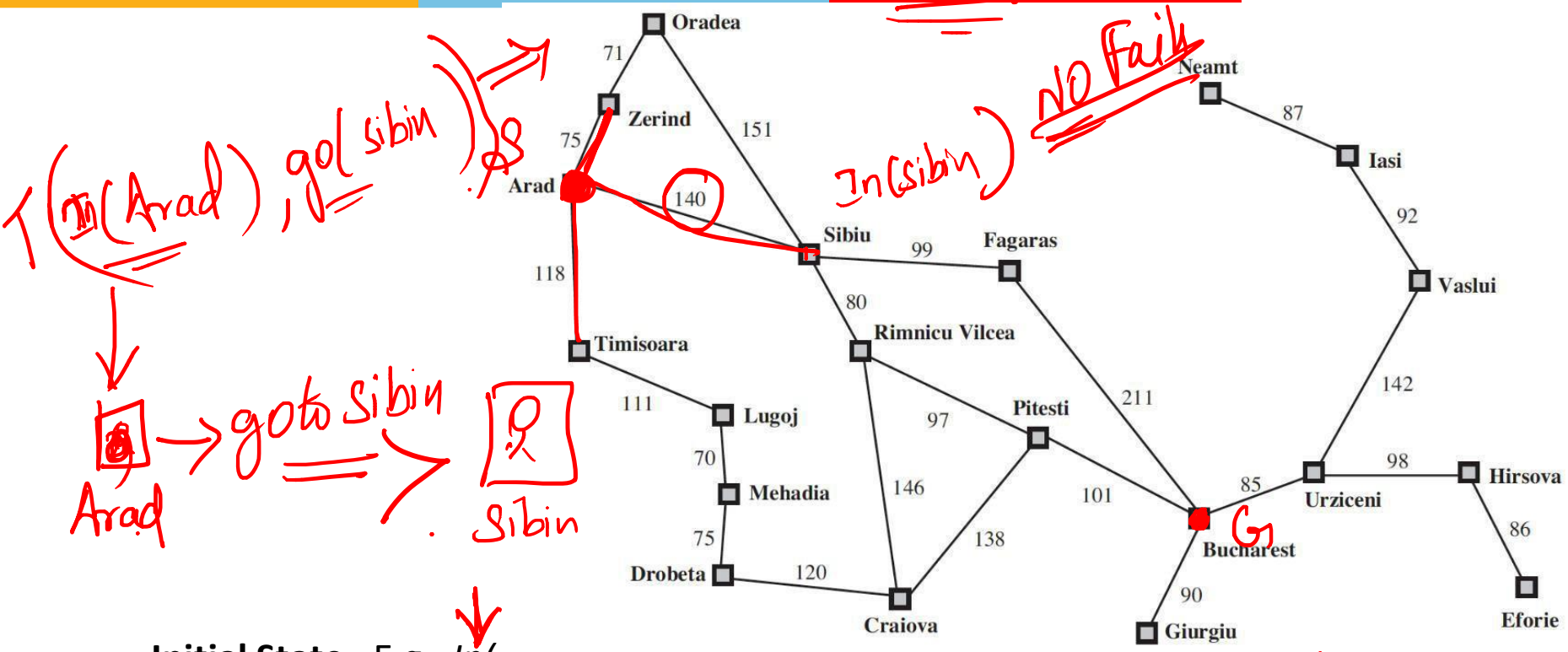



A function that assigns a numeric cost to each path. A path is a series of actions.

Each action is given a cost depending on the problem.

**Solution = Path Cost Function + Optimal Solution**

# Problem Solving Agents – Problem Formulation: BookExample




**Initial State** –E.g., *In(Arad)*

**Possible Actions** – *ACTIONS(s)* → *{Go(Sibiu), Go(Timisoara), Go(Zerind)}*

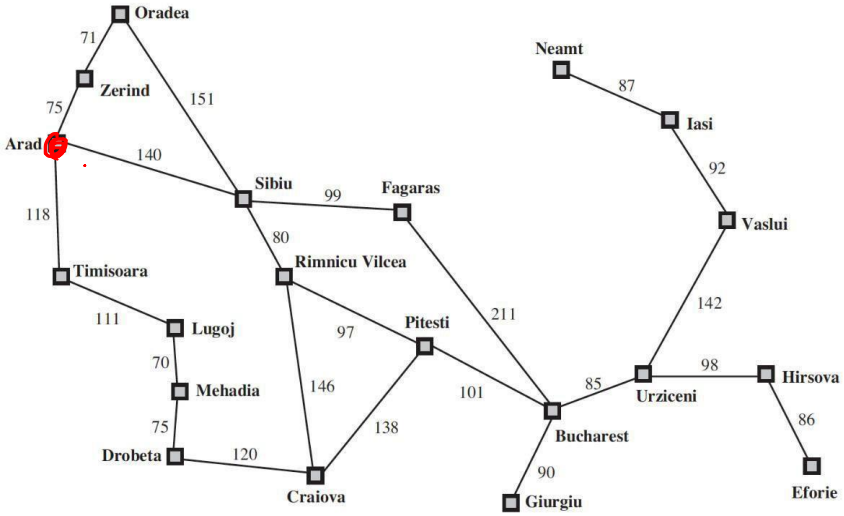
**Transition Model** – *RESULT( In(Arad), Go(Sibiu) ) = In(Sibiu)*

