ARTIFICIAL INTELLIGENCE

PROFESSIONAL ELECTIVE-1

SUBJECT CODE- 19CSE303

UNIT-1
INTRODUCTION

What is Artificial Intelligence?

- One of the booming technologies of computer science is Artificial Intelligence.
- Simulation of human intelligence in m/c that are programmed to think like humans and mimic their actions.

(OR)

- is the capability of a m/c to imitate intelligent human behaviour**.**
- currently working with a variety of subfields, ranging from general to specific,
  - **playing chess**
  - **proving theorems**
  - **playing music**
  - **Painting**
  - **Self driving cars**

Cont…

- Artificial Intelligence is composed of two words **Artificial** and **Intelligence**

- Artificial defines *"man-made,"* and intelligence defines *"thinking power"*, hence AI means *"a man-made thinking power.“*

So, we can define AI ,

- "It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions."

Cont…

Why Artificial Intelligence?

- With the help of AI, one can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.

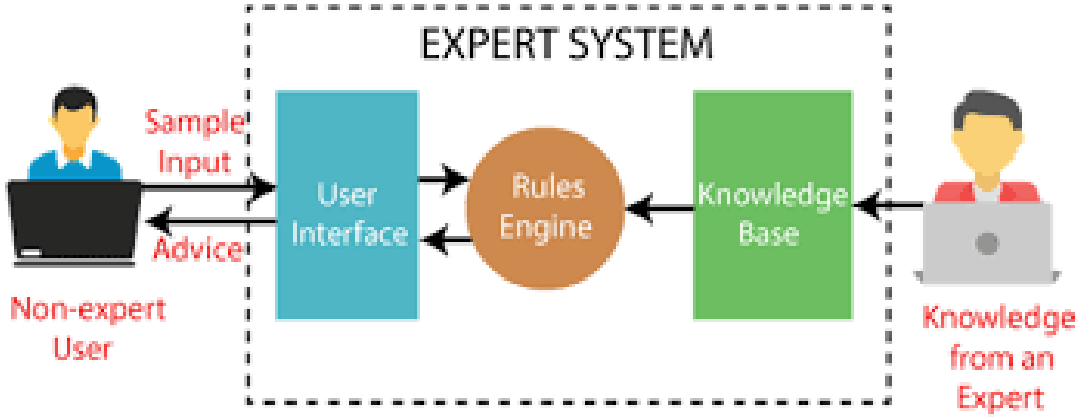
- With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.

- With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.
- Self driving cars need control signals as input . AI systems use Vehicles radar, cameras, apps and cloud services.

GOALS OF ARTIFICIAL INTELIGENCE

- To create an EXPERT SYSTEM

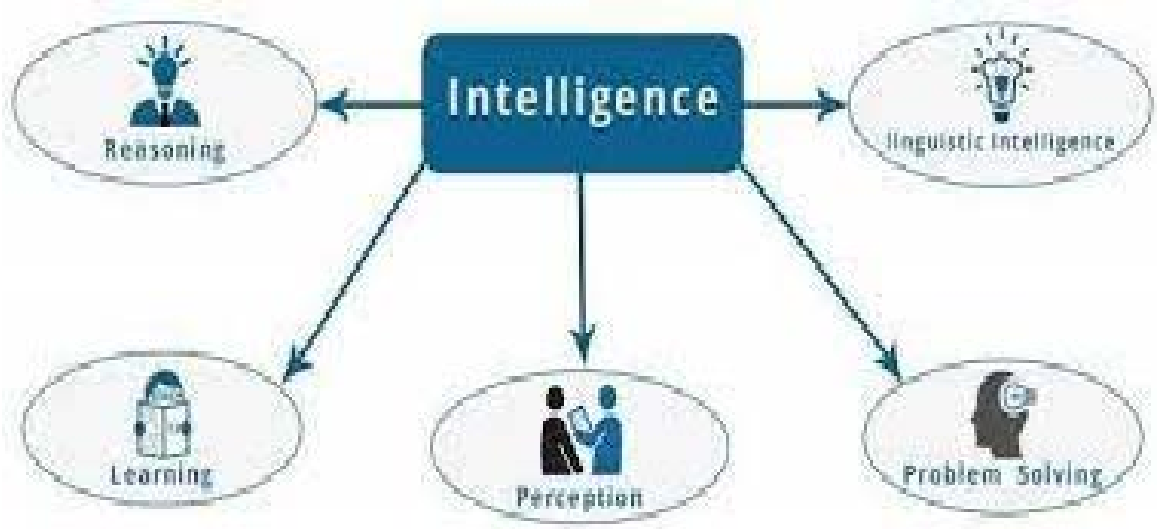
The system which exhibits intelligent behavior learn, demonstrate, explain, and advice its user



cont.

- To replicate human intelligence
- An intelligent connections of perceptions and actions
- Building a machine which can perform tasks that requires human intelligent
  - proving a theorem
  - playing chess
  - plan some surgical operations
  - driving a car in traffic

WHAT IS INTELLIGENCE COMPOSED OF?



Cont.

- Linguistic Intelligence:

The ability to speak, recognize, and use mechanisms of phonology (speech sounds), syntax (grammar), and semantics (meaning).

- **Reasoning** – It is the set of processes that enables us to provide basis for judgement, making decisions, and prediction.

- **Learning** – It is the activity of gaining knowledge or skill by studying, practising, being taught, or experiencing something. Learning enhances the awareness of the subjects of the study.

Cont.

- **Problem Solving** – It is the process in which one perceives and tries to arrive at a desired solution from a present situation by taking some path, which is blocked by known or unknown hurdles.

- **Perception** – It is the process of acquiring, interpreting, selecting, and organizing sensory information.

- Perception presumes **sensing**. In humans, perception is aided by sensory organs. In the domain of AI, perception mechanism puts the data acquired by the sensors together in a meaningful manner.

AI – PROBLEM SOLVING

- Problem solving in games such as Sudoku.eg. It is done by building an AI system to solve a particular problem.
- To do this one need to define the problem statement first, and then generating the solution by keeping the condition in mind.
- Some of the most popularly used problem solving techniques with the help of AI are:

  Towers of Hanoi

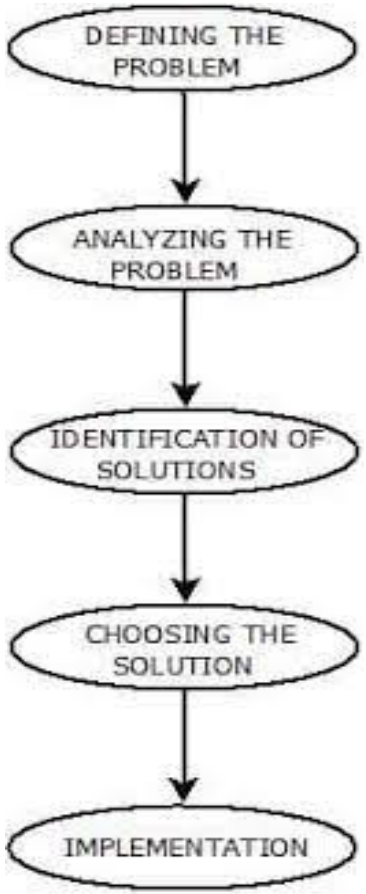
  playing chess

  N-queen problem

  water jug problem

  Travelling salesman problem

5 steps for problem solving

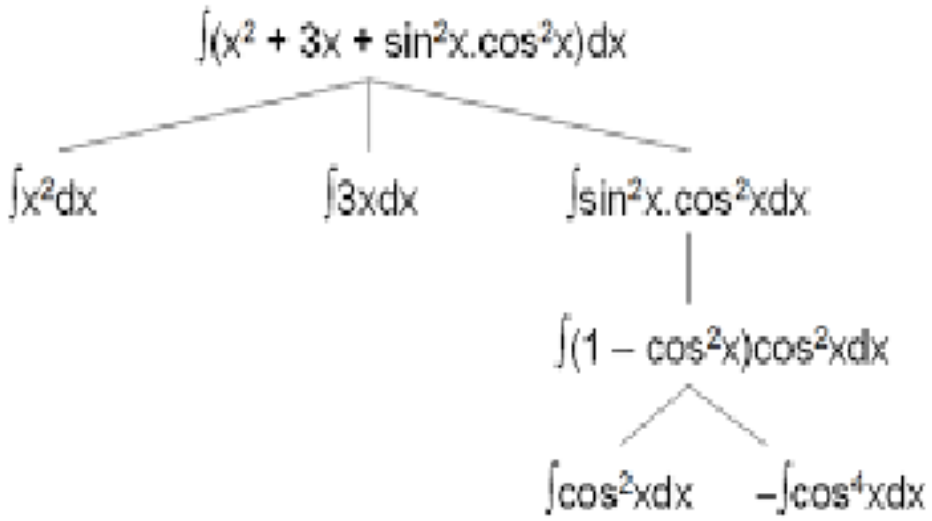


Problem Characteristics

- Is the problem decomposable?
  - if yes, the problem becomes simpler to solve because each lesser problem can be tackled and the solutions combined together at the end
- Can solution steps be undone or ignored?
  - a game for instance often does not allow for steps to be undone (can you take back a chess move?)
- Is the problem's universe predictable?
  - will applying the action result in the state we expect? for instance, in the monkey and banana problem, waving the stick on a chair does not guarantee that a banana will fall to the ground!
- Is a good solution absolute or relative?
  - for instance, do we care how many steps it took to get there?
- Is the desired solution a state or a path?
  - is the problem solved by knowing the steps, or reaching the goal?
- Is a large amount of knowledge absolutely required?
- Is problem solving interactive?

Powered by DeSiaMore 6

Is the problem decomposable?



Can solution steps be ignored or undone?

- **Ignorable:** theorem proving

  Solution steps can be ignored

- Recoverable: 8 puzzle problem

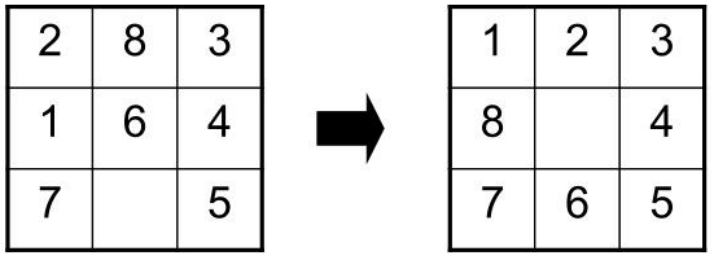
  Solution steps may be undone (backtracking)

- Irrecoverable: chess
- Solution steps cannot be undone.

Cont..

Can solution steps be ignored or undone?

The 8-Puzzle

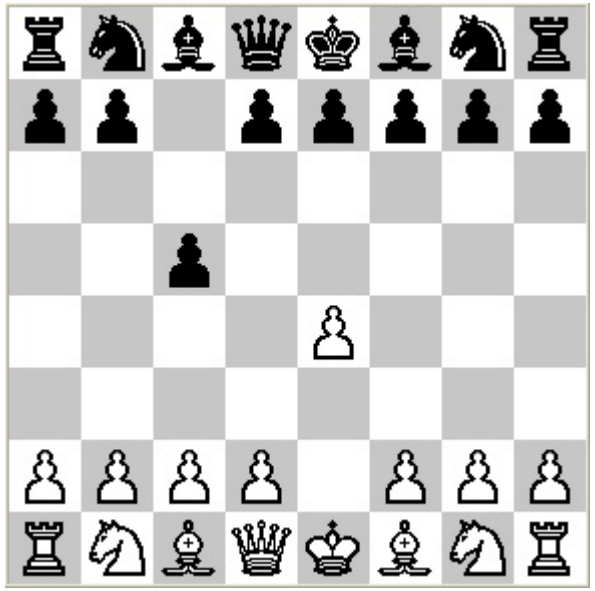


Moves can be undone and backtracked.

Recoverable!

33

Cont...

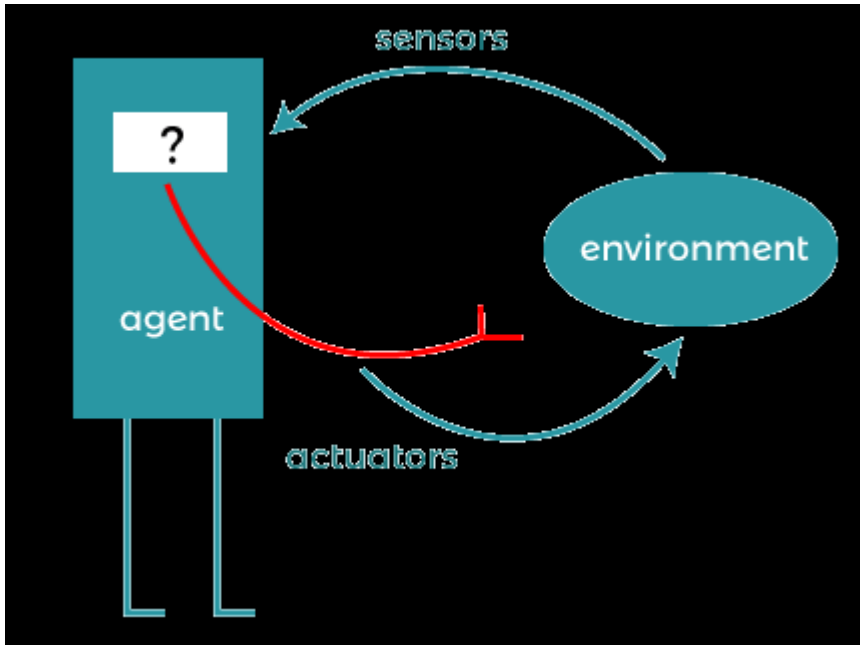


AGENT

- What is an Intelligent Agent?

An agent is anything that

- **Perceives** its environment through **sensors**
- **Acts** upon that environment through **Actuators**
- **STRUCTURE OF AN AGENT**
-



Examples of Agent

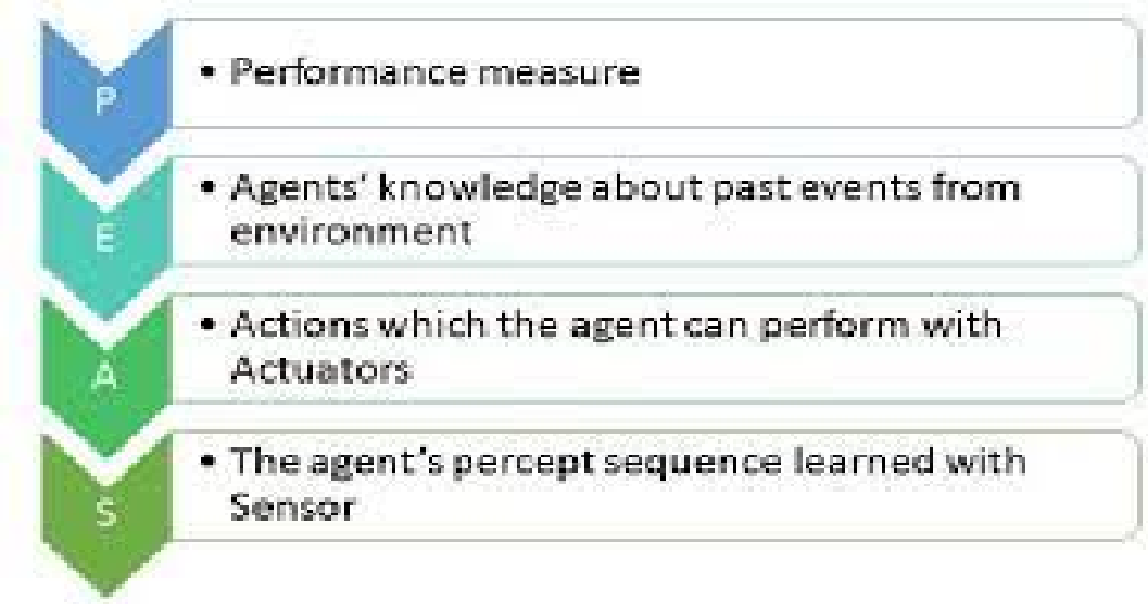
TABLE 1. AGENT MODE OF PERCEPTION

	Different Types of Agent Mode of Perception		
	Environment /Agent	*Sensor*	*Actuator*
1.	Human Agent[a]	Eyes, Ear etc	Legs, Hand, Mouth
2.	Robotic Agent	Cameras, IR range finder	Motor
3.	Software Agent	Key strokes, File contents	Displays to screen, write files

A Agent Architecture

Rational Agent

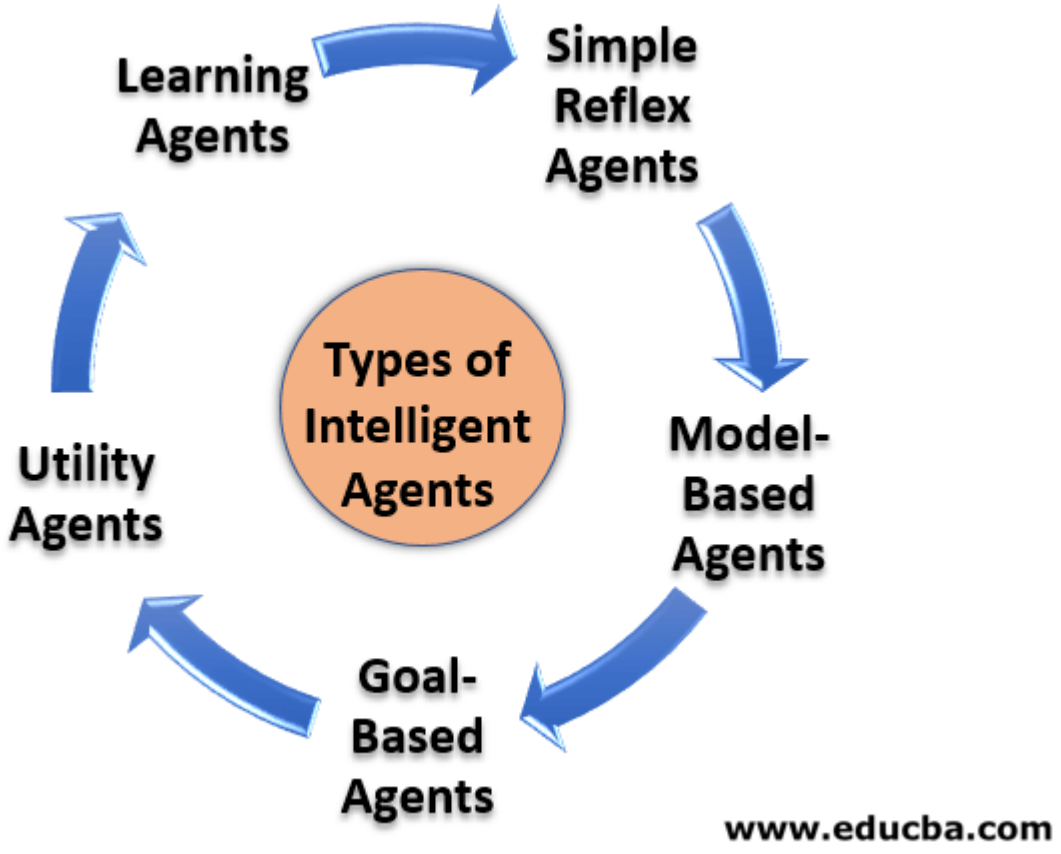
- A **rational agent** could be **anything that makes decisions, as a person, firm, machine, or software**.
- It carries out an action with the best outcome after considering past and current percept's.
- The rationality agents depends on



Example

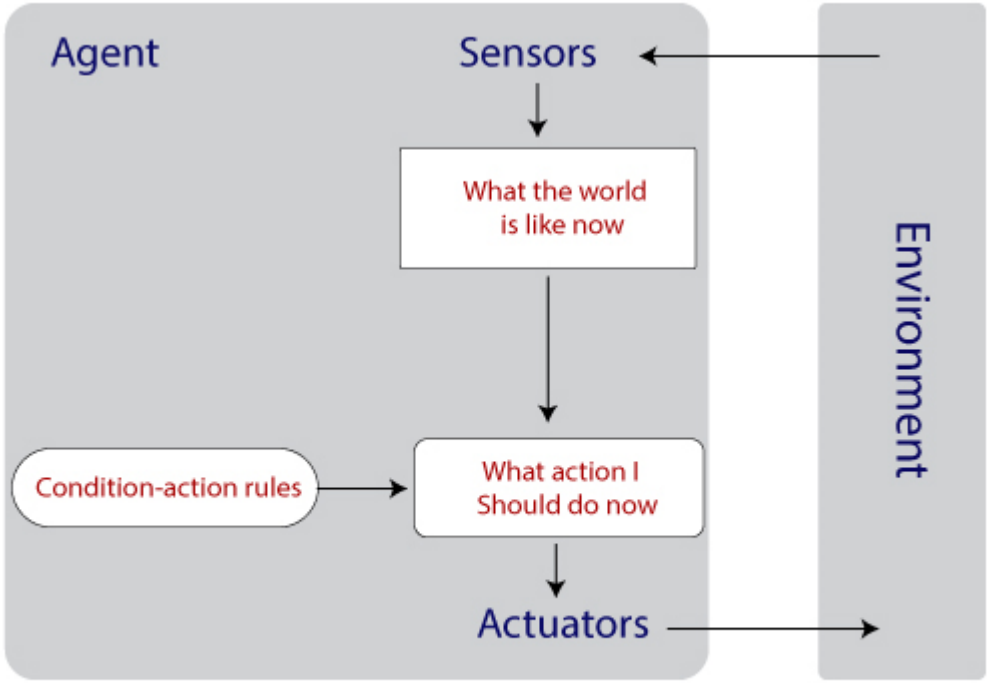
- Consider, e.g., the task of designing an automated taxi driver:
  - Performance measure: safe, fast, legal etc.
  - Environment: roads, traffic, pedestrians, etc.
  - Actuators: steering, accelerator, brake, signal, etc.
  - Sensors: cameras, speedometer, GPS, odometer, etc.

TYPES OF AGENT



SIMPLE REFLEX AGENT

- They choose actions based on the current situation ignoring the history of perception.
- They can perform action only on simple situation.
- The agent function is performed based on “condition- action rule” if condition then action.
- They will work only if the environment is fully observable

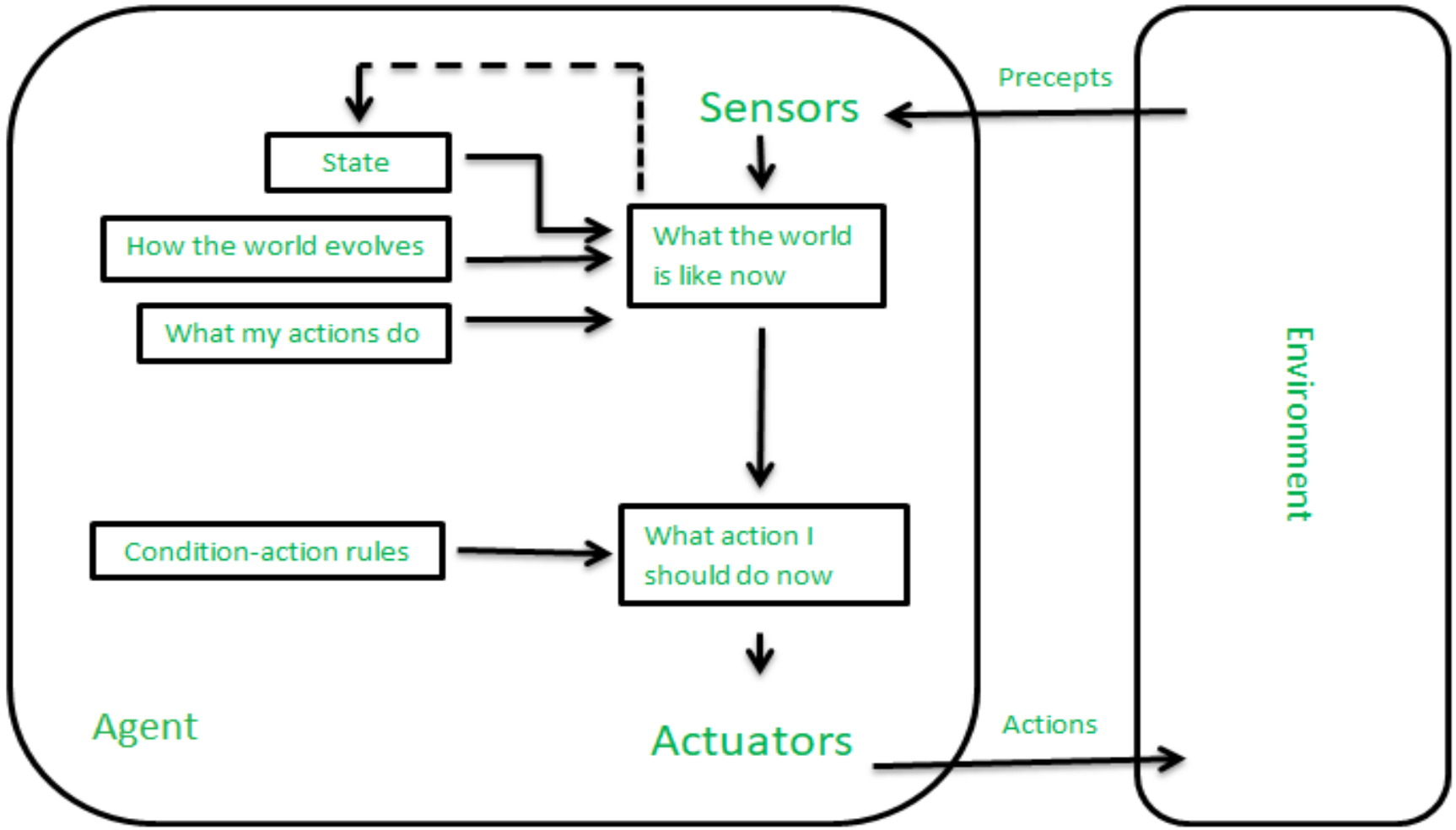


Schematic diagram of a simple-reflex agent

MODEL BASED AGENT

- The Model-based agent can work in a partially observable environment, and track the situation.
- A model-based agent has two important factors:
  - **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent.
  - **Internal State:** It is a representation of the current state based on percept history.
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.
- Updating the agent state requires information about:
  - How the world evolves
  - How the agent's action affects the world.

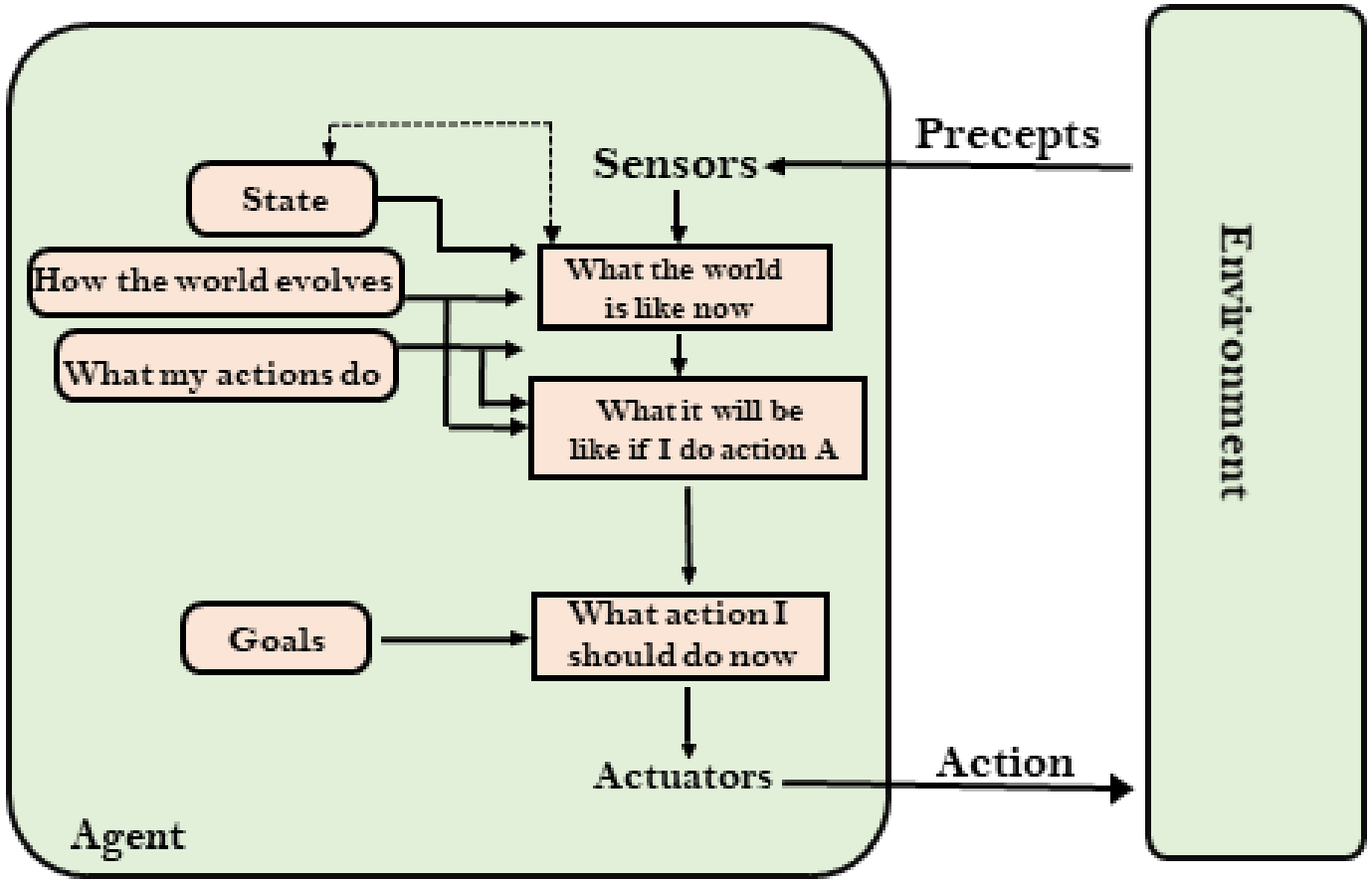
Diagram


Precepts
Sensors
State
How the world evolves
What my actions do
What the world is like now
Condition-action rules
What action I should do now
Agent
Actuators
Actions
Environment

GOAL BASED AGENT

- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.
- The agent needs to know its goal which describes desirable situations.
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.
- They choose an action, so that they can achieve the goal.
- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.

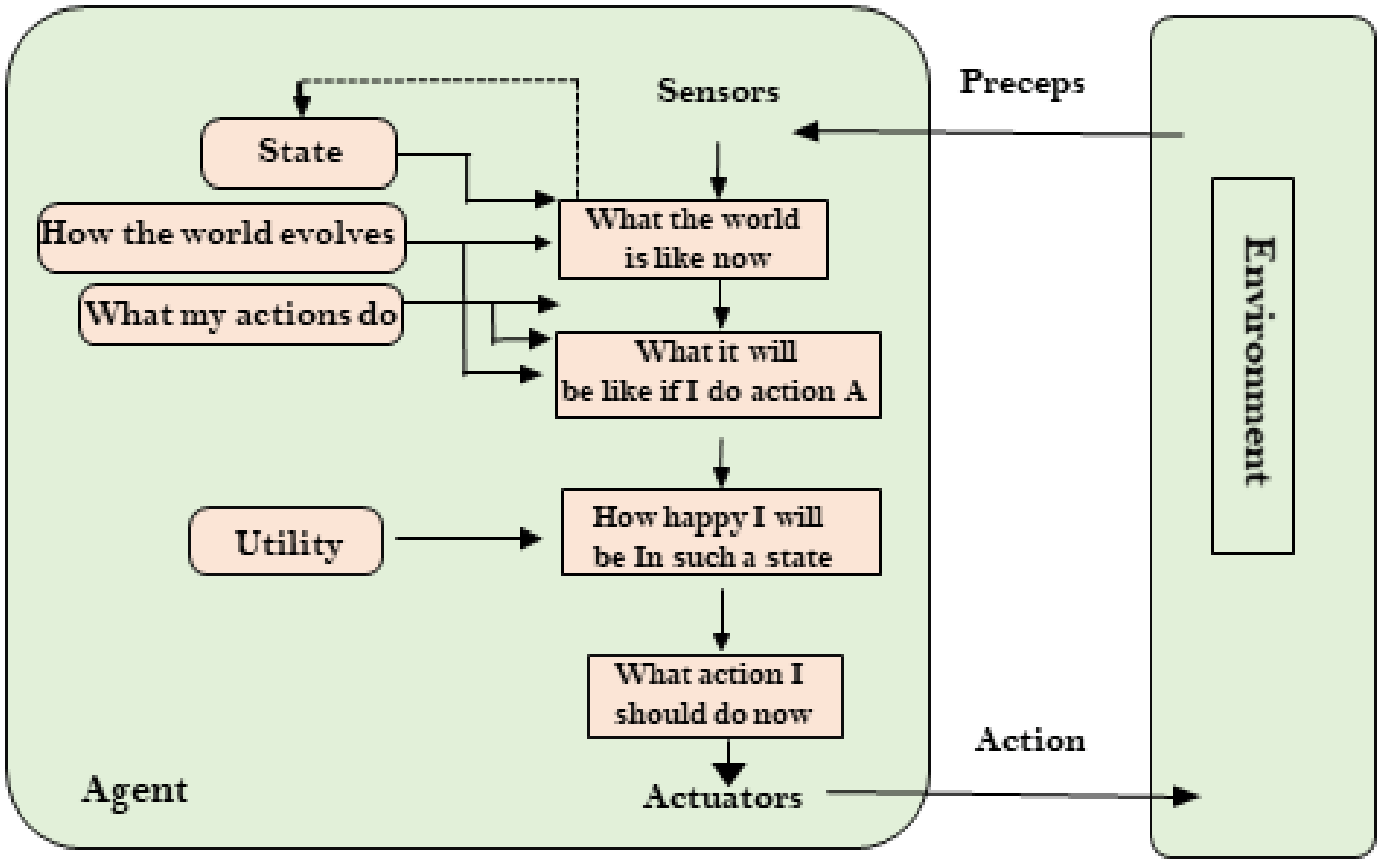
Diagram


Precepts
Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Actuators
Action
Agent
Environment

UTILITY BASED AGENT

- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.
- Utility-based agent act based not only goals but also the best way to achieve the goal.
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goals.

Diagram

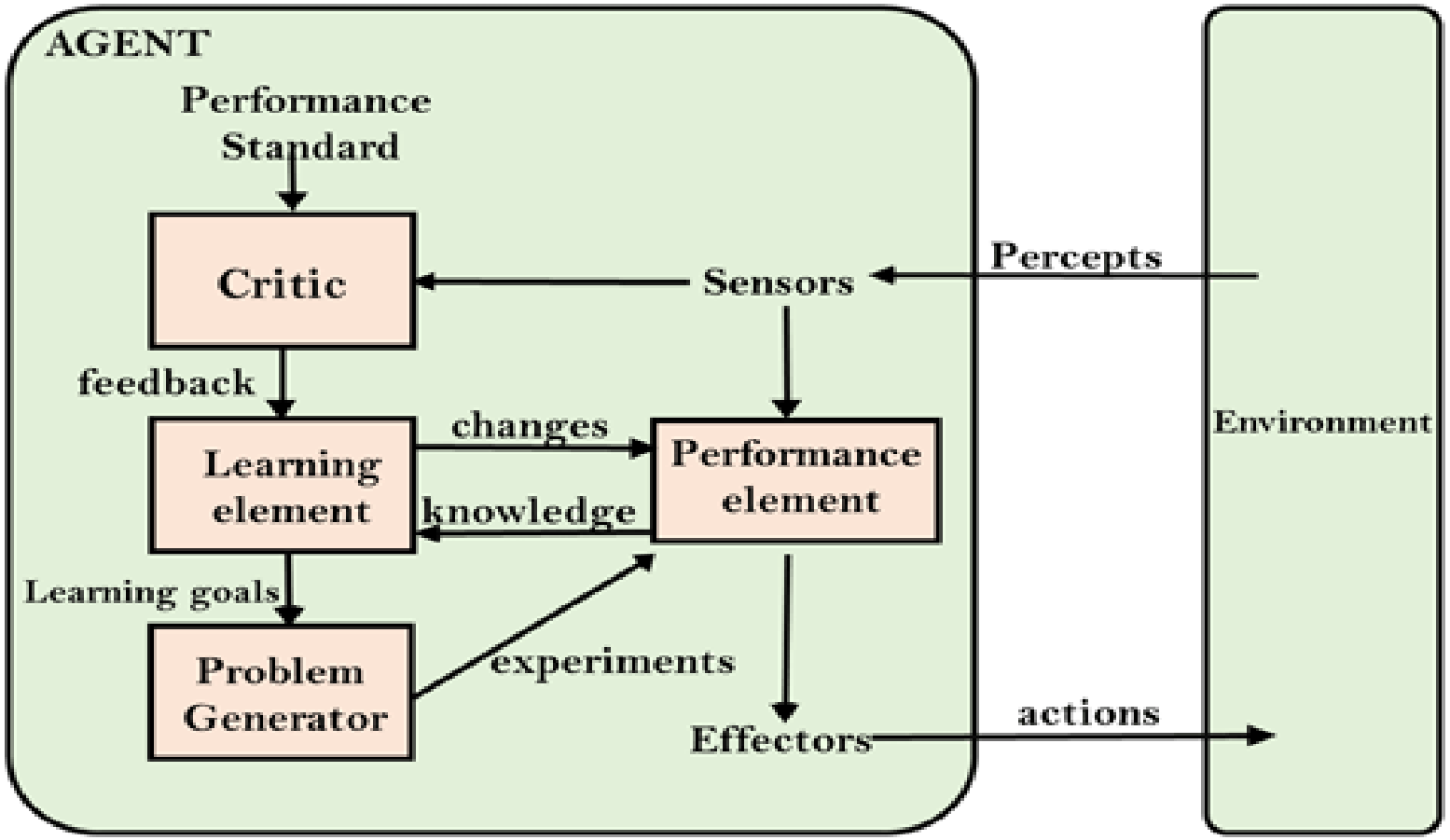

Sensors
Preceps
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Utility
How happy I will be In such a state
What action I should do now
Agent
Actuators
Action
Environment

LEARNING BASED AGENT

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities.
- It starts to act with basic knowledge and then able to act and adapt automatically through learning.

- A learning agent has mainly four conceptual components, which are:
  - **Learning element:** It is responsible for making improvements by learning from environment
  - **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
  - **Performance element:** It is responsible for selecting external action
  - **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.

Diagram


AGENT
Performance Standard
Critic
Sensors
Percepts
feedback
changes
Learning element
Performance element
knowledge
Environment
Learning goals
Problem Generator
experiments
Effectors
actions

TCS NQT LOGIC QUESTIONS

- What will be the output of the following Java code?

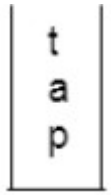
```
1.      import java.util.*;
2.      class stack
3.      {
4.          public static void main(String args[])
5.          {
6.              Stack obj = new Stack();
7.              obj.push(new Integer(3));
8.              obj.push(new Integer(2));
9.              obj.pop();
10.             obj.push(new Integer(5));
11.             System.out.println(obj);
12.         }
13.     }
```

OPTIONS

- [3, 5]
- [3, 2]
- [3, 2, 5]
- [3, 5, 2]

CONT...

- What will be result if the given stack is popped?
  - pat
  - tap
  - atp
  - apt



CONT...

```
INT MAIN()
{
INT A= 10,X;
X=A--  +  ++A;
PRINTF(%D”, X);
}
```

OPTIONS

1. 19
2. 20
3. 21
4. 22

TCS NQT Data Structure MCQS

1.Consider the following operation performed on the stack of size 5.

Push(1);

Pop();

Push(2);

Push(3);

Pop();

Push(4);

Pop();

Pop();

Push(5);

After the completion of all operations , the number of elements present in the stack is?

Answers: 1 ,2,3,4 ?

Cont…

2. If the elements of A,B,C and D are placed in a stack and are deleted one at a time, what is the order of deletion?

A. ABCD

B. DCBA Answer: ?

C. BCAD

D. CABD

3. Which of the real world scenario would u associate with a stack data structure.

a. Piling of chairs one above the other

b. People standing in a line to be serviced at the counter

c. Offer services based on the priority of the customer

d. Tatkal ticket booking in IRCTC

Problem formulation

- **One of the core steps of problem-solving which decides what action should be taken to achieve the formulated goal.**

- In AI this core part is dependent upon software agent which consisted of the following 5 components to formulate the associated problem.
- Problem Statement
  - Definition
  - Limitation or Constraints or Restrictions
- Problem Solution
- Solution Space
- Operators

Cont…

- **Definition of Problem**

Why it is important to build AI system? What will be the advantages of proposed system? For example "**I want to predict the price of house using AI system**".

- **Problem Limitation**

There always some limitations while solving problems. All these limitations or constraints must be fulfil while creating system.

Goal or Solution

What is expected from system? The Goal state or final state or the solution of problem is defined here. This will help us to proposed appropriate solution for problem. For example "**we can use some machine learning technique to solve this problem".**

Cont…

- **Solution Space**

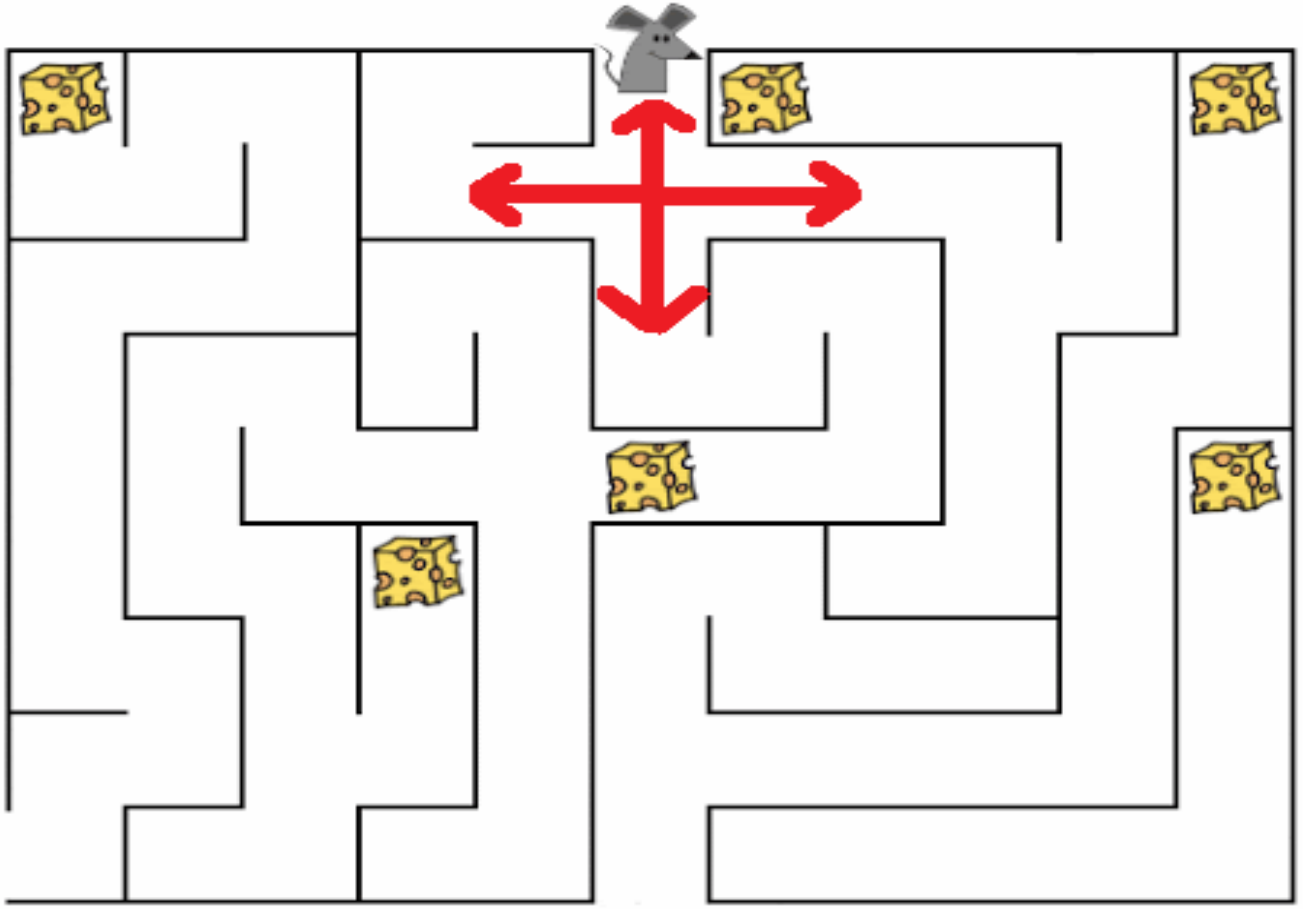
Problem can be solve in many ways. Many possible ways with which we can solve problem is known as Solution Space. For example "**price of house can be predict using many machine learning algorithms**".

- **Operators**

Operators are the actions taken during solving problem. Complete problem is solved using tiny steps or actions and all these consecutive actions leads to solution of problem.

- **Examples of Problem Formulation**

Cont...



Cont...

- **Mouse Path Problem**
- **Problem Statement**
  - **Problem Definition:** Mouse is hungry, mouse is in a puzzle where there are some cheese. Mouse will only be satisfied if mouse eat cheese
  - **Problem Limitation:** Some paths are close i-e dead end, mouse can only travel through open paths
- **Problem Solution**: Reach location where is cheese and eat minimum one cheese. There are possible solutions (cheese pieces)
- **Solution Space:** To reach cheese there are multiple paths possible
- **Operators**: Mouse can move in four possible directions, these directions are operators or actions which are UP, DOWN, LEFT and RIGHT

Well-defined Problems and Solution

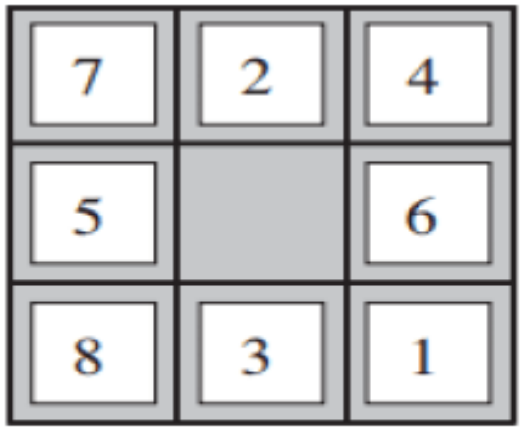
- Problem solving components discussed above are applicable to any problem. For AI system implementation, problem must be well defined. A well-defined problem must have five components:-

- **Initial State**: Start point of problem
- **Final State**: The finish point of problem. Goal or solution state
- **States**: Total states in problem
- **Transition Model**: How one can shift from one state to another
- **Actions**: Actions set, used to move from one state to another
- **Path Cost**: What is total effort (cost) from initial state to final state

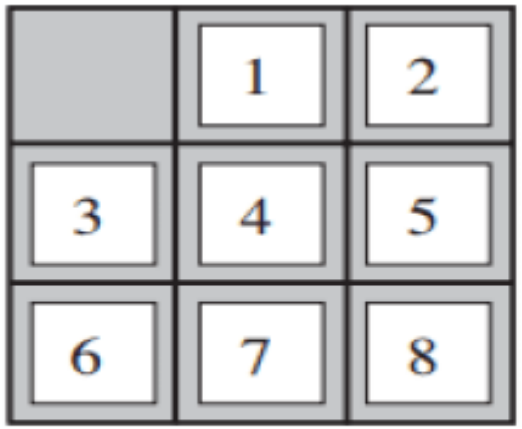
8 Puzzle or Slide Puzzle

- **States**: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.
- **Initial state**: Any random shuffled state can be designated as initial state
- **Actions**:
  - Slide Left
  - or Slide Right
  - or Slide Up
  - And Slide Down
- **Transition model**: Given a state and action, this returns the resulting state
- **Goal test**: This checks whether the state matches the goal
- **Path cost**: Each step costs 1

Cont…



Start State



Goal State

TCS NQT DATA STRUCTURE

1.While inserting the elements 71,65,84,69,67,83 in a empty binary search tree (BST), in the sequence shown, find the lowest level.

2. The five items A,B,C,D,E are pushed on to the stack one after the other starting from A. The stack is popped 4 items and each element is inserted into queue. The 2 elements are deleted from the queue and pushed back on to the stack. Now one item is popped from the stack. What is the popped item?

Uninformed Search Algorithms in AI

- AI systems use various search algorithms for a particular goal state(if it exists).
- As the name 'Uninformed Search' means the machine blindly follows the algorithm regardless of whether right or wrong, efficient or in-efficient.
- These algorithms are brute force operations, and they don't have extra information about the search space;
- The only information they have is on how to traverse or visit the nodes in the tree.
- Uninformed search algorithms are also called blind search algorithms.

Cont...

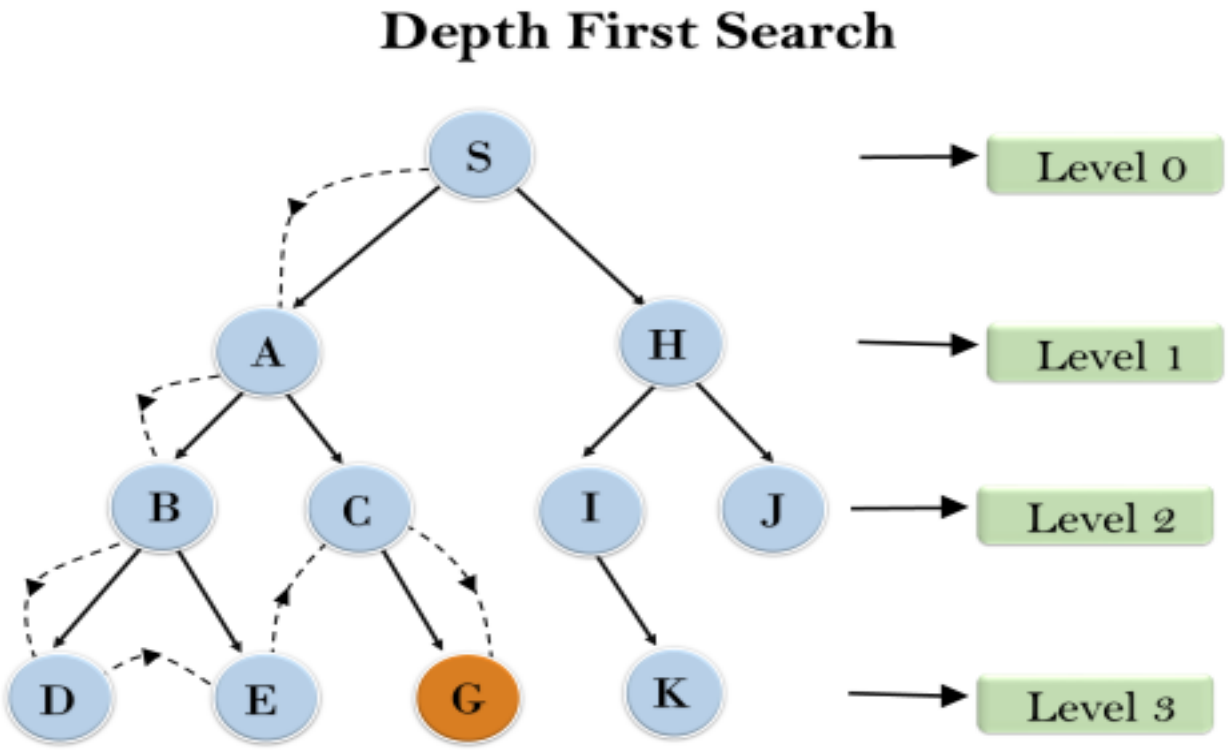
The different types of search algorithms are as follows:

- 1. Depth First Search (DFS)
- 2. Breadth-First Search(BFS)
- 3. Uniform Cost Search(UCS)
- 4. Depth Limited Search(DLS)
- **5. Iterative Deepening Depth First Search(IDDFS)**
- 6. Bidirectional Search(BS)

DFS(DEPTH FIRST SEARCH)

- It is a search algorithm where the search tree will be traversed from the root node.
- It will be traversing, searching for a key at the leaf of a particular branch.
- If the key is not found the searching retraces its steps back to the point from where the other branch was left unexplored and the same procedure is repeated for that other branch.

Cont…


Depth First Search
S
A
H
B
C
I
J
D
E
G
K
Level 0
Level 1
Level 2
Level 3

Cont…

- In the above search tree, we have shown the flow of depth-first search, and it will follow the order as:

- Root node--->Left node ----> right node.

- It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found.

- After backtracking it will traverse node C and then G, and here it will terminate as it found goal node.

Cont…

- **Completeness:** DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.

- **Time Complexity:** Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

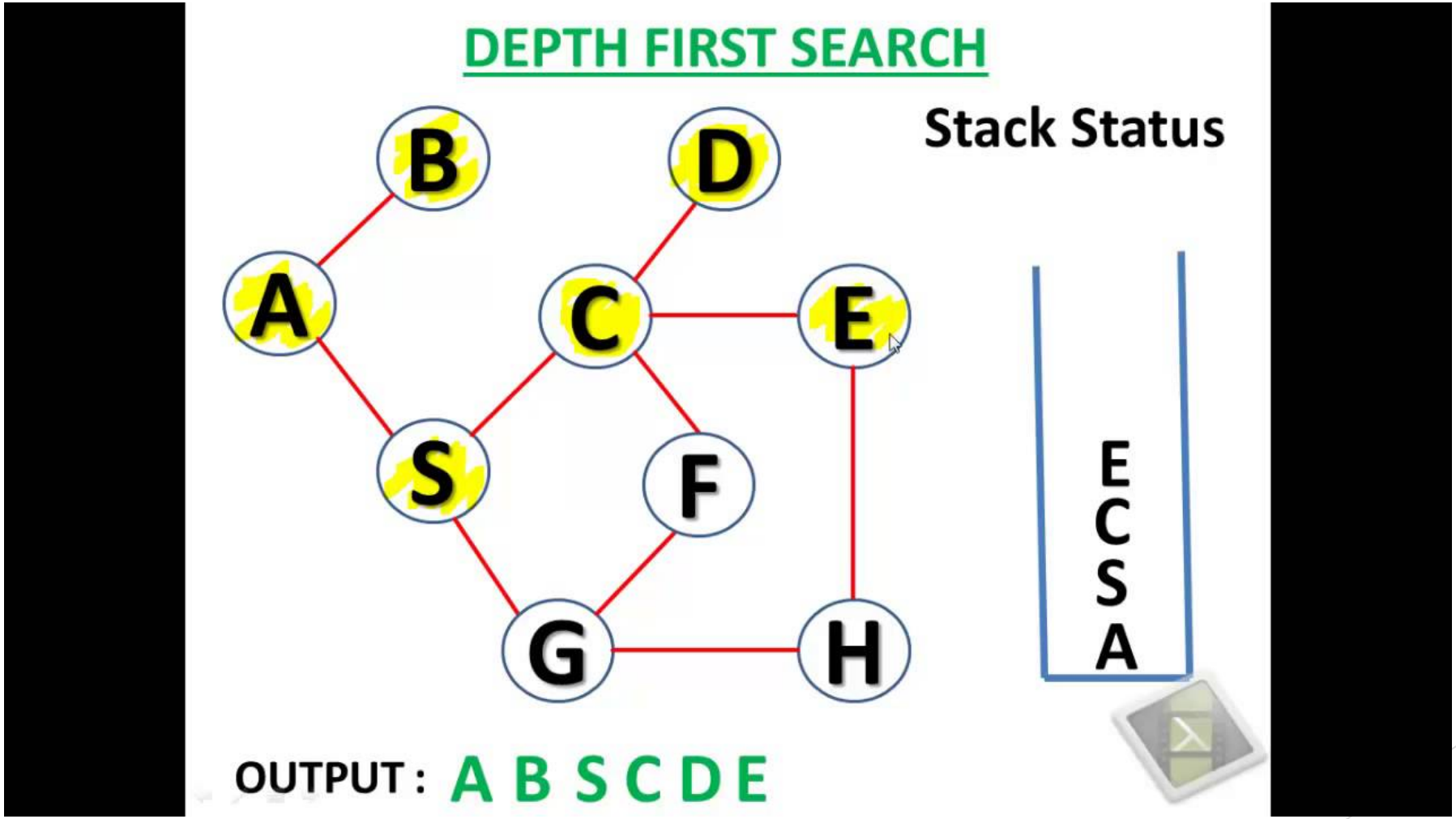
$$T(n)= \mathbf{O(n^m)}$$

Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.

- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

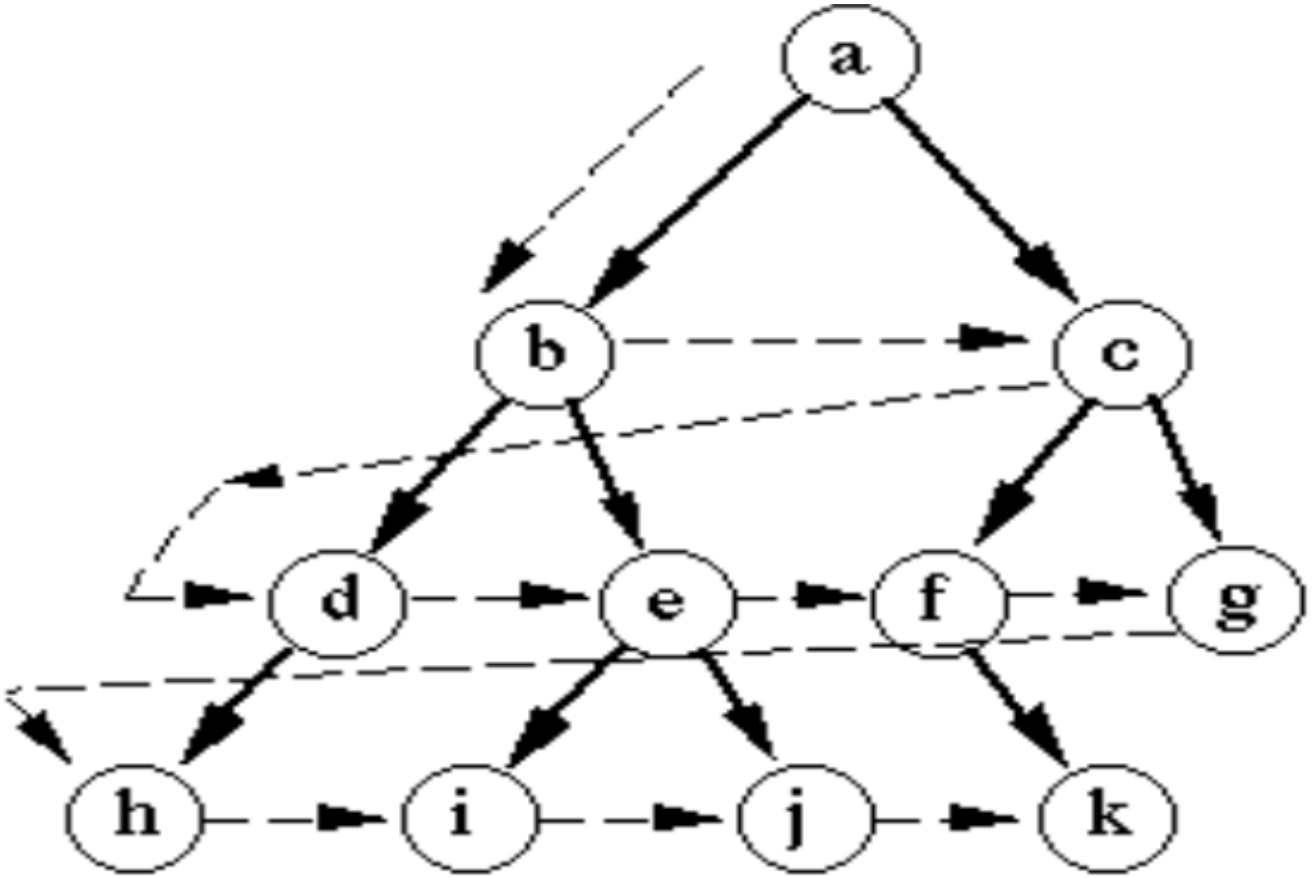
DFS using stack



BFS ALGORITHM

- It is another search algorithm in AI which traverses breadthwise to search the goal in a tree.
- It begins searching from the root node and expands the successor node before going expanding it further expands along breadthwise and traverses those nodes rather than searching depth-wise.
- BFS puts every vertex of the graph into two categories - visited and non-visited. It selects a single node in a graph and, after that, visits all the nodes adjacent to the selected node.

CONT...



Breadth-first search

CONT...

- The above figure is an example of a BFS Algorithm.

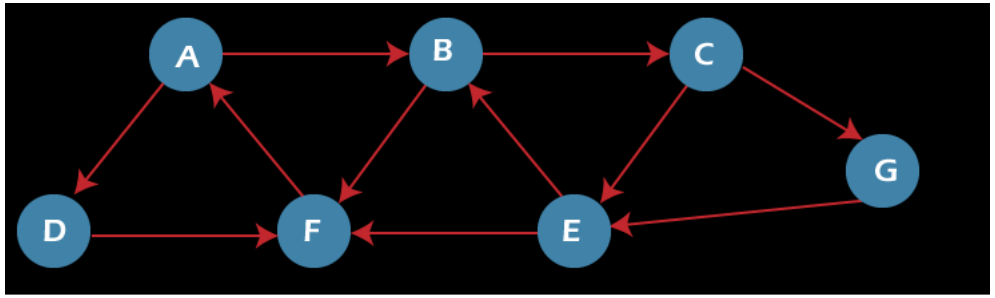
- It starts from the root node A and then traverses node B. Till this step it is the same as DFS. But here instead of expanding the children of B as in the case of DFS we expand the other child of A i.e. node C because of BFS and then move to the next level and traverse from D to G and then from H to K in this typical example. To traverse here we have only taken into consideration the lexicographical order. This is how the BFS Algorithm is implemented.

Applications of BFS algorithm

- The applications of breadth-first-algorithm are given as follows -
- BFS can be used to find the neighboring locations from a given source location.
- In a peer-to-peer network, BFS algorithm can be used as a traversal method to find all the neighboring nodes. Most torrent clients, such as BitTorrent, uTorrent, etc. employ this process to find "seeds" and "peers" in the network.
- BFS can be used in web crawlers to create web page indexes. It is one of the main algorithms that can be used to index web pages. It starts traversing from the source page and follows the links associated with the page. Here, every web page is considered as a node in the graph.
- BFS is used to determine the shortest path and minimum spanning tree.

Example of BFS algorithm

- Now, let's understand the working of BFS algorithm by using an example. In the example given below, there is a directed graph having 7 vertices.
- Adj:vertices



A: B,D

B: C,F

C: E,G

G: E

E: B,F

F: A

D: F

- In the above graph, minimum path 'P' can be found by using the BFS that will start from Node A and end at Node E. The algorithm uses two queues, namely QUEUE1 and QUEUE2. QUEUE1 holds all the nodes that are to be processed, while QUEUE2 holds all the nodes that are processed and deleted from QUEUE1.

Cont…

- Now, let's start examining the graph starting from Node A.
- **Step 1** - First, add A to queue1 and NULL to queue2.

1. QUEUE1 = {A}
2. QUEUE2 = {NULL}

- **Step 2** - Now, delete node A from queue1 and add it into queue2. Insert all neighbors of node A to queue1.

1. QUEUE1 = {B, D}
2. QUEUE2 = {A}

- **Step 3** - Now, delete node B from queue1 and add it into queue2. Insert all neighbors of node B to queue1.

1. QUEUE1 = {D, C, F}
2. QUEUE2 = {A, B}

- **Step 4** - Now, delete node D from queue1 and add it into queue2. Insert all neighbors of node D to queue1. The only neighbor of Node D is F since it is already inserted, so it will not be inserted again.

Cont…

1. QUEUE1 = {C, F}
2. QUEUE2 = {A, B, D}

- **Step 5** - Delete node C from queue1 and add it into queue2. Insert all neighbours of node C to queue1.

1. QUEUE1 = {F, E, G}
2. QUEUE2 = {A, B, D, C}

- **Step 5** - Delete node F from queue1 and add it into queue2. Insert all neighbours of node F to queue1. Since all the neighbours of node F are already present, we will not insert them again.

1. QUEUE1 = {E, G}
2. QUEUE2 = {A, B, D, C, F}

- **Step 6** - Delete node E from queue1. Since all of its neighbours have already been added, so we will not insert them again. Now, all the nodes are visited, and the target node E is encountered into queue2.

1. QUEUE1 = {G}
2. QUEUE2 = {A, B, D, C, F, E}

Cont…

- The time complexity of BFS algorithm is **O(V+E)**

- The space complexity of BFS can be expressed as **O(V)**

Heuristics

What is Heuristics?

- A heuristic is a technique that is used to solve a problem faster than the classic methods.
- These techniques are used to find the approximate solution of a problem when classical methods do not.
- Heuristics are said to be the problem-solving techniques that result in practical and quick solutions.
- Heuristics are strategies that are derived from past experience with similar problems.
- The most common heuristic methods are - trial and error, guesswork, the process of elimination, historical data analysis.

Cont…


Heuristic Search in Artificial Intelligence
Hill Climbing
Constraint Satisfaction Problems
Simulated Annealing
Best-First Search (BFS)

HILL CLIMBING

- It is a technique for optimizing the mathematical problems.
- Hill Climbing is widely used when a good heuristic is available.
- It is a local search algorithm that continuously moves in the direction of increasing elevation/value to find the mountain's peak or the best solution to the problem.
- It terminates when it reaches a peak value where no neighbour has a higher value.
- Traveling-salesman Problem is one of the widely discussed examples of the Hill climbing algorithm, in which we need to minimize the distance travelled by the salesman.

Cont…

The steps of a simple hill-climbing algorithm are listed below:

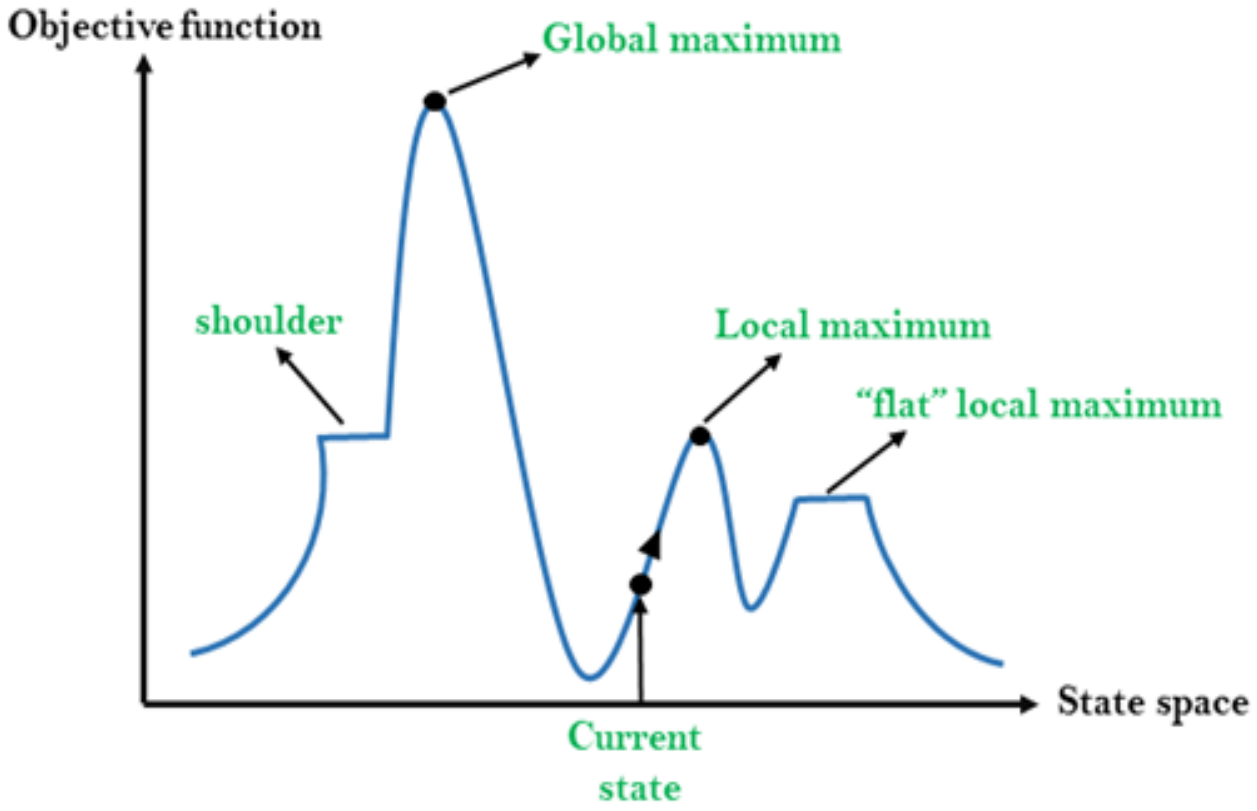
- **Step 1:** Evaluate the initial state. If it is the goal state, then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:

  If it is a goal state, then return to success and quit.

  Else if it is better than the current state, then assign a new state as a current state.

  Else if not better than the current state, then return to step2.
- **Step 5:** Exit.

State-space Diagram for Hill Climbing



Cont…

- The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.

- **Local Maximum:** Local maximum is a state which is better than its neighbour states, but there is also another state which is higher than it.

- **Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.

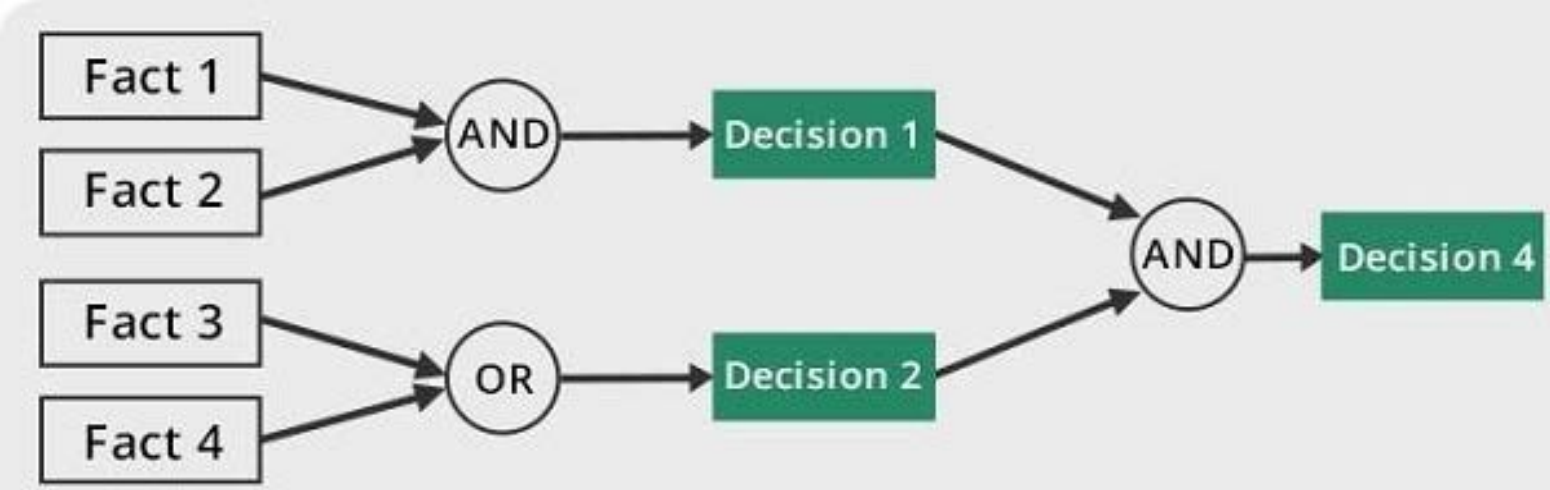
- **Current state:** It is a state in a landscape diagram where an agent is currently present.

- **Flat local maximum:** It is a flat space in the landscape where all the neighbour states of current states have the same value.

- **Shoulder:** It is a plateau region which has an uphill edge.

Forward chaining And Backward chaining

- Forward chaining is a method of reasoning in artificial intelligence in which inference rules are applied to existing data to extract additional data until an endpoint (goal) is achieved.
- In this type of chaining, the inference engine starts by evaluating existing facts, derivations, and conditions before deducing new information.
- An endpoint (goal) is achieved through the manipulation of knowledge that exists in the knowledge base.



Cont…

- **Properties of forward chaining**
- The process uses a down-up approach (bottom to top).
- It starts from an initial state and uses facts to make a conclusion.
- This approach is data-driven.
- It's employed in expert systems and production rule system.
- A simple example of forward chaining can be explained in the following sequence.

A

A->B

B

A is the starting point. A->B represents a fact. This fact is used to achieve a decision B.

A practical example will go as follows;

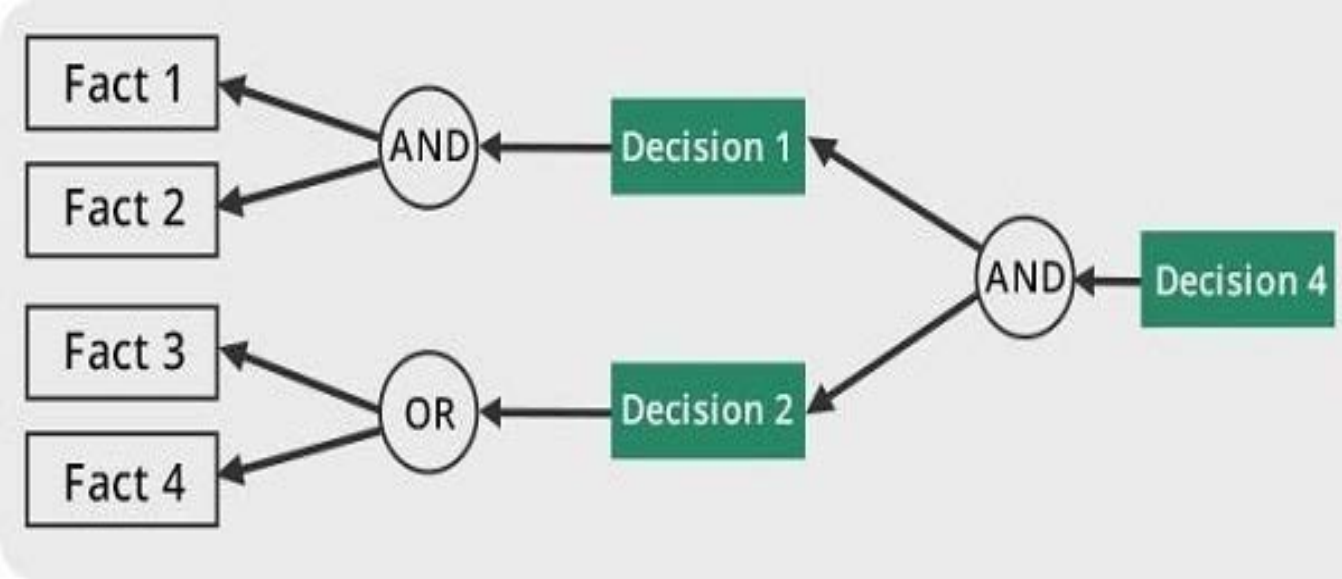
Tom is running (A)

If a person is running, he will sweat (A->B)

Therefore, Tom is sweating. (B)

BACKWARD CHAINING

- Backward chaining is a concept in artificial intelligence that involves backtracking from the endpoint or goal to steps that led to the endpoint. This type of chaining starts from the goal and moves backward to comprehend the steps that were taken to attain this goal.
- The backtracking process can also enable a person establish logical steps that can be used to find other important solutions.



Cont…

- **Properties of backward chaining**
- The process uses an up-down approach (top to bottom).
- It’s a goal-driven method of reasoning.
- The endpoint (goal) is subdivided into sub-goals to prove the truth of facts.
- A backward chaining algorithm is employed in inference engines, game theories, and complex database systems.

Cont...

- Example of backward chaining
- The information provided in the previous example (forward chaining) can be used to provide a simple explanation of backward chaining. Backward chaining can be explained in the following sequence.
- B
- A->B
- A
- B is the goal or endpoint, that is used as the starting point for backward tracking. A is the initial state. A->B is a fact that must be asserted to arrive at the endpoint B.
- A practical example of backward chaining will go as follows:
- Tom is sweating (B).
- If a person is running, he will sweat (A->B).
- Tom is running (A).

III UNIT

PLANNING WITH STATE SPACE SEARCH

INTRODUCTION

- The most straight forward approach of planning algorithm is , state space search

- Two types of search

  Forward state Space Search Algorithm (Progression)

  Backward state space Search Algorithm(Regression)

- Because the descriptions of actions in a planning problem specify both **preconditions and effects**

- It is possible to search in both directions: forward means initial state and backward means from the goal state.

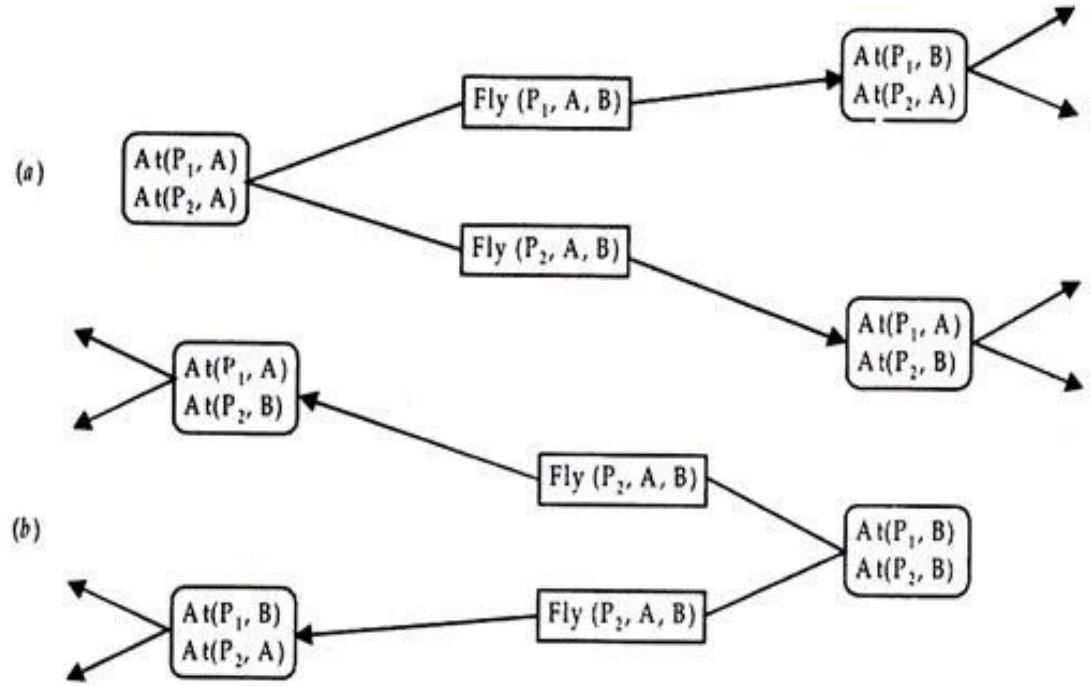


Fig. 8.5. *Two approaches to searching for a plan, (a) Forward (Progression) state-space search, starting in the initial state and using the problem's actions to search forward for the goal state, (b) Backward (regression) state-space search: a belief-state search starting at the goal state(s) and using the inverse of the actions to search backward for the initial state.*

Forward State-Space Search:

- Planning with forward state-space search is similar to the problem-solving approach. It is sometimes **called progression planning**, because it moves in the forward direction.
- We start with the problem's initial state, considering sequences of actions until we reach a goal state.

The formulation of planning problem as state-space search problems is as follows:

- i. The initial state of the search is the initial state from the planning problem. In general each state will be set of positive ground literals; literals not appearing are false.
- ii. The actions which are applicable to a state are all those whose preconditions are satisfied. The successor state resulting from an action is generated by adding the positive effect literals and deleting the negative effect literals.
- iii. The goal test checks whether the state satisfies the goal of the planning problem.
- iv. The step cost of each action

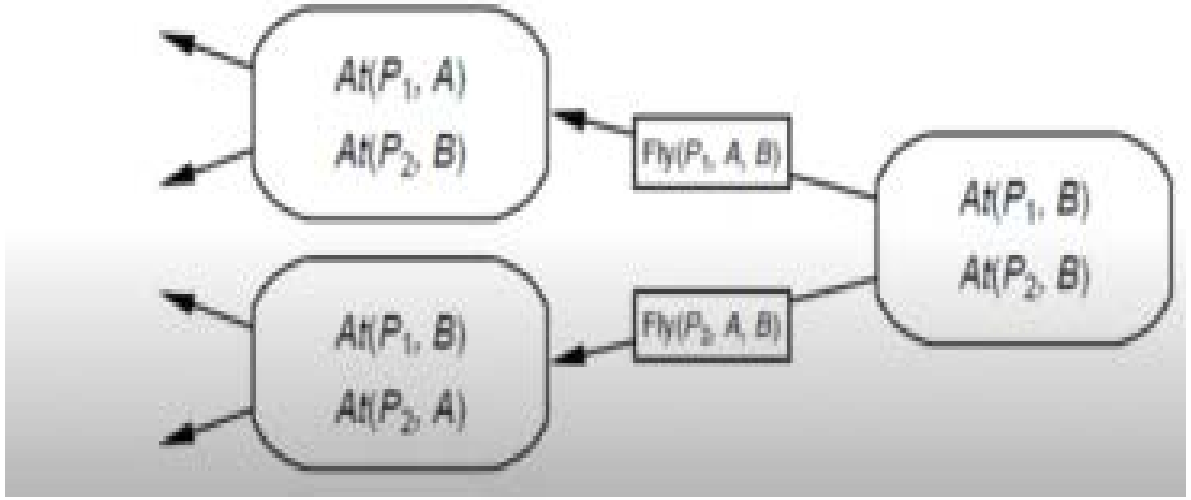
Cont…

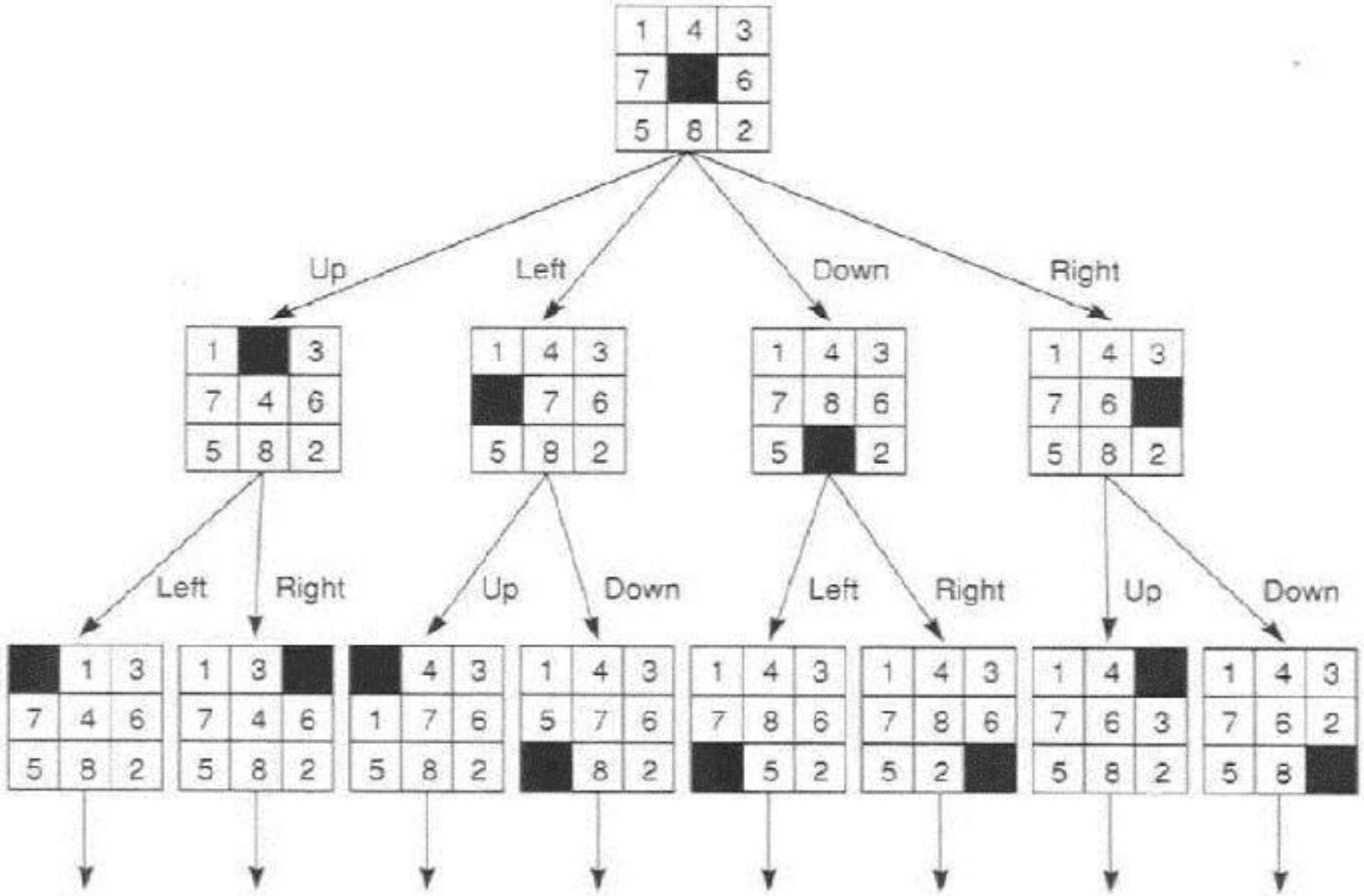
- From the early days of planning research it is known that forward state-space search is too inefficient to be practical. Mainly, this is because of a big branching factor since forward search does not address only relevant actions, (all applicable actions are considered).
- Consider for example, an air cargo problem with 10 airports, where each airport has 5 planes and 20 pieces of cargo.
- The goal is to move all the cargo at airport A to airport B. There is a simple solution to the problem: load the 20 pieces of cargo into one of the planes at A, fly the plane to B, and unload the cargo. But finding the solution can be difficult because the average branching factor is huge: each of the 50 planes can fly to 9 other airports, and each of the 200 packages can be either unloaded (if it is loaded), or loaded into any plane at its airport (if it is unloaded).
- On average, let's say there are about 1000 possible actions, so the search tree up to the depth of the obvious solution has about 1000 nodes. It is thus clear that a very accurate heuristic will be needed to make this kind of search efficient.

Backward (regression)state space search

- In regression state we start at the goal and apply the actions backward until we find the sequence of steps that reaches the initial state.

- It is called relevant states search because we only consider actions that are relevant to the goal (or current state).




Up
Left
Down
Right
Left
Right
Up
Down
Left
Right
Up
Down

Partial order planning

- A **partial-order plan** or **partial plan** is a plan which specifies all actions that need to be taken, but only specifies the order between actions when necessary. It is the result of a partial-order planner. A partial-order plan consists of four components:
- A set of **actions** (also known as **operators**).
- A **partial order** for the actions. It specifies the conditions about the order of some actions.
- A set of **causal links**. It specifies which actions meet which preconditions of other actions. Alternatively, a set of **bindings** between the variables in actions.
- A set of **open preconditions**. It specifies which preconditions are not fulfilled by any action in the partial-order plan.
- In order to keep the possible orders of the actions as open as possible, the set of order conditions and causal links must be as small as possible.
- A plan is a solution if the set of open preconditions is empty.

Cont…

Example

- For example, a plan for baking a cake might start:
- go to the store
- get eggs; get flour; get milk
- pay for all goods
- go to the kitchen
- This is a partial plan because the order for finding eggs, flour and milk is not specified, the agent can wander around the store reactively accumulating all the items on its shopping list until the list is complete.

Cont…

Disadvantages to partial-order planning

- One drawback of this type of planning system is that it requires a lot more computational power for each node.
- This higher per-node cost occurs because the algorithm for partial-order planning is more complex than others.
- When coding a robot to do a certain task, the creator needs to take into account how much energy is needed.
- Though a partial-order plan may be quicker it may not be worth the energy cost for the robot.
- The creator must be aware of and weigh these two options to build an efficient robot.

Cont…

Total Order planning (TOP)

- FSSS and BSSS are examples of TOP.
- They only explore linear sequences of actions from start to goal state, They cannot take advantage of problem decomposition,
- i.e. splitting the problem into smaller sub-problems and solving them individually.

Partial Order Planning (POP)

- It works on problem decomposition.
- It will divide the problem into parts and achieve these sub goals independently.

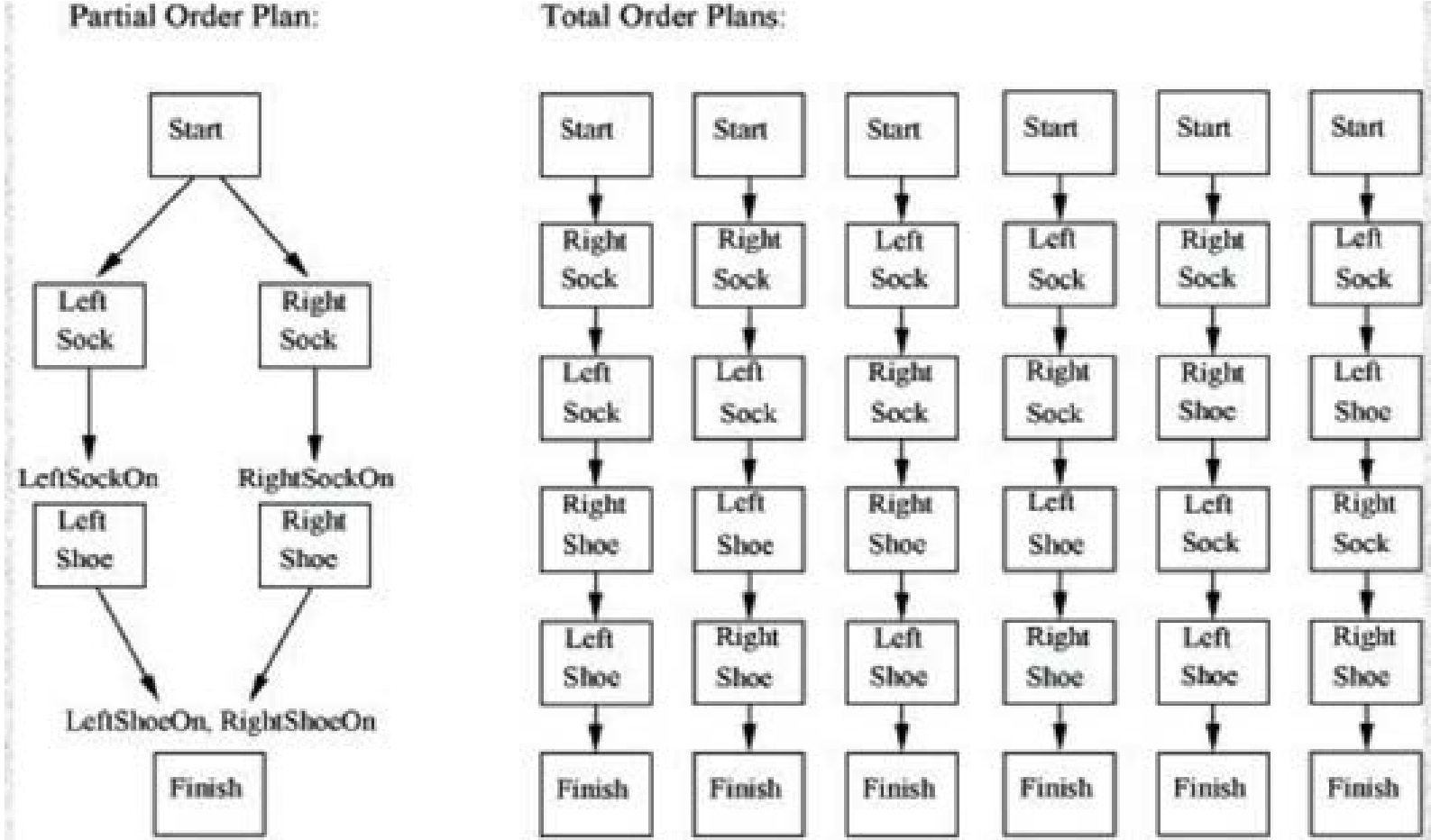
Cont…

- It solves the sub problems with sub plans and then combines these sub plans and reorders them based on requirements.
- In POP, ordering of the actions is partial. It does not specify which action will come first out of the two actions which are placed in the plan
- Let's look at this with the help of an example.
- The problem of wearing shoes can be performed through total order or partial order planning.

Cont…

- *Init: Barefoot*
- *Goal: RightShoeOn ^ LeftShoeOn*
- *Action: 1. RightShoeOn*
- *Precondition: RightSockOn*
- *Effect: RightShoeOn*
- *2. LeftShoeOn*
- *Precondition: LeftSockOn*
- *Effect: LeftShoeOn*
- *3. LeftSockOn*
- *Precondition: Barefoot*
- *Effect: LeftSockOn*
- *4. RightSockOn*
- *Precondition: Barefoot*
- *Effect: RightSockOn*

Cont…


Partial Order Plan:
Start
Left Sock
Right Sock
LeftSockOn
RightSockOn
Left Shoe
Right Shoe
LeftShoeOn, RightShoeOn
Finish
Total Order Plans:
Start
Right Sock
Left Sock
Left Shoe
Right Shoe
Finish

Cont…

- The TOP consists of six sequences, one of which can be taken in order to reach the finish state.
- However, the POP is less complex.
- It combines two action sequences.
- The first branch covers the left sock and left shoe.
- To wear left shoe, wearing the left sock is a precondition.
- Similarly the second branch covers the right sock and right show. Once these actions are taken, we achieve our goal and reach the finish state.

PLANNING GRAPHS

- A planning graph is a sequence of levels corresponding to “time steps,” alternating between “state levels” Si and “action levels” Ai.
- It starts with state level S0, which contains the literals true of the initial state.
- An action is in Ai if its preconds. are in Si.
- It has edges from its preconds. in Si and to its effects in Si+1.
- Also, each literal in Si has a persistence edge to the same literal in Si+1.

CONT...

- Mutual Exclusion (Mutex) Links Two actions in Ai have a mutex link if a precondition or effect of one action conflicts with a precondition or effect of the other action.

- Two literals in Si+1 have a mutex link if one negates the other or if every pair of actions in Ai achieving them are mutex. An action cannot be in Ai if any two preconds. in Si are mutex

CONT...

- “Simple” Planning Graph
- Example
- Init(have(Cake))
- Goal(have(Cake) ∧ eaten(Cake))
- Action(eat(Cake),
- Precond: have(Cake)
- Effect: eaten(Cake) ∧ ¬have(Cake))
- Action(bake(Cake),
- Precond: ¬have(Cake)
- Effect: have(Cake))

Planning and Acting in the Real World

- *Planning and scheduling the operations*
  - Spacecraft
  - Factories
  - Military campaigns

CONT...

- **Time, Schedules, and Resources**
- Classical planning representation is about
  - *What to do*
  - *In what order*
  - *Cannot talk about time*
    - How long an action takes
    - When the action occurs

CONT...

- Planning to produce schedules
  - *Assign initial and final times*
  - *Schedule for an airline, assign planes to flights, departure and arrival times*
- Resource constraints
  - *Limited number of staff in an airline (i.e. pilots)*
  - *Staf cannot be in two places at the same time*
-

CONT...

- A **job-shop scheduling problem**
  - *Consists of a set of **jobs***
    - Each job consists of a set of **actions**
  - *Actions have ordering constraints among them*
  - *Actions (each of them) have a **duration** and a set of **resource constraints***

CONT...

$$Jobs(\{AddEngine1 \prec AddWheels1 \prec Inspect1\},$$
$$\{AddEngine2 \prec AddWheels2 \prec Inspect2\})$$

$$Resources(EngineHoists(1), WheelStations(1), Inspectors(2), LugNuts(500))$$

$$Action(AddEngine1, \text{DURATION:}30,$$
$$\text{USE:}\ EngineHoists(1))$$
$$Action(AddEngine2, \text{DURATION:}60,$$
$$\text{USE:}\ EngineHoists(1))$$
$$Action(AddWheels1, \text{DURATION:}30,$$
$$\text{CONSUME:}\ LugNuts(20), \text{USE:}\ WheelStations(1))$$
$$Action(AddWheels2, \text{DURATION:}15,$$
$$\text{CONSUME:}\ LugNuts(20), \text{USE:}\ WheelStations(1))$$
$$Action(Inspect_i, \text{DURATION:}10,$$
$$\text{USE:}\ Inspectors(1))$$

Figure 1 A job-shop scheduling problem for assembling two cars, with resource constraints. The notation $A \prec B$ means that action A must precede action B.

UNCERTAIN KNOLEDGE AND REASONING

IV UNIT

Uncertainty

- We have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates.

- With this knowledge representation, we might **write A→B, which means if A is true then B is true,** but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called **Uncertainty.**

- So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

CONT...

Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

1. Information occurred from unreliable sources.
2. Experimental Errors
3. Equipment fault
4. Temperature variation
5. Climate change.

Probabilistic Reasoning

Probabilistic reasoning:

- Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

- In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as

1. "It will rain today,“
2. "behavior of someone for some situations,“3.
3. "A match between two teams or two players.“

- These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Cont...

- **Need of probabilistic reasoning in AI:**
- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.
- In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:
- **Bayes' rule**
- **Bayesian Statistics**

Cont…

- let's understand some common terms:

- **Probability:** Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

1. $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
2. $P(A) = 0$, indicates total uncertainty in an event A.
3. $P(A) = 1$, indicates total certainty in an event A.

Cont…

- We can find the probability of an uncertain event by using the below formula.

$$\text{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- P(¬A) = probability of a not happening event.
- P(¬A) + P(A) = 1.
- **Event:** Each possible outcome of a variable is called an event.

Cont…

- Conditional probability:
- Conditional probability is a probability of occurring an event when another event has already happened.
- Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

- **Where P($A \wedge B$)= Joint probability of a and B**
- **P(B)= Marginal probability of B.**

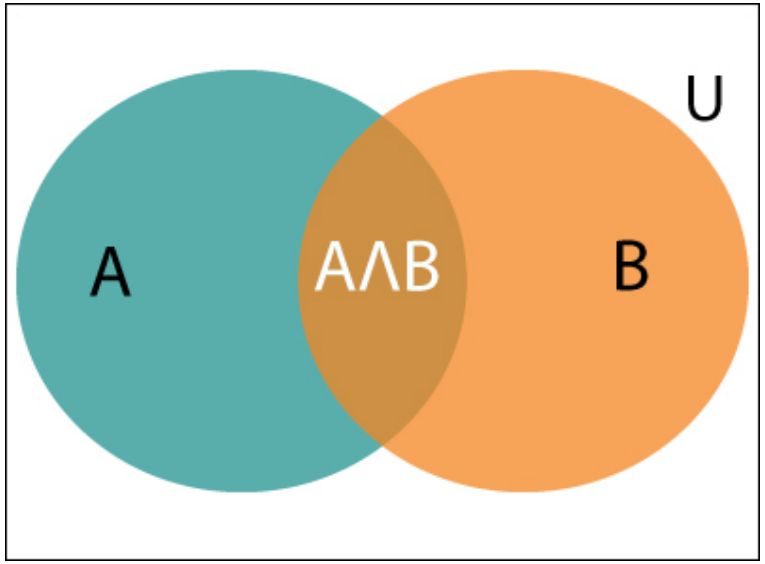
Cont...

- If the probability of A is given and we need to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

- It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A∧*B*) by P(B)**.

Cont…



Cont…

- In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

- Solution??????????????????

Cont…

- **Solution:**
- Let, A is an event that a student likes Mathematics
- B is an event that a student likes English.

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

- **Hence, 57% are the students who like English also like Mathematics.**

Bayes' theorem in Artificial intelligence

Bayes' theorem:

Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which **determines the probability of an event with uncertain knowledge.**

- In probability theory, it relates the conditional probability and marginal probabilities of two random events.
- Bayes' theorem was named after the British mathematician **Thomas Bayes**. The **Bayesian inference** is an application of Bayes' theorem, which is fundamental to Bayesian statistics.
- It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

Cont…

- **Example**: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.
- Bayes' theorem can be derived using product rule and conditional probability of event A with known event B:
- As from product rule we can write:

$$P(A \wedge B) = P(A|B)\, P(B) \text{ or}$$

- Similarly, the probability of event B with known event A:

$$P(A \wedge B) = P(B|A)\, P(A)$$

- Equating right hand side of both the equations, we will get:

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)} \quad \ldots\ldots(a)$$

- The above equation (a) is called as **Bayes' rule** or **Bayes' theorem**. This equation is basic of most modern AI systems for **probabilistic inference**.
- It shows the simple relationship between joint and conditional probabilities

Cont…

- **Question: From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King|Face), which means the drawn face card is a king card.**
- **Solution:**

$$P(king|face) = \frac{P(Face|king) * P(King)}{P(Face)} \quad .....(i)$$

- P(king): probability that the card is King= 4/52= 1/13
- P(face): probability that a card is a face card= 3/13
- P(Face|King): probability of face card when we assume it is a king = 1
- Putting all values in equation (i) we will get:

$$P(king|face) = \frac{1 * (\frac{1}{13})}{(\frac{3}{13})} = 1/3,$$ **it is a probability that a face card is a king card.**

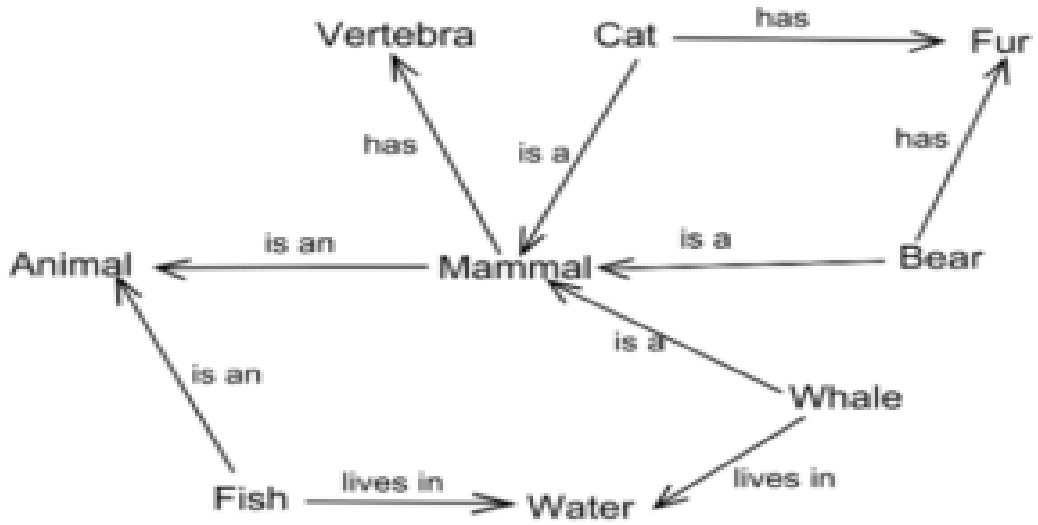
Cont…

- **Following are some applications of Bayes' theorem:**
- It is used to calculate the next step of the robot when the already executed step is given.
- Bayes' theorem is helpful in weather forecasting.

Semantic Networks

- How Important Are Semantic Networks In Artificial Intelligence
- A semantic network is a graphic notation for representing knowledge in patterns of interconnected nodes.
- Semantic networks became popular in artificial intelligence and natural language processing only because it represents knowledge or supports reasoning.
- These act as another alternative for predicate logic in a form of knowledge representation.
- The structural idea is that knowledge can be stored in the form of graphs, with nodes representing objects in the world, and arcs representing relationships between those objects.

Cont…

- AI agents have to store and organize information in their memory. One of the ways they do this is by using *semantic networks*.
- Semantic networks are a way of representing relationships between objects and ideas.
- For example, a network might tell a computer the relationship between different animals (a cat IS A mammal, a cat HAS whiskers). Below is an example image of a semantic network.
- 

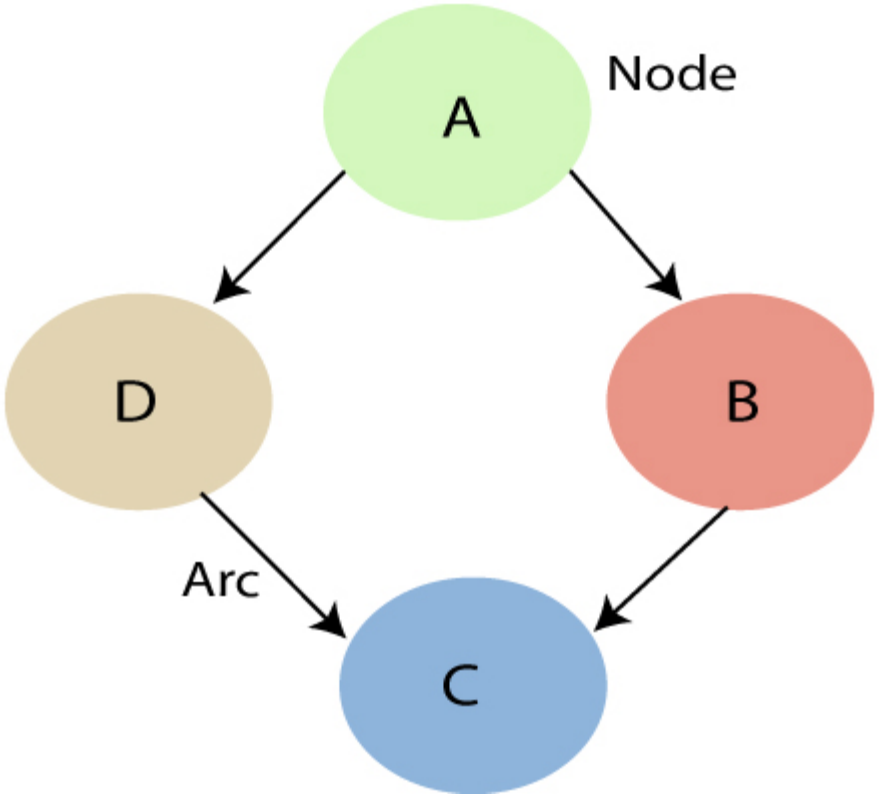
Bayesian Belief Network in artificial intelligence

- Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:
- "A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."
- It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.
- Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

Cont…

- Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.
- Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:
- **Directed Acyclic Graph**
- **Table of conditional probabilities.**
- The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.

Cont…



Cont...

- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.
  These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
  - **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
  - **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
  - **Node C is independent of node A.**

Cont…

- The Bayesian network has mainly two components:
- **Causal Component**
- **Actual numbers**
- Each node in the Bayesian network has condition probability distribution **$P(X_i \mid Parent(X_i))$**, which determines the effect of the parent on that node.
- Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

Joint probability distribution:

- If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.

Cont…

- Joint probability distribution:
- If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.
- **$P[x_1, x_2, x_3,....., x_n]$**, it can be written as the following way in terms of the joint probability distribution.
- **$= P[x_1| x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]$**
- **$= P[x_1| x_2, x_3,....., x_n]P[x_2|x_3,....., x_n]....P[x_{n-1}|x_n]P[x_n].$**
- In general for each variable Xi, we can write the equation as:

$$P(Xi|Xi\text{-}1,\ldots\ldots\ldots, X1) = P(Xi\ |Parents(Xi\ ))$$

Cont…

- Explanation of Bayesian network:
- Let's understand the Bayesian network through an example by creating a directed acyclic graph:

- **Example:** Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Cont…

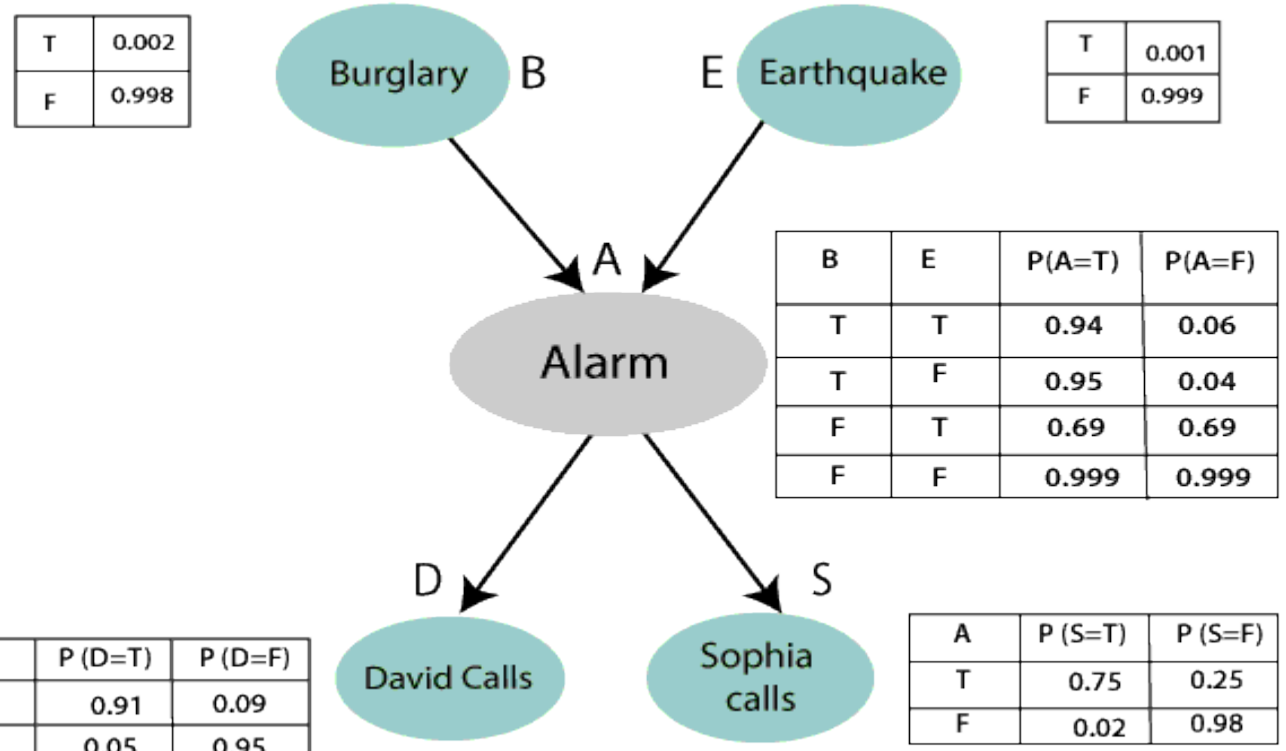
- **Problem:**
- **Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.**
- **Solution:**

- The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.
- The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.
- The conditional distributions for each node are given as conditional probabilities table or CPT.
- Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
- In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

Cont…

- **List of all events occurring in this network:**
- **Burglary (B)**
- **Earthquake(E)**
- **Alarm(A)**
- **David Calls(D)**
- **Sophia calls(S)**
- We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:
- **P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]**
- **=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]**
- **= P [D| A]. P [S| A, B, E]. P[A, B, E]**
- **= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]**
- **= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]**

Cont…



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

Cont…

- Let's take the observed probability for the Burglary and earthquake component:

P(B= True) = 0.002, which is the probability of burglary.

P(B= False)= 0.998, which is the probability of no burglary.

P(E= True)= 0.001, which is the probability of a minor earthquake

P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

- **Conditional probability table for Alarm A:**
- The Conditional probability of Alarm A depends on Burglar and earthquake:

Cont…

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Cont...

- **Conditional probability table for David Calls:**
- The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

Cont...

- **Conditional probability table for Sophia Calls:**
- The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

Cont…

- From the formula of joint distribution, we can write the problem statement in the form of probability distribution:
- **P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).**
- = 0.75* 0.91* 0.001* 0.998*0.999
- **= 0.00068045.**
- **Hence, a Bayesian network can answer any query about the domain by using Joint distribution.**
- **The semantics of Bayesian Network:**

Cont…

- There are two ways to understand the semantics of the Bayesian network, which is given below:
- **1. To understand the network as the representation of the Joint probability distribution.**
- It is helpful to understand how to construct the network.
- **2. To understand the network as an encoding of a collection of conditional independence statements.**
- It is helpful in designing inference procedure.

Unit V

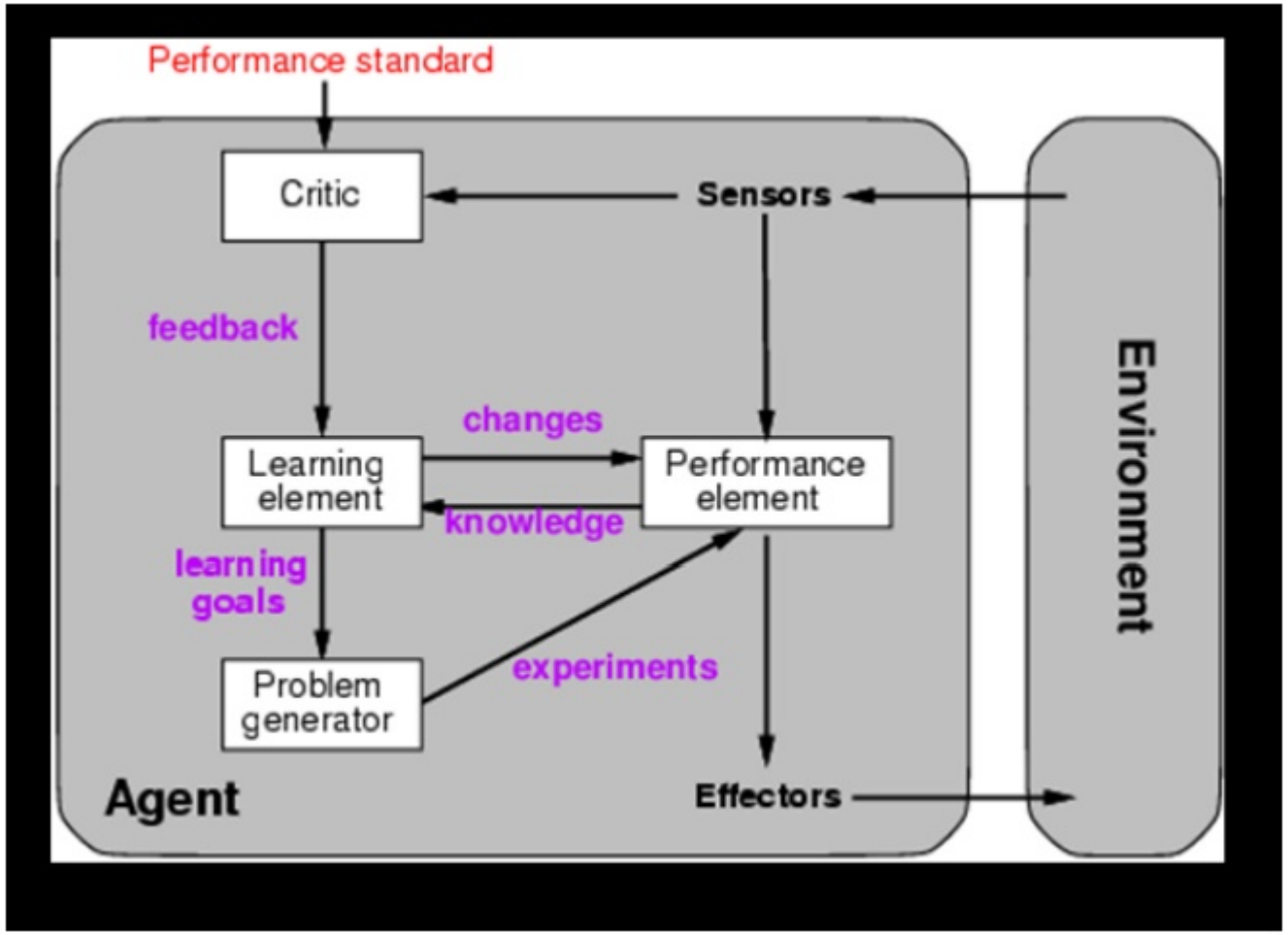
Learning

LEARNING FROM OBSERVATIONS:

What is learning?

- Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more effectively the next time.
- Learning is essential for unknown environments,
- i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
- i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Cont…


Performance standard
Critic
Sensors
feedback
changes
Learning element
Performance element
knowledge
learning goals
Problem generator
experiments
Agent
Effectors
Environment

Inductive Learning in Artificial Intelligence

- Inductive Learning, also known as Concept Learning, is how A.I. systems attempt to use a generalized rule to carry out observations.
- Inductive Learning Algorithms (APIs) are used to generate a set of classification rules. These generated rules are in the "If this, then that" format.
- **The Fundamental Concept of Inductive Learning**
- There are two methods for obtaining knowledge in the real world: first, from domain experts, and second, from machine learning.
- Domain experts are not very useful or reliable for large amounts of data. As a result, we are adopting a machine learning approach for this project.

Cont…

Some practical examples of induction are:

Credit risk assessment.

- The x is the property of the customer.
- The f(x) is credit approved or not.

Disease diagnosis.

- The x is the characteristics of a given patient.
- The f(x) is the patient's disease.

Face recognition.

- The x are bitmaps of the faces we want to recognize.
- The f(x) is a name assigned to that face.

Cont…

Inductive Learning may be helpful in the following four situations:

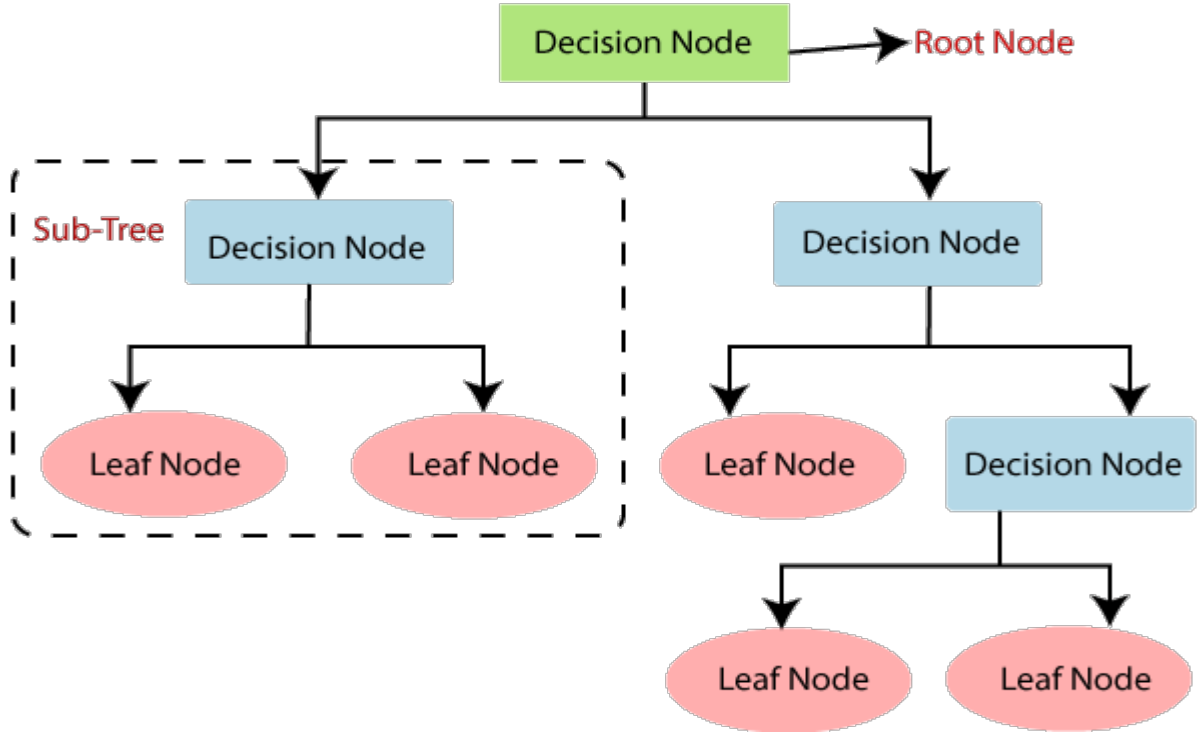
- **Problems in which no human expertise is available**. People cannot write a program to solve a problem if they do not know the answer. These are areas ripe for exploration.
- **Humans can complete the task, but no one knows how to do it.** There are situations in which humans can do things that computers cannot or do not do well. Riding a bike or driving a car are two examples.
- **Problems where the desired function is frequently changing**. Humans could describe it and write a program to solve it, but the problem changes too frequently. It is not economical. The stock market is one example.
- **Problems where each user requires a unique function.** Writing a custom program for each user is not cost-effective. Consider Netflix or Amazon recommendations for movies or books.

Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**

Cont…

- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub trees.
- Below diagram explains the general structure of a decision tree:



Cont…

Decision Tree Terminologies

- Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- Branch/Sub Tree: A tree formed by splitting the tree.
- Pruning: Pruning is the process of removing the unwanted branches from the tree.
- Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

Cont...

- **How does the Decision Tree algorithm Work?**
- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

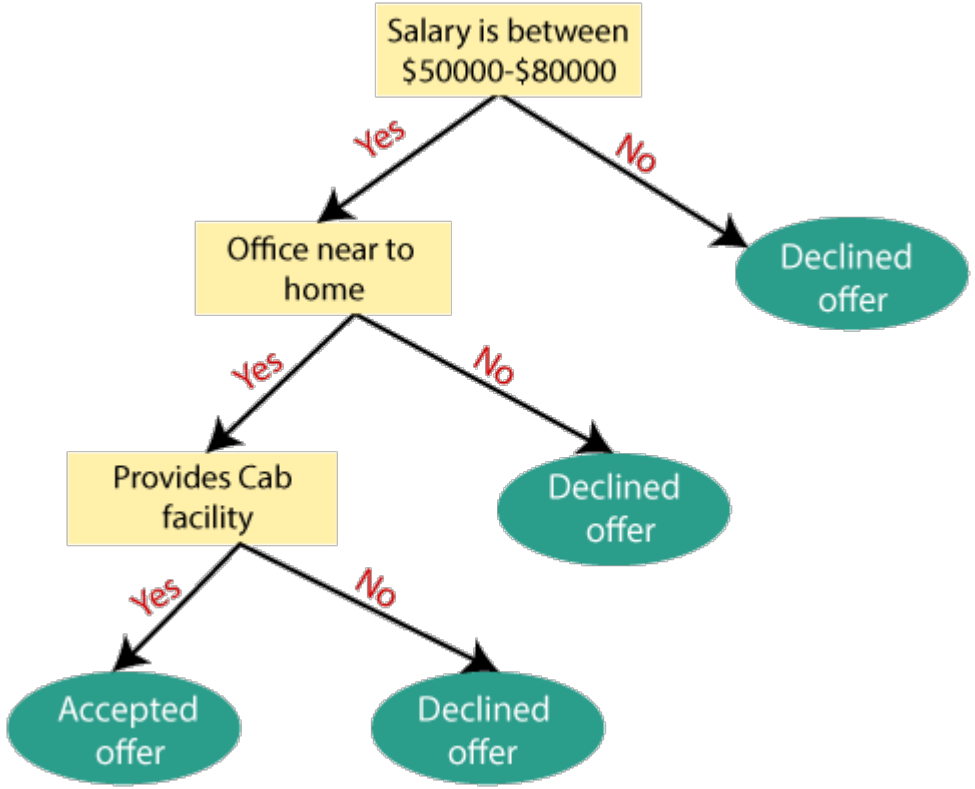
Cont…

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Cont…

- **Example:** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:
-

Cont…


Salary is between $50000-$80000
Yes
No
Office near to home
Declined offer
Yes
No
Provides Cab facility
Declined offer
Yes
No
Accepted offer
Declined offer

Explanation based Learning

- Explanation based learning has ability to learn from a single training instance. Instead of taking more examples the explanation based learning is emphasized to learn a single, specific example.
- For example, consider the Ludoo game. In a Ludoo game, there are generally four colors of buttons. For a single color there are four different squares.
- Suppose the colors are red, green, blue and yellow. So maximum four members are possible for this game.
- Two members are considered for one side (suppose green and red) and other two are considered for another side (suppose blue and yellow).
- So for any one opponent the other will play his game. A square sized small box marked by symbols one to six is circulated among the four members.

Cont…

- The number one is the lowest number and the number six is the highest for which all the operations are done.
- Always any one from the 1st side will try to attack any one member in the 2nd side and vice versa.
- At any instance of play the players of one side can attack towards the players of another side.
- Likewise, all the buttons may be attacked and rejected one by one and finally one side will win the game.
- Here at a time the players of one side can attack towards the players of another side.
- So for a specific player, the whole game may be affected. Hence we can say that always explanation based learning is concentrated on the inputs like a simple learning program, the idea about the goal state, the idea about the usable concepts and a set of rules that describes relationships between the objects and the actions.

Cont…

- Consider the problem of learning the concept bucket. We want to generalize from a single example of a bucket. At first collect the following informations.

1. **Input Examples:**

$$\text{Owner (object, X)} \wedge \text{has part (object, Y)} \wedge \text{is(object, Deep)} \wedge \text{Color (Object, Green)}$$

$$\wedge \ldots\ldots \text{(Where Y is any thin material)}$$

2. **Domain Knowledge:**

$$\text{is (a, Deep)} \wedge \text{has part (a, b)} \wedge \text{is a(b, handle)} \rightarrow \text{liftable (a)}$$

$$\text{has part (a, b)} \wedge \text{is a (b, Bottom)} \wedge \text{is (b, flat)} \rightarrow \text{Stable (a)}$$

$$\text{has part (a, b)} \wedge \text{is a (b, Y)} \wedge \text{is (b, Upward} - \text{pointing)} \rightarrow \text{Open} - \text{vessel (a)}$$

Cont...

3. **Goal:** Bucket

B is a bucket if B is liftable, stable and open-vessel.

4. Description of Concept: These are expressed in purely structural forms like Deep, Flat, rounded etc.

Cont…

4. **Description of Concept:** These are expressed in purely structural forms like Deep, Flat, rounded etc.

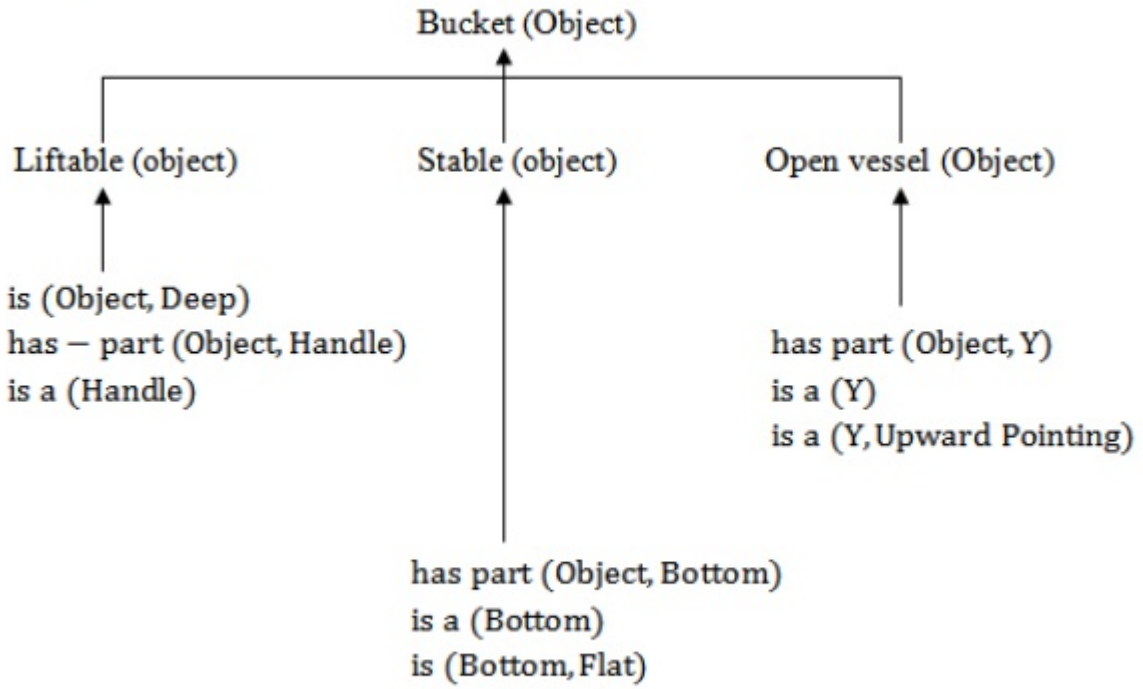


Figure An explanation of BUCKET Object

Statistical Learning Method

- Statistical Learning is a set of tools for understanding data.
- These tools broadly come under two classes: supervised learning & unsupervised learning.
- Generally, supervised learning refers to predicting or estimating an output based on one or more inputs.
- Unsupervised learning, on the other hand, provides a relationship or finds a pattern within the given data without a supervised output.

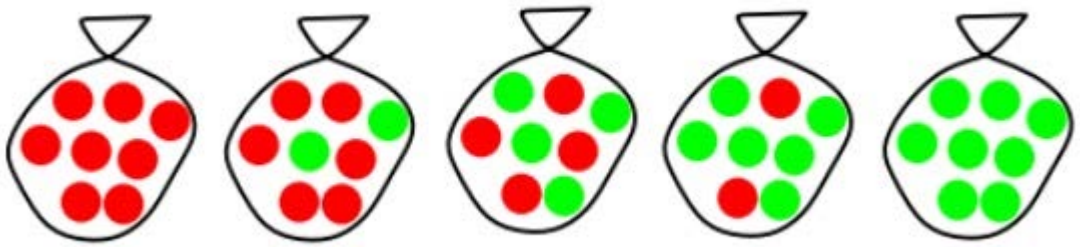
Cont…

- View learning as Bayesian updating of a probability distribution over the hypothesis space
- H is the hypothesis variable, values h1, h2, . . ., prior P(H) jth observation dj gives the outcome of random variable Dj training data d = d1, . . . , dN
- Given the data so far, each hypothesis has a posterior probability:
- $P(h_i|d) = \alpha P(d|h_i)P(h_i)$
- where P(d|hi) is called the likelihood

cont...

- Example
- Suppose there are five kinds of bags of candies: 10% are h1: 100% cherry candies
- 20% are h2: 75% cherry candies + 25% lime candies 40% are h3: 50% cherry candies + 50% lime candies 20% are h4: 25% cherry candies + 75% lime candies 10% are h5: 100% lime candies

Cont…



Cont...

1.The true hypothesis eventually dominates the Bayesian prediction given that the true hypothesis is in the prior

2.The Bayesian prediction is optimal, whether the data set be small or large

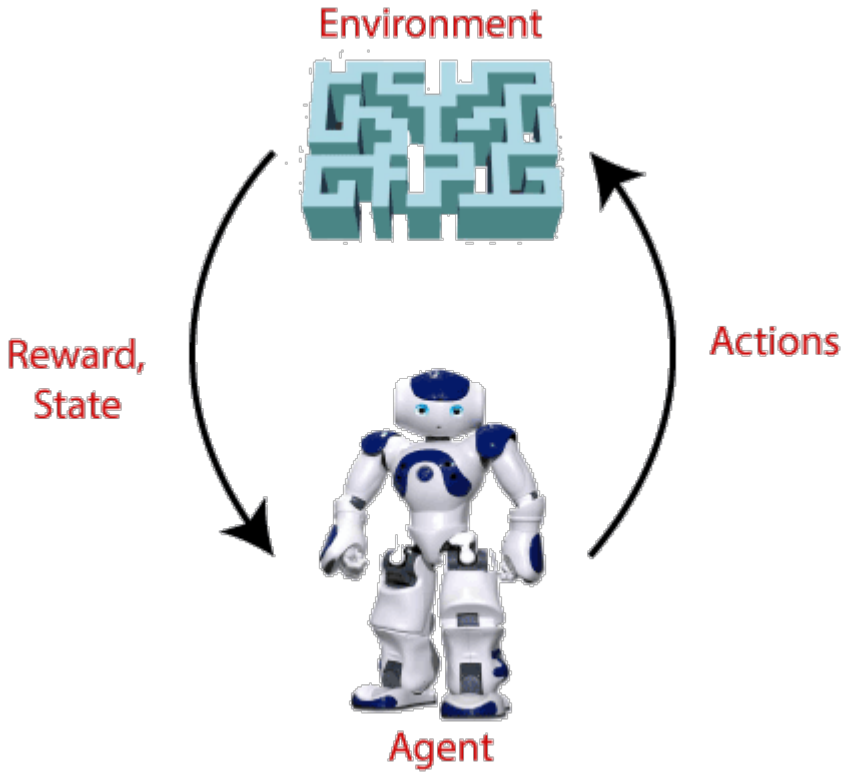
Reinforcement Learning

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.

Cont…

- **Example:** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
- The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.

Cont...



Cont...

- Terms used in Reinforcement Learning
- Agent(): An entity that can perceive/explore the environment and act upon it.
- Environment(): A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- Action(): Actions are the moves taken by an agent within the environment.
- State(): State is a situation returned by the environment after each action taken by the agent.

Cont…

- Reward(): A feedback returned to the agent from the environment to evaluate the action of the agent.
- Policy(): Policy is a strategy applied by the agent for the next action based on the current state.
- Value(): It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- Q-value(): It is mostly similar to the value, but it takes one additional parameter as a current action (a).

Neural Net Learning

- **Neural Networks** is one of the most significant discoveries in history.
- Neural Networks can solve problems that can't be solved by algorithms:
- Medical Diagnosis
- Face Detection
- Voice Recognition
- **Neural Networks** is the essence of **Deep Learning**.

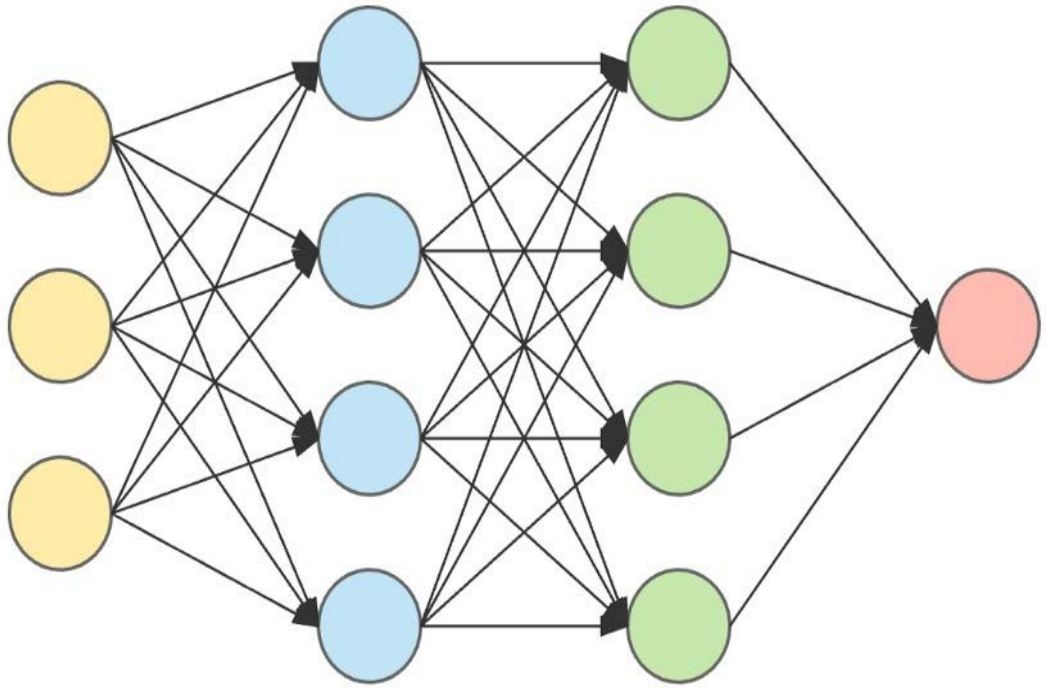
Neurons

- Scientists agree that our brain has around 100 billion neurons.
- These neurons have hundreds of billions connections between them.

Cont…

Cont…

- The Neural Network Model
- Input data (Yellow) are processed against a hidden layer (Blue) and modified against another hidden layer (Green) to produce the final output (Red).



genetic algorithm

- A genetic algorithm is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature."
- It is used to solve optimization problems in machine learning.
- It is one of the important algorithms as it helps solve complex problems that would take a long time to solve.