**Goal Test** – *IsGoal( In(Bucharest) ) = Yes*

**Path Cost** – *cost( In(Arad), go(Sibiu)) = 140 kms*

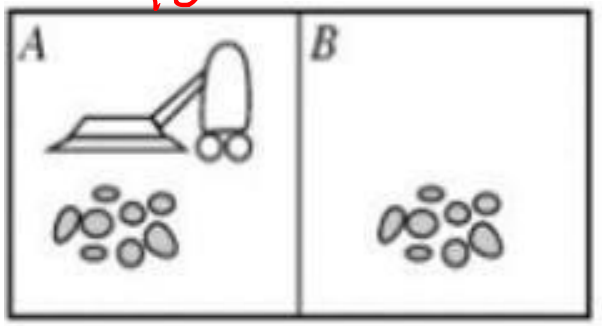
# Example Problem Formulation

| | Travelling Problem |
|---|---|
| Initial State | Based on the problem |
| Possible Actions | Take a flight \| Train \| Shop |
| Transition Model/ Successor Function | [A, Go(A->S)] = [S] |
| Goal Test | Is current = B (destination) |
| Path Cost | Cost + Time + Quality |



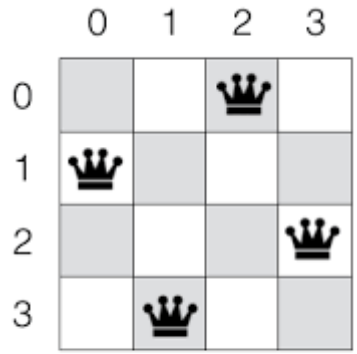
# Example Problem Formulation

| | Vacuum World |
|---|---|
| Initial State | Any |
| Possible Actions | [Move Left, Move Right, Suck, NoOps] |
| Transition Model/ Successor Function | [A, ML] = [B , Dirty]<br>[A, ML] = [B, Clean] |
| Goal Test | Is all room clean? [A, Clean] [B, Clean] |
| Path Cost | No of steps in path |



# Example Problem Formulation

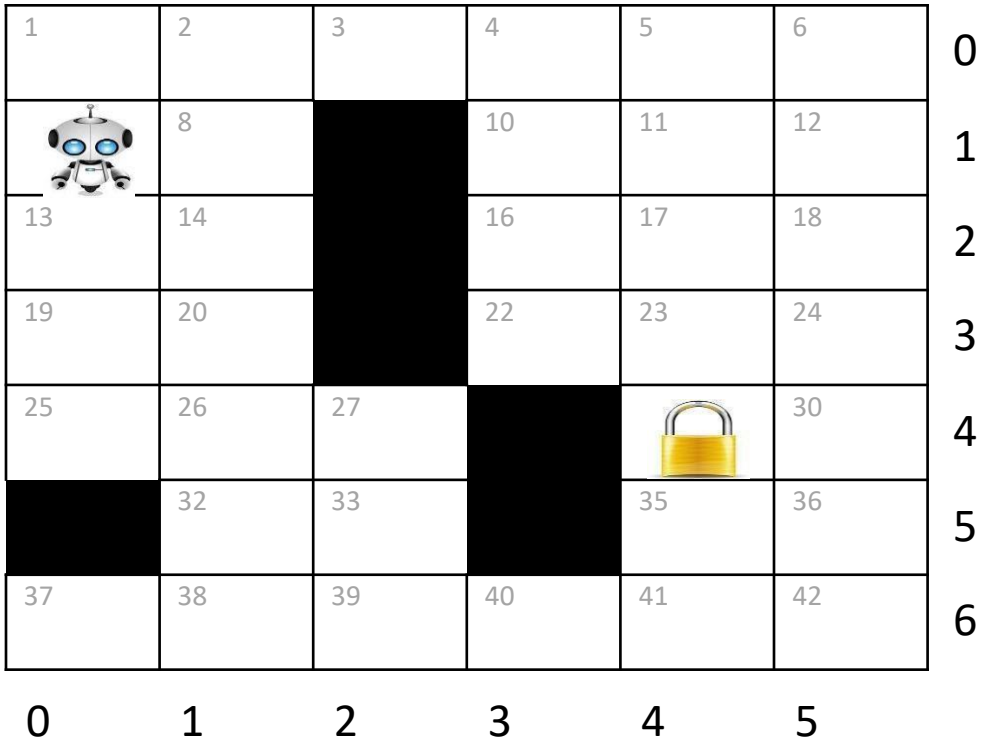
| | N-Queen |
|---|---|
| Initial State | Empty \| Partial \| Full |
| Possible Actions | |
| Transition Model/ Successor Function | |
| Goal Test | |
| Path Cost | |




board[r][c]

# Path finding
# Robot

## Successor Function Design




**N-W-E-S**

**Required Reading:** AIMA - Chapter #1, 2, 3.1, 3.2, 3.3

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials