

Chapter 1

Introduction To Artificial Intelligence

Intelligence

The ability to reason

The ability to understand

The ability to create

The ability to Learn from experience

The ability to plan and execute complex tasks

The intelligent behavior may include

>Everyday Tasks:

recognize a friend, recognize who is calling, translate from one language to another, interpret a photograph, talk, and cook a dinner

>Formal Tasks:

prove a logic theorem, geometry, calculus, play chess, checkers, or Go

>Expert Tasks:

engineering design, medical designers, financial analysis

Artificial Intelligence

- AI is the branch of computer science concerned with making computers behave like humans.
- AI is the science and engineering of making intelligent machines, especially intelligent computer programs.
- The process may include
 - ✗ Learning (Gaining of information and rules for using the information)
 - ✗ Reasoning (Using the rules to reach approximate or definite conclusions)
 - ✗ Self-Correction

According to Barr and Feigenbaum:

“Artificial Intelligence is the part of computer science concerned with designing intelligence computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior.”

According to Elaine Rich:

“AI is the study of how to make computers do things at which, at the moment, people are better”

An AI system should have

- Capability to provide reason about something
- Capability of natural language processing
- Capability of learning past experience
- Capability of self-correction

Approach / View of AI fall into four categories

Thinking humanly

Acting humanly

Thinking rationally

Acting rationally

	Humanly	Rationally
Thinking	Thinking humanly – cognitive modeling. Systems should solve problems the same way humans do.	Thinking rationally – the use of logic. Need to worry about modeling uncertainty and dealing with complexity.
Acting	Acting humanly – the Turing Test approach.	Acting rationally – the study of rational agents: agents that maximize the expected value of their performance measure given what they currently know.

The major problem with AI is what's known as
‘garbage in, garbage out’

The Turing Test approach

The Turing Test is a method for determining whether or not a computer is capable of thinking like a human.

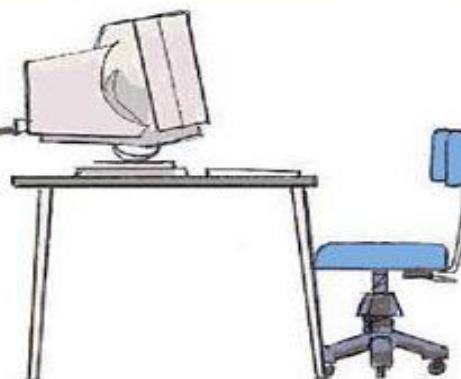
The test is named after Alan Turing, an English mathematician who pioneered artificial intelligence during the 1940s and 1950s, and who is credited with devising the original version of the test.

According to this kind of test, a computer is deemed to have artificial intelligence if it can mimic human responses under specific conditions.

Interrogator



Respondent A



Respondent B

Consider the following setting.

There are two rooms, A and B. One of the rooms contains a computer. The other contains a human. The interrogator is outside and does not know which one is a computer. He can ask questions through a teletype and receives answers from both A and B. The interrogator needs to identify whether A or B are humans. To pass the Turing test, the machine has to fool the interrogator into believing that it is human.

To pass a Turing test, a computer must have following capabilities:

- Natural Language Processing:
Must be able to communicate in English successfully
- Knowledge representation:
To store what it knows and hears.
- Automated reasoning:
Answer the Questions based on the stored information.
- Machine learning:
Must be able to adapt in new circumstances

On one side of the argument, **human-like interaction is seen as absolutely essential** to human-like intelligence. **A successful AI is worthless if its intelligence lies trapped in an unresponsive program.**

Turing's test deliberately avoided direct physical interaction between the interrogator and the computer, because physical simulation of a person is unnecessary for intelligence.

So, **total Turing Test includes a video signal** so that the interrogator can test the subject's perceptual abilities, as well as the opportunity for the interrogator to pass physical objects "through the hatch."

To pass the total Turing Test, the computer will need

- computer vision** to perceive objects, and
- robotics** to move them about

AI and related fields/ Foundations of AI

Philosophy:

Logic, reasoning, mind as a physical system, foundations of learning, language and rationality.

Mathematics:

Formal representation and proof algorithms, computation, undesirability, intractability, probability.

Psychology:

Adaptation, phenomena of perception and motor control.

Economics:

Formal theory of rational decisions, game theory.

Linguistics:

Knowledge representation, grammar

Neuro science:

Physical substrate for mental activities

Control theory:

Homeostatic systems, stability, optimal agent design

Brief History of AI

The term “Artificial Intelligence” was used for the first time in 1956 by an American scientist John McCarthy who is referred to as the Father of AI.

McCarthy also came up with a programming language called LISP (i.e. List-Processing), which is still used to program computer in AI that allow the computer to learn.

Further, the major achievements can be listed as below:

1943	First electronic computer “Colossus” was developed.
1949	First commercial stored program computer was developed.
1950	<ul style="list-style-type: none"> - Alan Turing proposes the Turing test as a measure of machine intelligence. - Claude Shannon published a detail analysis of chess playing as search. - Isaac Asimov published his three laws of Robotics.
1951	The first working AI programs were written to run on the Ferranti Mark machine of the University of Manchester; a checkers-playing program written by Christopher Stavechey and a chess-playing program is written by Dietrich Prinz
1955	The first Dartmouth college summer AI conference is organized by John McCarthy, Marvin Minsky, Nathan Rochester of IBM and Claude Shannon.
1956	<ul style="list-style-type: none"> - The name artificial intelligence is used for the 1st time as the topic of the second Dartmouth Conference, organized by John McCarthy.

	<ul style="list-style-type: none"> - The first demonstration of the Logic Theorist (LT) written by Allen Newell, J.C. Shaw and Merbart Simon pus is called the first AI program
1957	The general problem Solver (GPS) demonstrated by Newell, Shaw and Simon
1958	John McCarthy at MIT invented the Lisp Programming Language.
1959	<ul style="list-style-type: none"> - John McCarthy and Marvin Minsky founded the MIT AI Lab. - First industrial robot company, animation was established.
1972	Prolog programming language was developed by Alain Colmerauer
1980	First National Conference of the American Association for Artificial Intelligence (AAAI) was held at Stratford.
Mid 1980's	Neural networks become widely used with the Back propagation algorithm.
1994	AI system exist in real environments with real sensory inputs (i.e. Intelligent Agents)
1997	First time AI system controlled a spacecraft named “Deep Space II”
2007	Checkers is solved by a team of researchers of the University of Alberta.
Present	Programmers are still trying to develop a computer which can successfully pass the “Turing Test”.

Difference between AI and NI

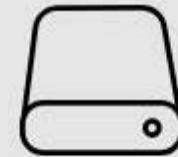
1#. Energy Efficiency

**Human
Intelligence**



25
watts human brain

**Artificial
Intelligence**



2 watts for modern machine
learning machine.

Difference between AI and NI

2#. Universal

Human Intelligence



Humans usually learn how to manage hundreds of different skills during life.

Artificial Intelligence



While consuming kilowatts of energy, this machine is usually designed for a few tasks.

Difference between AI and NI

3#. Multi Tasking

Human Intelligence



Human
worker work on multiple
responsibilities.

Artificial Intelligence



The time needed to teach system on
each and every responsibility is
considerably high.

Difference between AI and NI

4#. Decision Making

Human Intelligence



Humans have the ability to learn decision making from experienced scenarios.

Artificial Intelligence

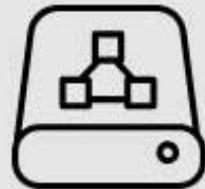


Even the most advanced robots can hardly compete in mobility with 6 years old child. And this results we have after 60 years of research and development.

Difference between AI and NI

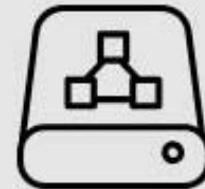
5#. State

**Human
Intelligence**



Brains
are Analogue

**Artificial
Intelligence**



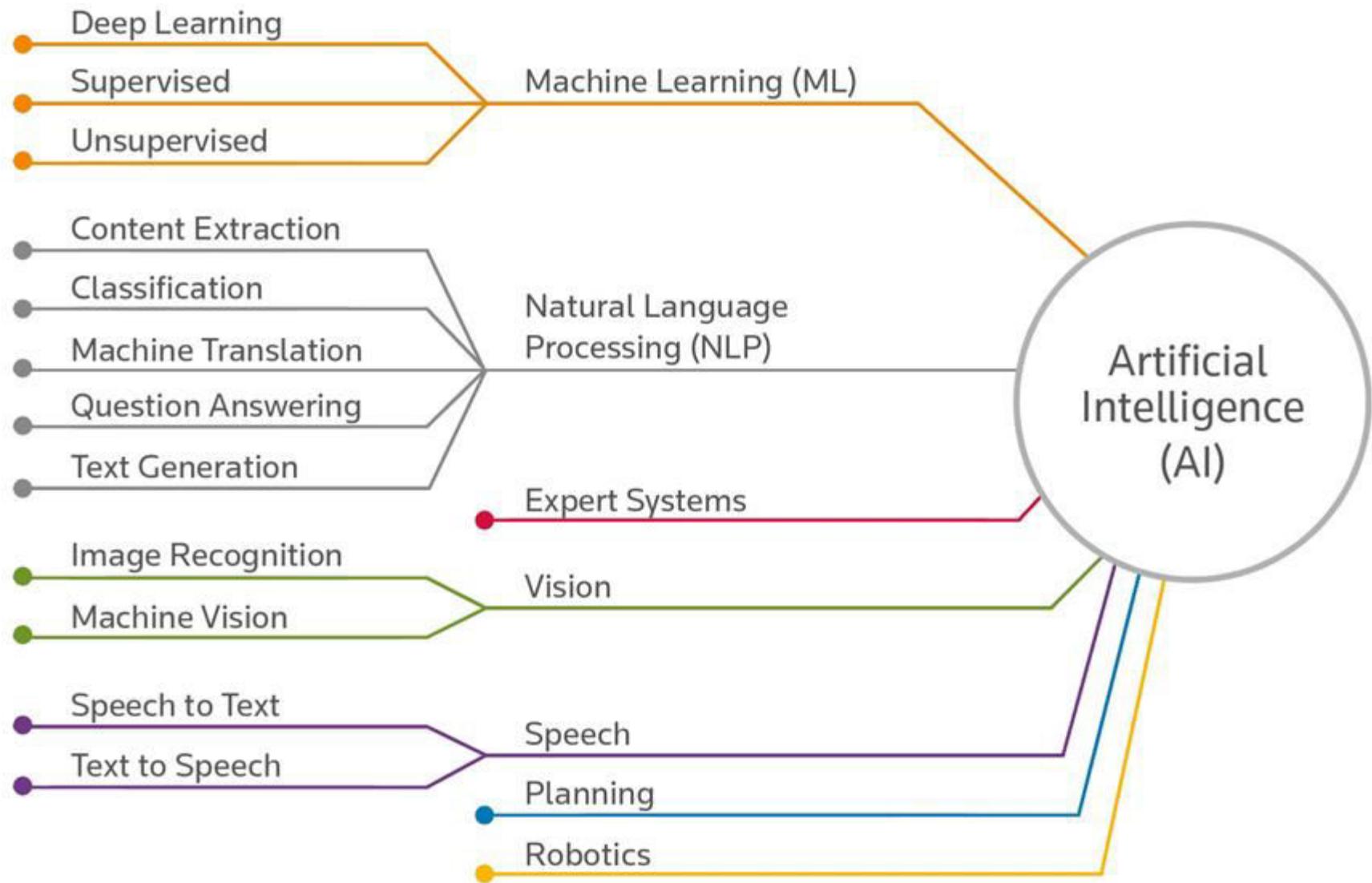
Computers
are digital

Application of AI

Artificial intelligence has been used in a wide range of fields including medical diagnosis, stock trading, robot control, law, remote sensing, scientific discovery and toys.

Many thousands of AI applications are deeply embedded in the infrastructure of every industry.

In the late 90s and early 21st century, AI technology became widely used as elements of larger systems, but the field is rarely credited for these successes.



Game Playing

Machines can **play master level chess**. There is some AI in them, but they well against people mainly through **brute force** method, looking at hundreds of thousands of positions.

Speech Recognition

It is possible to instruct some computers using speech. In 1990s, computer speech recognition reached a practical level for limited purposes.

Understanding Natural Language

To perform many natural language processing tasks such as machine translation, summarization, information extraction, word sense disambiguation need the AI in machine.

Computer Vision

Computer vision is concerned with the theory behind artificial system that extract information from images. The image data can take many forms such as videos sequences views from multiple cameras and data from a medical scanner.

Expert System

Expert system needs the AI to perform its task. One of the first expert system was MYCIN in 1974 which diagnosis bacterial infections of the blood and suggests treatments.

Finance

Financial institutions have long used artificial neural network systems to detect charges or claims outside of the norm, flagging these for human investigation.

Use of AI in banking can be traced back to 1987 when Security Pacific National Bank in USA set-up a Fraud Prevention Task force to counter the unauthorized use of debit cards.

Hospitals and medicine

Artificial neural networks are used as clinical decision support systems for medical diagnosis, such as in Concept Processing technology.

Other tasks in medicine that can potentially be performed by artificial intelligence include:

- ✗ Computer-aided interpretation of medical images. Such systems help scan digital images, *e.g.* from computed tomography, for typical appearances and to highlight conspicuous sections, such as possible diseases. A typical application is the detection of a tumor.
- ✗ Heart sound analysis
- ✗ Companion robots for the care of the elderly

Music

The evolution of music has always been affected by technology. With AI, scientists are trying to make the computer emulate the activities of the skillful musician. Composition, performance, music theory, sound processing are some of the major areas on which research in Music and Artificial Intelligence are focusing.

Aviation

The Air Operations Division (AOD) uses AI for the rule based expert systems. The AOD has use for artificial intelligence for replacement operators for fighting and training simulators, mission management aids, support systems for tactical decision making, and post processing of the simulator data into symbolic summaries.

Ethical issues in artificial intelligence

❑ Unemployment.

What happens after the end of jobs?

❑ Humanity.

How do machines affect our behaviour and interaction?

❑ Racist robots.

How do we eliminate AI bias?

❑ Security.

How do we keep AI safe from adversaries?

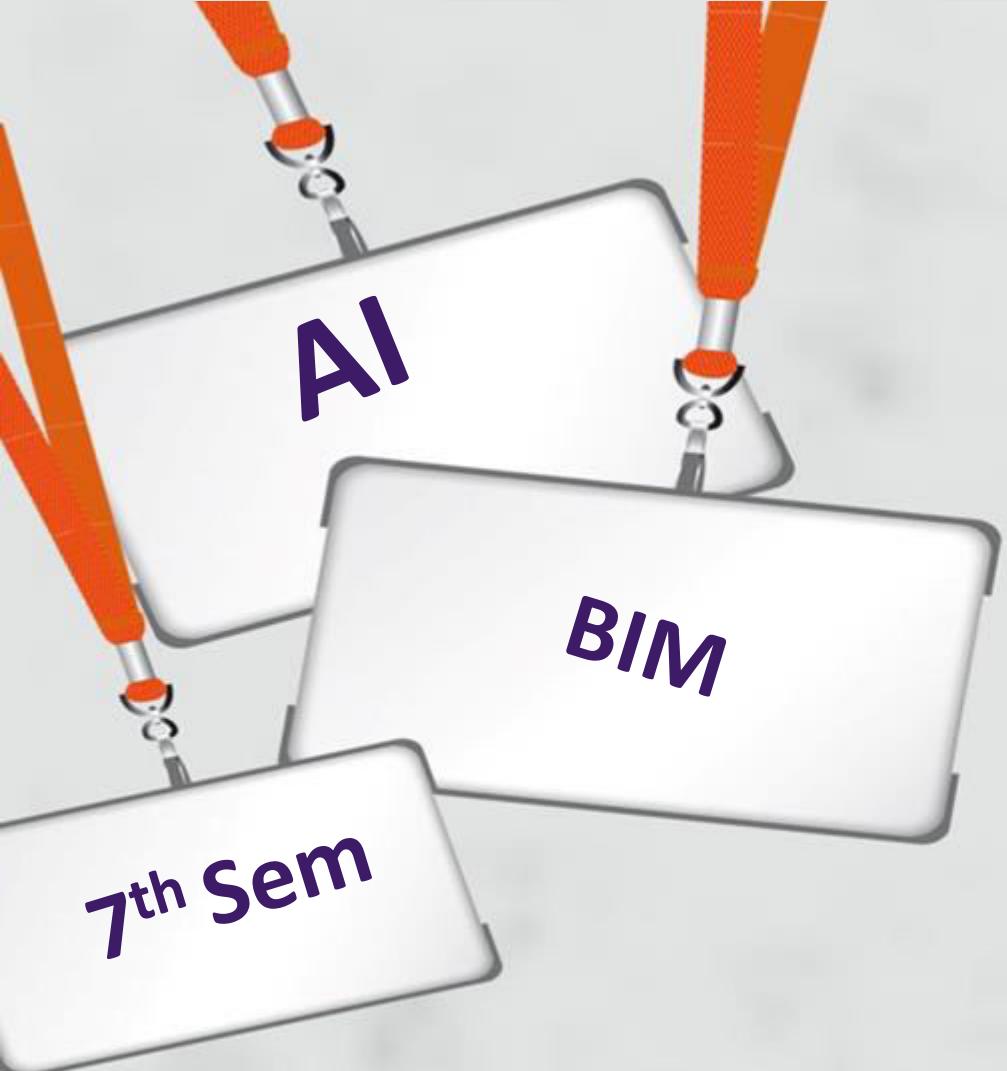
❑ Singularity.

How do we stay in control of a complex intelligent system?

Omniscience is the property of having complete or maximal knowledge. Along with supremacy and perfect goodness, it is usually taken to be one of the central attributes of someone.

Since omniscience is maximal or complete knowledge, it is typically defined in terms of knowledge of all true propositions, namely, as

S is omniscient = For every proposition p, if p is true then S knows p.

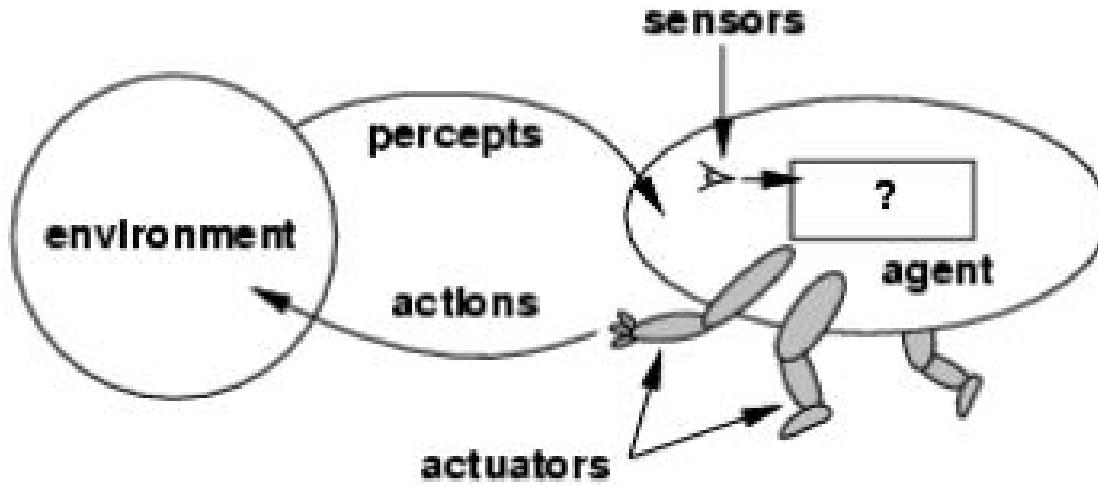


Chapter 2

Agent and Environment

Agent

- An AI system is composed of an agent and its environment.
- The agents act in their environment.
- The environment may contain other agents
- An agent is anything that can be viewed as:
 - perceiving its environment through sensors and
 - acting upon that environment through actuators



Architecture of Agent

Agent Terminologies

- **Performance Measure:**

- determines how successful the agent is.

- **Behavior:**

- activities the agent performs to achieve the goal

- **Percepts:**

- formation of concepts based on the inputs.

- **Percept Sequence:**

- set of all the perceptions till date

- **Agent Function:**

- mapping of perception

Agent's structure

Agent = Architecture + Agent Program

- ✓Architecture = the machinery that an agent executes on.
- ✓Agent Program = an implementation of an agent function

Agent Function

The agent function maps from percept sequences to actions:

$$[f: \mathcal{P}^* \rightarrow \mathcal{A}]$$

The agent program runs on the physical architecture to produce f

Example of Agents

□ Human agent(Powered by muscles)

- Eyes, ears, skin, taste buds, etc. for sensors
- Hands, fingers, legs, mouth, etc. for actuators

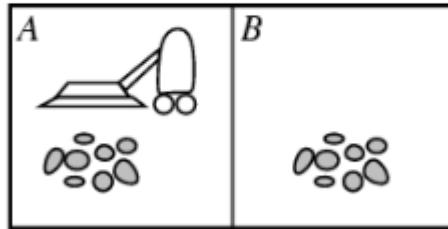
□ Robot (Powered by motors)

- Camera, infrared, bumper, etc. for sensors
- Grippers, wheels, lights, speakers, etc. for actuators

□ Software agent (Powered By Information and Results)

- Functions as sensors
- Functions as actuators

Vacuum-cleaner world



› **Percepts:**

location and state of the environment, e.g., [A,Dirty], [B,Clean]

› **Actions:**

Left, Right, Suck, No Op

Properties of Agent

- › Autonomous
- › Interacts with other agents plus the environment
- › Reactive to the environment
- › Pro-active (goal- directed)

Environment types

➤ Fully observable :

An agent's sensors give it access to the complete state of the environment at each point in time

➤ Deterministic :

The next state of the environment is completely determined by the current state and the action executed by the agent.

➤ Episodic :

The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action)

➤ Static :

The environment is unchanged while an agent is deliberating.

Intelligent agents

An Intelligent Agent is an **autonomous entity** which observes through sensors and acts upon an environment using actuators and directs its activities towards achieving the goal.

Hence, an agent gets percepts one at a time, and maps this percept sequence to actions.

Types of Agent

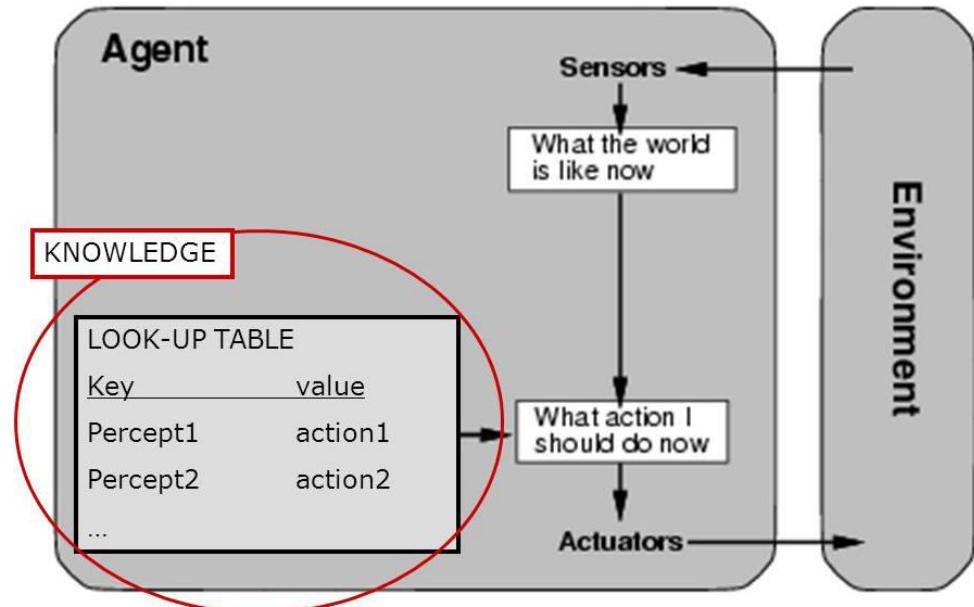
There are several kinds of agent programs which try to meet these conditions:

- Table Lookup Agent
- Simple Reflex Agent
- Model-based Reflex Agent
- Goal-based Agent
- Utility-based Agent

Table lookup of percept-action pairs mapping from every possible perceived state to optimal action for it

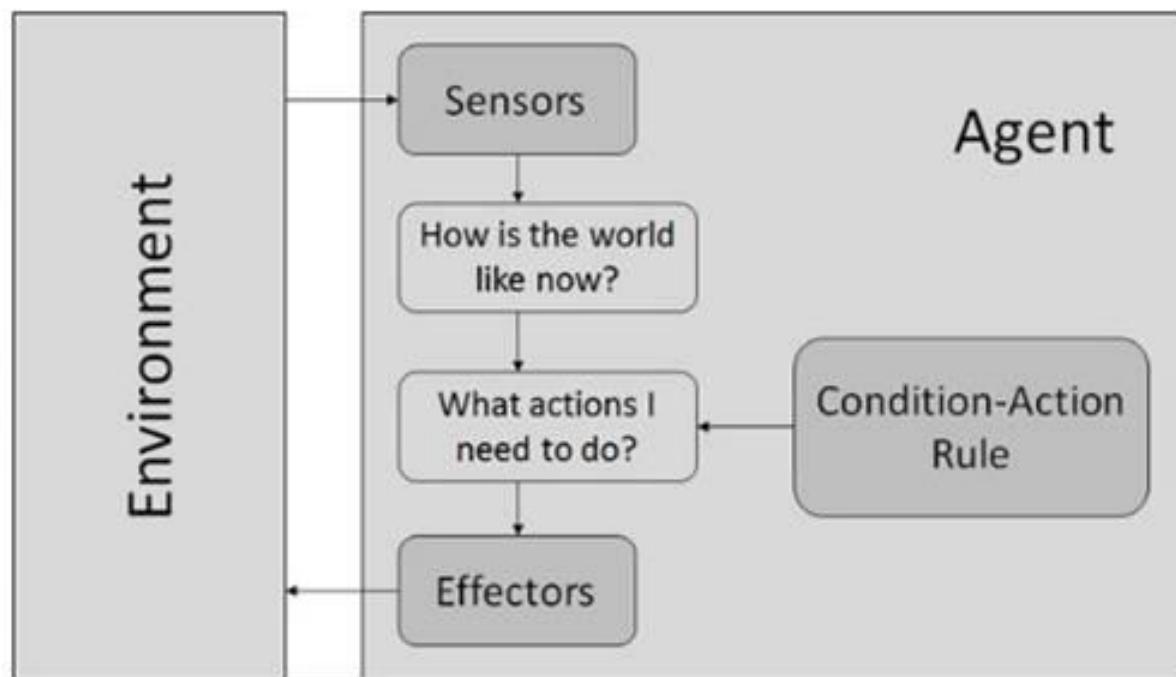
Problems:

- ✓ Too big to generate and to store (Chess has about states, for example)
- ✓ No knowledge of non-perceptual parts of the current state
- ✓ Not adaptive to changes in the environment; entire table must be updated if changes occur
- ✓ Looping: Can't make actions conditional on previous actions/states



- Simple Reflex Agent

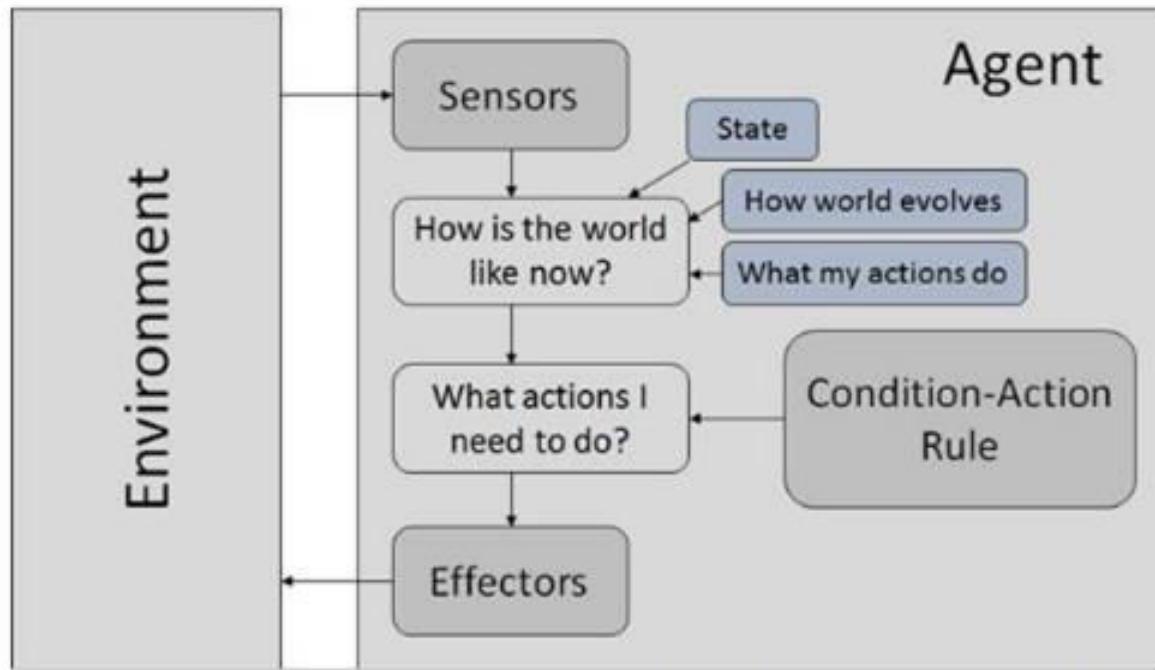
- They choose actions only based on the current percept or situation.
- This is useful when a quick automated system is required. Human have very similar reaction to fire.



(Condition- Action Rule/IF-THEN Rule IF hands on fire, THEN pull away)

- **Model Based Agents (Reflex Agent with Internal State)**

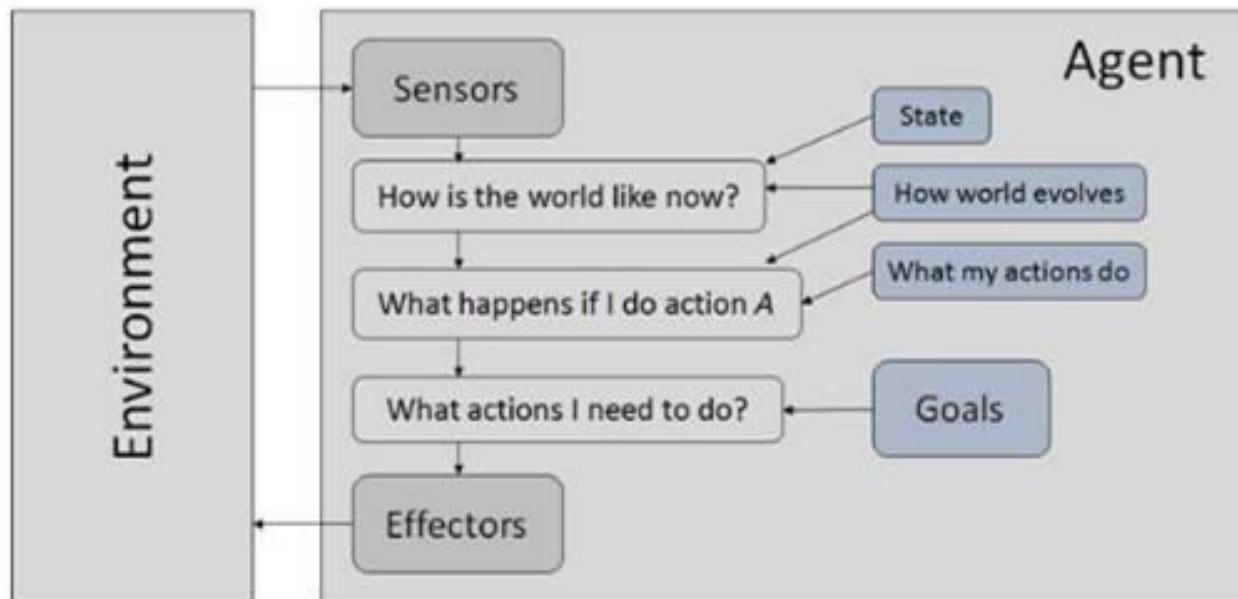
- They choose action only based on their model and maintain their internal state where model means the information about perception function and Internal States are



the representation of unobserved aspects of current state depending on percept history.

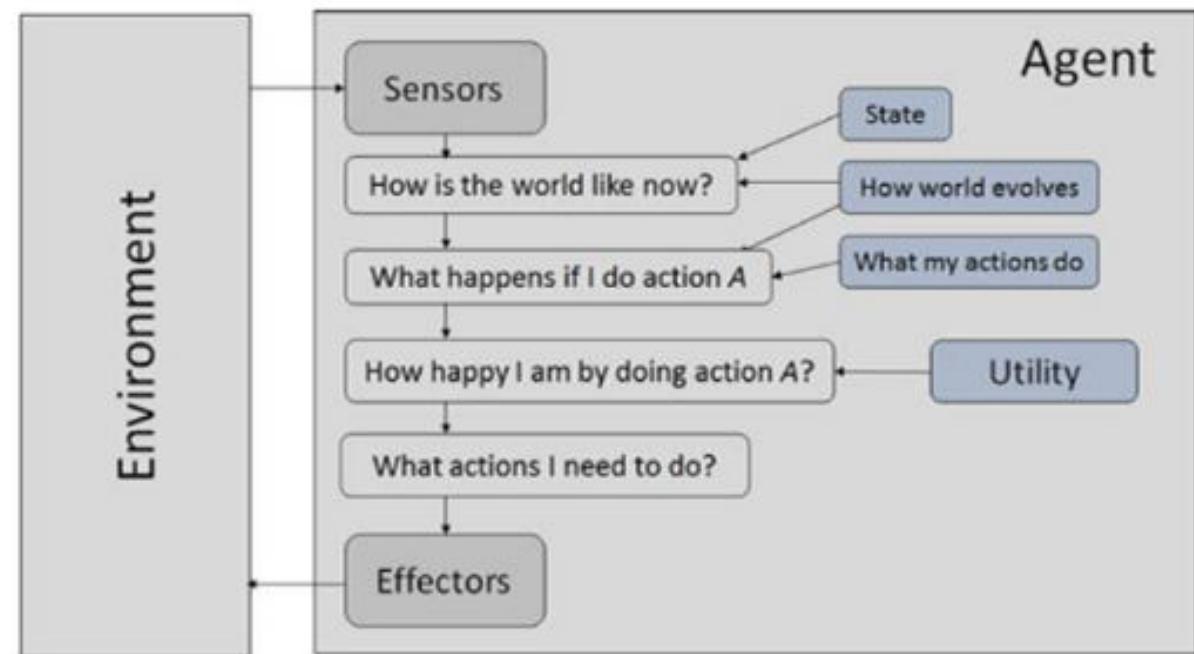
- Goal Based Agents

- They choose their actions in order to achieve goals. Goal-based approach is more flexible than reflex agent since the knowledge/information is modeled in such a way that they are easy for modifications.



- Utility Based Agents

- They choose actions based on a utility for each state.
- Utility function maps each state after each action to achieve the goal. This is useful when we either have



many actions for same goal or we have many goals for same actions.

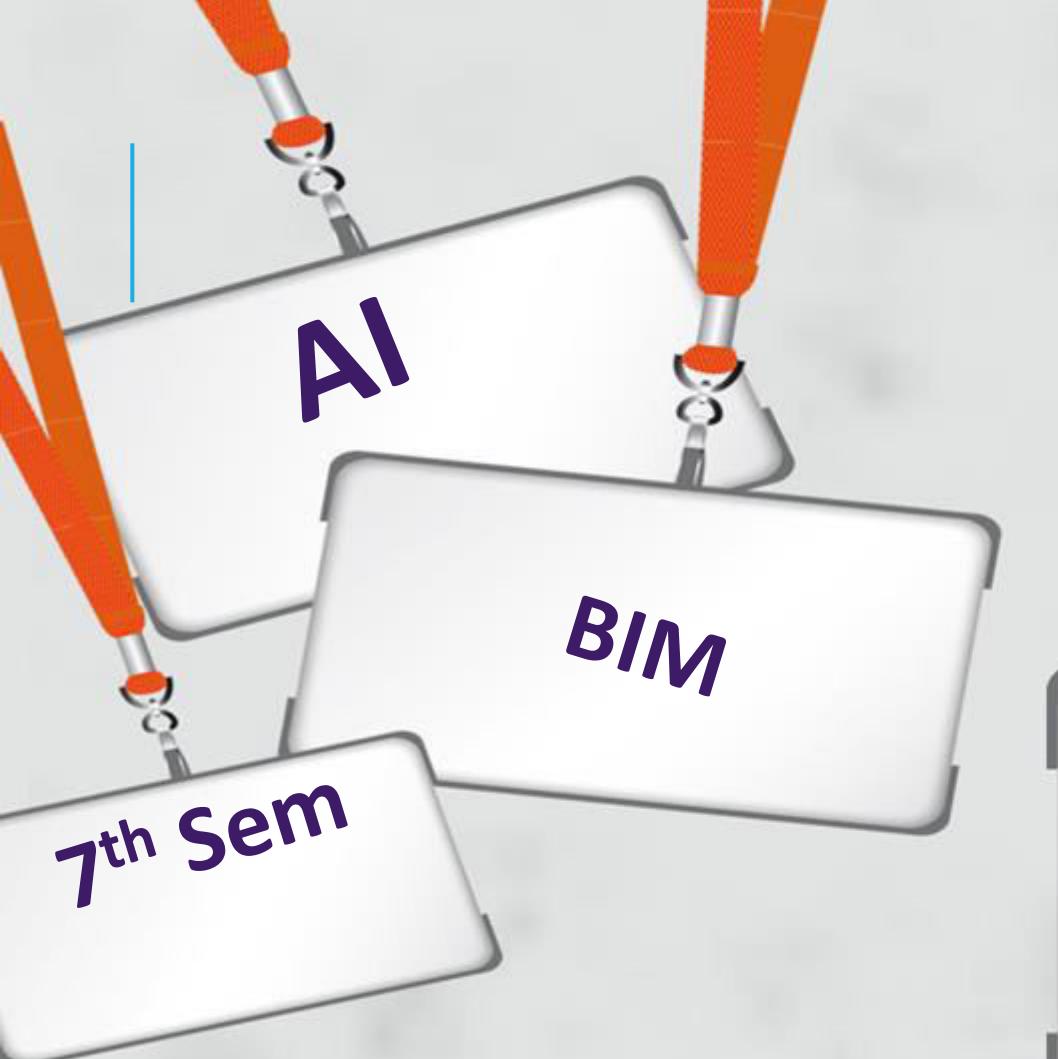
PEAS

- To design a rational agent we must specify its task environment.
- Standing for performance, environment, actuators and sensors, PEAS define task environments about formulating the performance of intelligent agents.

Q. Point out the task of designing an automated taxi driver according to PEAS description.

Q. Point out the task of designing a Medical diagnosis system according to PEAS description.

Agent type	P	E	A	S
Medical Diagnosis System	Healthy patient, minimize costs, lawsuits	Patient, Hospital Staff	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's Answers.
Interactive English Tutor	Maximize student's score on test.	Set of students, testing agency.	Display exercises, suggestions, corrections	Typed words.
Satellite Image Analysis System	Correct categorization	Downlink from orbiting satellite	Display categorization of scene.	Colour pixel arrays.
Refinery Controller	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Taxi Driver	Safe: fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display.	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard.



Chapter 3

Searching

SEARCHING

- The general concept of ‘searching’ is about **looking for something**.
- In computer science, searching techniques are strategies that look for solutions to a problem in a search space.
- The solutions or ‘goal states’ could sometimes be an object, a goal, a sub-goal or a path to the searched item.
- Search is a universal problem-solving technique.

SEARCHING

- In general, a computer search problem has the following characteristics:
 - **A goal state** – the definition of a need
 - **An initial state** – a current position or a set of conditions
 - **A set of actions** - strategies and actions to select and evaluate an option in the search space
 - **Goal test criteria** – criteria used to test if an option is a solution
 - **A path cost** - the cost of the actions in a search path.

SEARCH SPACE

- The search space **can be represented by a ‘graph’ which is based on graph theory.**
- A **computer ‘tree’ data structure** is a special kind of graph. A tree has a **root node** on the top of the structure and it has **at most one path to each node**. Each node may be connected to a lower level of neighbors which are called **child nodes** (successors). Nodes that have no children are called **leaf nodes**.



Why is search important in Artificial Intelligence?

Modeling and solving problems on a computer in this manner dates back to Alan Turing and many AI problems can be easily modeled as state spaces.

Solving these problems can "simply" be reduced to exploring the state space, and identifying the correct answer.

Every problem can be reduced to search. Every problem has an input within some range (the domain) and an output in some other range (codomain). That is, every problem can be formulated as a kind of map from one space to another, where the source is the givens of the problem, and the destination is the solution to the problem.

SEARCH TERMINOLOGY

- **Problem Space:** Environment in which the search takes place.
- **Problem Instance:** It is Initial state + Goal state
- **Problem Space Graph:** It represents problem state. States are shown by nodes and operators are shown by edges.
- **Depth of a problem:** Length of a shortest path or shortest sequence of operators from Initial State to goal state.

- **Space Complexity:** The maximum number of nodes that are stored in memory.
- **Time Complexity:** The maximum number of nodes that are created.
- **Admissibility:** A property of an algorithm to always find an optimal solution.
- **Branching Factor:** The average number of child nodes in the problem space graph.

SEARCH STRATEGIES

- A search strategy is defined by picking the order of node expansion
- Strategies are evaluated along the following dimensions:

Completeness:

does it generate to find a solution if there is any?

Optimality:

does it always find the highest quality (least-cost) solution?

Time complexity:

How long does it take to find a solution?

Space complexity:

How much memory does it need to perform the search?

Time and space complexity are measured in terms of

- ❖ b : maximum branching factor of the search tree
- ❖ d : depth of the least-cost solution
- ❖ m : maximum depth of the state space (may be ∞)

CLASSIFICATION

Uninformed Search (Blind Search/Brute force search)

The search algorithms that do not use any extra information regarding the problem

- ✓ Depth First Search
- ✓ Breath First Search
- ✓ Depth Limit Search
- ✓ Iterative deepening
- ✓ Uniform Cost
- ✓ Bidirectional Search

Informed search or Heuristic search

Informed search have problem specific knowledge apart from problem definition

- ✓ Hill climbing Search
- ✓ Best first Search
 - Greedy Best First Search
 - A* Search
- ✓ Simulated Annealing

DEPTH FIRST SEARCH (DFS)

- Proceeds down a single branch of the tree at a time
 - Expands the root node, then the leftmost child of the root node
 - Always expands a node at the deepest level of the tree
 - Only when the search hits a dead end (a partial solution which can't be extended), the search backtrack and expand nodes at higher levels.

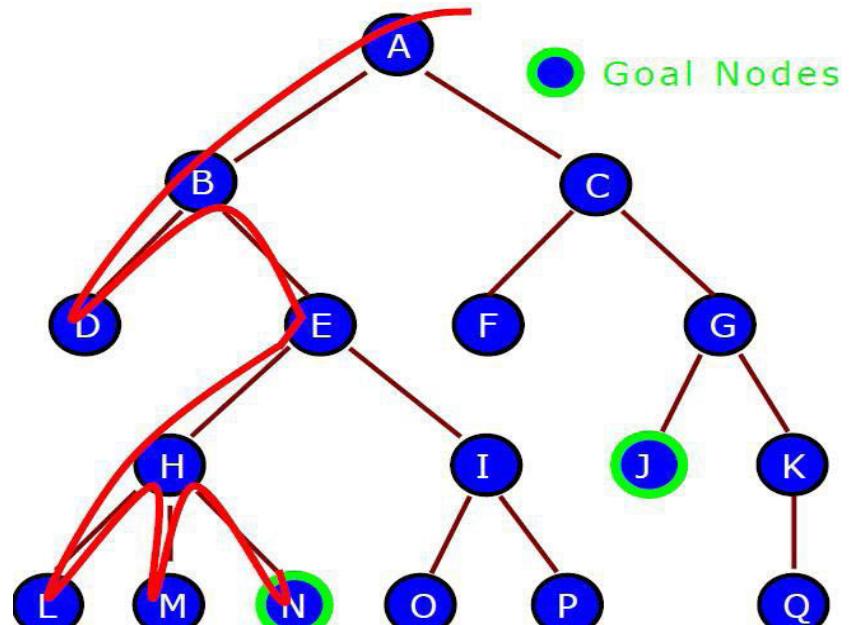


Fig. Depth-first search (DFS)

DEPTH FIRST SEARCH (DFS)

❑ *Completeness:*

- ✓ Can get stuck going down the wrong path when a different choice would lead to a solution near the root of the search tree
- ✓ Not complete

❑ *Optimality:*

- ✓ The strategy might return a solution path that is longer than the optimal solution, if it starts with an unlucky path
- ✓ Not optimal

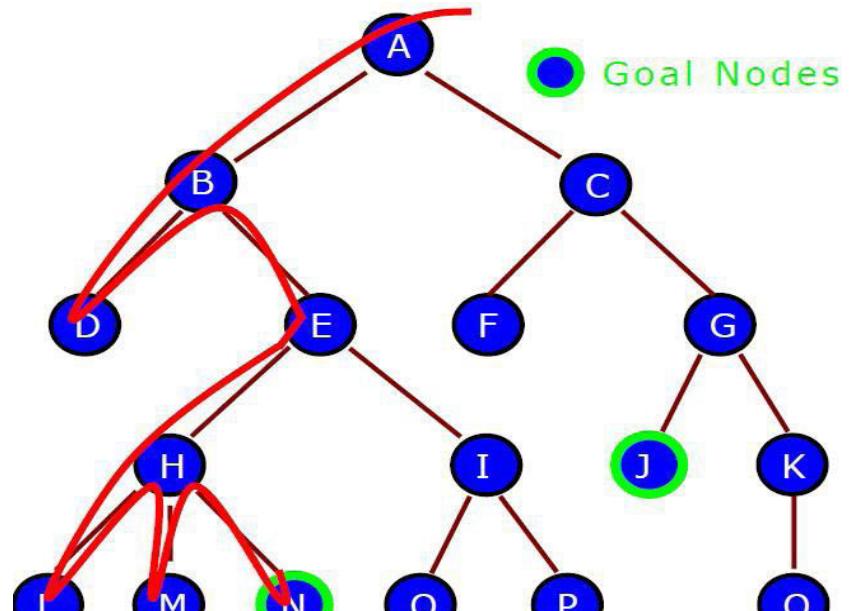


Fig. Depth-first search (DFS)

DEPTH FIRST SEARCH (DFS)

□ *Space complexity:*

- It needs to store a single path from root to a leaf node and the remaining unexpanded sibling nodes for each node in the path
- Once a node has been expanded, it can be removed from memory as soon as all its decedents have been fully explored
- For a search tree of branching factor ‘b’ and maximum tree depth ‘m’, only the storage of $bm+1$ node is required
- Hence,

$$\begin{aligned}\text{SpaceComplexity} &= O(bm+1) \\ &= O(bm)\end{aligned}$$

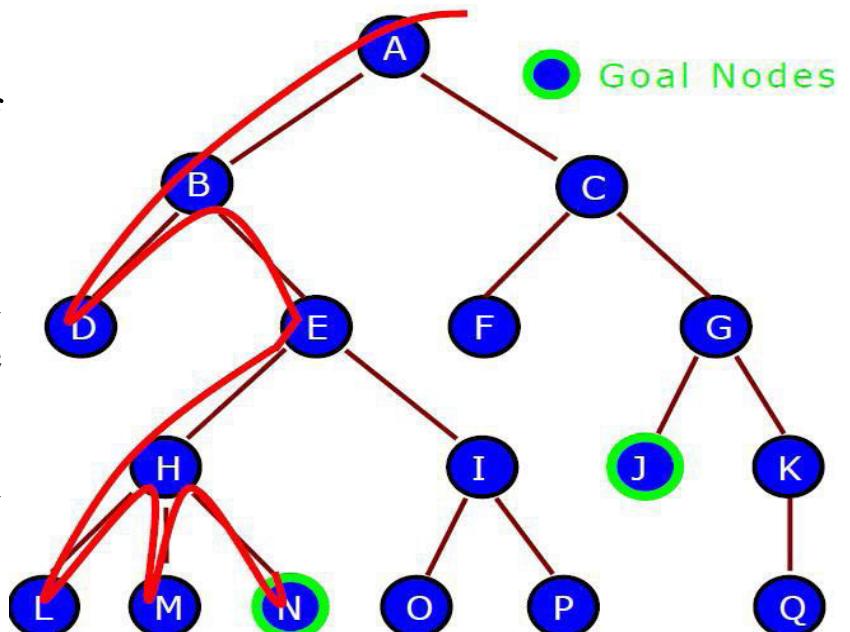


Fig. Depth-first search (DFS)

DEPTH FIRST SEARCH (DFS)

□ *Time Complexity:*

If you can access each node in $O(1)$ time, then with branching factor of b and max depth of m , the total number of nodes in this tree would be $= b * b * b \dots m$ times $= b^m$, resulting in total time to visit each node proportional to b^m . Hence the complexity = $O(b^m)$

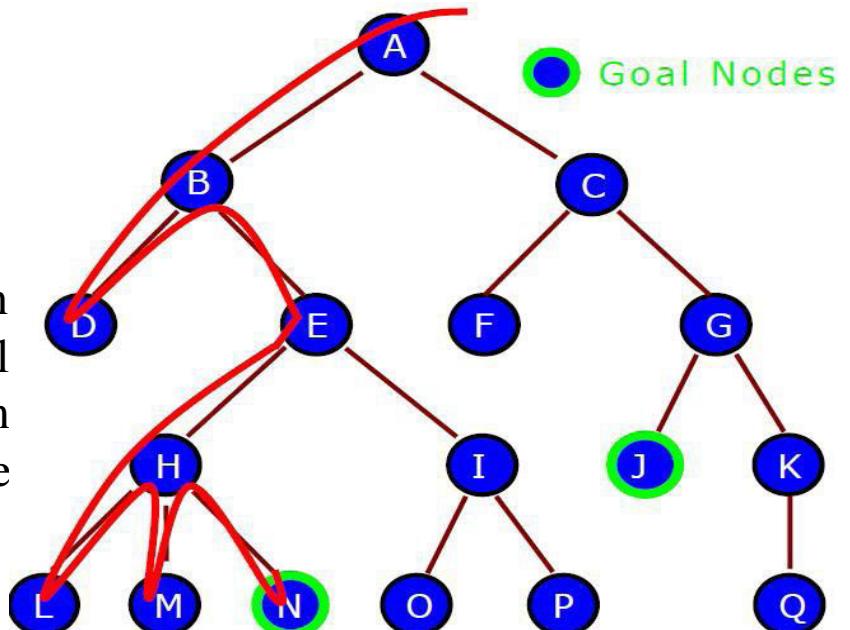
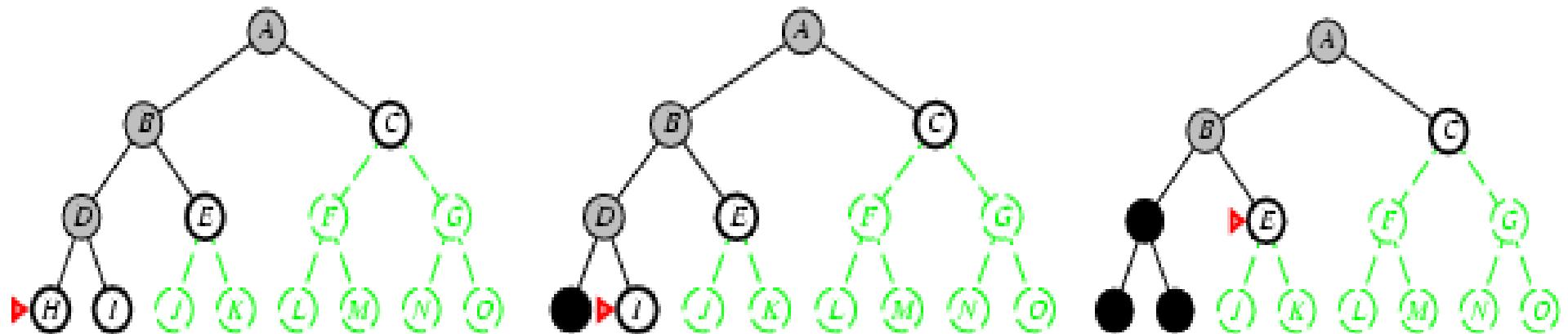
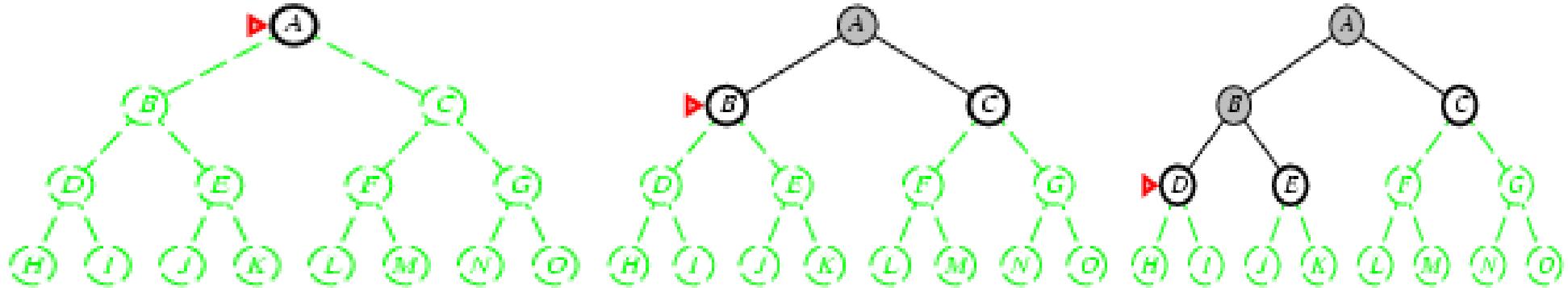
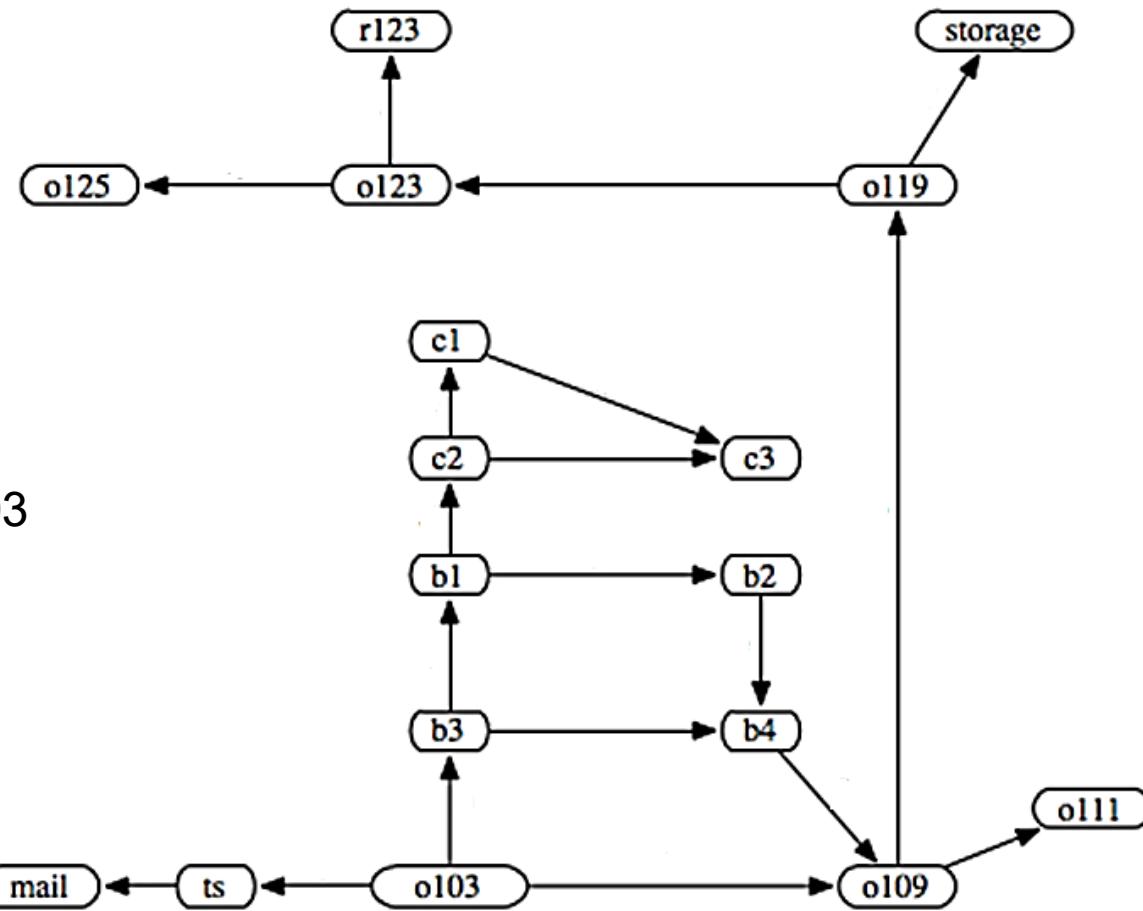


Fig. Depth-first search (DFS)

DFS example (find path from A to E)





Q. Find the Path from o103
to r123 using DFS

Breadth-First Search (BFS)

- Proceeds level by level down the search tree
- Starting from the root node (initial state) explores all children of the root node, left to right
- If no solution is found, expands the first (leftmost) child of the root node, then expands the second node and so on

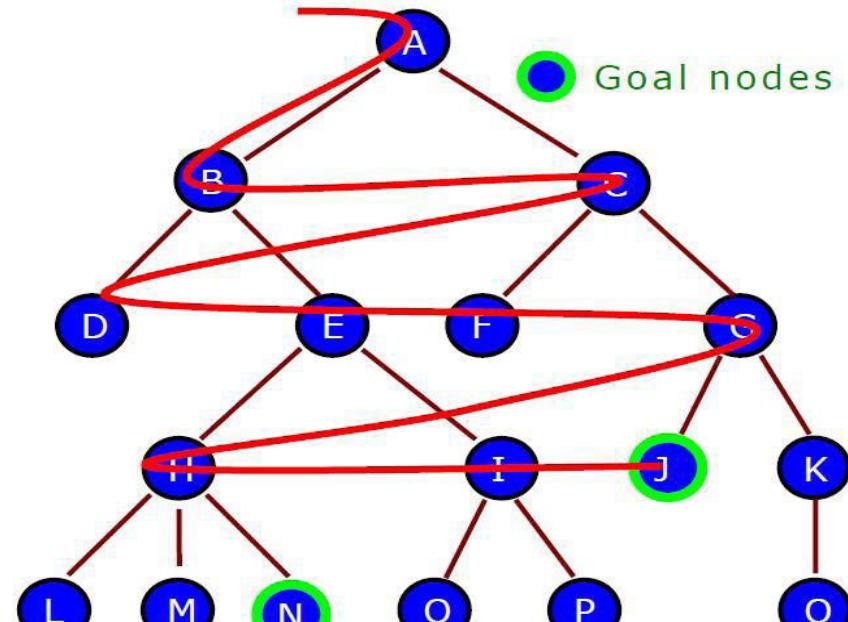


Fig. Breadth-first search (BFS)

Breadth-First Search (BFS)

□ *Completeness:*

- This search strategy finds the shallowest goal first
- Complete, if the shallowest goal is at some finite depth

□ *Optimality*

- The shallowest goal node is not necessarily the optimal one
- Optimal, if the path cost is a non-decreasing function of the path of the node (For example: when all the actions have the same cost)

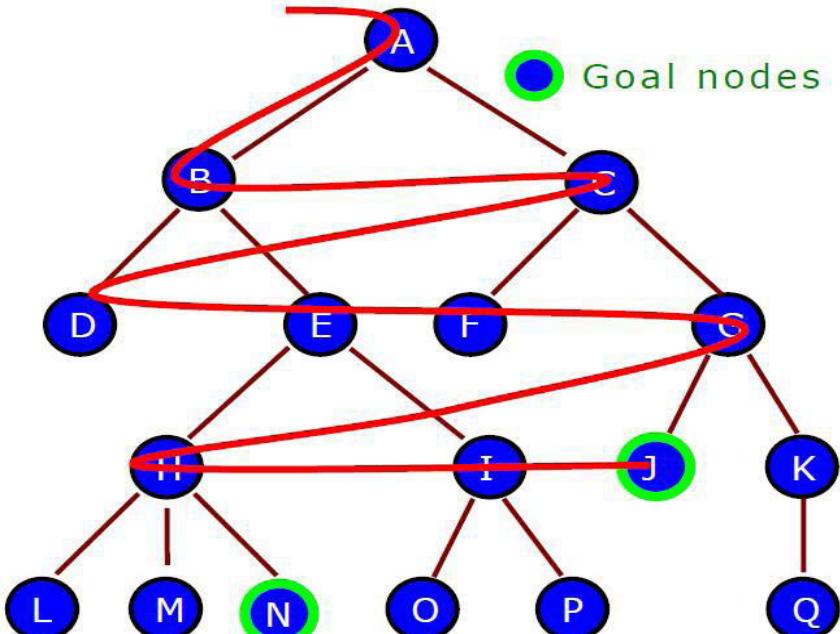


Fig. Breadth-first search (BFS)

Breadth-First Search (BFS)

□ *Time Complexity:*

- For a search tree a branching factor ‘ b ’ expanding the root yields ‘ b ’ nodes at the first level.
- Expanding ‘ b ’ nodes at first level yields b^2 nodes at the second level.
- Similarly, expanding the nodes at d^{th} level yields b^{d+1} node at $(d+1)^{th}$ level

Hence, time complexity is $O(b^{d+1})$
where,

b = branching factor and

d = level of goal node in the search table

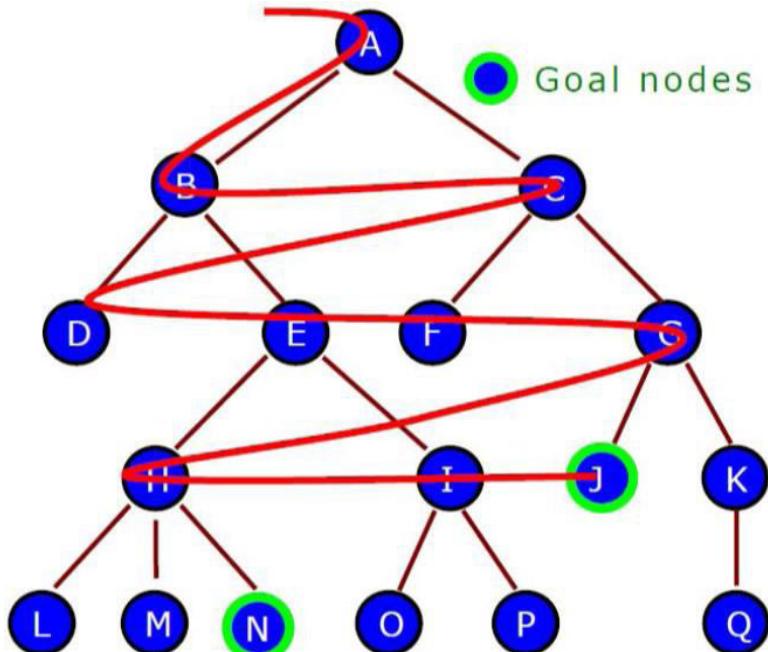


Fig. Breadth-first search (BFS)

Breadth-First Search (BFS)

□ *Space Complexity*

- Same as time complexity
- i.e. $O(b^{d+1})$

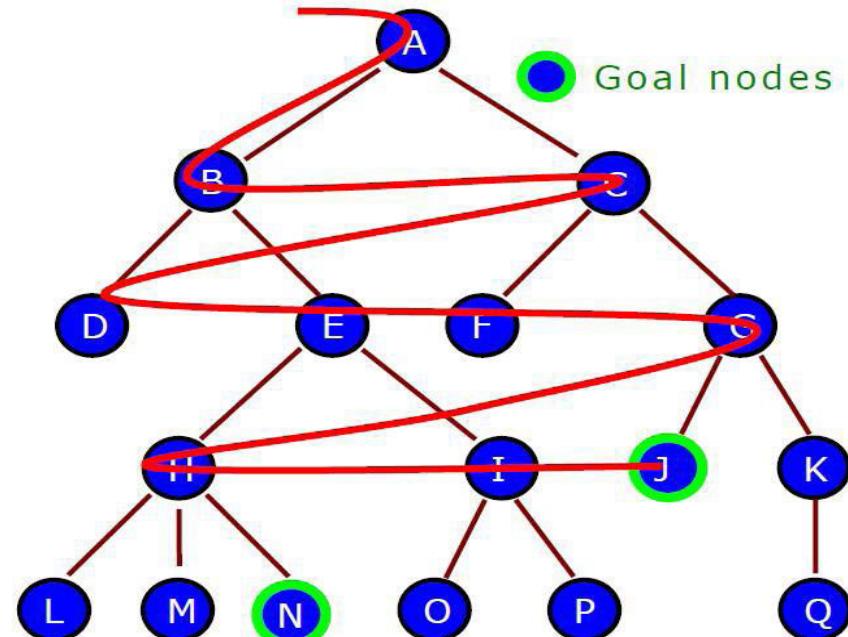
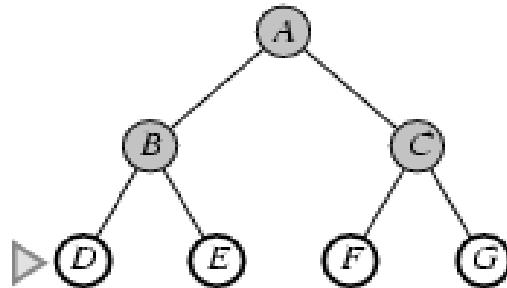
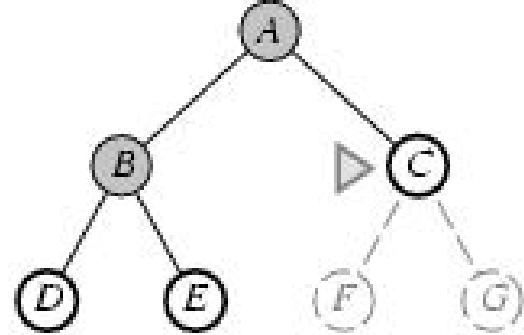
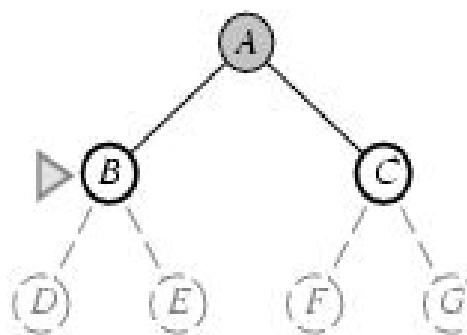
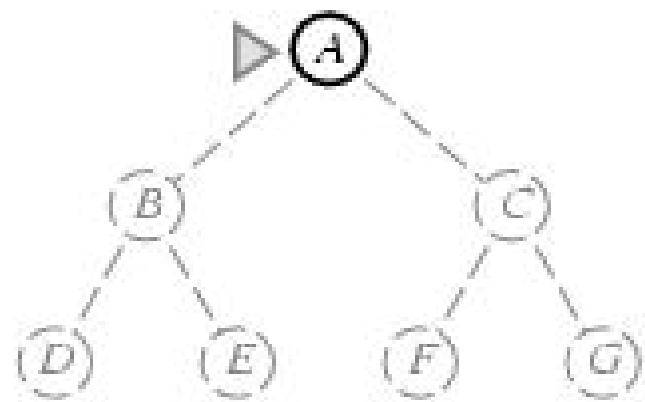
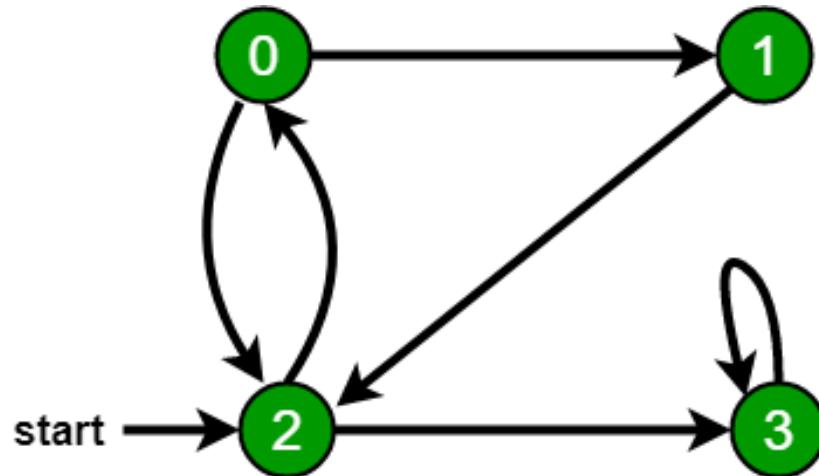


Fig. Breadth-first search (BFS)

BFS example (Find path from A to D)

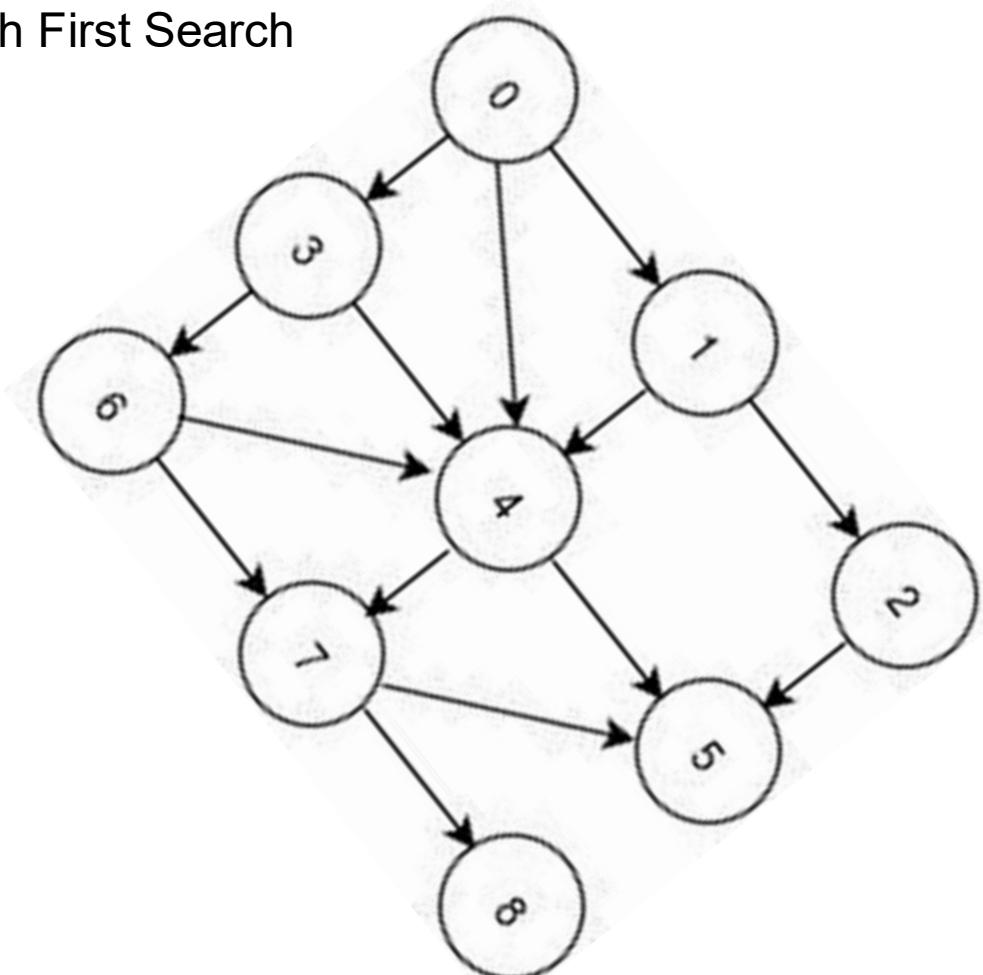
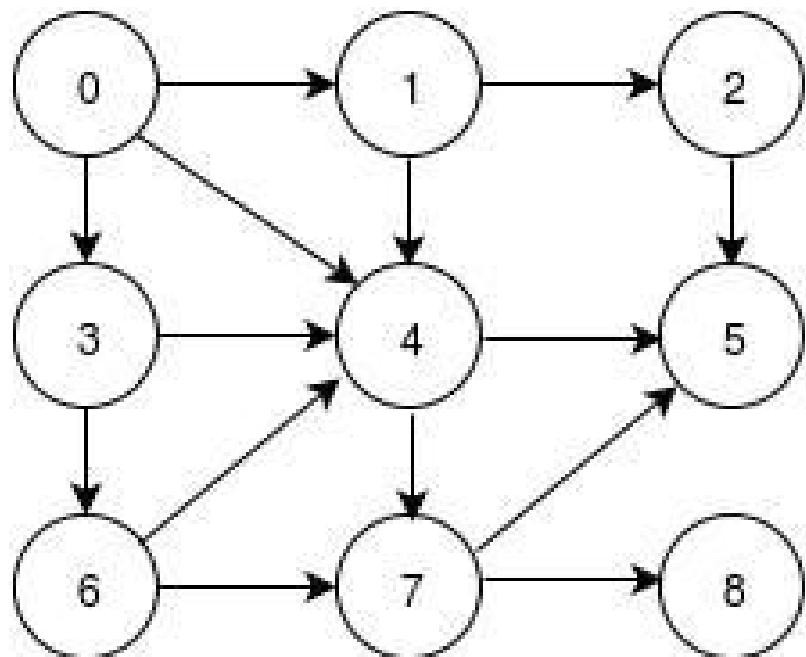


Perform the Breadth First Search



2, 0, 3, 1.

Q. Find the Path from 0 to 8 using Breadth First Search



0 1 4 3 2 6 5 7

Uniform Cost Search

Uniform-cost is guided by path cost rather than path length, the algorithms starts by expanding the root, then expanding the node with the lowest cost from the root, the search continues in this manner for all nodes.

- **Completeness:**

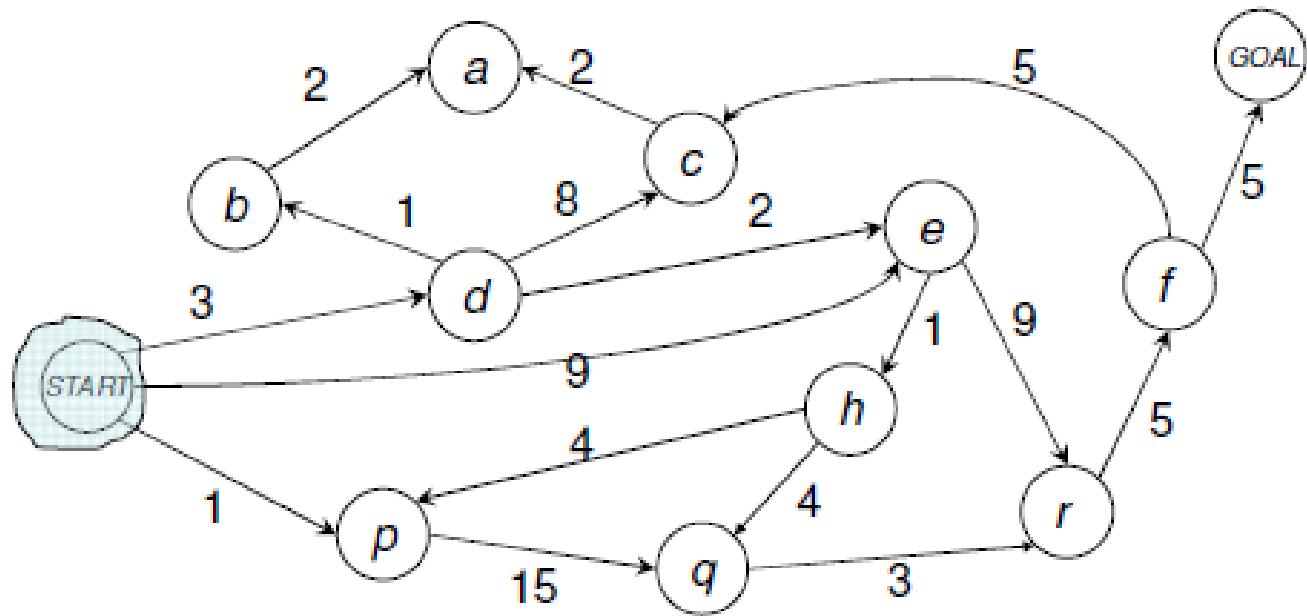
Complete if the cost of each step exceeds some small positive integer, this to prevent infinite loops.

- **Optimality:**

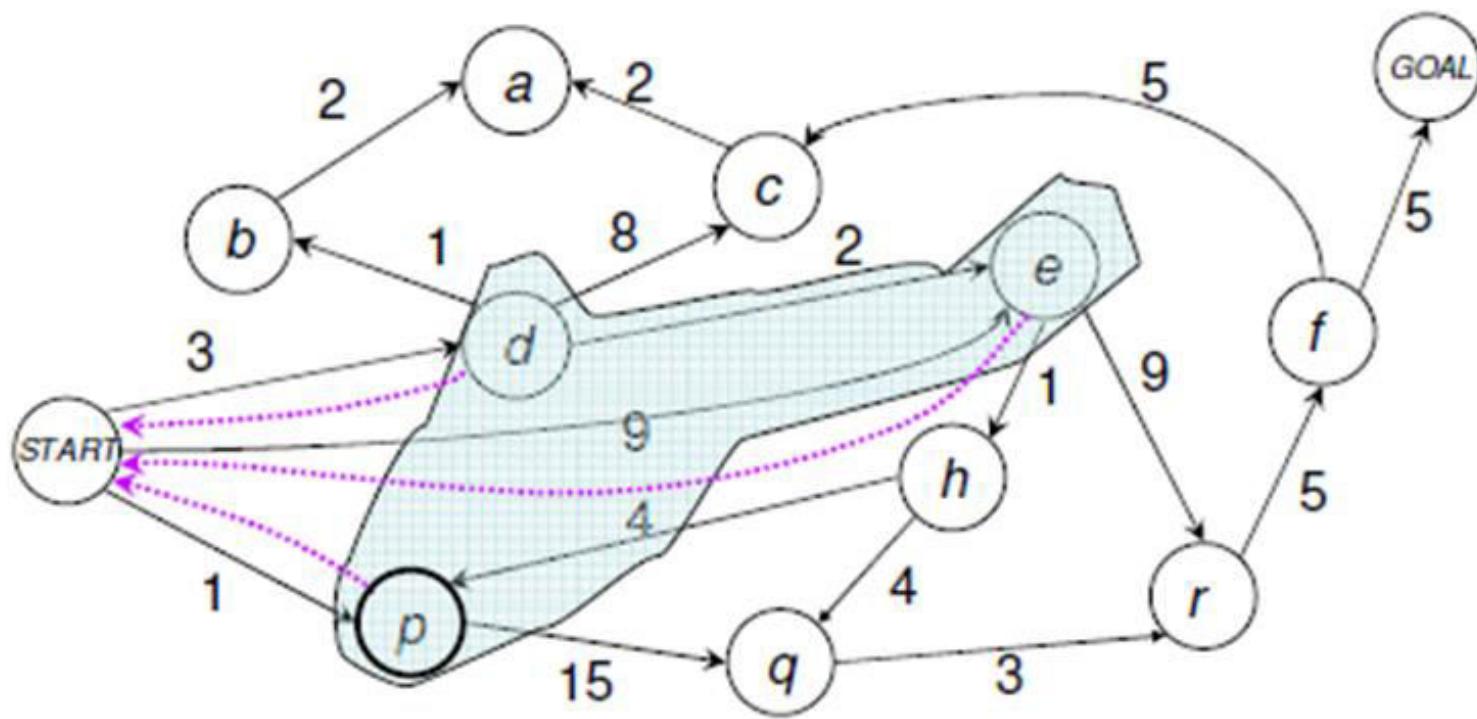
Optimal in the sense that the node that it always expands is the node with the least path cost.

- **Time Complexity:** $O(b^{C/e})$.

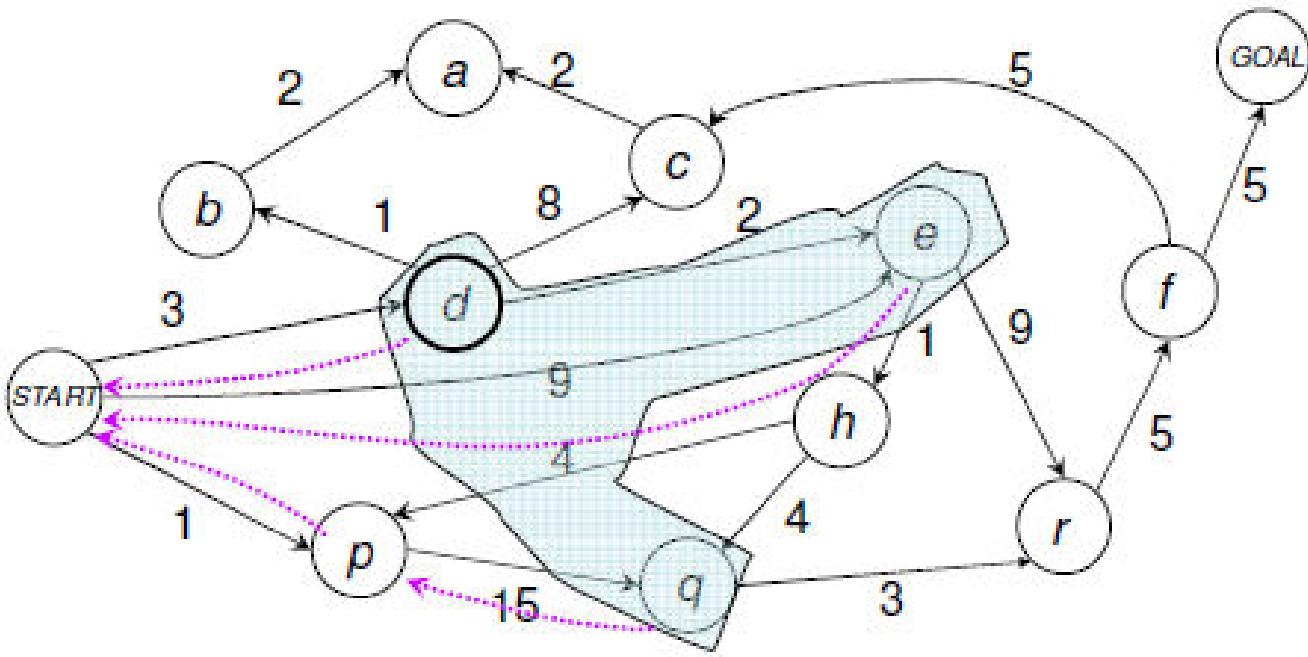
- **Space Complexity:** $O(b^{C/e})$



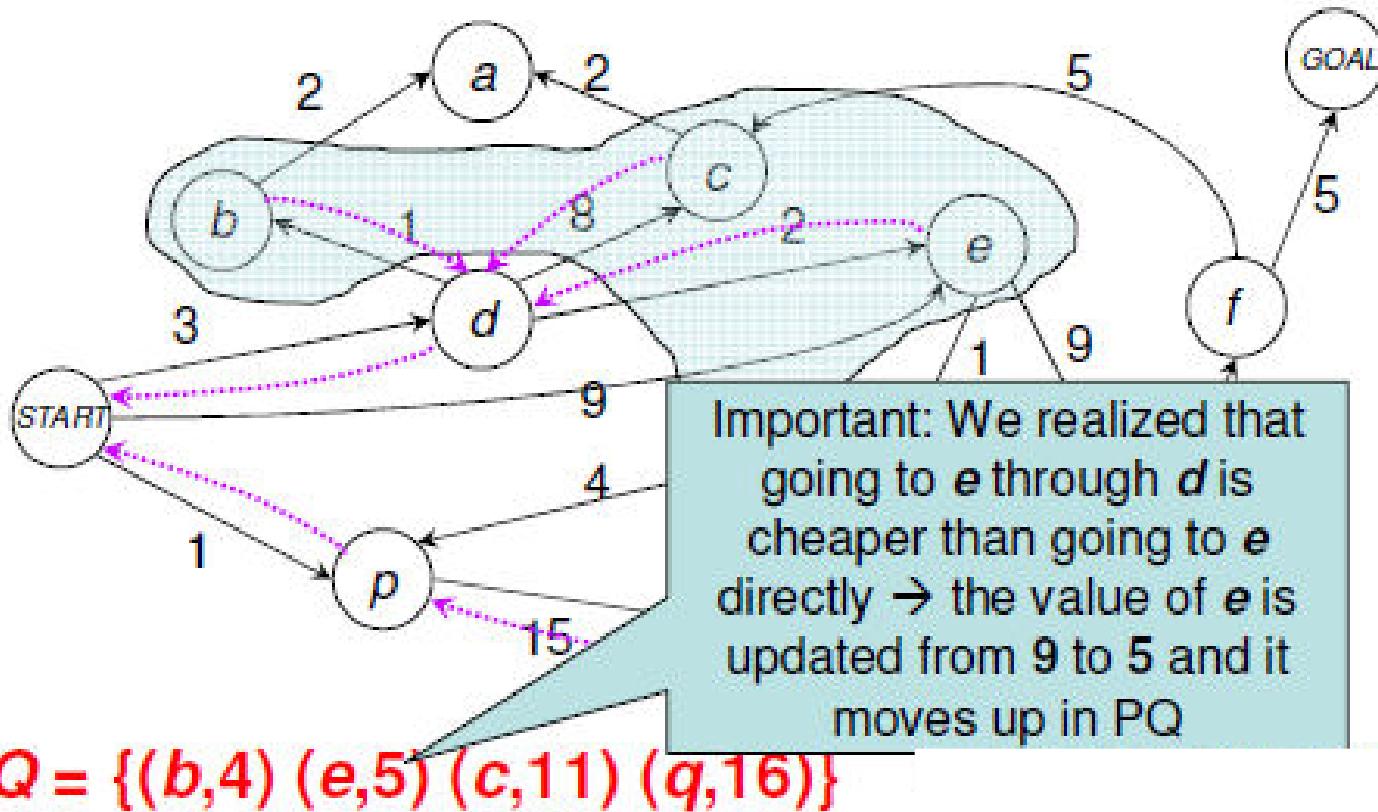
$$PQ = \{(START, 0)\}$$

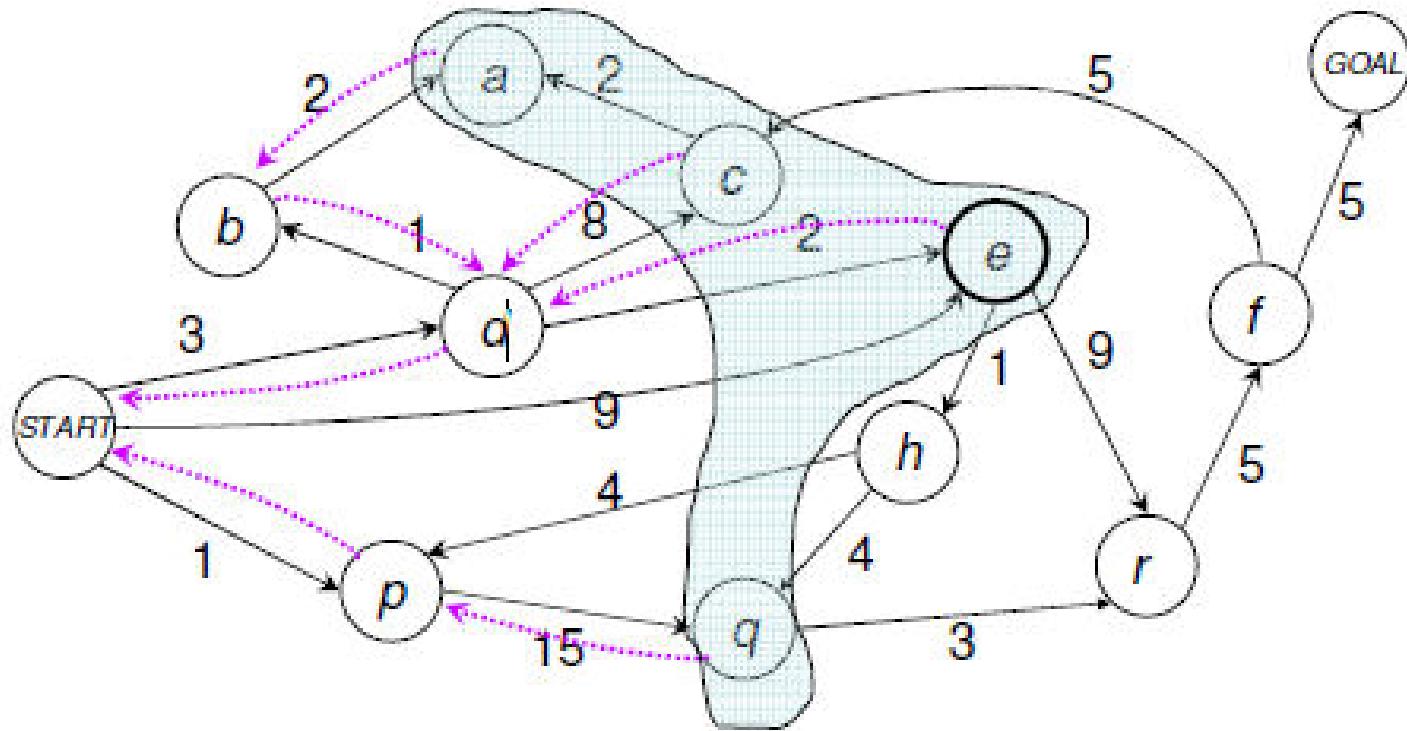


$$PQ = \{(p, 1) (d, 3) (e, 9)\}$$

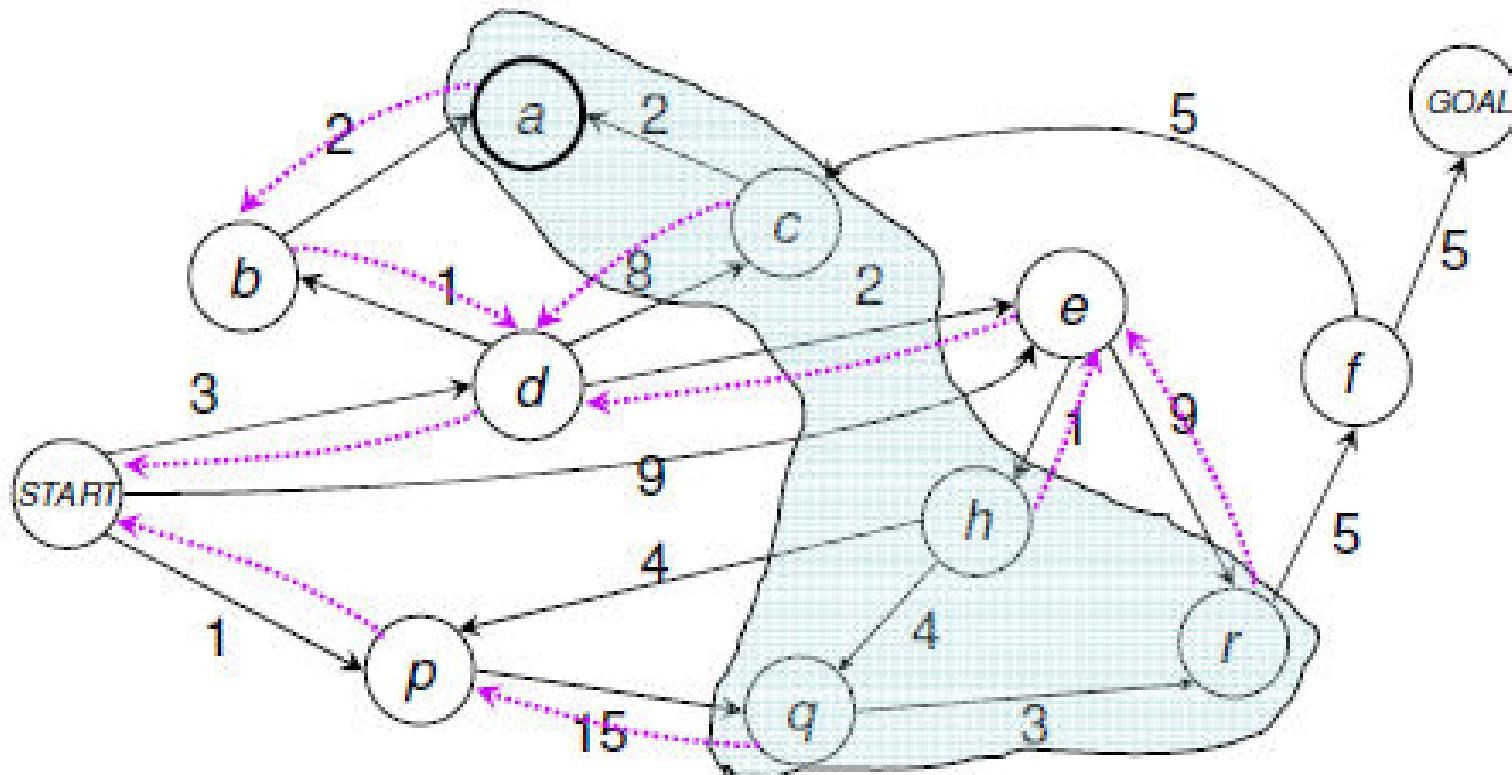


$$PQ = \{(d, 3) (e, 9) (q, 16)\}$$

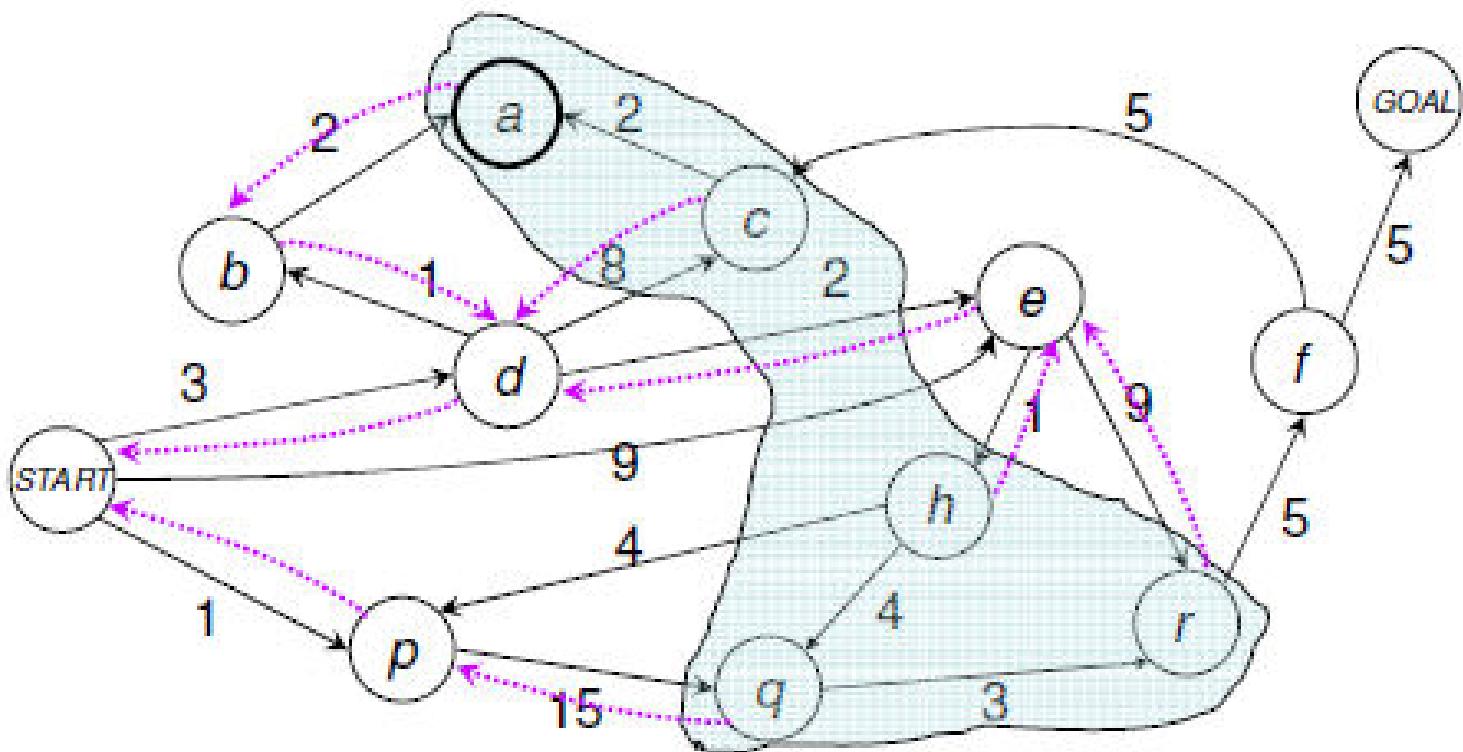




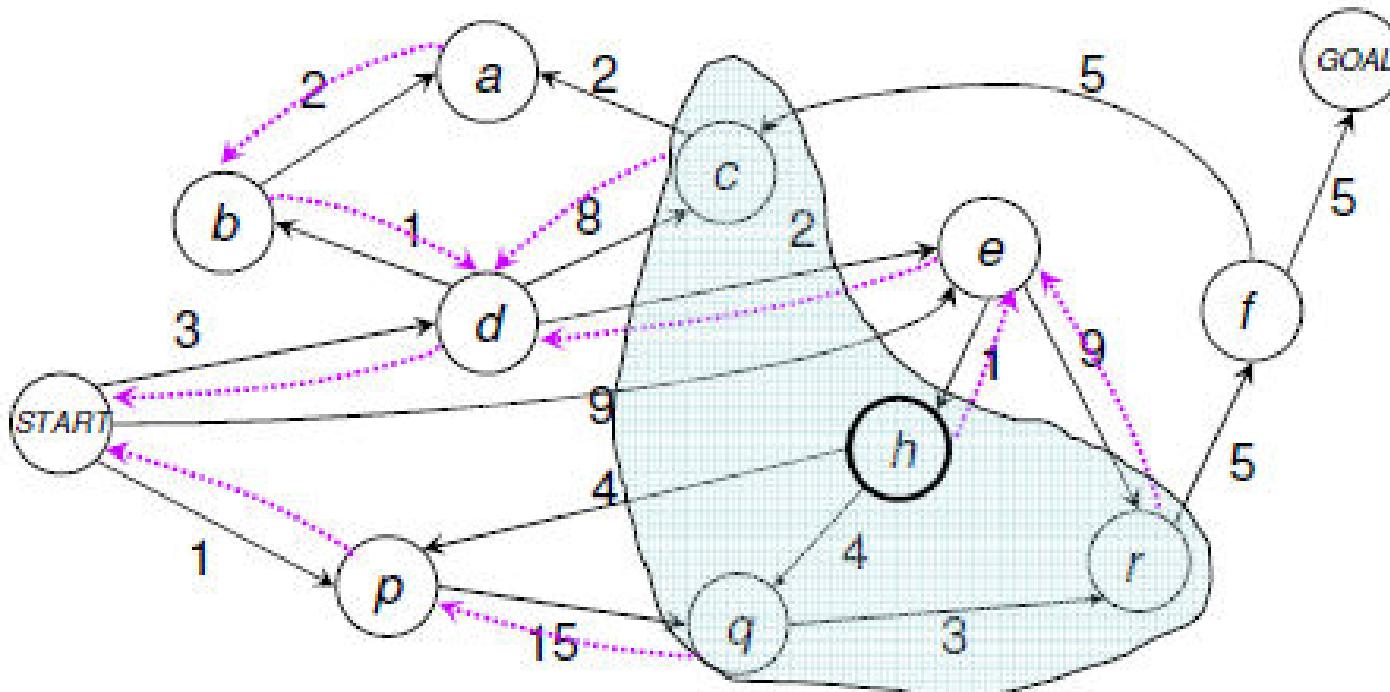
$$PQ = \{(e,5) (a,6) (c,11) (q,16)\}$$



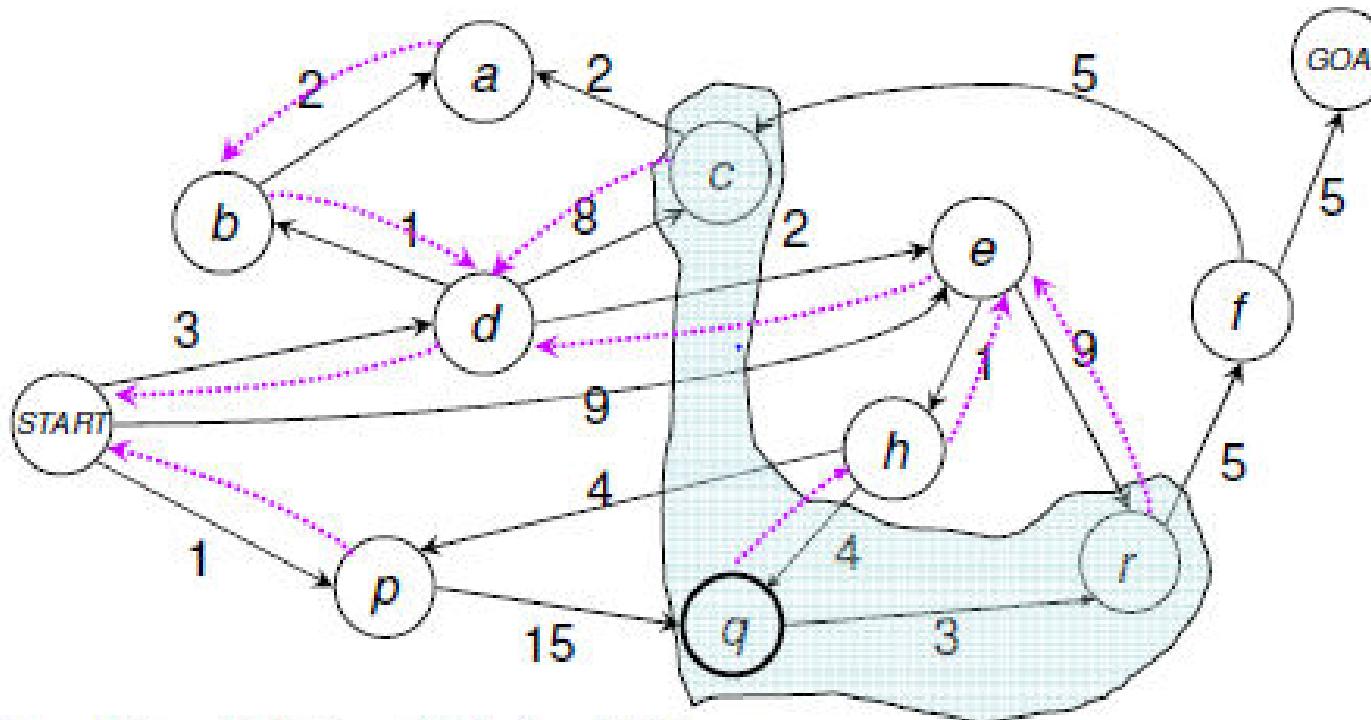
$$PQ = \{(a, 6) (h, 6) (c, 11) (r, 14) (q, 16)\}$$



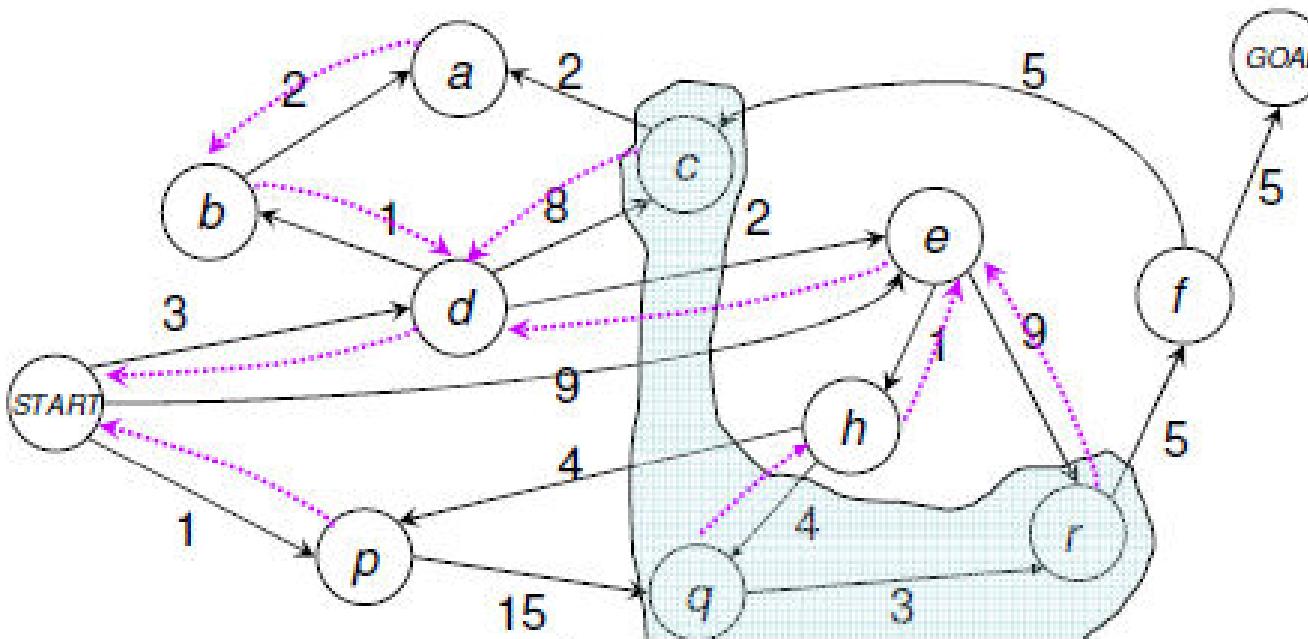
$$PQ = \{(a,6) (h,6) (c,11) (r,14) (q,16)\}$$



$$PQ = \{(h,6) (c,11) (r,14) (q,16)\}$$

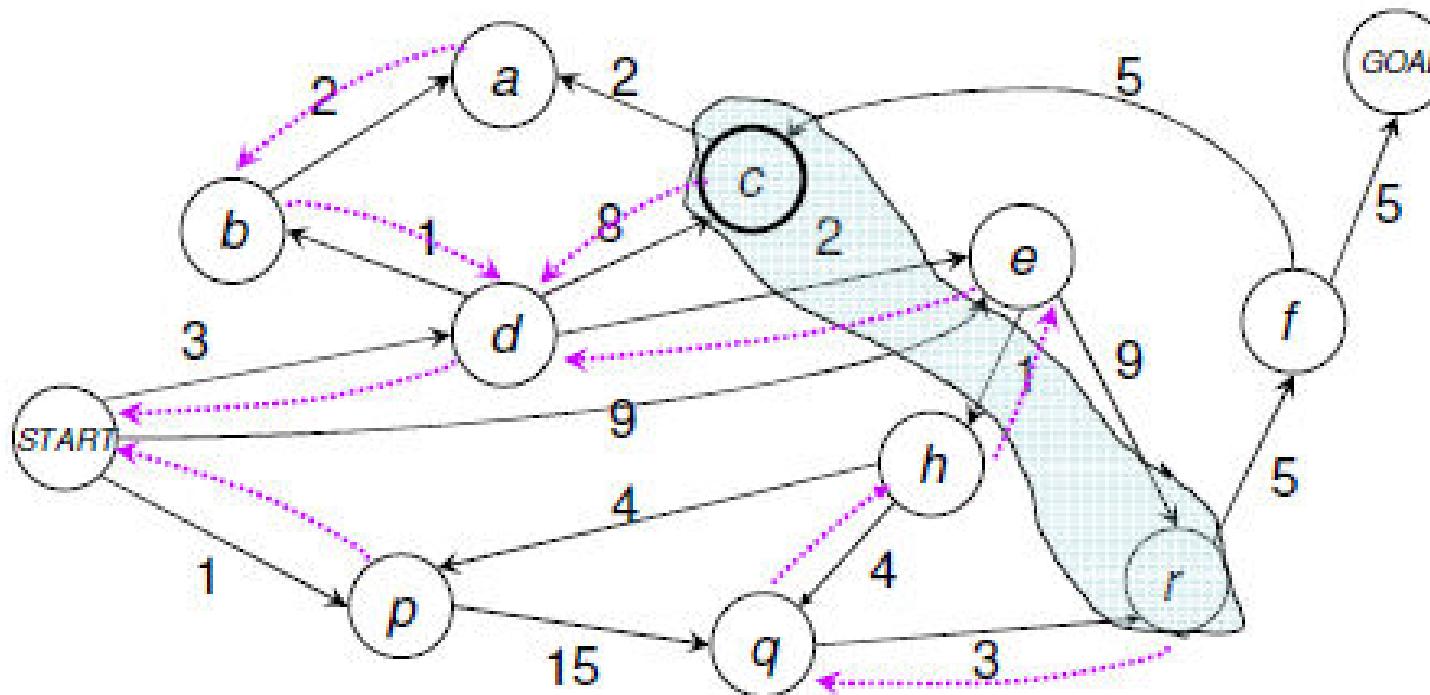


$$PQ = \{(q, 10) (c, 11) (r, 14)\}$$

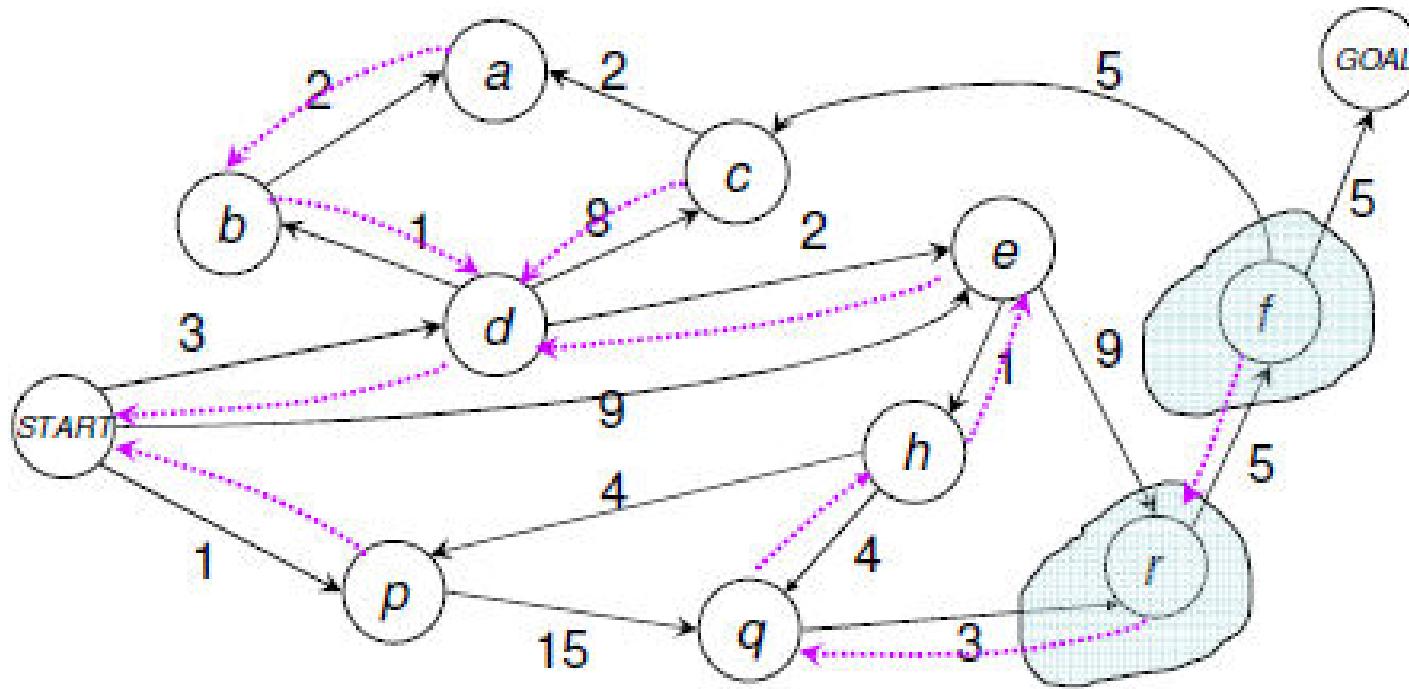


$$PQ = \{(q, 10), (c, 11), \dots\}$$

Important: We realized that going to q through h is cheaper than going through $p \rightarrow$ the value of q is updated from 16 to 10 and it moves up in PQ

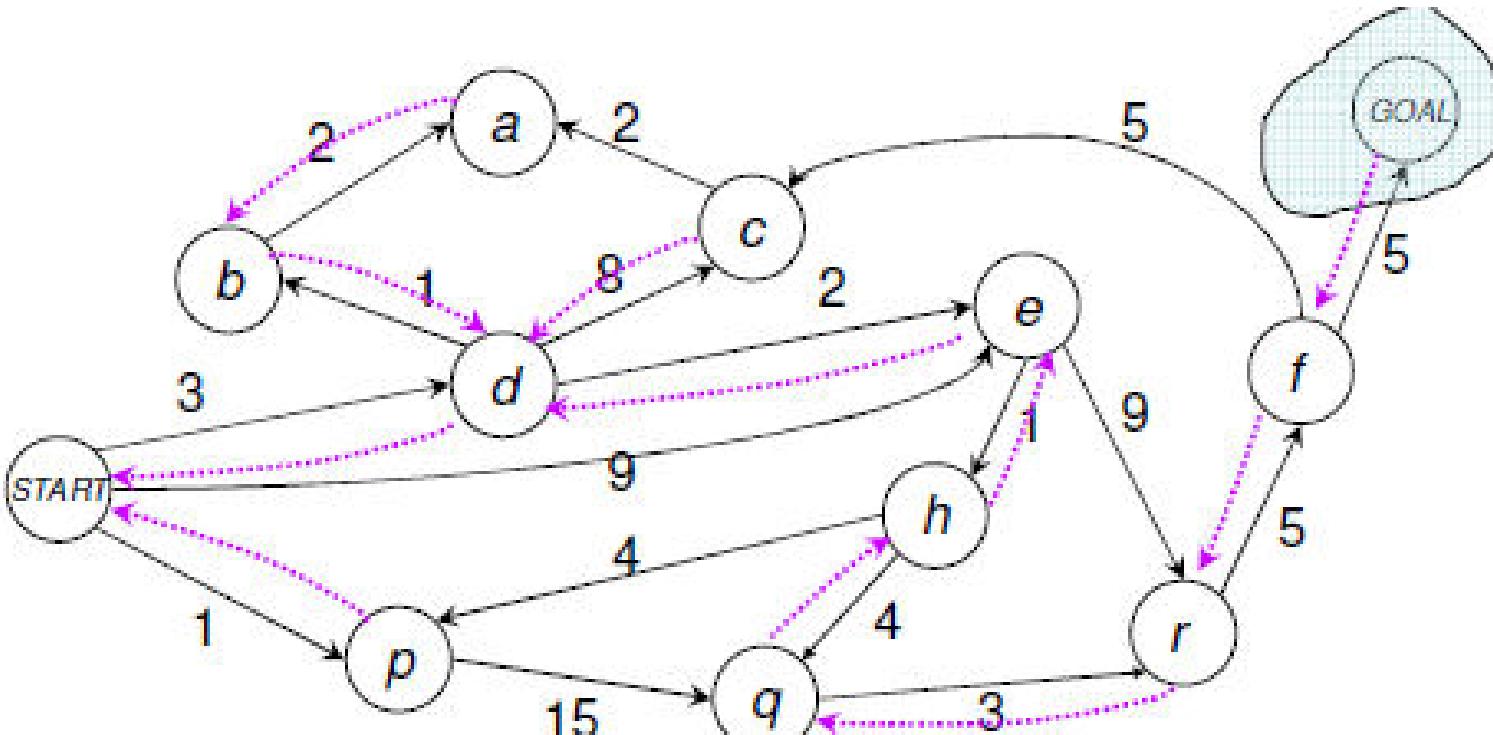


$$PQ = \{(c, 11) (r, 13)\}$$

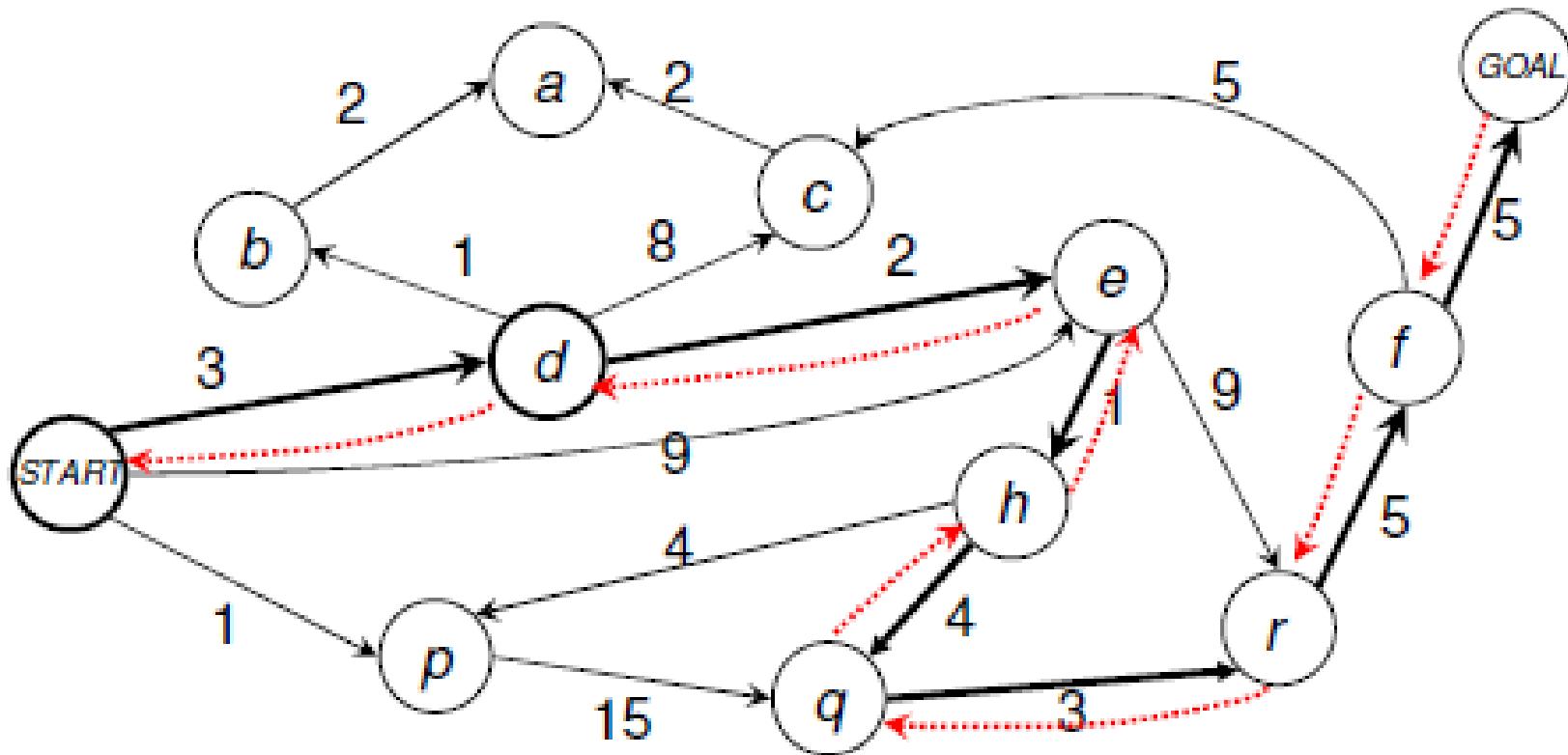


$$PQ = \{(r, 13)\}$$

$$PQ = \{(f, 18)\}$$

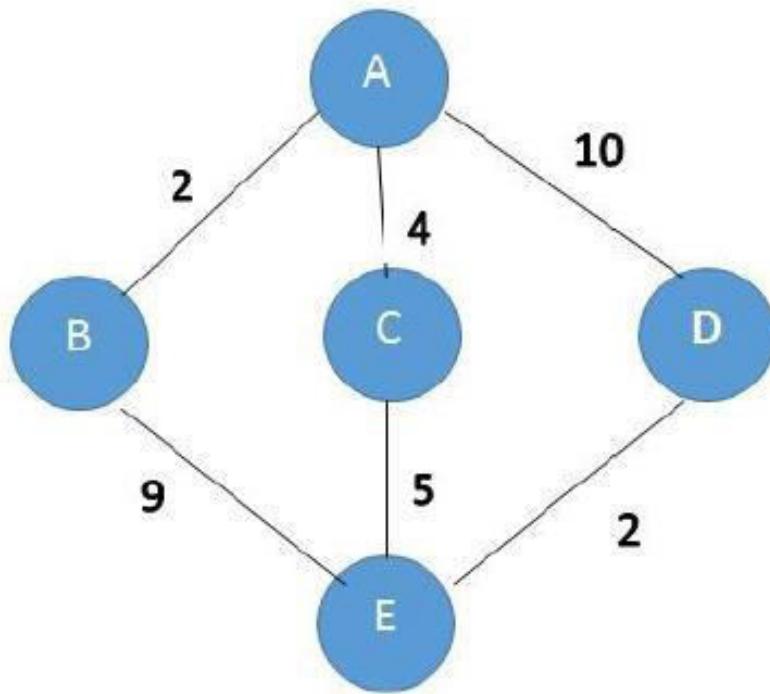


$$PQ = \{(GOAL, 23)\}$$



Final path: {START, d, e, f, GOAL}

Find path from A to E using UCS



Depth Limit Search

- Depth-first search will not find a goal if it searches down a path that has infinite length. So, in general, depth-first search is not guaranteed to find a solution, so it is not complete.
- This problem is eliminated by limiting the depth of the search to some value L . However, this introduces another way of preventing depth-first search from finding the goal: if the goal is deeper than L it will not be found.
- Perform depth first search but only to a pre-specified depth limit L .
- No node on a path that is more than L steps from the initial state

Depth Limit Search

□ *Completeness:*

Incomplete as solution may be beyond specified depth level

□ *Optimality:*

not optimal

□ *Space complexity:*

b as branching factor and L as tree depth level , $O(b \cdot L)$

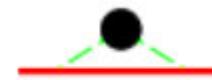
□ *Time Complexity:*

$O(bL)$

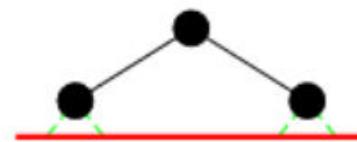
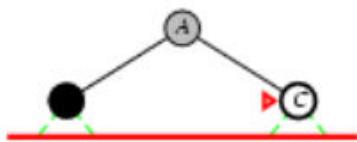
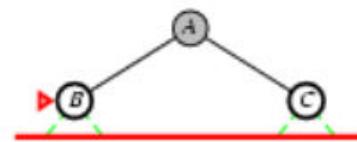
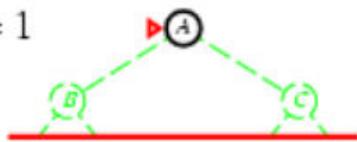
Iterative Deepening Search (IDS)

- › Iterative deepening search is a strategy that sidesteps the issue of choosing the best depth limit by trying all possible depth limits.
- › Starting at depth limit $L = 0$, we iteratively increase the depth limit, performing a depth limited search for each depth limit.
- › Stop if no solution is found, or if the depth limited search failed without cutting off any nodes because of the depth limit.
- › Search is helpful only if the solution is at given depth level.

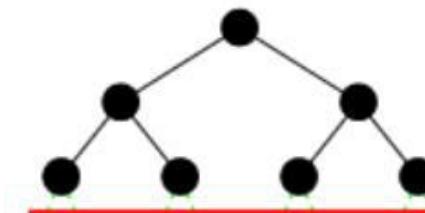
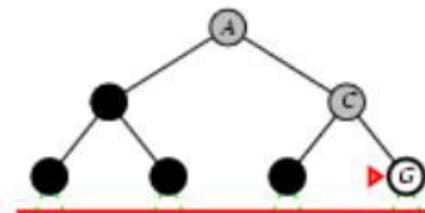
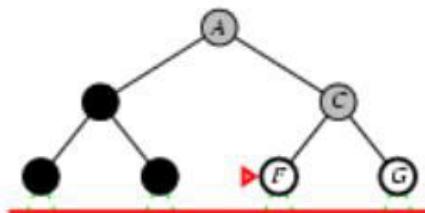
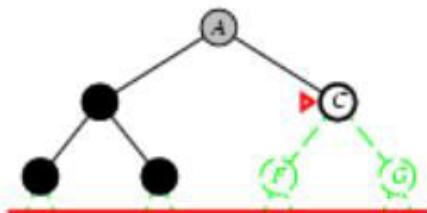
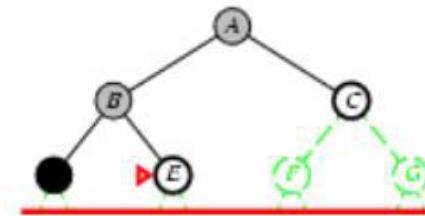
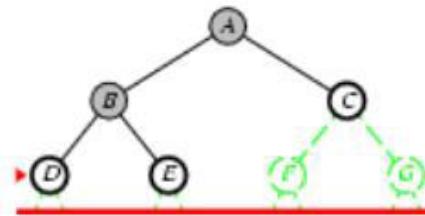
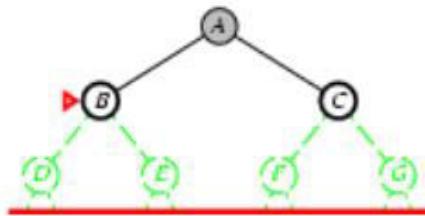
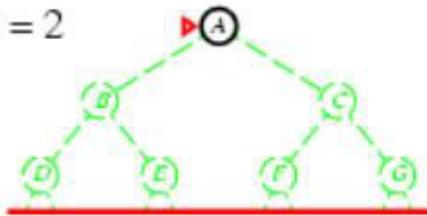
Limit = 0



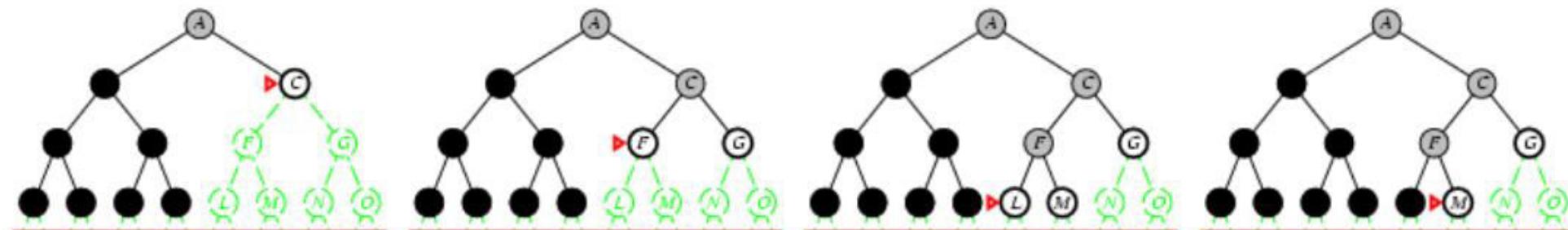
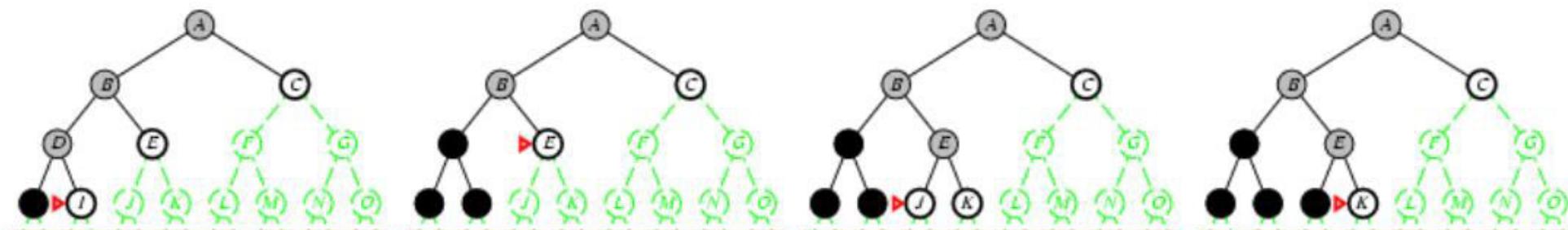
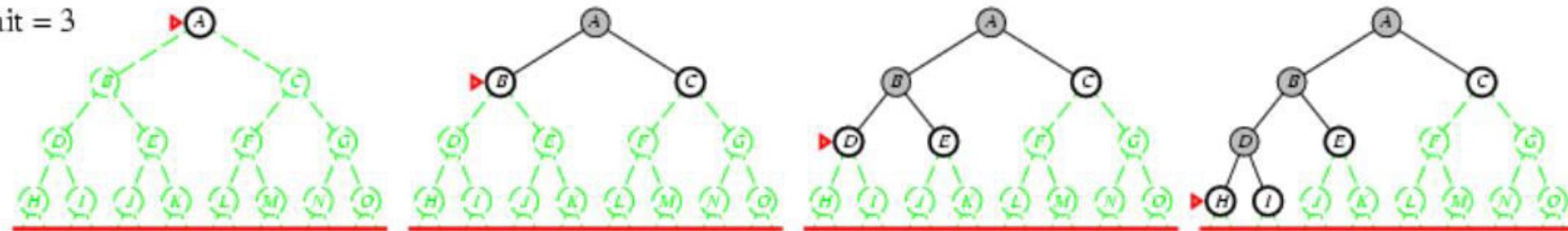
Limit = 1



Limit = 2



Limit = 3



What is the difference between DLS and Iterative Deepening?

DLS is a search strategy resulting when we limit the depth of the Search and use the modest memory requirements of DFS.

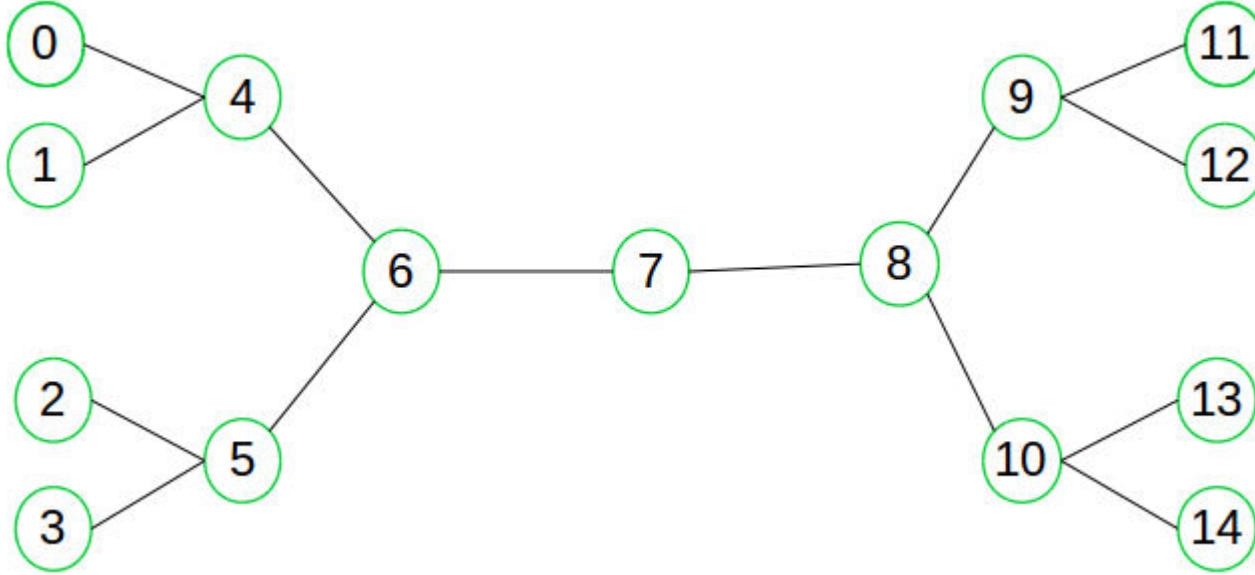
IDS works by looking for the depth d , thus starting with depth limit 0 and make a DFS and if the search failed it increase the depth limit by 1 and try a DFS again with depth 1

Bidirectional Search

- As the name suggests, bidirectional search **suggests** to run 2 simultaneous searches
- It runs two simultaneous search –
 - Forward search from source/initial vertex toward goal vertex
 - Backward search from goal/target vertex toward source vertex
- Bidirectional search replaces single search graph(which is likely to grow exponentially) with two smaller sub graphs – one starting from initial vertex and other starting from goal vertex. **The search terminates when two graphs intersect.**

Bidirectional Search

- Completeness:
Bidirectional search is complete when we use BFS in both searches
- Optimality:
Like the completeness, bidirectional search is optimal when BFS is used.
- Time/Space Complexity : $O(b^{d/2})$



Suppose we want to find if there exists a path from vertex 0 to vertex 14. Here we can execute two searches, one from vertex 0 and other from vertex 14. When both forward and backward search meet at vertex 7, we know that we have found a path from node 0 to 14 and search can be terminated now

Assignment:

Compare all the Search techniques

Informed Search (Heuristic Search)

- Informed search have problem specific knowledge apart from problem definition.
- They use experimental algorithm which improves efficiency of search process.
- The idea is to develop a domain specific heuristic function $h(n)$ where $h(n)$ guesses the cost of getting to the goal from node n .

Heuristic Function

The heuristic function is a way to inform the search about the direction to a goal. It provides an informed way to guess which neighbor of a node will lead to a goal

The heuristic function is denoted by $h(n)$

A heuristic function is said to be **consistent**, or monotone, if it's estimate is always less than or equal to the estimated distance from any neighboring vertex to the goal.

A heuristic is consistent if for every node n , every successor n' of n if

$$f(n') \geq f(n)$$

Best-First Search (BFS)

- Best-First search is a graph-based heuristic search algorithm
- The name “best-first” refers to the method of exploring the node with the best “score” first.
- An evaluation function is used to assign a score to each candidate node. The evaluation function must represent some estimate of the cost of the path from state to the closest goal state

Algorithm

1. Put the initial node on a list START
2. If $\text{START} = \text{GOAL}$ or $\text{START} = \text{EMPTY}$, then terminate search
3. Assign the next node as START and call this node-A
4. If $A = \text{GOAL}$, terminate the search with success
5. Else-if, node has successor and generate all of them. Find out how far they are from the GOAL node.
6. Sort all the children generated so far by remaining distance from the goal. Name the list as START-1. Replace $\text{START} = \text{START-1}$
7. Go to step-2

Types of BFS

- Greedy Best First Search
- A* Search

Greedy Best First Search

- It tries to get as close as it can to the goal.
- It expands the node that appears to be closest to the goal
- It evaluates the node by using heuristic function only.
- Evaluation function $f(n) = h(n)$

$h(n)$ - is estimate of cost from n to goal
 - is 0 for goal state

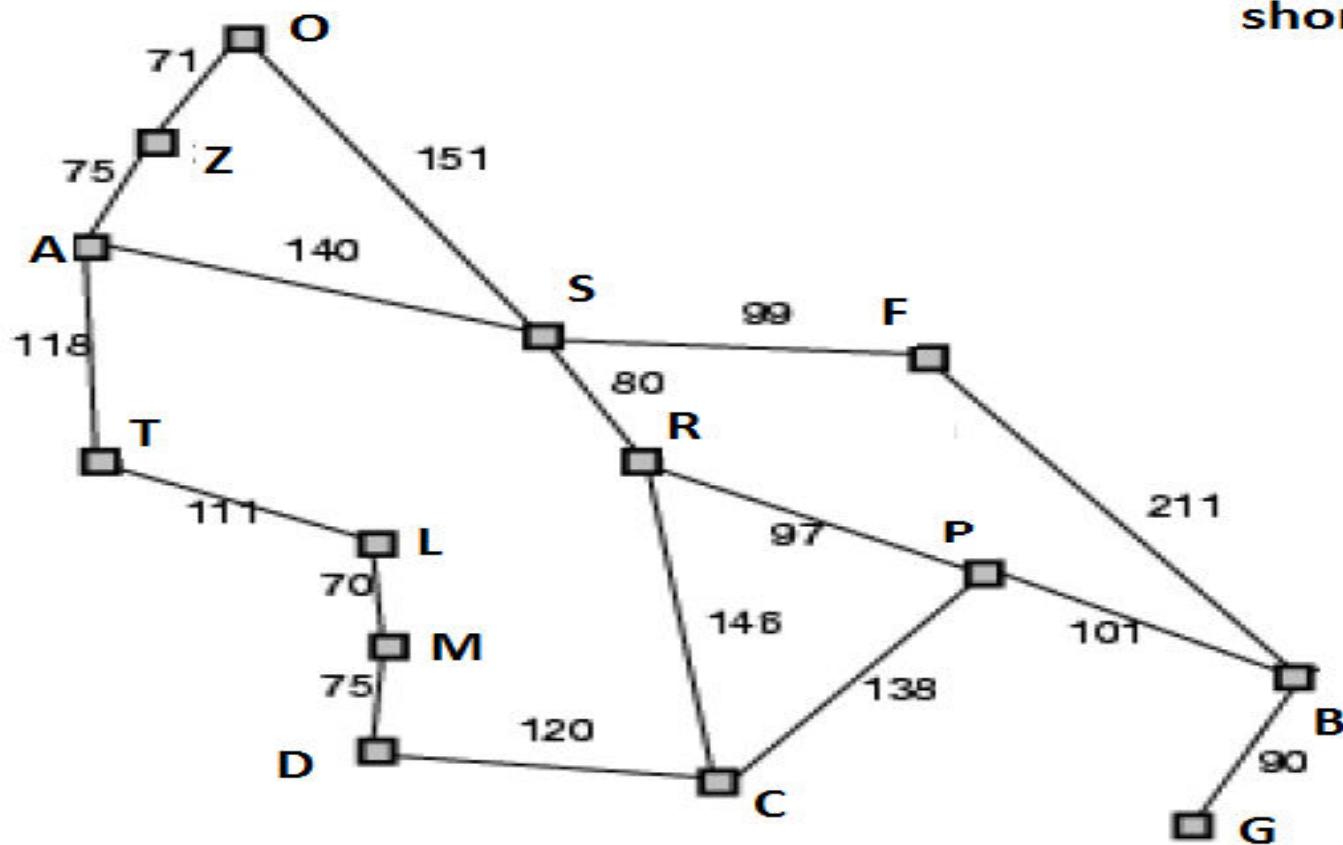
Properties

- ***Completeness:*** No – can get stuck in loops
- ***Time Complexity:*** $O(b^m)$, but a good heuristic can give dramatic improvement
- ***Space Complexity:*** $O(b^m)$, keeps all nodes in memory
- ***Optimality:*** No

Applications:

This algorithm is used in Huffman encoding, minimum spanning tree, Dijkstra's algorithm etc.

Given following graph of cities, starting at City “A”, problem is to reach to the “B”

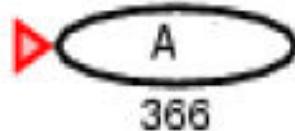


shortest line distance to B

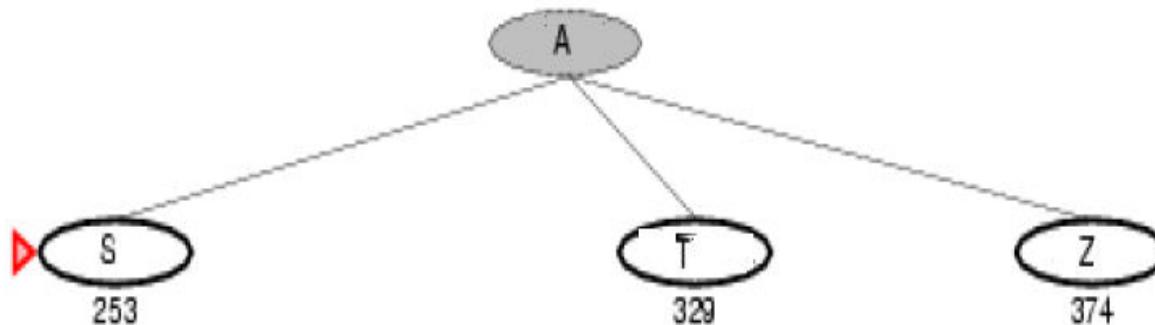
A	366
B	0
C	160
D	242
F	176
G	77
L	244
M	241
O	380
P	101
R	193
S	253
T	329
Z	374

Solution using greedy best first can be as below:

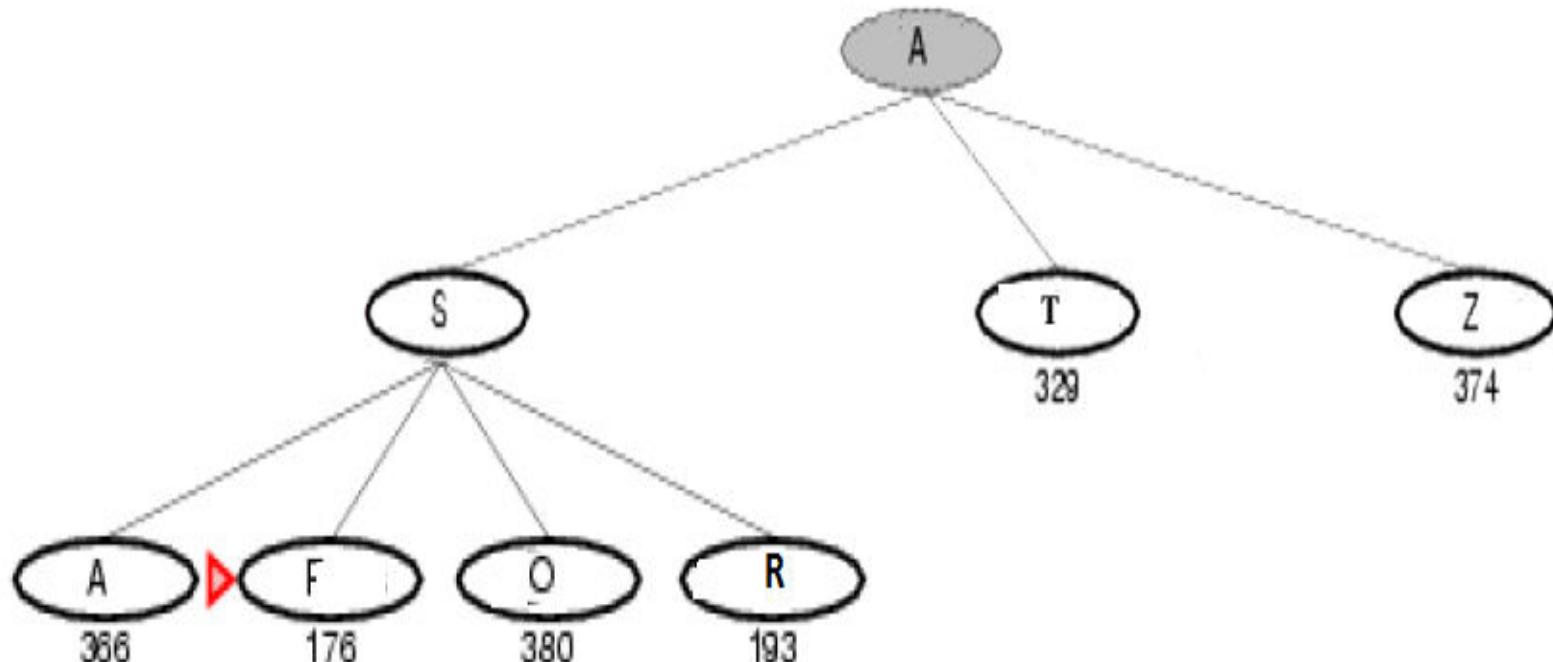
Step 1: Initial State



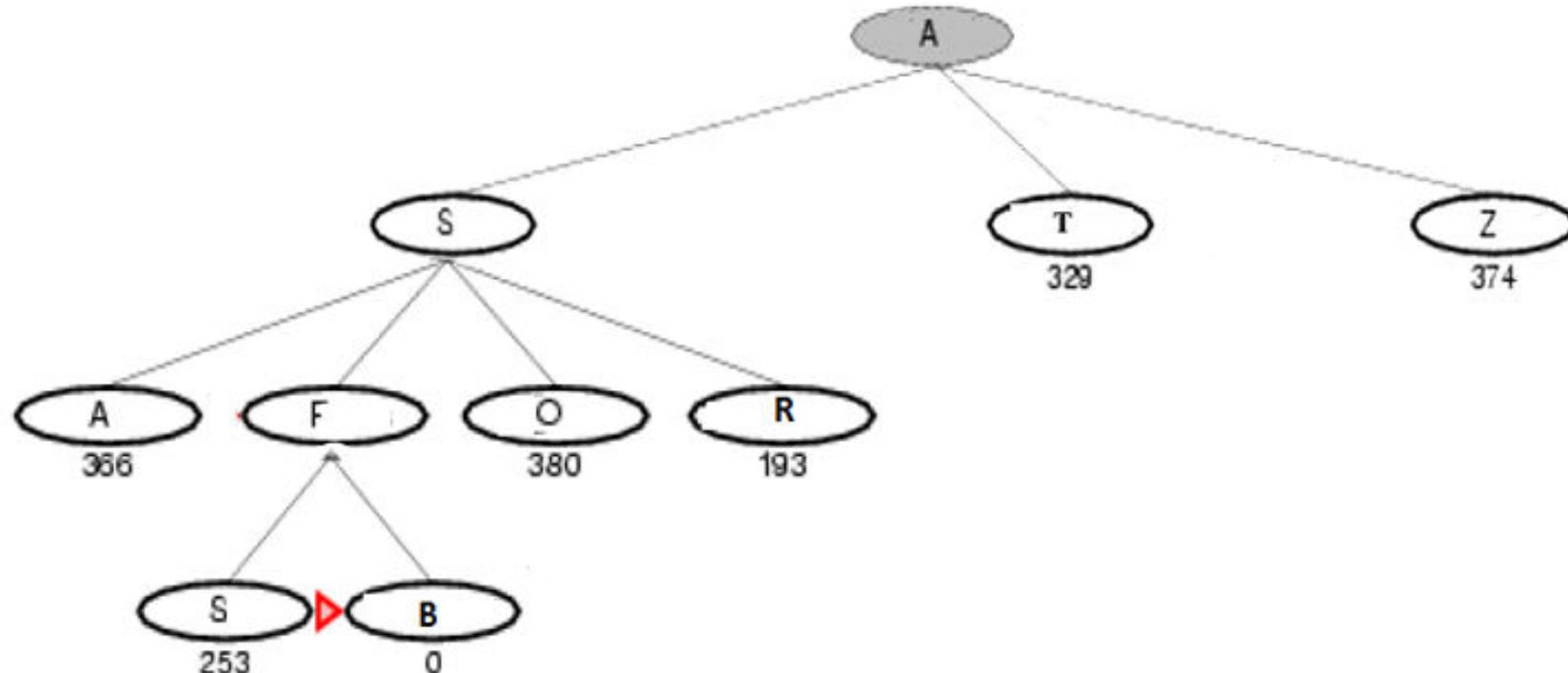
Step 2: After expanding A



Step 3: After expanding S



Step 3: After expanding F



A* Search

- It finds a minimal cost-path joining the start node and a goal node for node n.
- Evaluation function: $f(n) = g(n) + h(n)$

Where,

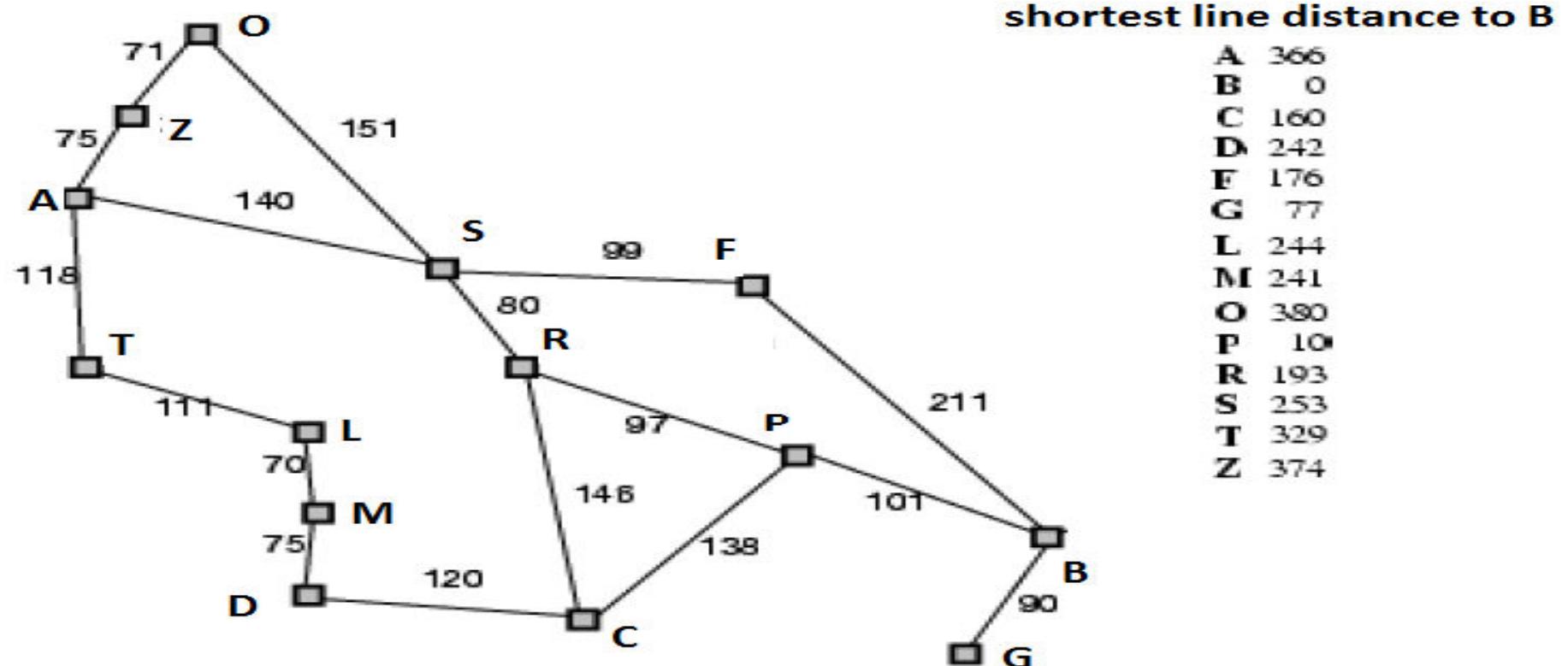
□ $g(n)$ = cost so far to reach n from root

□ $h(n)$ = estimated cost to goal from n

$f(n)$ = estimated total cost of path through n to goal.

- Avoid expanding paths that are already expensive
- The main drawback of A* algorithm and indeed of any best-first search is its memory requirement.

A* search example (Find path from A to B)

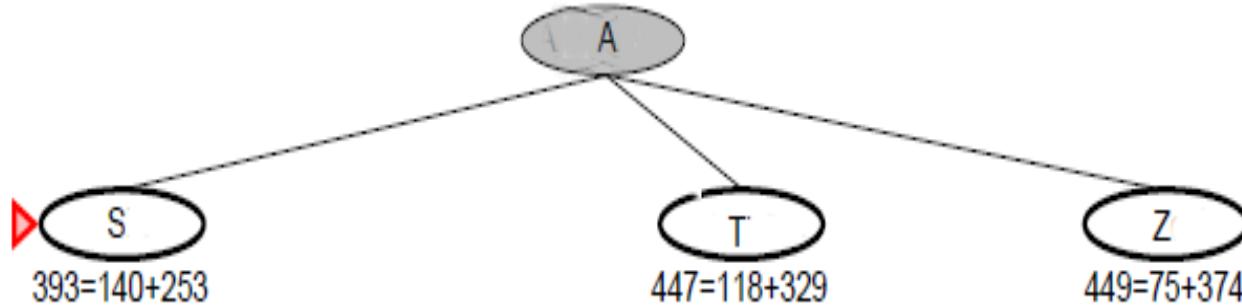


Here, evaluate nodes connected to source. Evaluate $f(n) = g(n) + h(n)$ for each node. Select node with lowest $f(n)$ value.

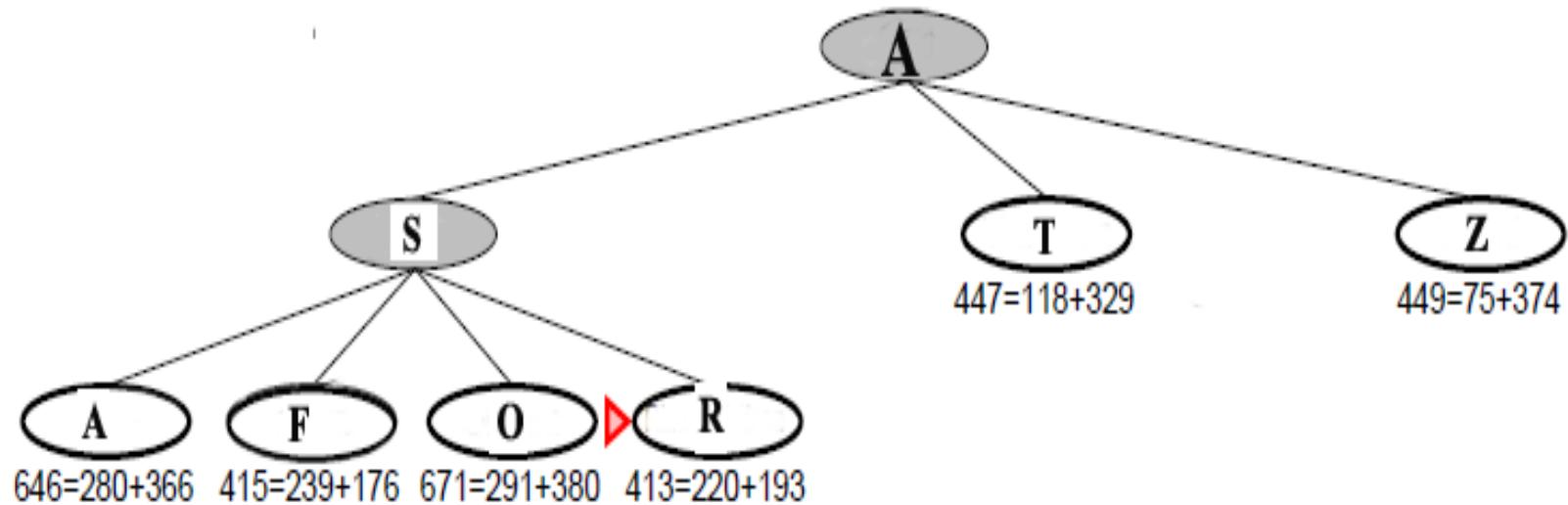
Step 1:



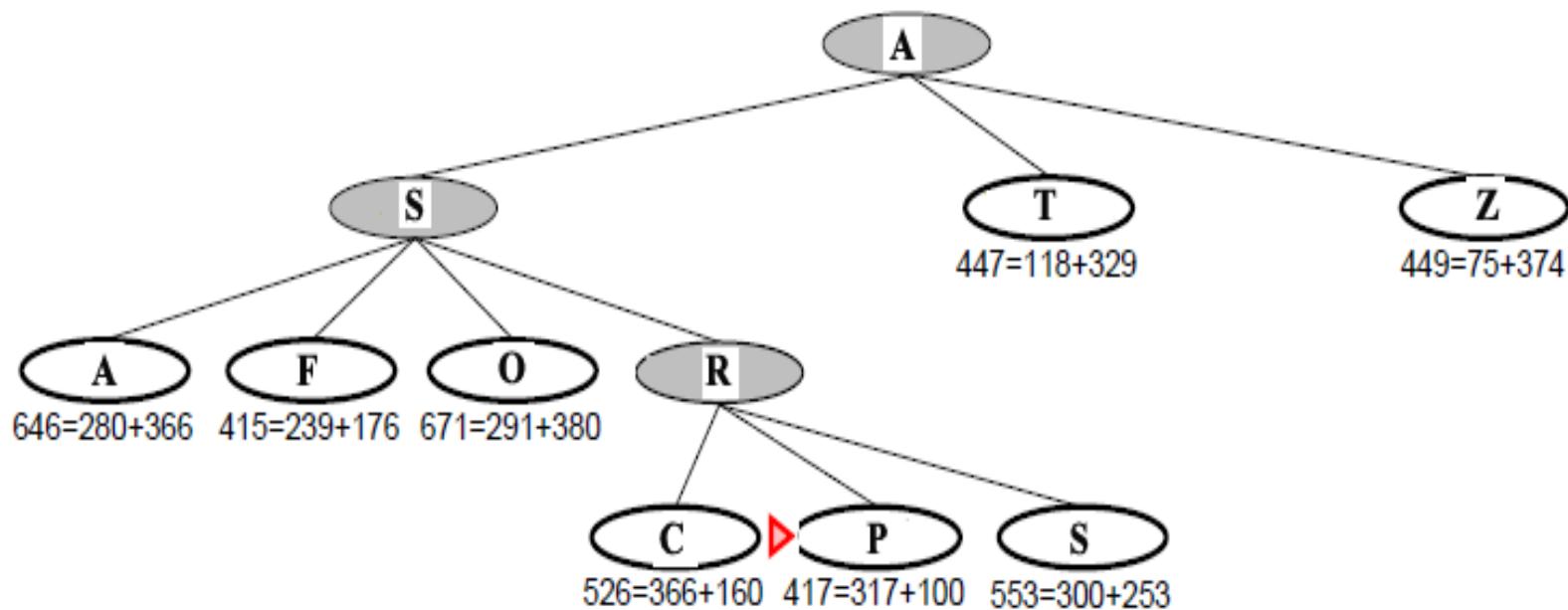
Step 2



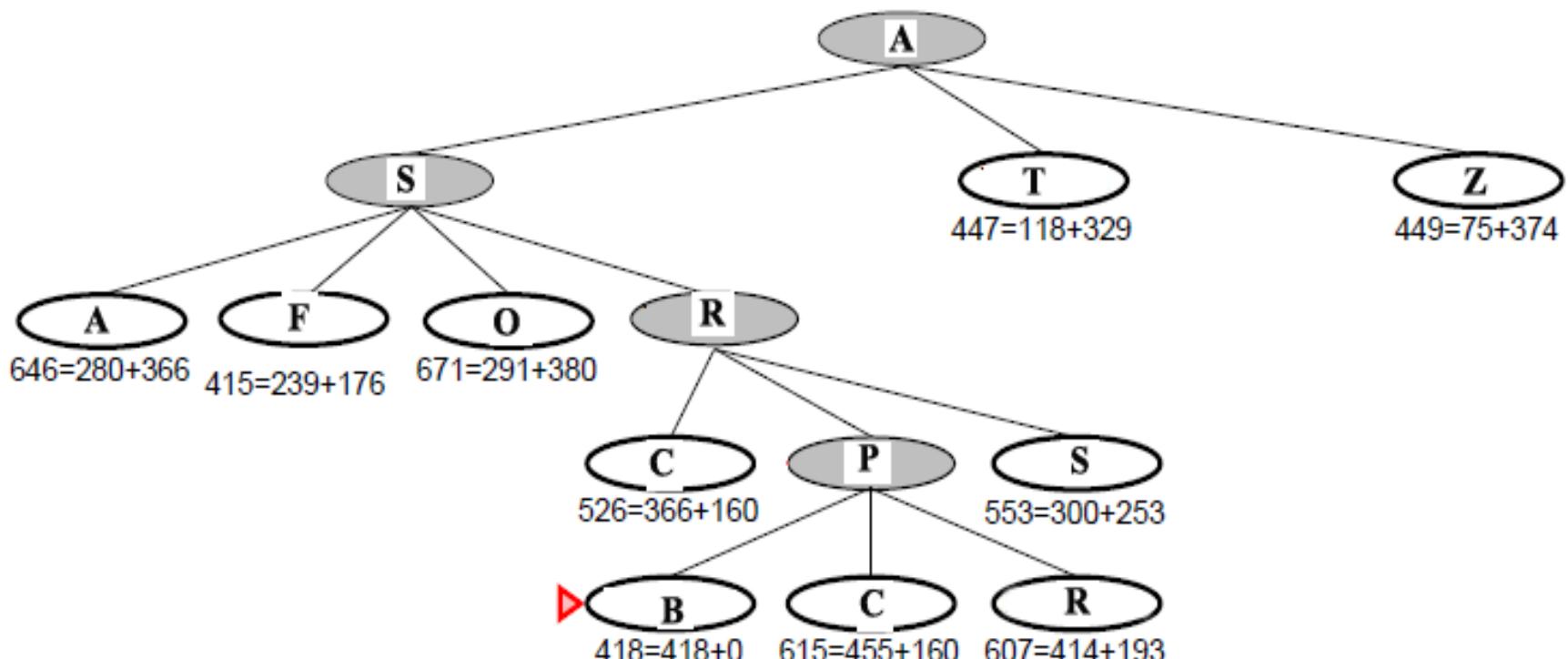
Step 3:



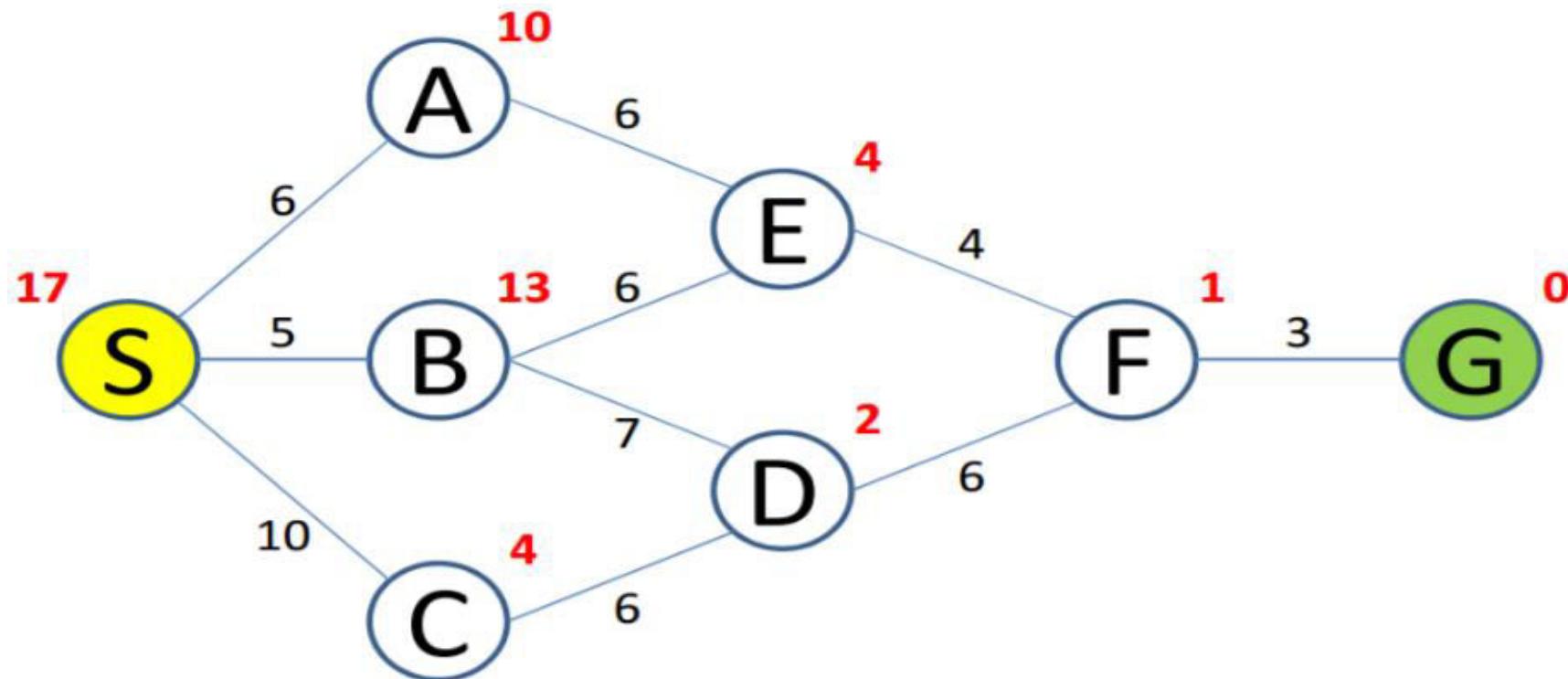
Step 4:



Step 5:

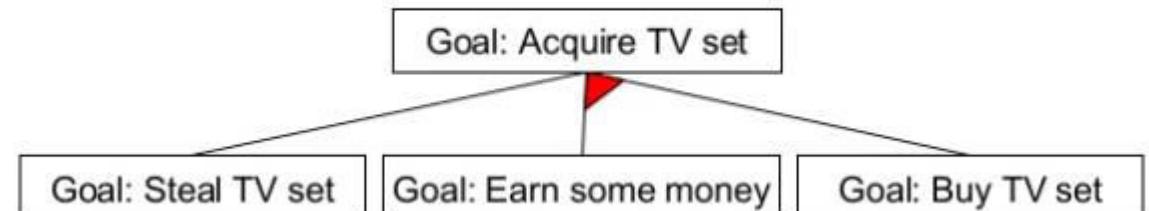


Perform Greedy and A* Search

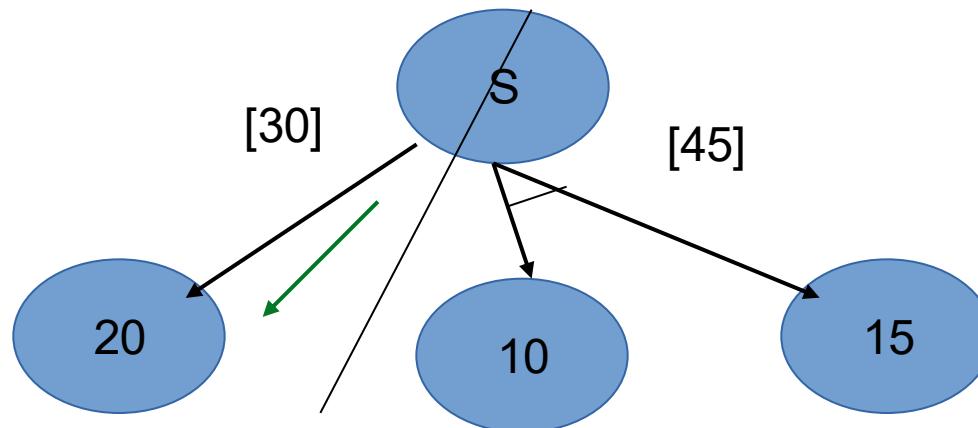


AO* Searching

- AO* Search is a type of heuristic search algorithm .
- AO* Search is used when problems can be divided into sub parts and which can be combined
- AO* in artificial intelligence is represented using AND OR graph or AND OR tree
- AO* have one or more and arc in it

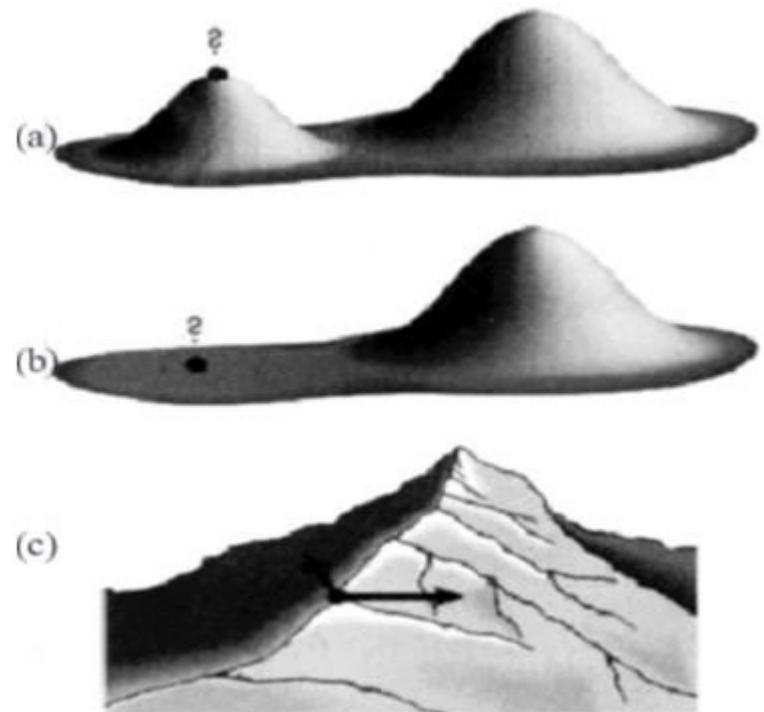


Edge = 10



Hill Climbing Search

- Hill climbing is an extension of depth-first search which uses some knowledge such as estimates of the distance of each node from the goal to improve the search
- It is simply a loop that continually moves in the direction of increasing value—that is, uphill. It terminates when it reaches a “peak” where no neighbor has a higher value.
- Hill climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead about where to go next



Hill Climbing Search

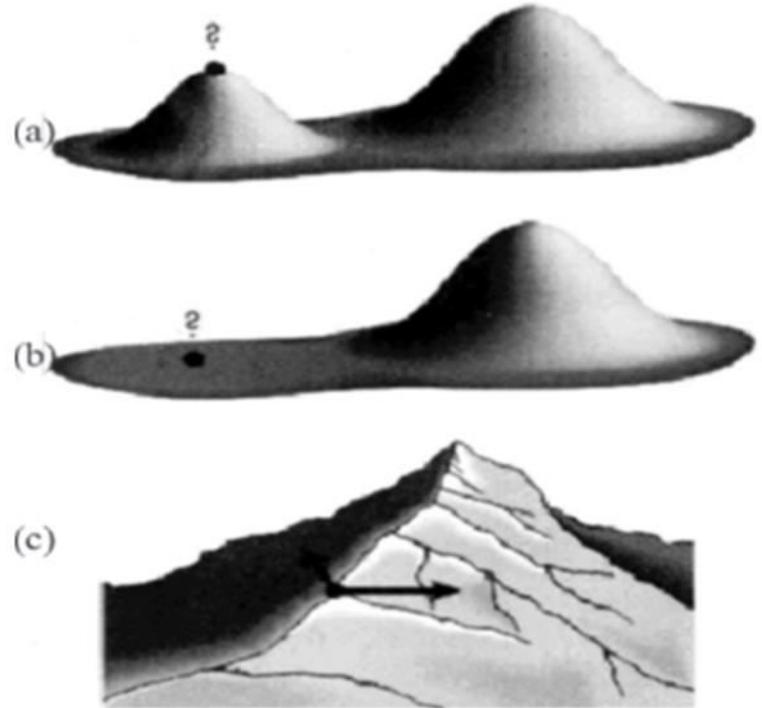
➤ Drawbacks

❑ Local Maxima

A local maximum is a peak that is higher than each of its neighboring states but lower than the global maximum.

❑ Plateaus: A plateau is an area of the search space where evaluation function is flat, thus requiring random walk

❑ Ridges: Where there are steep slopes and the search direction is not towards the top but towards the side



Hill Climbing Search

➤ Remedies

Back tracking for local maximum:

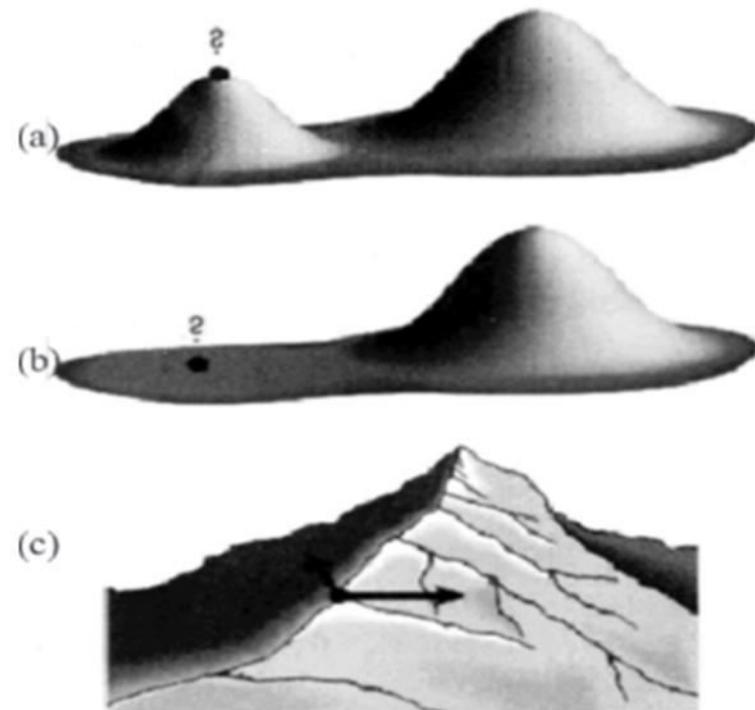
The back tracking help in undoing what is been done so far and permit to try totally different part to attain the global peak.

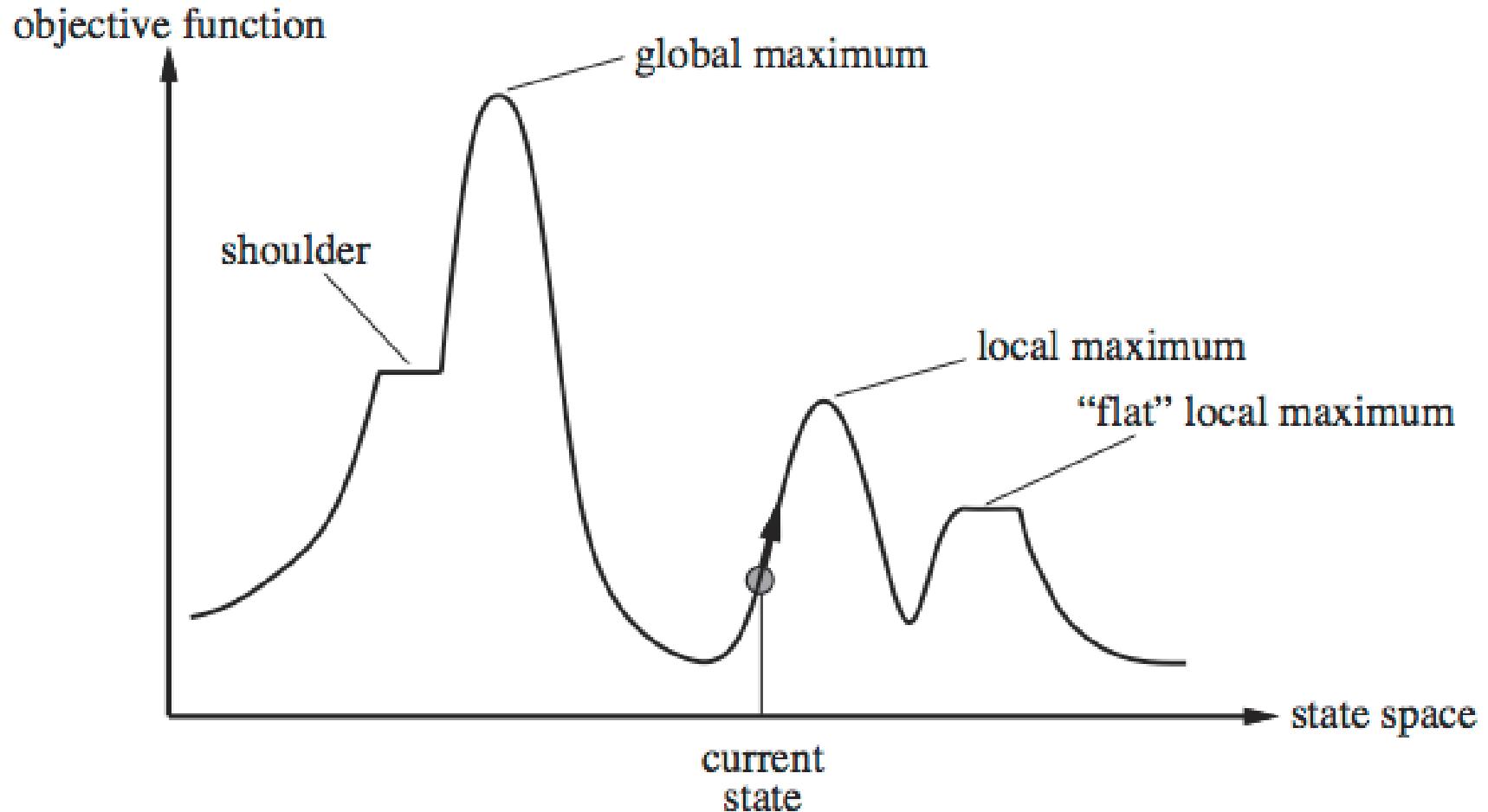
Big Jump:

A big jump is the solution to escape from plateaus because all neighbors' points have same value using the greedy approach

Random restart:

Keep restarting the search from random locations until a goal is found

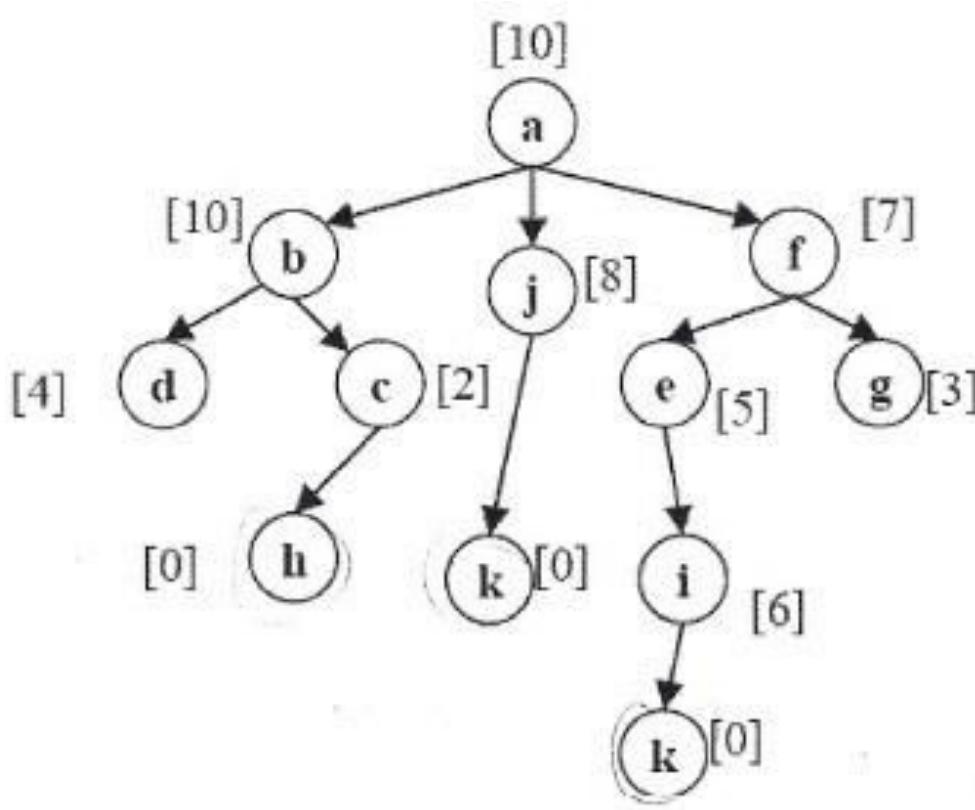




Why Hill Climbing is not Complete?

- Hill-climbing always attempts to make changes that improve the current state.
- The main problem that hill climbing can encounter is that of local maxima. This occurs when the algorithm stops making progress towards an optimal solution; mainly due to the lack of immediate improvement in adjacent states.

Problems in Hill Climbing



Here, "a" is initial and h and k are final states

- › We start a-> f-> g and then what ??finish(without result)
- › A common way to avoid getting stuck in local maxima with Hill Climbing is to use random restarts. In the example if G is a local maxima, the algorithm would stop there and then pick another random node to restart from.
- › So if J or C were picked (or possibly A, B, or D), this would find the global maxima in H or K

Simulated Annealing

- Simulated Annealing escapes local maxima by allowing some "bad" moves but gradually decrease their frequency.
- Instead of restarting from a random point, we allow the search to take some downhill steps to try to escape local maxima

Planning

- Planning is the process of **deciding** in detail how to do something before you actually start to do it.
- It is the task of coming up with a sequence of actions that will achieve a goal.
- **Planner** is viewed as the producer or generator of the solution.

Eg: STRIPS (STanford Research Institute Problem Solver)

➤ **Classical Planning**

Planning for which environments that are fully observable, deterministic, finite, static (change happens only when the agent acts), and discrete (in time, action, objects, and effects)

➤ **Non Classical Planning**

Planning for which environments for partially observable and involves a different set of algorithms and agent designs

PROBLEMS

- A problem is a situation (difficult/easy) experienced by an agent.
- Problem is solved by a sequence of actions that reduce the difference between the initial situation and the goal.

PROBLEM SOLVING

- Problem solving, particularly in artificial intelligence, may be characterized as a **systematic search through a range of possible actions in order to reach some predefined goal or solution.**
- Problem-solving methods divide into **special purpose and general purpose.**
- A special-purpose method is tailor-made for a particular problem and often exploits very specific features of the situation in which the problem is embedded.
- In contrast, a general-purpose method is applicable to a wide variety of problems.

Planning	Problem Solving
The task of coming up with a sequence of actions that will achieve a goal is called planning	Problem Solving is the systematic search through a range of possible actions in order to reach some predefined goal or solution
Planning is a generic term of problem solving	Problem solving comprises standard search process

Four general steps in problem solving:

- Goal formulation
 - What are the successful world states
- Problem formulation
 - What actions and states to consider given the goal
- Search
 - Determine the possible sequence of actions that lead to the states of known values and then choosing the best sequence.
- Execute
 - Give the solution perform the actions.

Problem formulation:

A problem is defined by:

- An initial state: State from which agent starts
- Successor function: Description of possible actions available to the agent.
- Goal test: Determine whether the given state is goal state or not
- Path cost: Sum of cost of each path from initial state to the given state.

A solution is a sequence of actions from initial to goal state. Optimal solution has the lowest path cost.

PROBLEM TYPES

Assignment

Explain Different types of problems

- Toy Problems
- Real-world Problems
- Monkey Banana Problem

Well Defined Problems

A problem when defined with its components is called well defined problem. The actions and rules should be defined in as general way as possible.

A problem can be defined formally by four components:

- **Initial State:**

state from where an agent starts

- **Successor Function:**

description of possible **actions** available to an agent, given a particular state x , a successor function returns a set of $\langle action, successor \rangle$ order pairs, where each action is one of the legal action in state x and each successor is a state that can be reached from x by applying the action.

- **Goal Test:**

determines whether a given state is a goal state

- **Path Cost:**

assigns numeric cost to each path

Defining problem as a state space search

A set of initial state, actions and the goal state for a given problem is known as state space for that problem. A state space represents a problem in terms of states and operators that changes the states. A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes representing the legal moves from one state to another, an initial state and a goal state

The major components of state space representation are:

- Initial state
- Goal state, and
- Operator or legal moves.

Q. Explain problem as a state space with
an example.

To explain problem as a state space representation in more detailed manner, let us consider a problem of 8-puzzle game. The puzzle consists of an 8-square frame and an empty slot. The tiles are numbered from 1-8. It is possible to move the tiles in the square field by moving tile into the empty slot. The objective is to get square in a numeric order.

- **Initial State**

1		2
4	5	3
7	8	6

- **Operators**

- UP: If the empty slot is not touching the up-frame, move it up.
- DOWN: If the empty slot is not touching down-frame, move it down.
- LEFT: If the empty slot is not touching left-frame, move it left.
- RIGHT: If the empty slot is not touching right-frame, move it right.

- **Final State**

1	2	3
4	5	6
7	8	

Q. Vacuum World Problem

The world has only two *locations*

Each location may or may not contain *dirt*

The agent may be in one location or the other

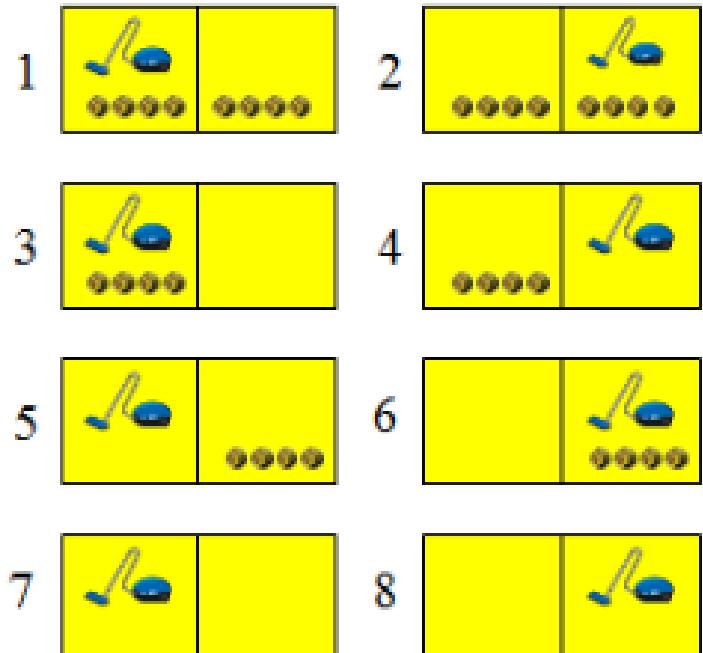
8 possible *world states*

Three possible actions: *Left, Right, Suck*

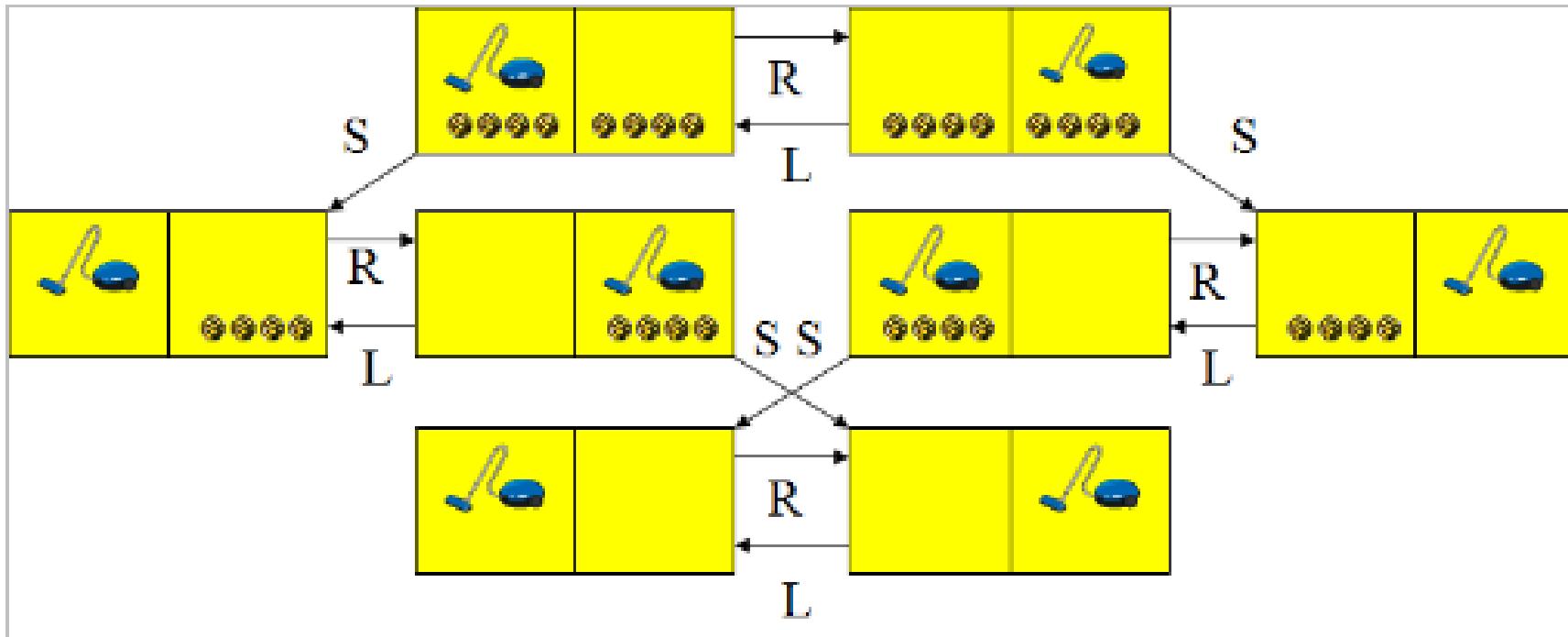
Goal: clean up all the dirt

Here,

- **Initial States:**
One of the 8 states
- **Operators:**
Move left, Move right, Suck
- **Final State:**
No dirt left in any square



- **Solution**



Q. You are given two jugs, a 4-gallon one and a 3-gallon one. Neither has any measuring mark on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into the 4-gallon jug

The state space for this problem can be described as the set of ordered pairs of integers (x,y)

Where,

X represents the quantity of water in the 4-gallon jug $X= 0,1,2,3,4$

Y represents the quantity of water in 3-gallon jug $Y=0,1,2,3$

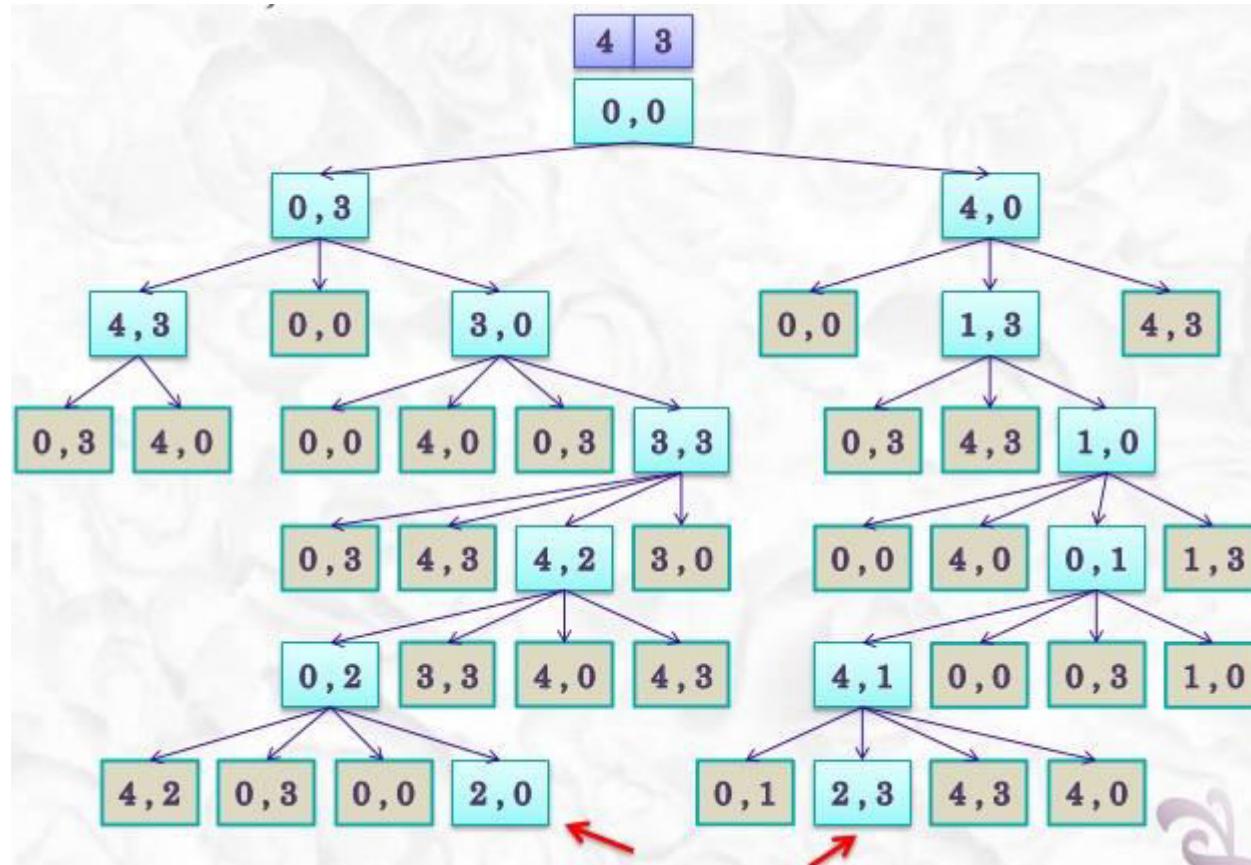
Start State: (0,0)

Goal State: (2,0)

Production Rule:

- Fill 4/3 gallon jug
- Empty 4/3 gallon jug
- Pour from either of the jug to another

Search tree structure Solution:



Q. You are given two jars of a six gallons and eight gallons capacity. There is no marking on the jars. There is a water tap, which can be used to fill jar. Your goal is to have exactly four gallons of water in eight gallons jar without taking any other jar or measuring device.

Q. River Crossing Problem

In a distinct land, bigamy is common. There are six people who want to cross a river in this land. This group consists of two men, each two wives. No man can tolerate any of his wives being in the company of another man unless yet least he or his next wife is present in the boat or in next land. There is a boat that holds two people to be used for crossing the river. How is the trip possible?

- **Initial State**
 $W \{ H_1, W_1, W_1' \text{ and } H_2, W_2, W_2' \}, E \{ \phi \}$
- **Operation**
 - No man can tolerate any of his wives being in the company of another man.
 - Crossing the river
- **Final State**
 $W \{ \phi \}, E \{ H_1, W_1, W_1' \text{ and } H_2, W_2, W_2' \}$

- **Solution**

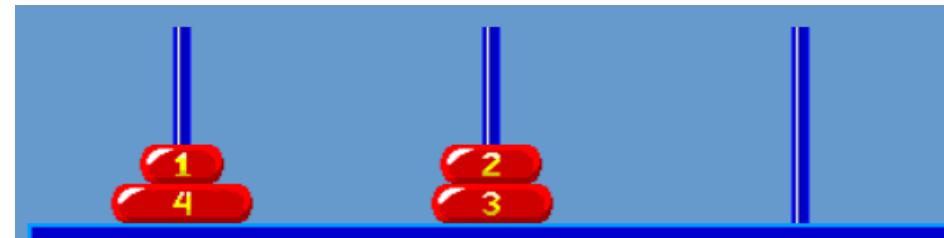
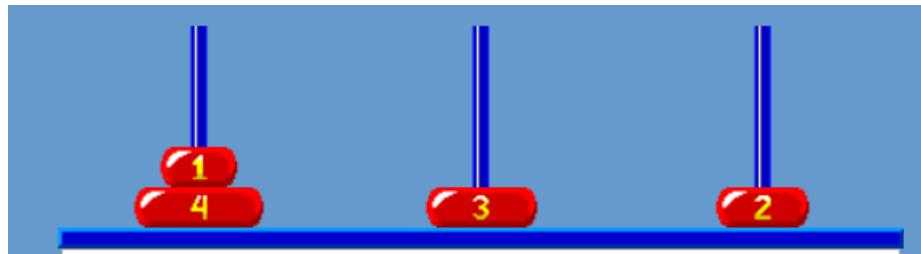
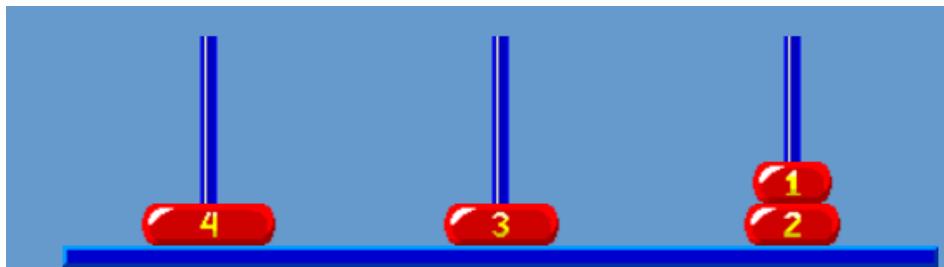
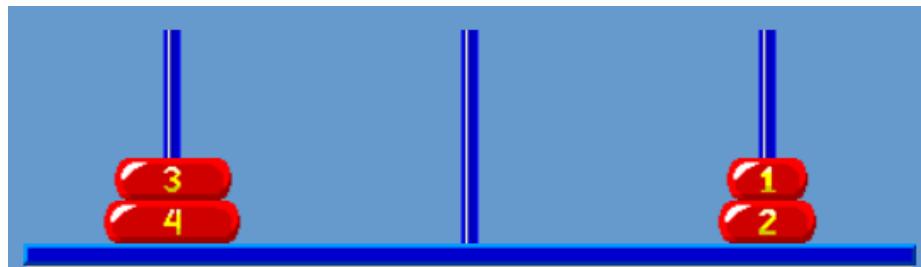
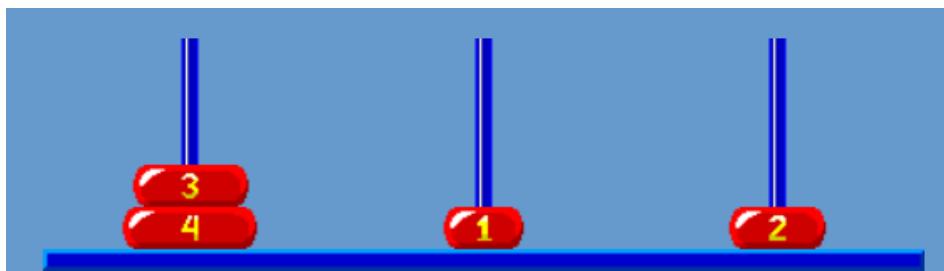
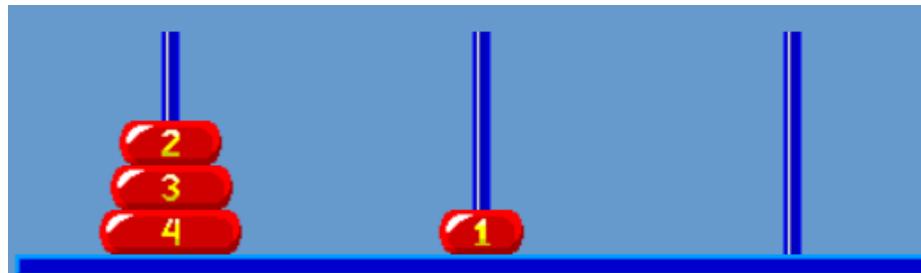
S. N	West (W)	River	East (E)
0.	H1,W1,W1', H2,W2,W2'		(\emptyset)
1.	H1,H2,W2,W2'	W1, W1'	W1,W1'
2.	H1,W1,H2,W2,W2'	W1,	W1'
3.	H1,W1,H2,	W2, W2'	W1',W2,W2'
4.	H1,W1,H2,W2	W2	W1',W2'
5.	H1, H2,	W1, W2	W1,W1',W2,W2'
6.	H1, H2, W2	W2	W1,W1', W2'
7.	H1,	H2, W2	W1,W1', H2,W2, W2'
8.	H1,H2,	H2,	W2
9.	(\emptyset)	H1, H2,	H1, W1,W1', H2,W2, W2'

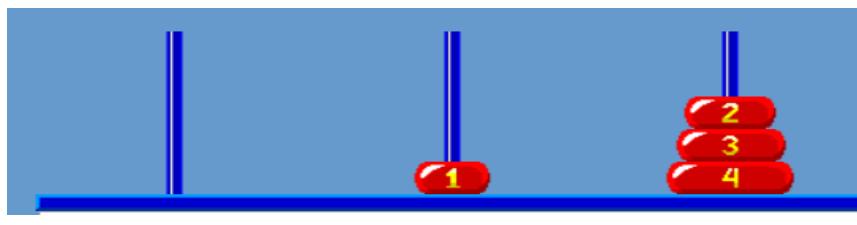
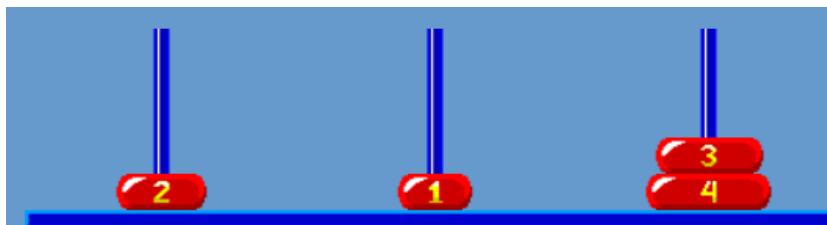
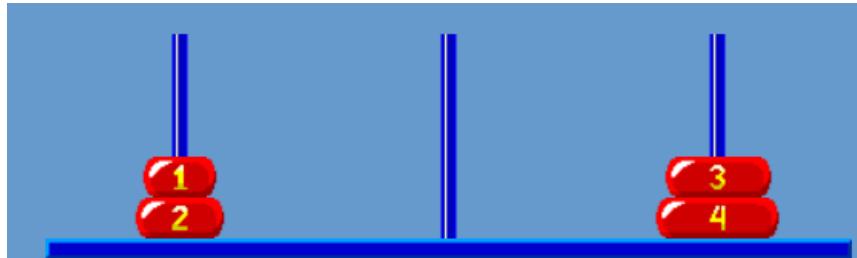
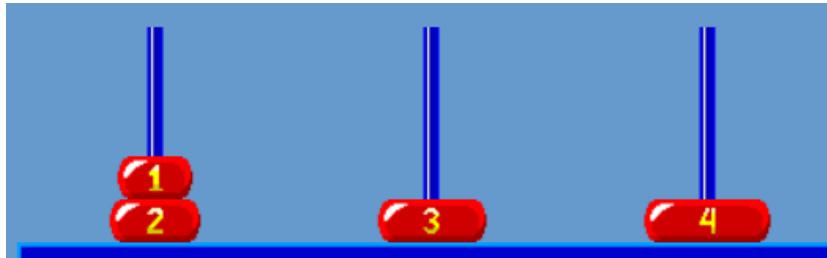
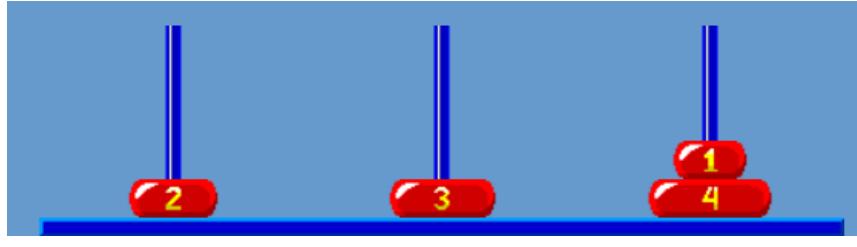
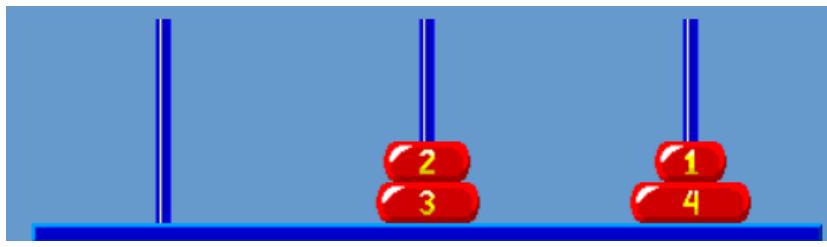
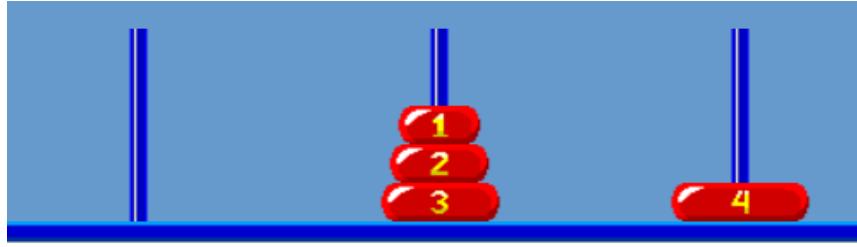
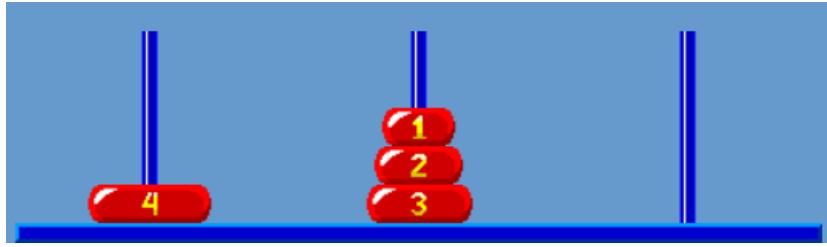
Q. Tic Tac Toe problem

Two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game

Q. Pegs and Disks problem





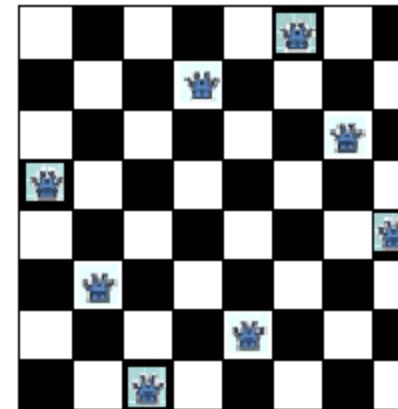
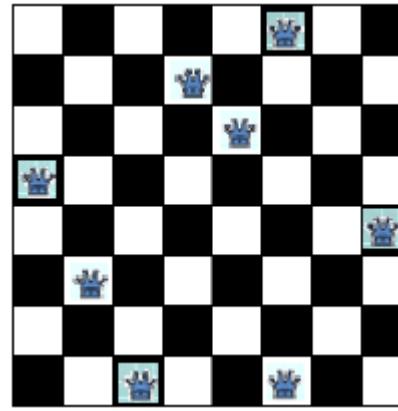


Q.8 queen's problem

The problem is to place 8 queens on a chessboard so that no two queens are in the same row, column or diagonal

Here,

- **Initial State**
- **Operation**
 - Add a queen in any square
- **Final State**



Constraint Satisfaction Problems

A **constraint satisfaction problem** (CSP) consists of

- a set of variables,
- a domain for each variable, and
- a set of constraints.

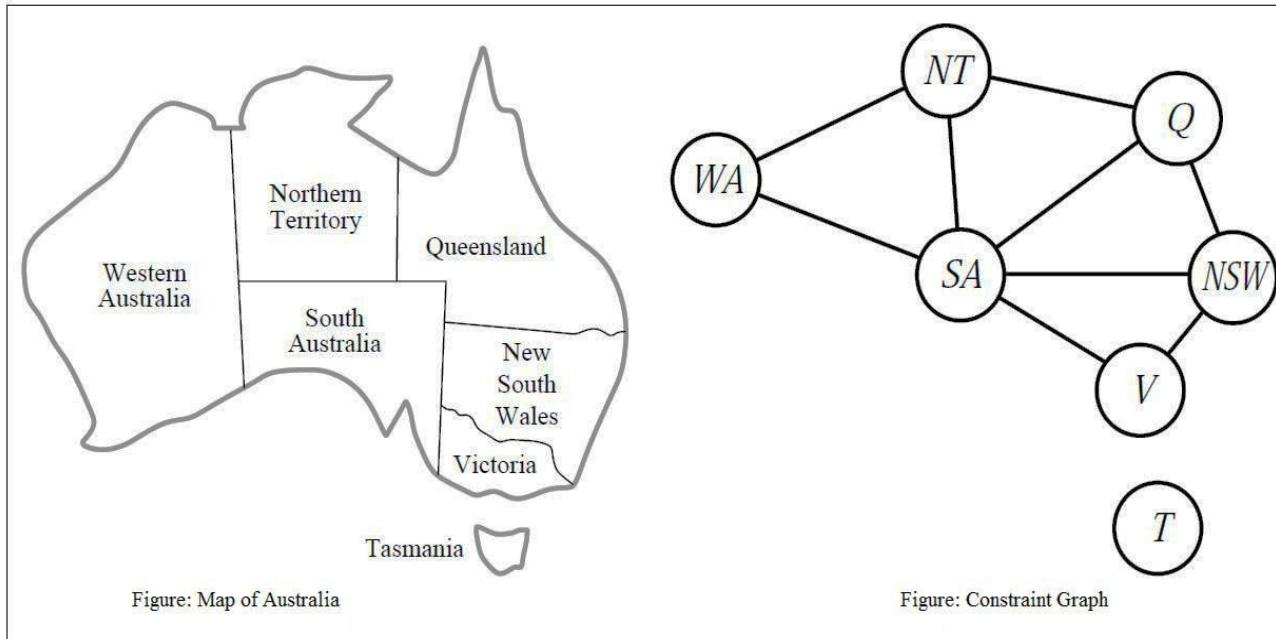
The aim is to choose a value for each variable so that the resulting possible world satisfies the constraints; we want a model of the constraints.

Constraint Satisfaction Problems

- ✓ The process of finding a solution to set of variables V_i (V_1, V_2, \dots, V_n) and a set of constraint C_i (C_1, C_2, \dots, C_m)
- ✓ There is no any specified rule to define the procedure to solve the CSP.
Eg : **8-queens problem**
- ✓ The CSP is a two-step process
 1. Constraints are discovered and propagated as far as possible
 2. There is still not found solution, then search begins

Constraint Satisfaction Problems

Problem: map of Australia showing each of its states and territories, and that we are given the task of coloring each region either red, green, or blue in such a way that no neighboring regions have the same color.



Constraint Satisfaction Problems

CSP Formulation:

- define the variables to be the regions: $X=\{WA, NT, Q, NSW, V, SA, T\}$
- domain of each variable is $D=\{red, green, blue\}$
- The constraints require neighboring regions to have distinct colors, for example, the allowable combinations for WA and NT are the pairs $\{(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)\}$



Constraint Satisfaction Problems



WA

NT

Q

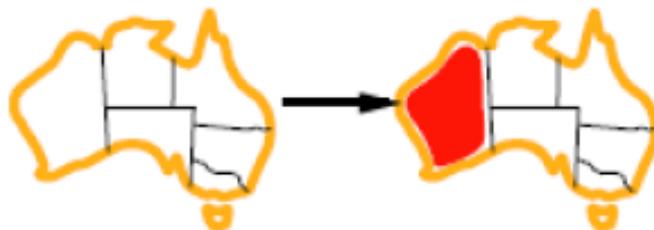
NSW

V

SA

T

Constraint Satisfaction Problems



WA

NT

Q

NSW

V

SA

T

A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.
A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.	A row of seven colored squares: red, green, blue, red, green, blue, red.

Constraint Satisfaction Problems



Figure: Possible solution

There are many possible solutions, such as:

$$\{WA=\text{red}, NT=\text{green}, Q=\text{red}, NSW=\text{green}, V =\text{red}, SA=\text{blue}, T =\text{green}\}$$

Crypto Arithmetic Problems

- › Crypt-Arithmetic Problems are substitution problems where digits representing a mathematical operation are replaced by unique digits.
- › Eg:

The diagram illustrates a mapping or transformation between two sets of four-digit numbers. On the left, a 4x4 grid contains the letters A through I. An arrow points to the right, leading to another 4x4 grid which contains the digits 0 through 9. This represents a one-to-one correspondence where each letter is mapped to a unique digit.

	A	B	C
+	D	E	F
	G	H	I

	0	2	4
+	5	8	9
	6	1	3

Solution:

$$1. \ A \neq B \neq C \neq D \neq E \neq F \neq G \neq H \neq I$$

$$2. \ C + F = I$$

$$C + F = 10 + I \text{ (I as carry)}$$

$$3. \ B + E = H$$

$$B + E = 10 + H$$

$$B + E + 1 = H$$

$$B + E + 1 = 10 + H \text{ (H as carry)}$$

$$4. \ A + D = G$$

$$A + D + 1 = G$$

Step 1:

Domain of C = {1, 2, 3, 4, 5, 6, 7, 8, 9}

Domain of F = {1, 2, 3, 4, 5, 6, 7, 8, 9}

So,

Domain of I = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Select C = 4 & F = 9 Then I = 3 (Carry = 1)

So,

	A	B	4
+	D	E	9
	G	H	3

Step 2:

Domain of B = {1, 2, 5, 6, 7, 8}

Domain of E = {1, 2, 5, 6, 7, 8}

So,

Domain of H = {0, 1, 2, 5, 6, 7, 8}

Select B = 2 & E = 8

Then H = 10+1 {previous carry = 1 + (Carry = 1)}

So,

	A	2	4
+	D	8	9
	G	1	3

Step 3:

Domain of A = {0, 5, 6, 7}

Domain of D = {0, 5, 6, 7}

So, Domain of G = {5, 6}

Select A = 0 & D = 5 Then G = 6 (with addition of Carry)

So,

Hence, the required solutions are:

A = 0, B = 2, C = 4, D = 5, E = 8, F = 9, G = 6, H = 1, I = 3.

T R U E

+ T R U E

F A L S E

Step 1:

Domain of F = {1}

So, Select F = 1

Now we have,

$$\begin{array}{r} \text{T} \quad \text{R} \quad \text{U} \quad \text{E} \\ + \quad \text{T} \quad \text{R} \quad \text{U} \quad \text{E} \\ \hline 1 \quad \text{A} \quad \text{L} \quad \text{S} \quad \text{E} \end{array}$$

Step 2:

Domain of E = {0}

So, Select E = 0

Now we have,

$$\begin{array}{r} \text{T} \quad \text{R} \quad \text{U} \quad 0 \\ + \quad \text{T} \quad \text{R} \quad \text{U} \quad 0 \\ \hline 1 \quad \text{A} \quad \text{L} \quad \text{S} \quad 0 \end{array}$$

Step 3:

Domain of U = {2, 3, 4, 6, 7, 8, 9}

So, Domain of S = {2, 4, 6, 8}

Select U = 4 Then S = 8

So,

$$\begin{array}{r} \text{T} \quad \text{R} \quad 4 \quad 0 \\ + \quad \text{T} \quad \text{R} \quad 4 \quad 0 \\ \hline 1 \quad \text{A} \quad \text{L} \quad 8 \quad 0 \end{array}$$

Step 4:

Domain of R = {3, 6}

So, Domain of L = {2, 6}

Select R = 6 Then L = 2 (Carry = 1)

So,

$$\begin{array}{r} \text{T} & 6 & 4 & 0 \\ + & \text{T} & 6 & 4 & 0 \\ \hline 1 & \text{A} & 2 & 8 & 0 \end{array}$$

Step 5:

Domain of T = {7}

So, Domain of A = {5}

Finally we have,

$$\begin{array}{r} 7 & 6 & 4 & 0 \\ + & 7 & 6 & 4 & 0 \\ \hline 1 & 5 & 2 & 8 & 0 \end{array}$$

Hence, the required solutions are:

T = 7, R = 6, U = 4, E = 0, F = 1, A = 5, L = 2, S = 8.

Class Work

$$\begin{array}{r} \text{S} \quad \text{E} \quad \text{N} \quad \text{D} \\ + \quad \text{M} \quad \text{O} \quad \text{R} \quad \text{E} \\ \hline \text{M} \quad \text{O} \quad \text{N} \quad \text{E} \quad \text{Y} \end{array}$$

$$\begin{array}{r} 9 \quad 5 \quad 6 \quad 7 \\ + \quad 1 \quad 0 \quad 8 \quad 5 \\ \hline 1 \quad 0 \quad 6 \quad 5 \quad 2 \end{array}$$

Class Work

	W	R	O	N	G
+	W	R	O	N	G
	R	I	G	H	T

Class Work

		B	A	S	E
	+	B	A	L	L
	G	A	M	E	S

Game Playing

A game can be formally defined as a kind of search problem as below:

- **Initial state:**

It includes the board position and identifies the player's to move.

- **Successor function:**

It gives a list of (move, state) pairs each indicating a legal move and resulting state.

- **Terminal test:**

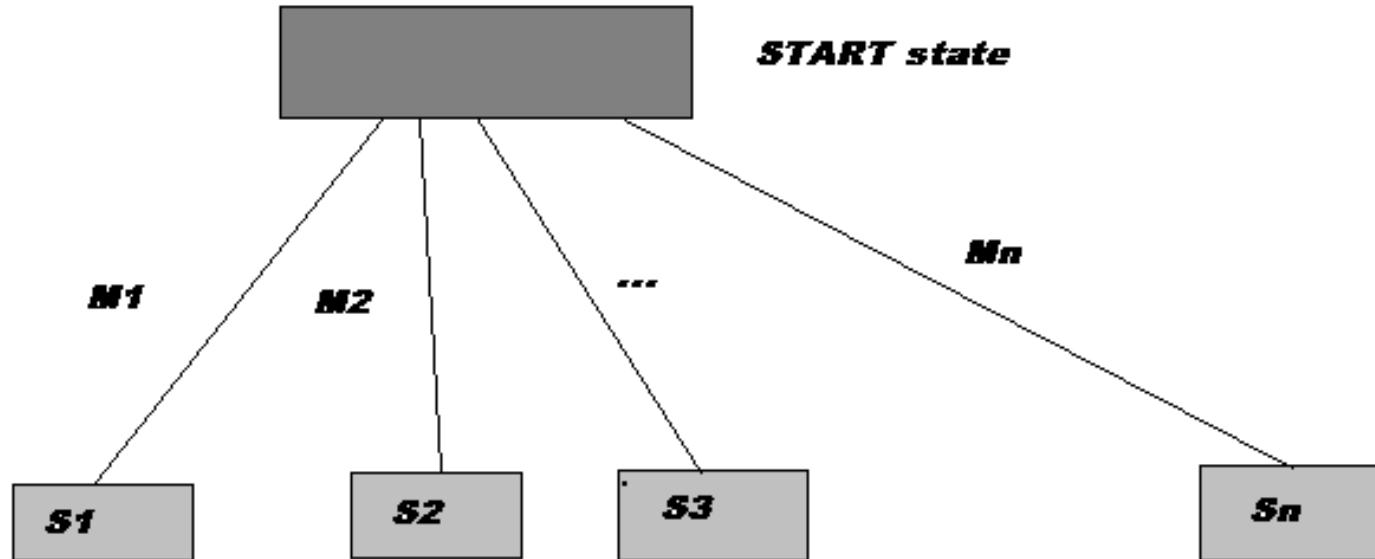
This determines when the game is over. States where the game is ended are called terminal states.

- **Utility function:**

It gives numerical value of terminal states. E.g. win (+1), loose (-1) and draw (0).

Game Trees

- › We can represent all possible games(of a given type) by a directed graph often called a game tree.
- › The nodes of the graph represent the states of the game.
- › The arcs of the graph represent possible moves by the players (+ and -)



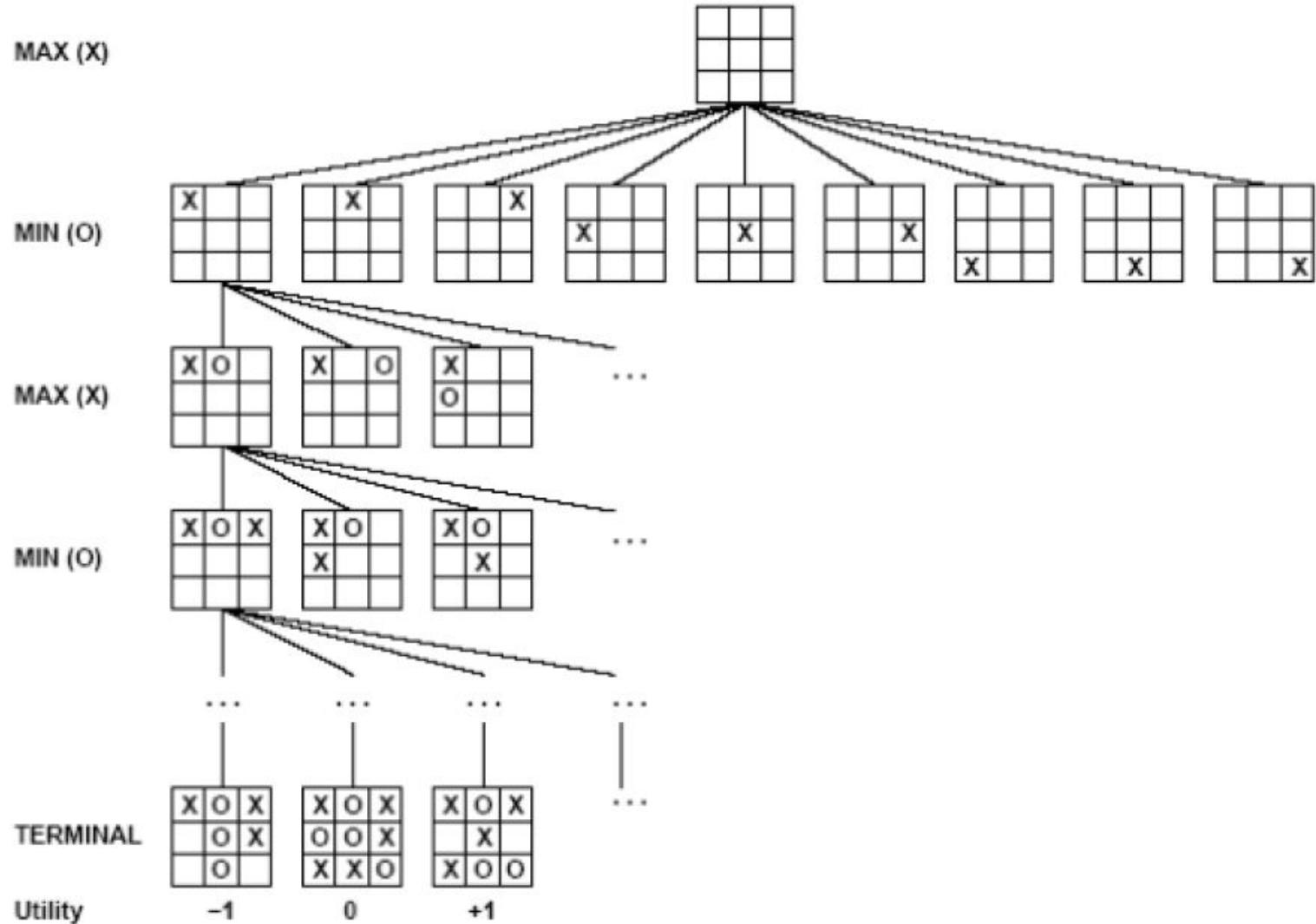
Example: Tic-tac-toe

There are two players denoted by X and O. They are alternatively writing their letter in one of the 9 cells of a 3 by 3 board. The winner is the one who succeeds in writing three letters in line.

The game begins with an empty board. It ends in a win for one player and a loss for the other, or possibly in a draw.

A complete tree is a representation of all the possible plays of the game. The root node is the initial state, in which it is the first player's turn to move (the player X).

The successors of the initial state are the states the player can reach in one move, their successors are the states resulting from the other player's possible replies, and so on.



Adversarial Search

- Competitive environments in which the agents' goals are in conflict, give rise to adversarial (oppositional) search, often known as games.
- In AI, games means fully observable environments in which there are two agents whose actions must alternate and in which utility values at the end of the game are always equal and opposite.
 - E.g. If first player wins, the other player necessarily loses.

Evaluation Function

An evaluation function, also known as a heuristic evaluation function or static evaluation function, is a **function used by game-playing programs to estimate the value or goodness of a position in the board game**. The function looks only at the current position and does not explore possible moves (therefore static). Typically, evaluate how good it is for the player, how good it is for the opponent, and then subtracts the opponent's score from the player's.

Typical values : –infinity (loss) to +infinity (win) or $[-1, +1]$ or $[0, +1]$.

Mini max Algorithm

- It is a recursive algorithm for choosing the next move in a n-player game, usually a two player game
- The value is computed by means of a position evaluation function and it indicates how good it would be for a player to reach the position.
- The player then makes the move that **maximizes the minimum value of the position from the opponents possible moves** called maximizing player and **other player minimize the maximum value of the position** called minimizing player.

Mini Max Game Search

- It is a Depth-first search with limited depth.
- Assume the opponent will make the best move possible.
- Algorithm
 - *mini max (player, board)*
 - *if(game over in current board position)
return winner*
 - *if(max's turn)
return maximal score of calling minimax*
 - *else (min's turn)
return minimal score of calling minimax*

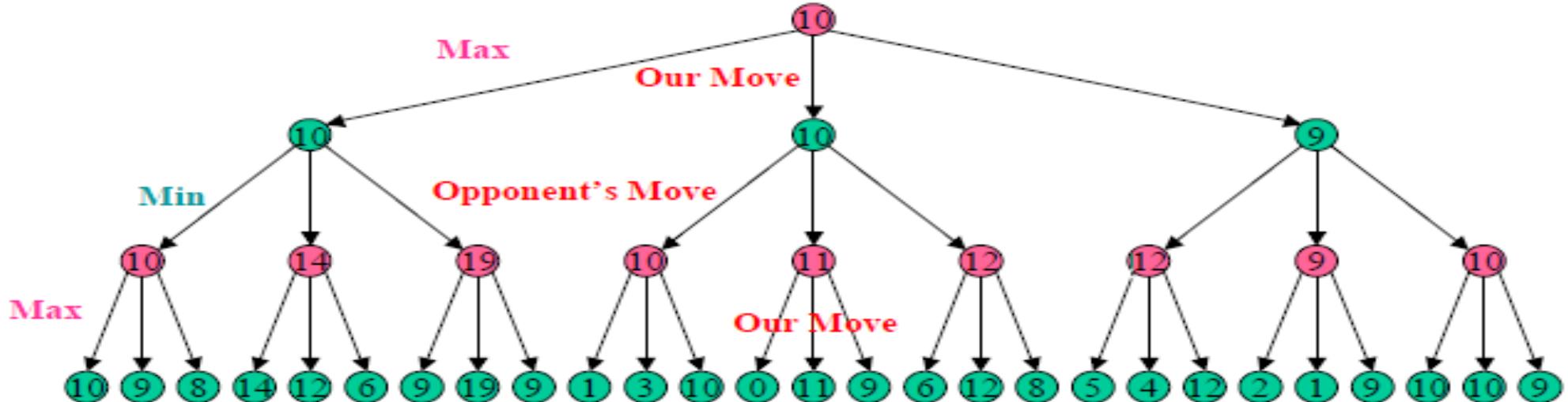
Properties of MiniMax

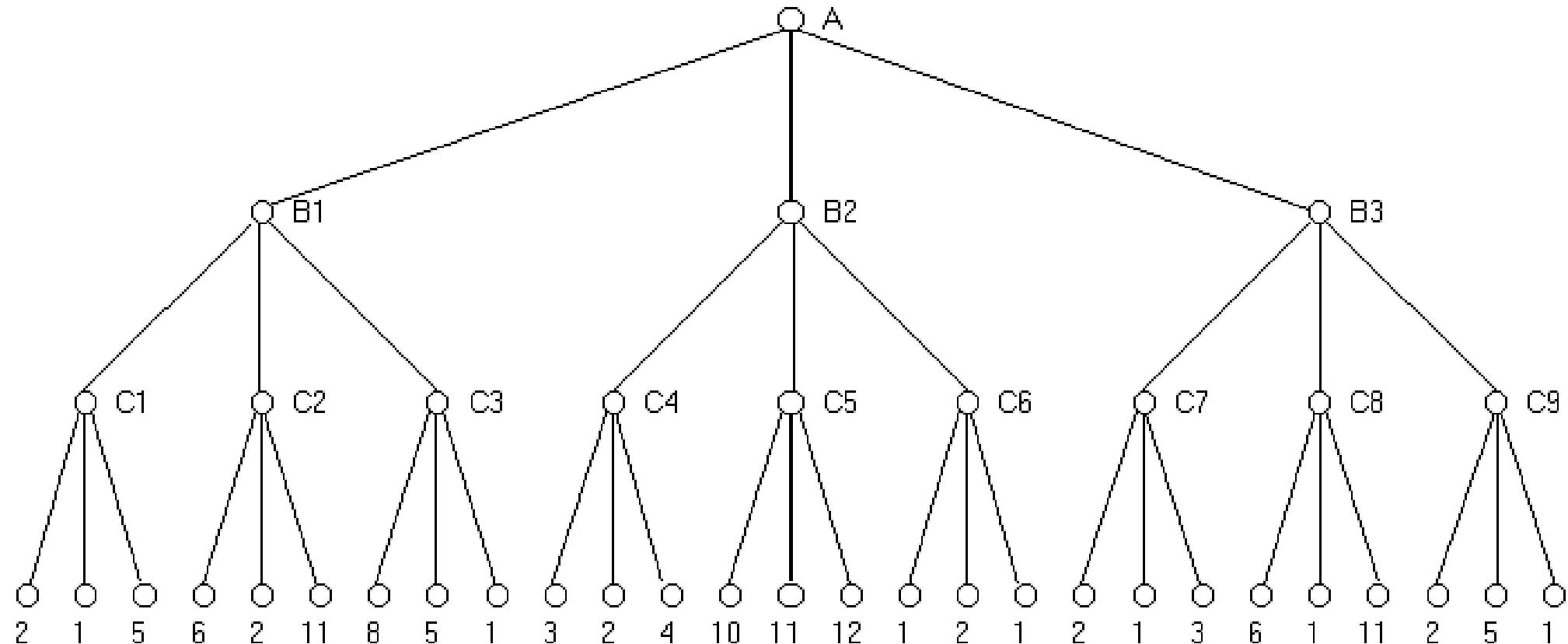
- *Complete?* Yes (if tree is finite)
- *Optimal?* Yes (against an optimal opponent)
- *Time complexity?* $O(b^m)$
- *Space complexity?* $O(b^m)$ (depth-first exploration)

We first consider games with two players; MAX and MIN. MAX moves first, and then they take turns moving until the game is over. Each level of the tree alternates, MAX is trying to maximize score, and MIN is trying to minimize MAX score in order to undermine success



We first consider games with two players; MAX and MIN. MAX moves first, and then they take turns moving until the game is over. Each level of the tree alternates, MAX is trying to maximize score, and MIN is trying to minimize MAX score in order to undermine success

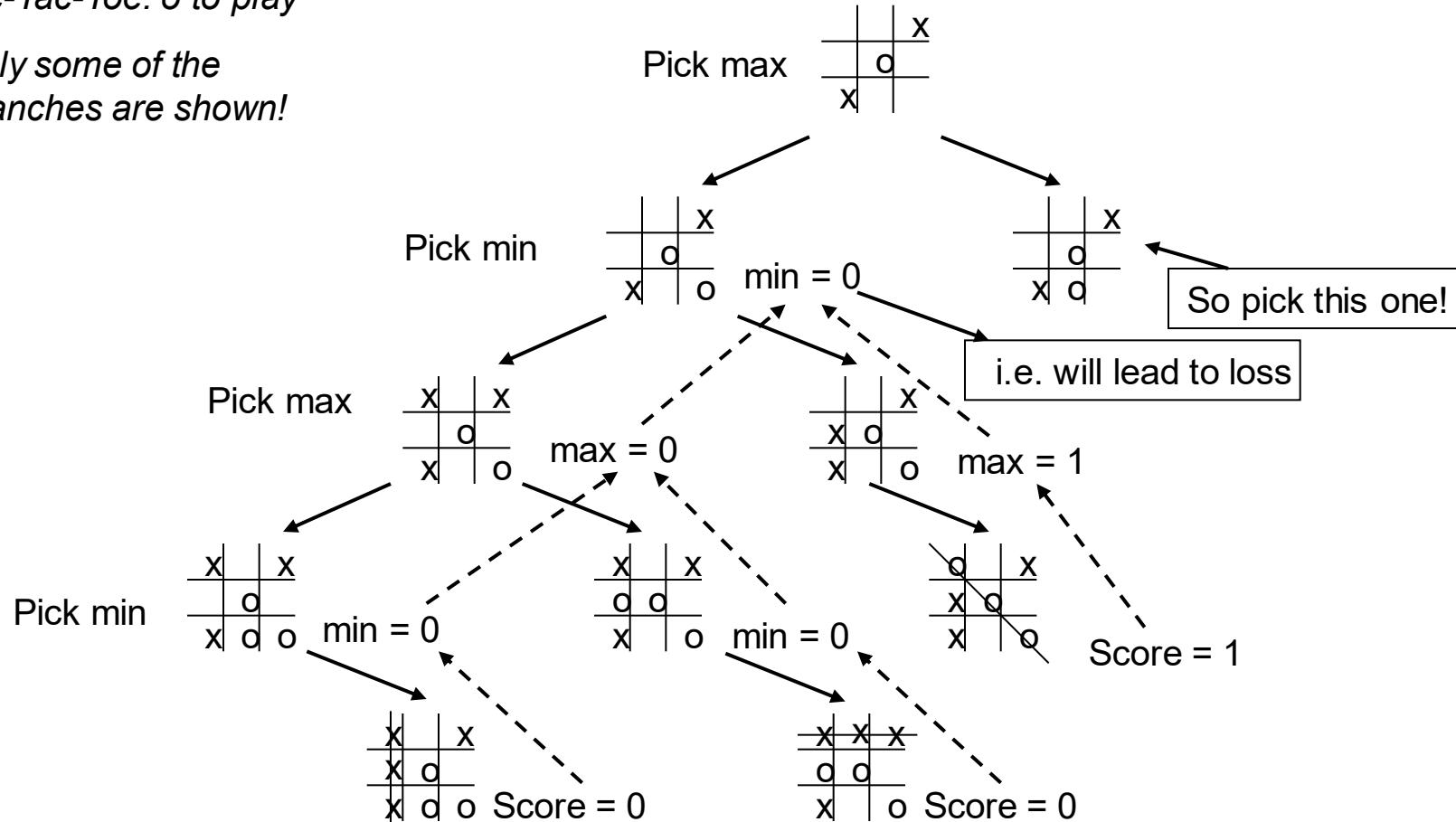




Example Max-Min Strategy

Tic-Tac-Toe: o to play

Only some of the
Branches are shown!



The mini max algorithm returns the best move for MAX under the assumption that MIN play optimally. What happens when MIN plays sub optimally ?

Optimality is still well defined, even if the opponent isn't playing well. Moreover, if the game tree is small enough that the agent can fully explore it, then the optimal player really doesn't care what the other one does.

Let's say Max goes first. What will Max do? He will look at every possible game sequence. He will then take the action which guarantees that he will get a score of X . No matter what Min does in subsequent moves, Min can never get a score less than X .

Alpha–beta pruning

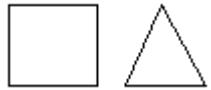
- Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the mini max algorithm in its search tree.
- It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, etc.)
- It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further.

β (A beta cutoff)

- the value of the best (i.e., highest-value) choice we have found so far at any choice point along the path for MAX.

α (An alpha cutoff)

- the value of the best (i.e., lowest-value) choice we have found so far at any choice point along the path for MIN.

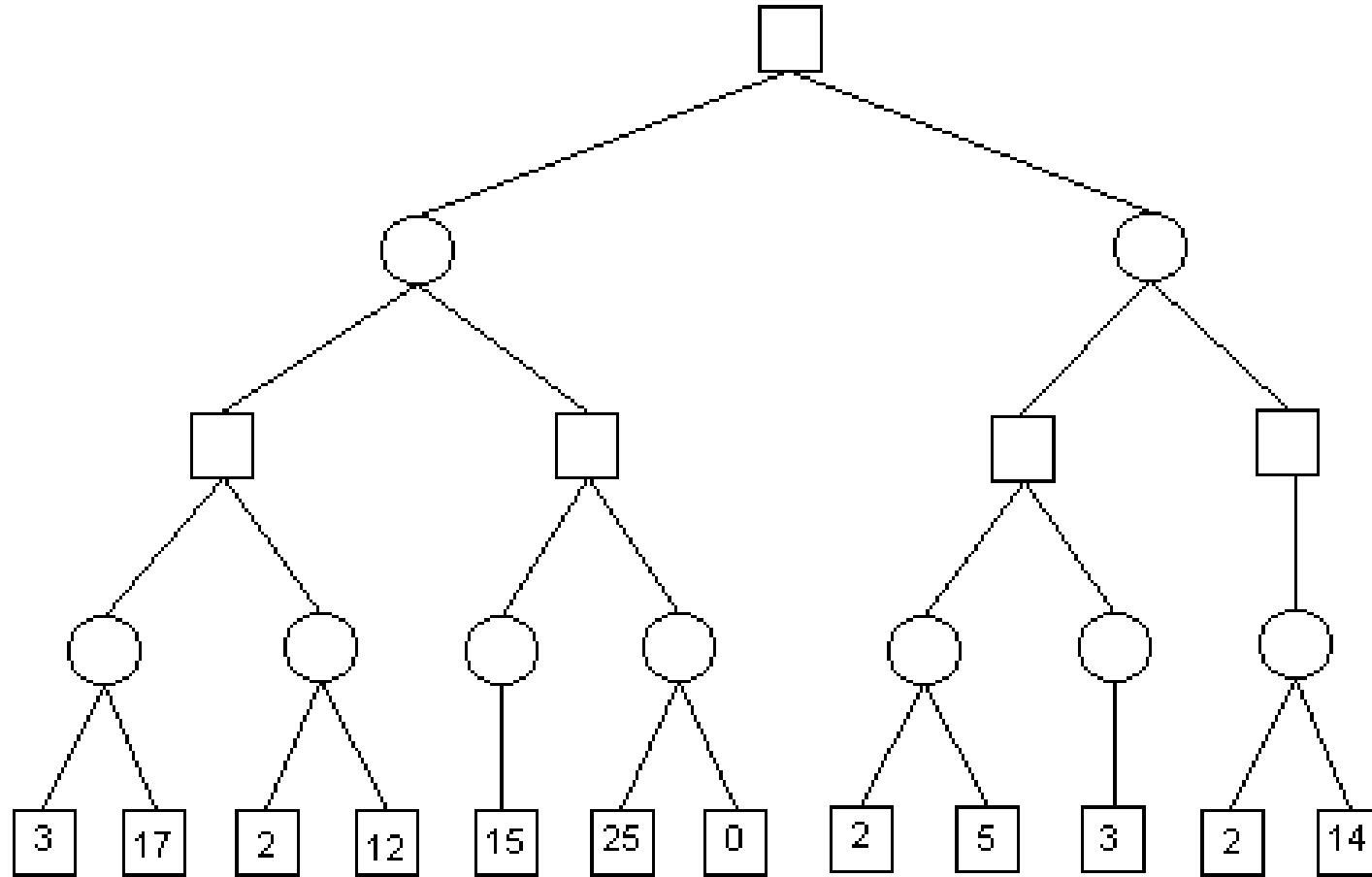


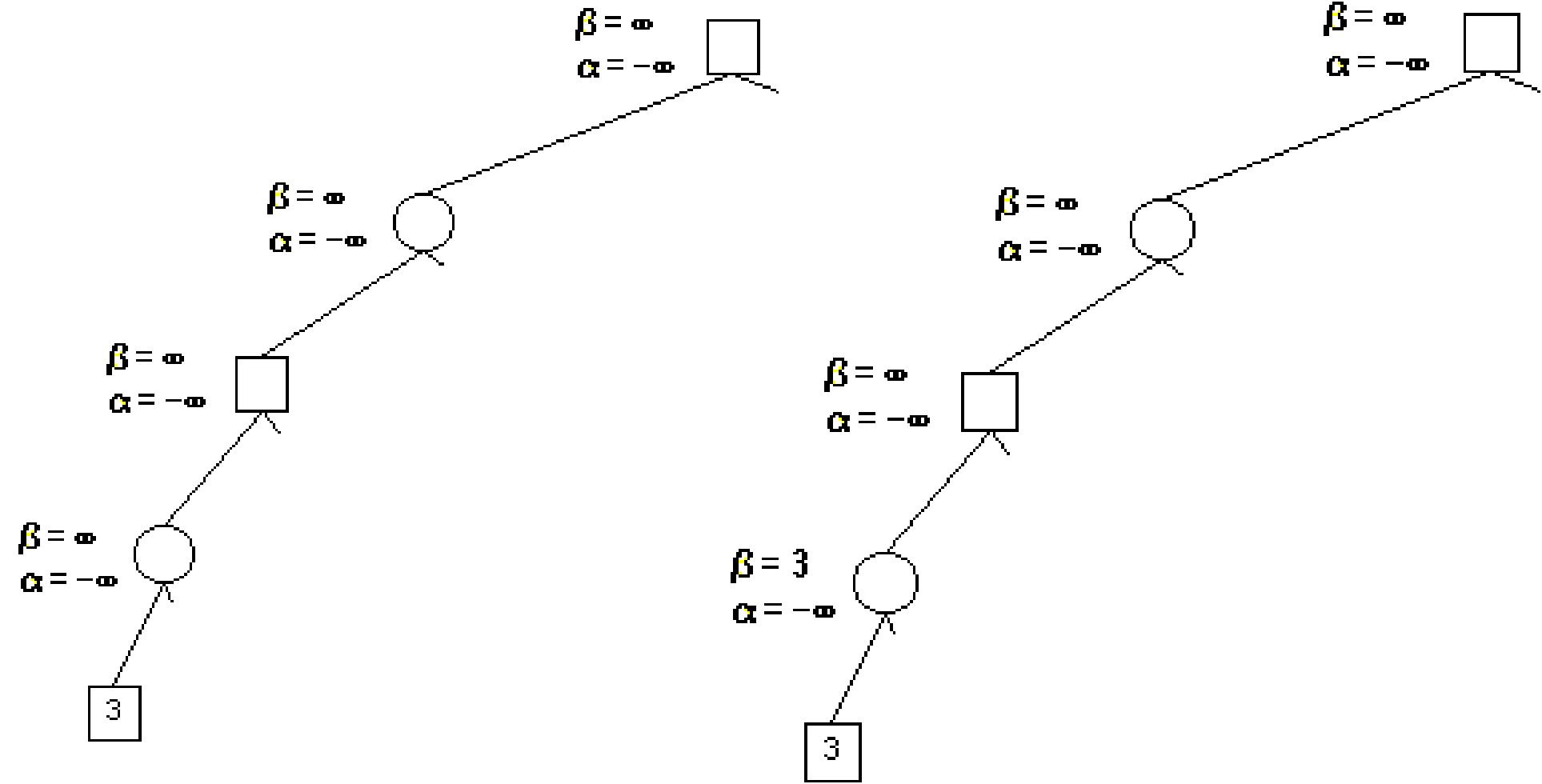
These are also called **MAX nodes**. The goal at a MAX node is to maximize the value of the subtree rooted at that node. To do this, a MAX node chooses the child with the greatest value, and that becomes the value of the MAX node

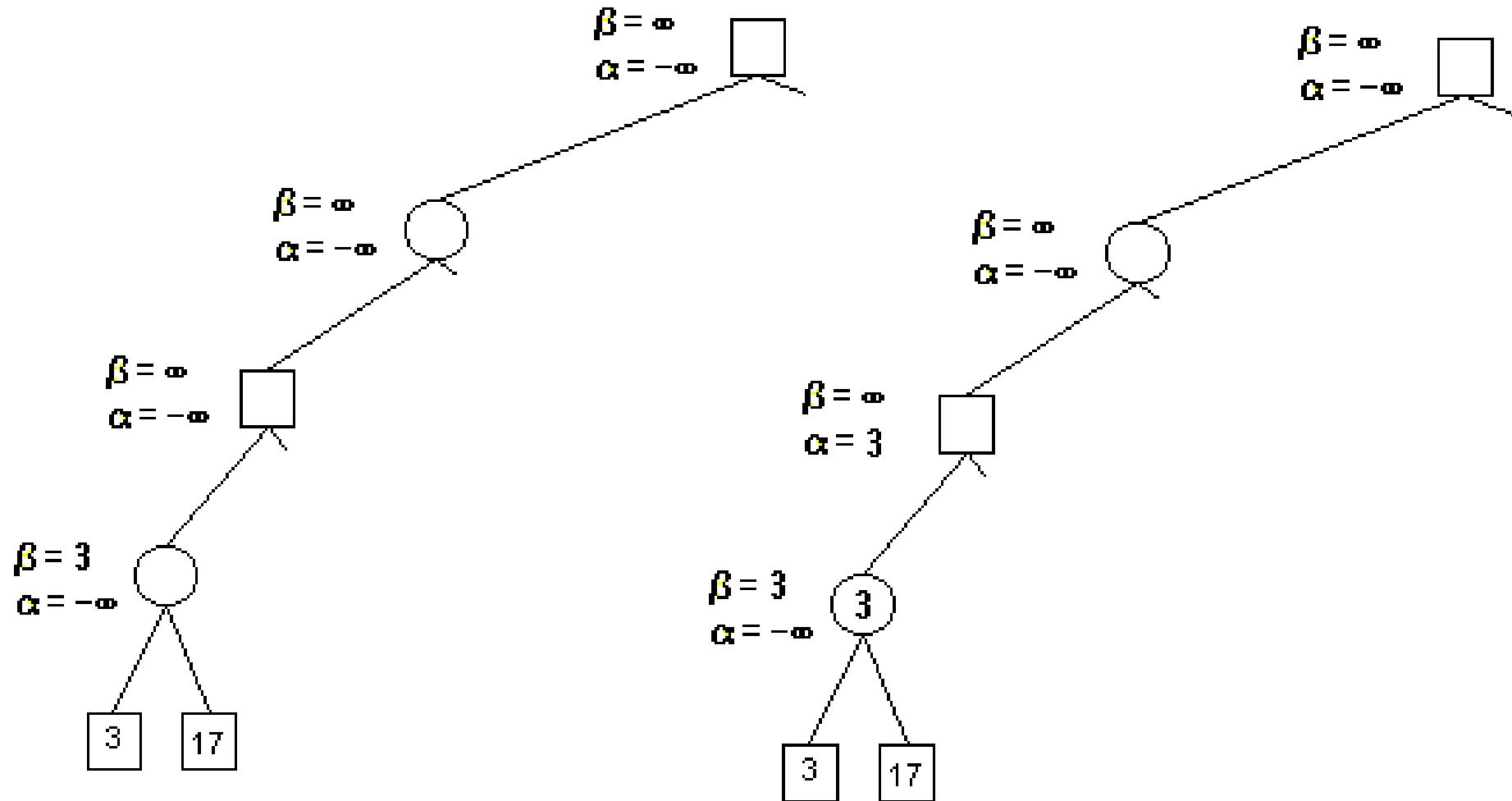


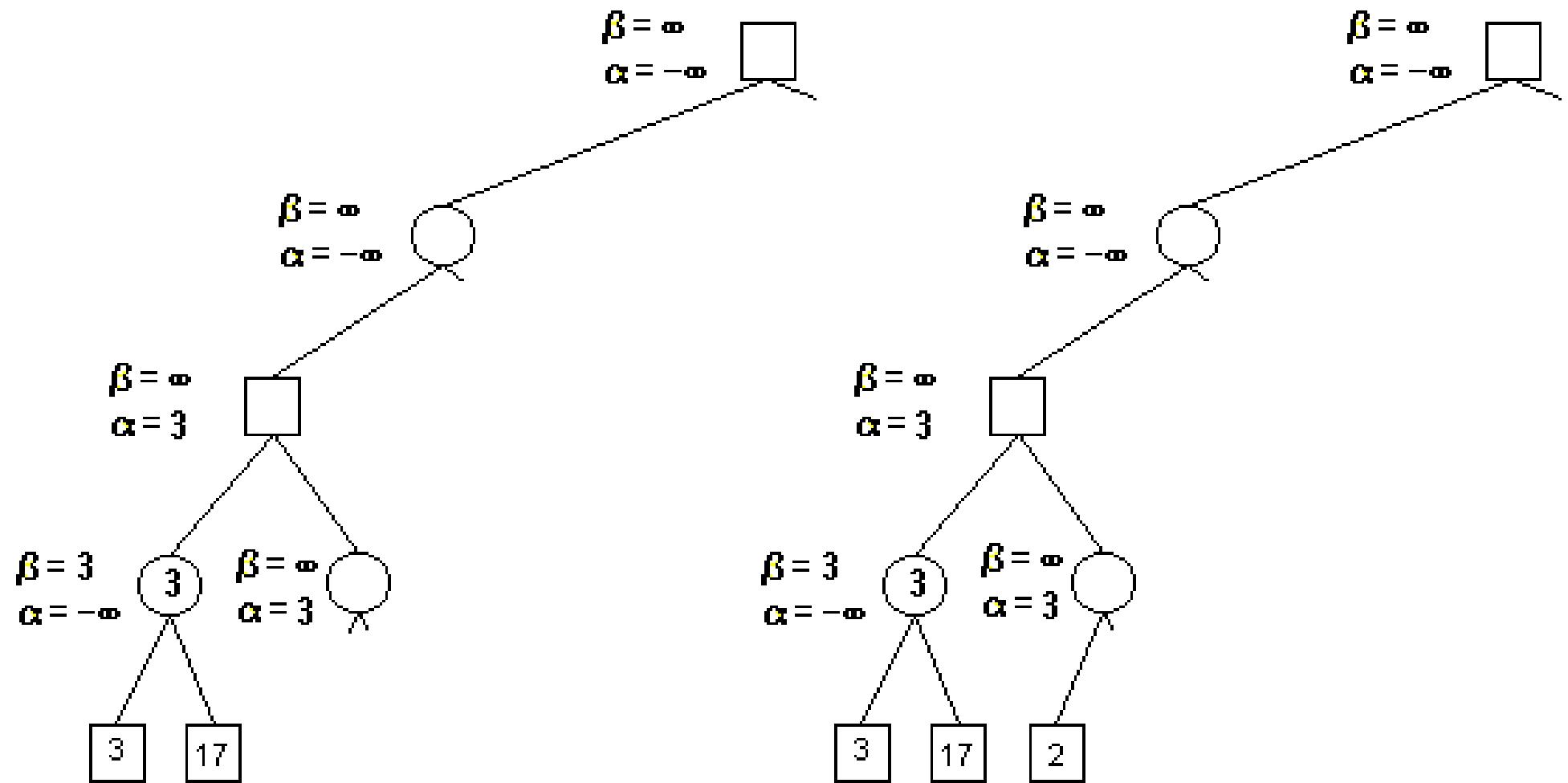
These are also called **MIN nodes**. The goal at a MIN node is to minimize the value of the subtree rooted at that node. To do this, a MIN node chooses the child with the least (smallest) value, and that becomes the value of the MIN node

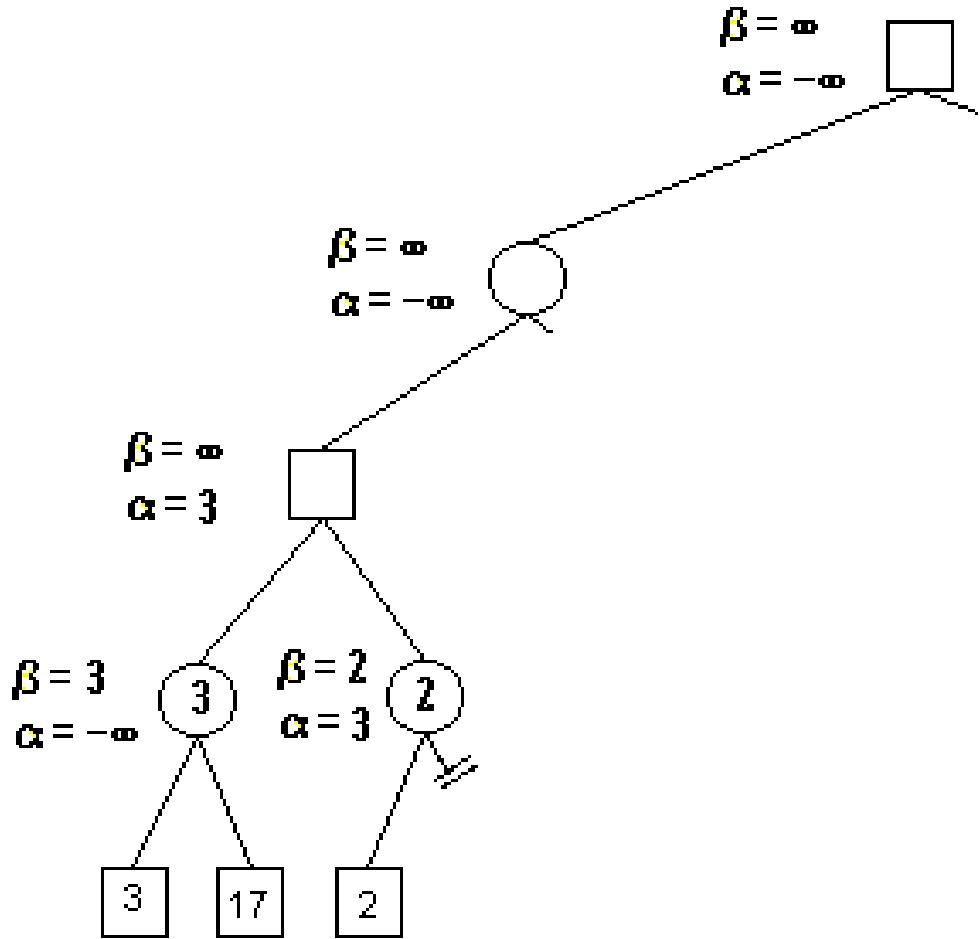
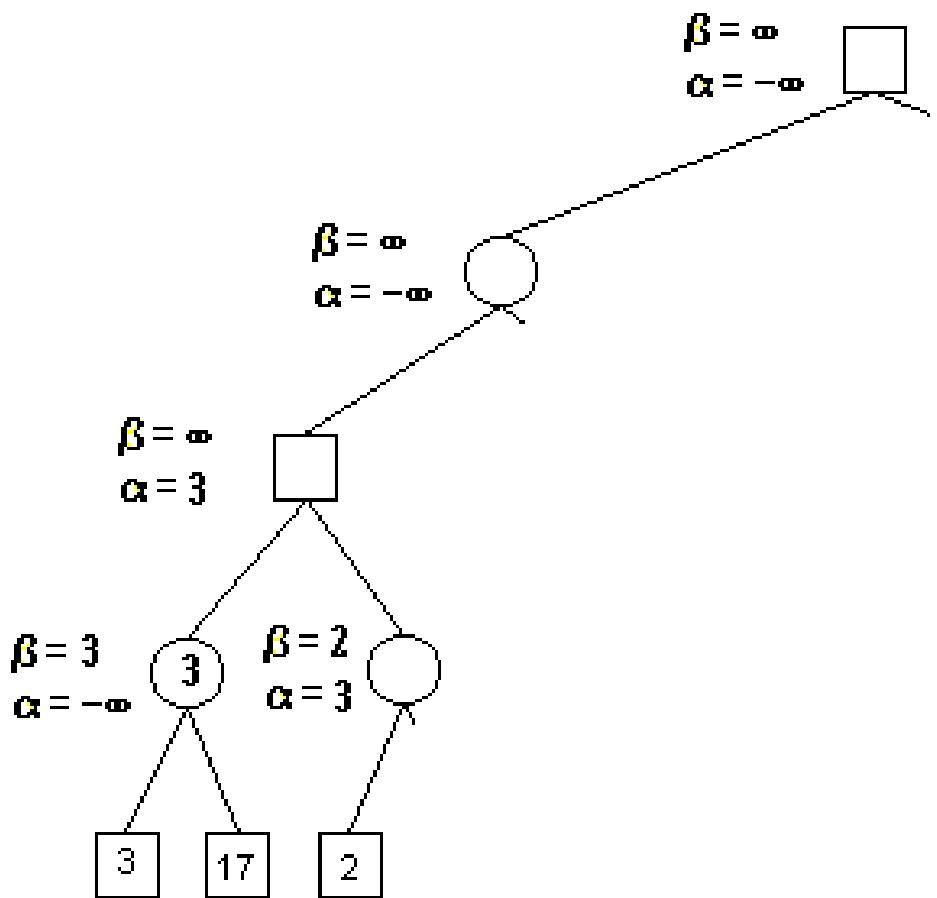
Demonstrate alpha-beta pruning

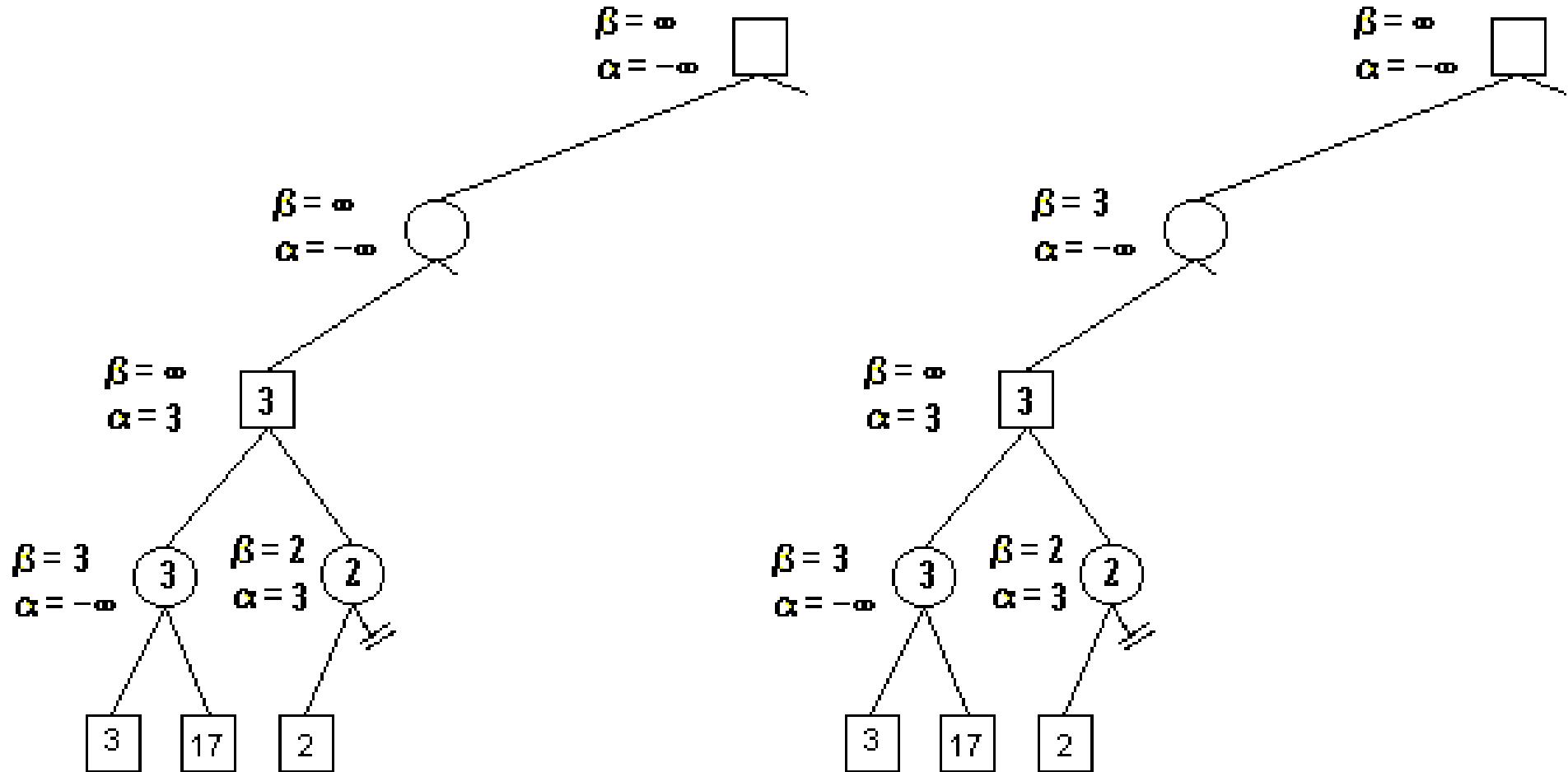


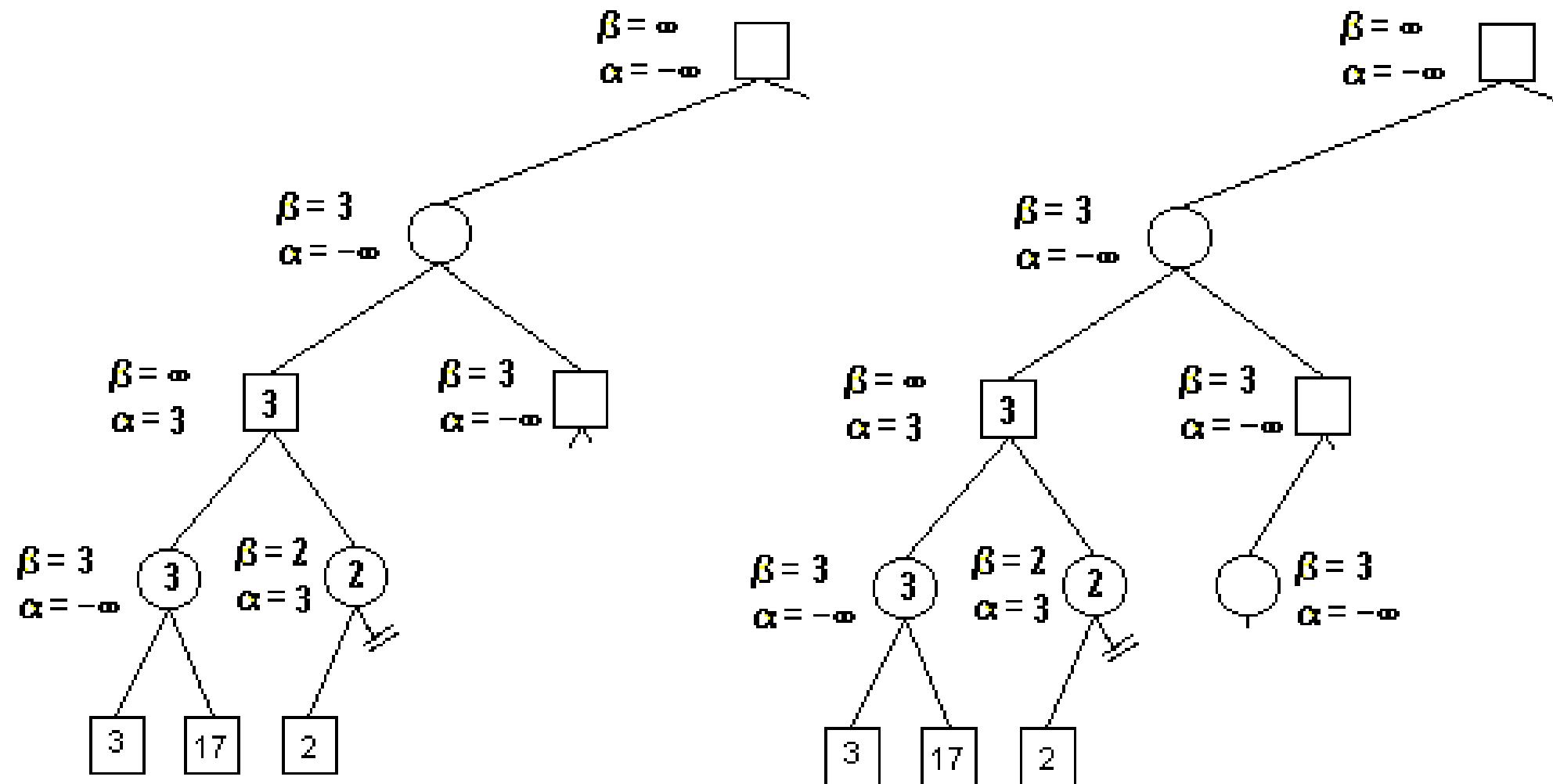


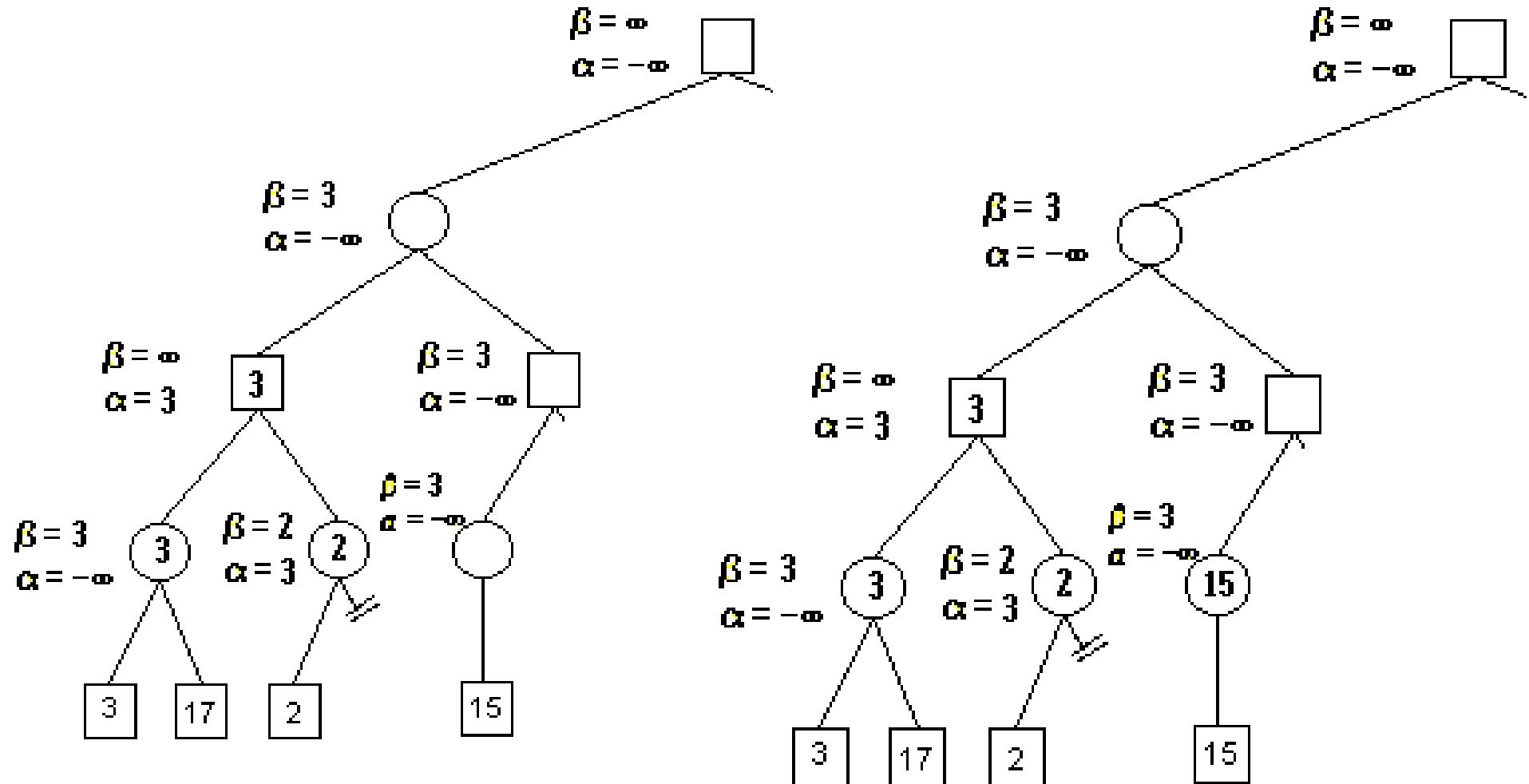


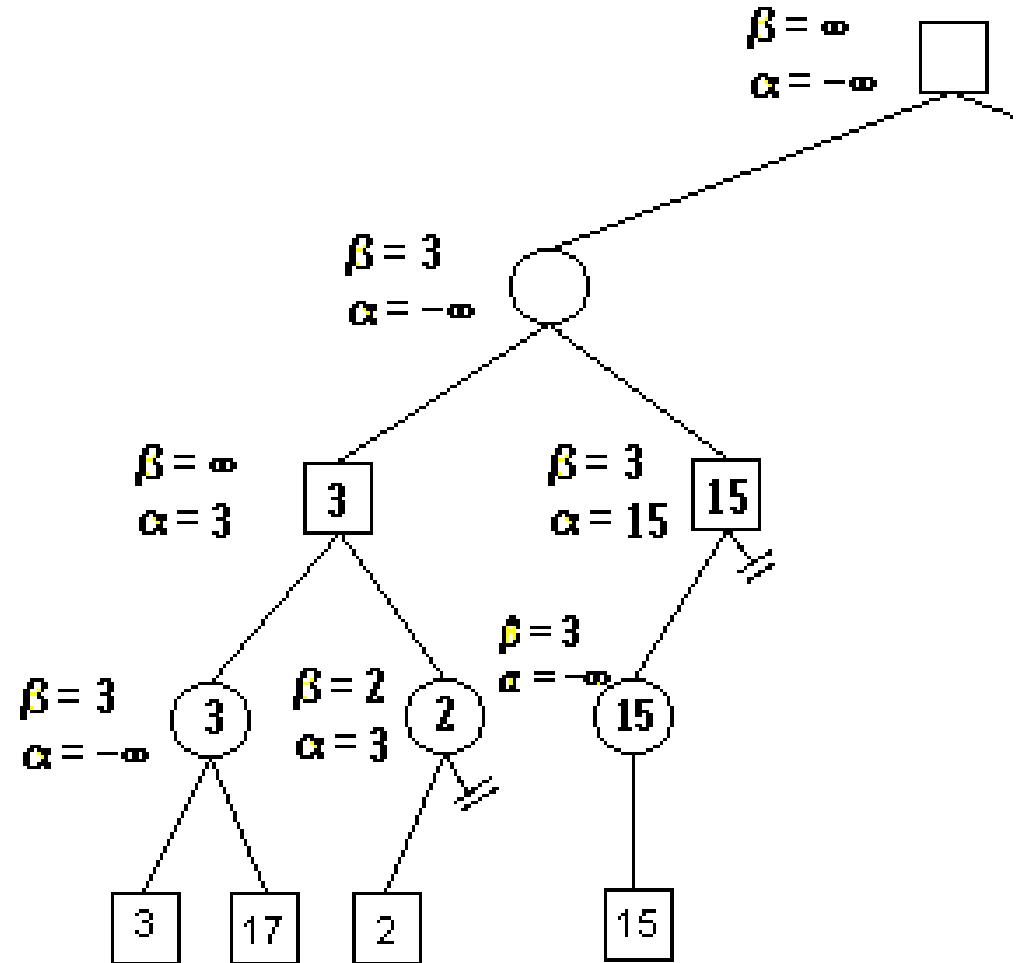
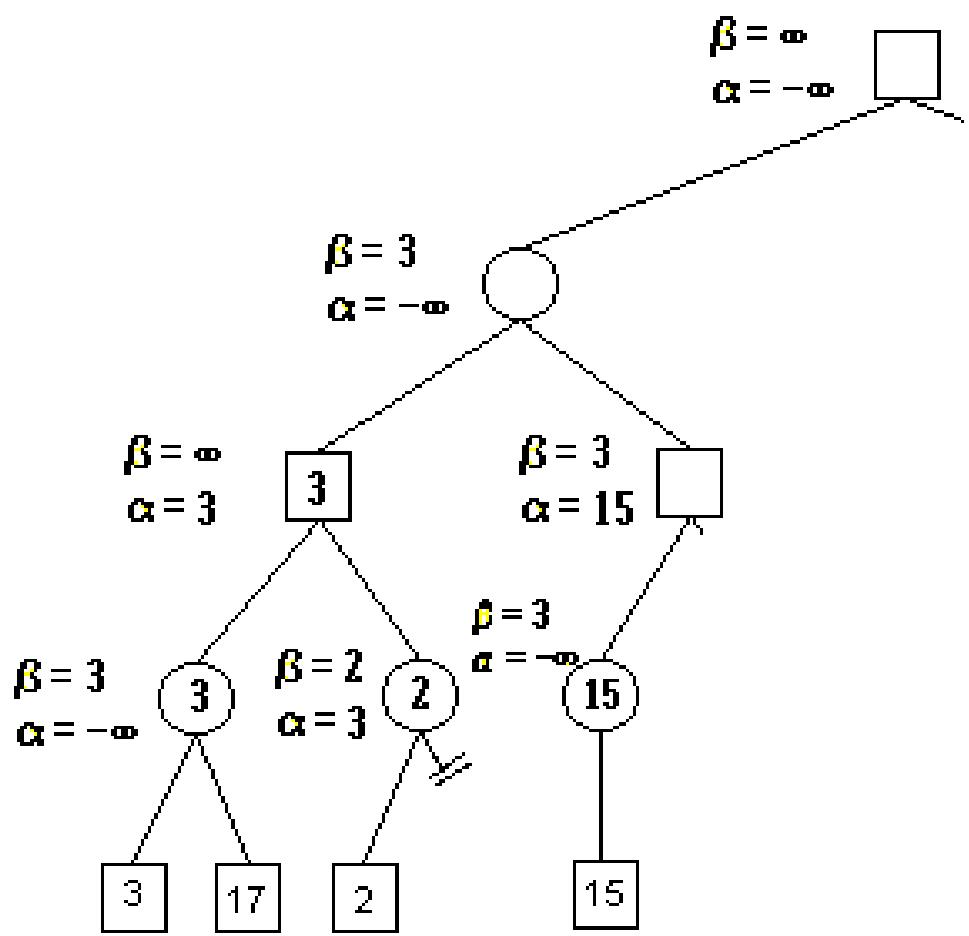


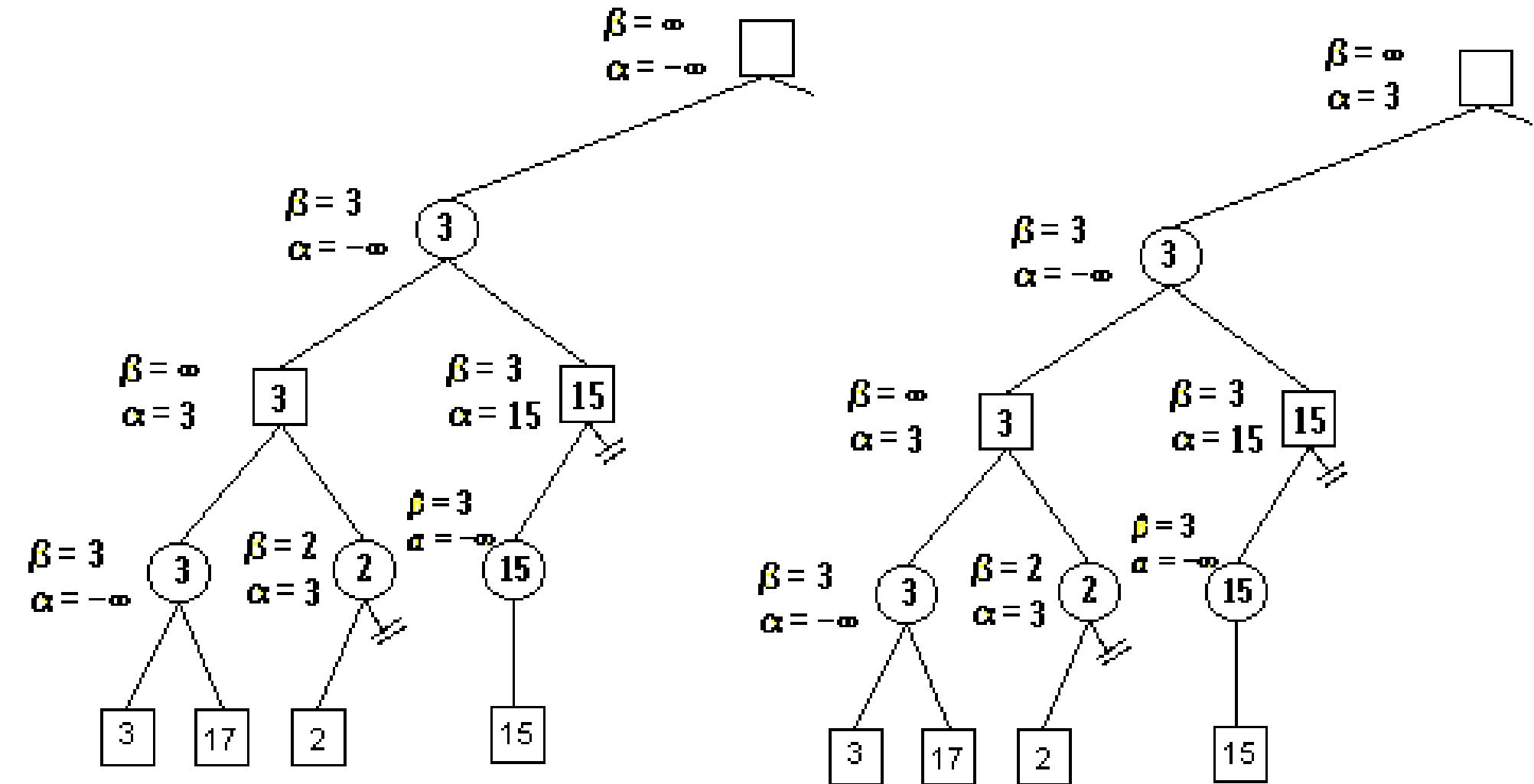


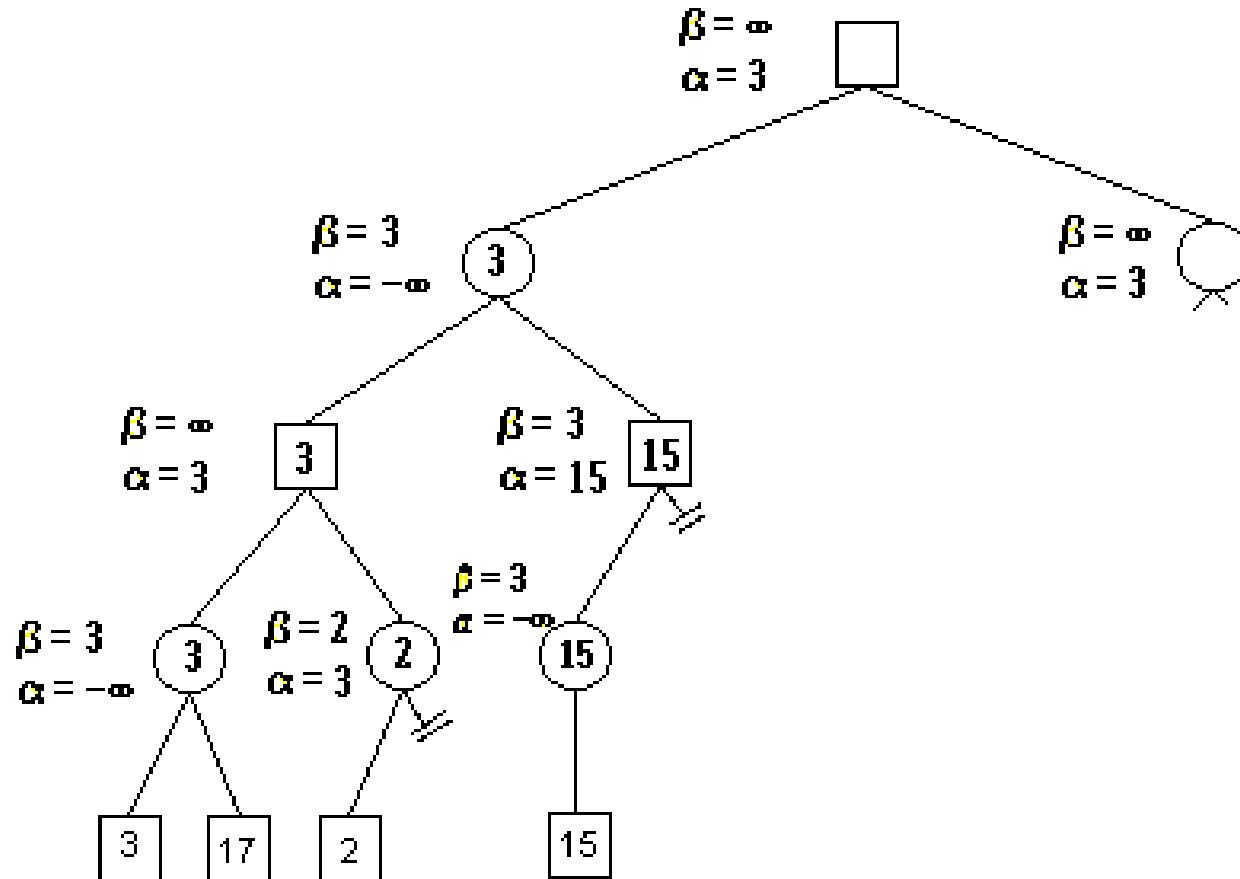


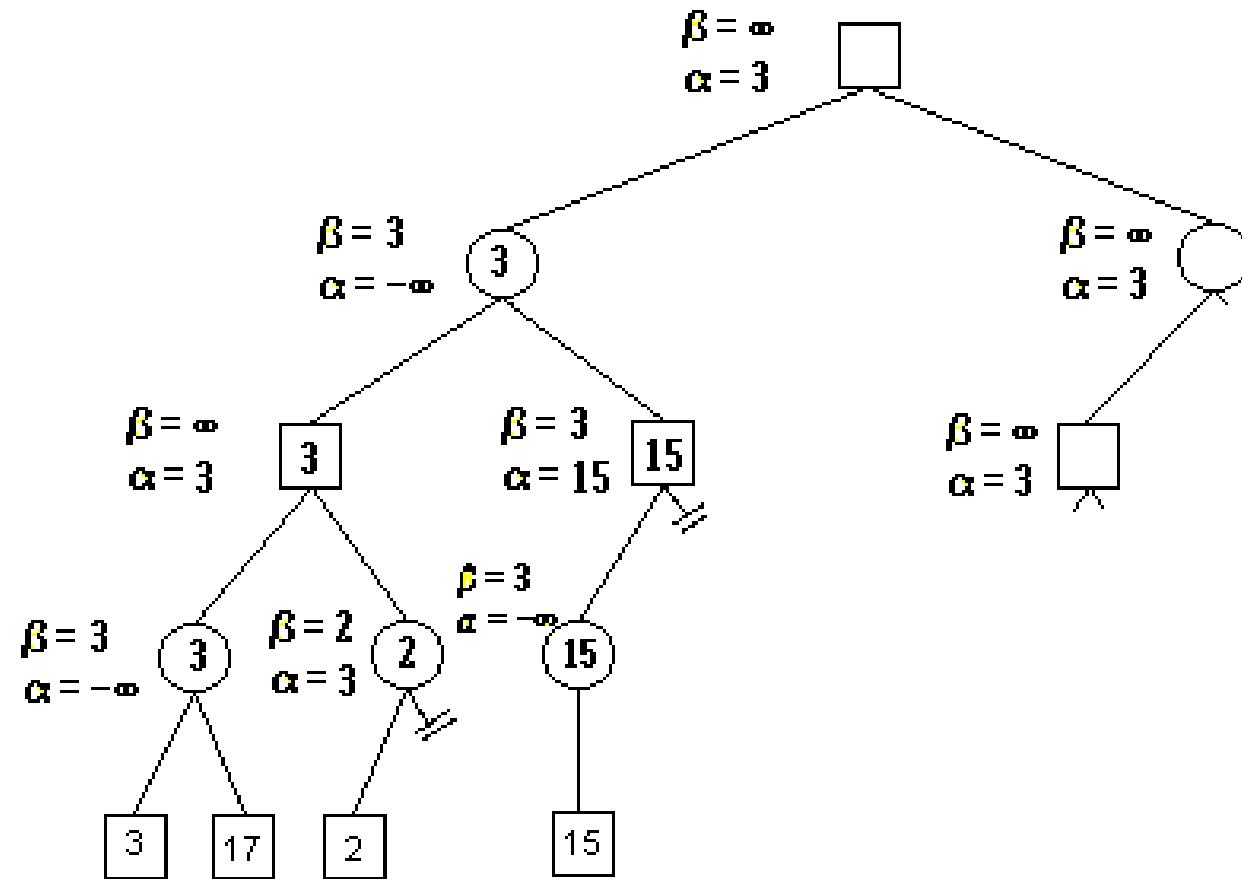


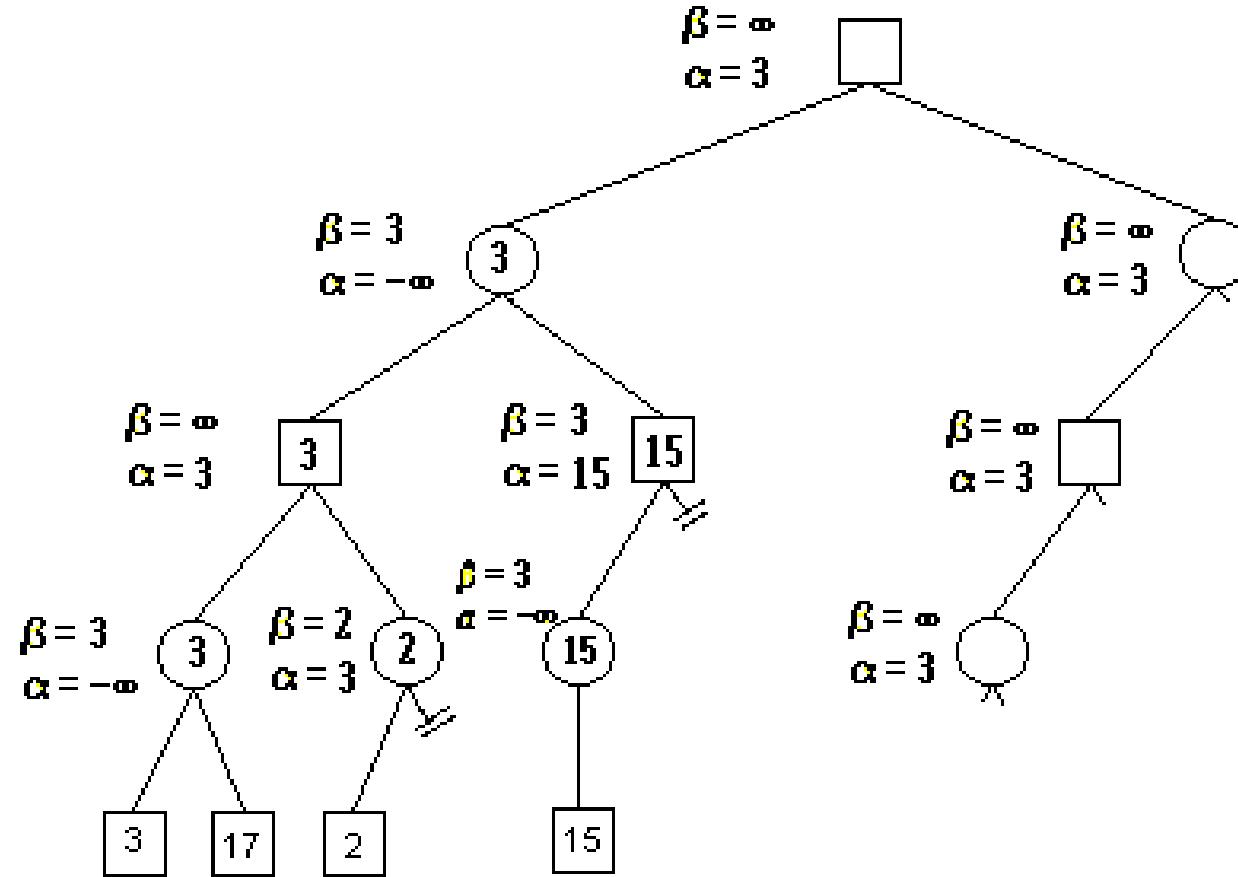


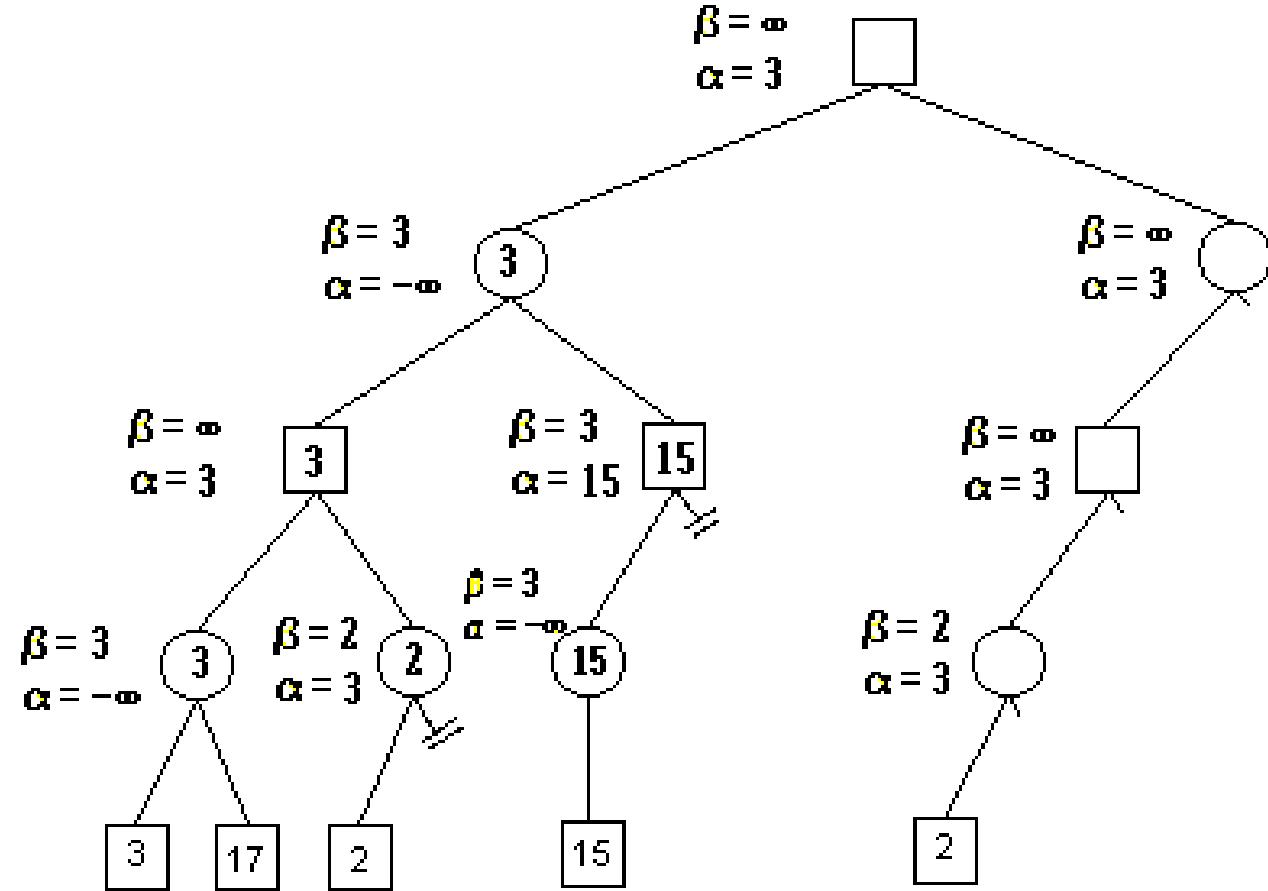


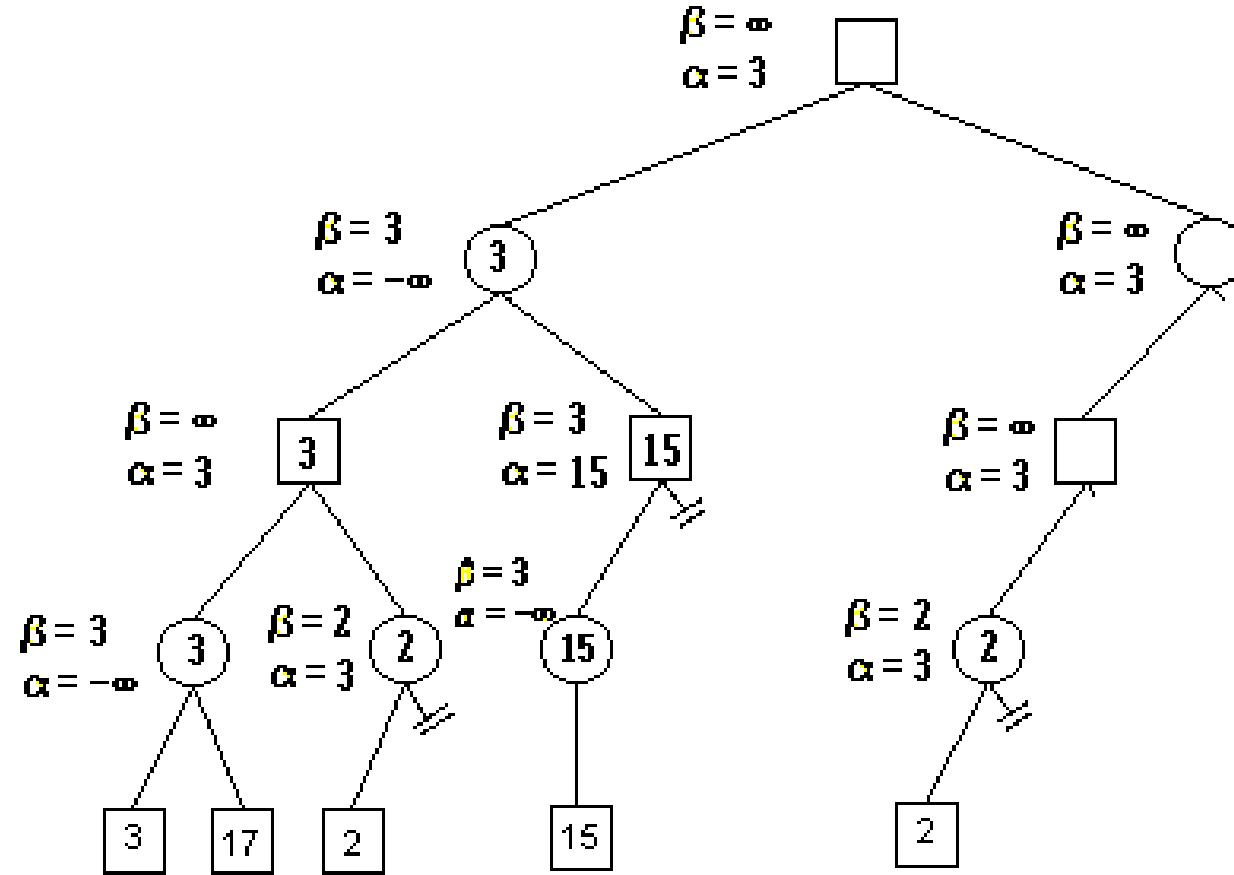


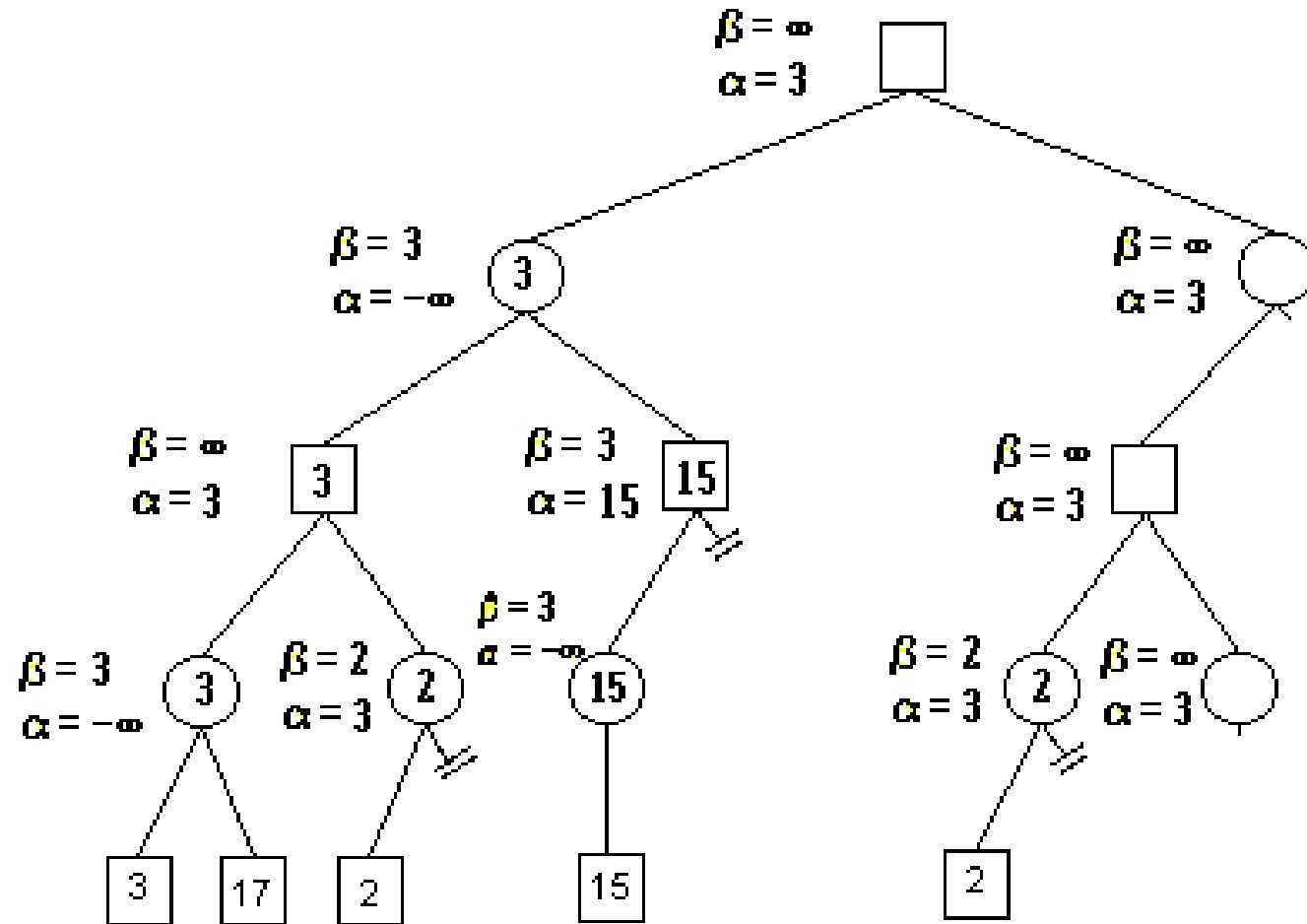


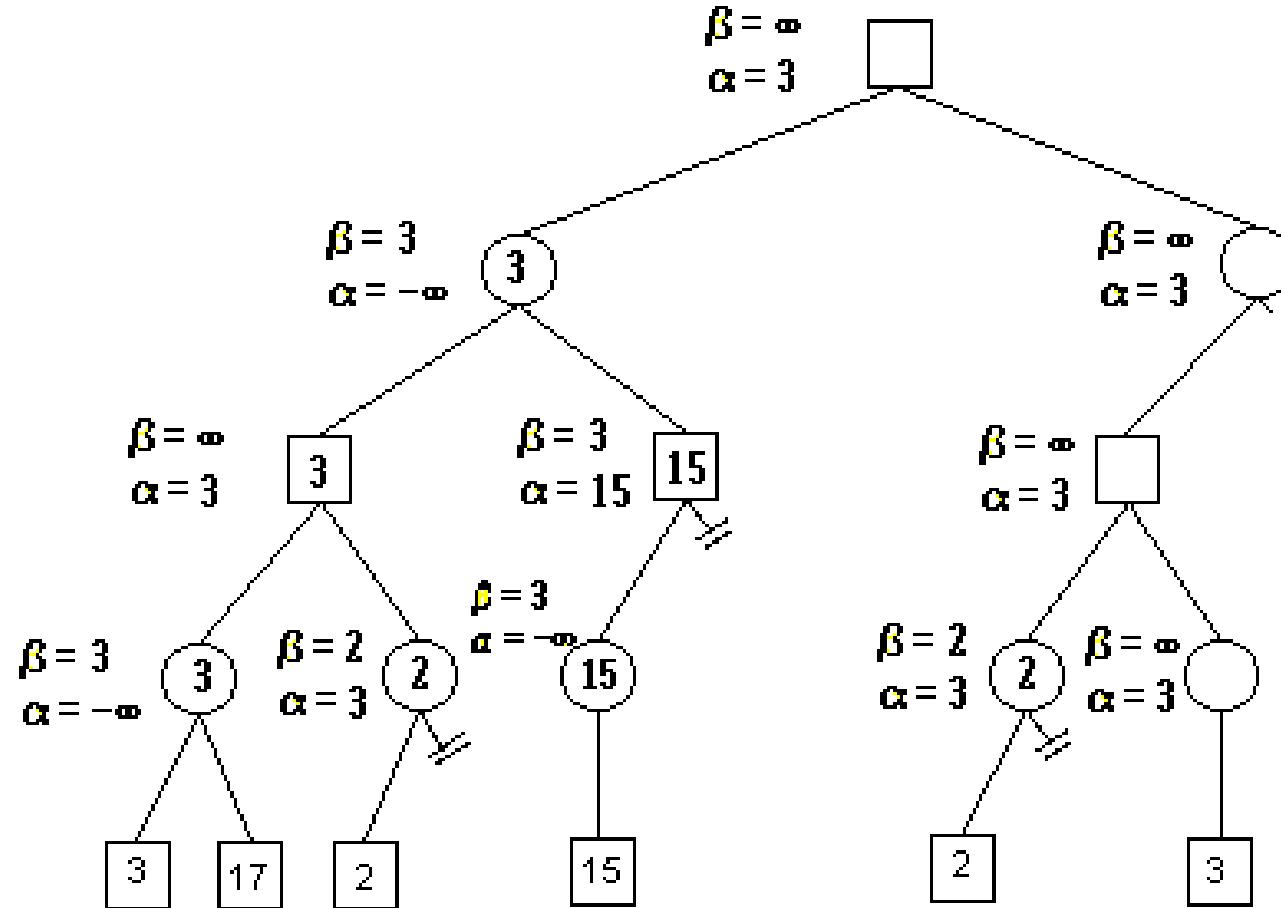


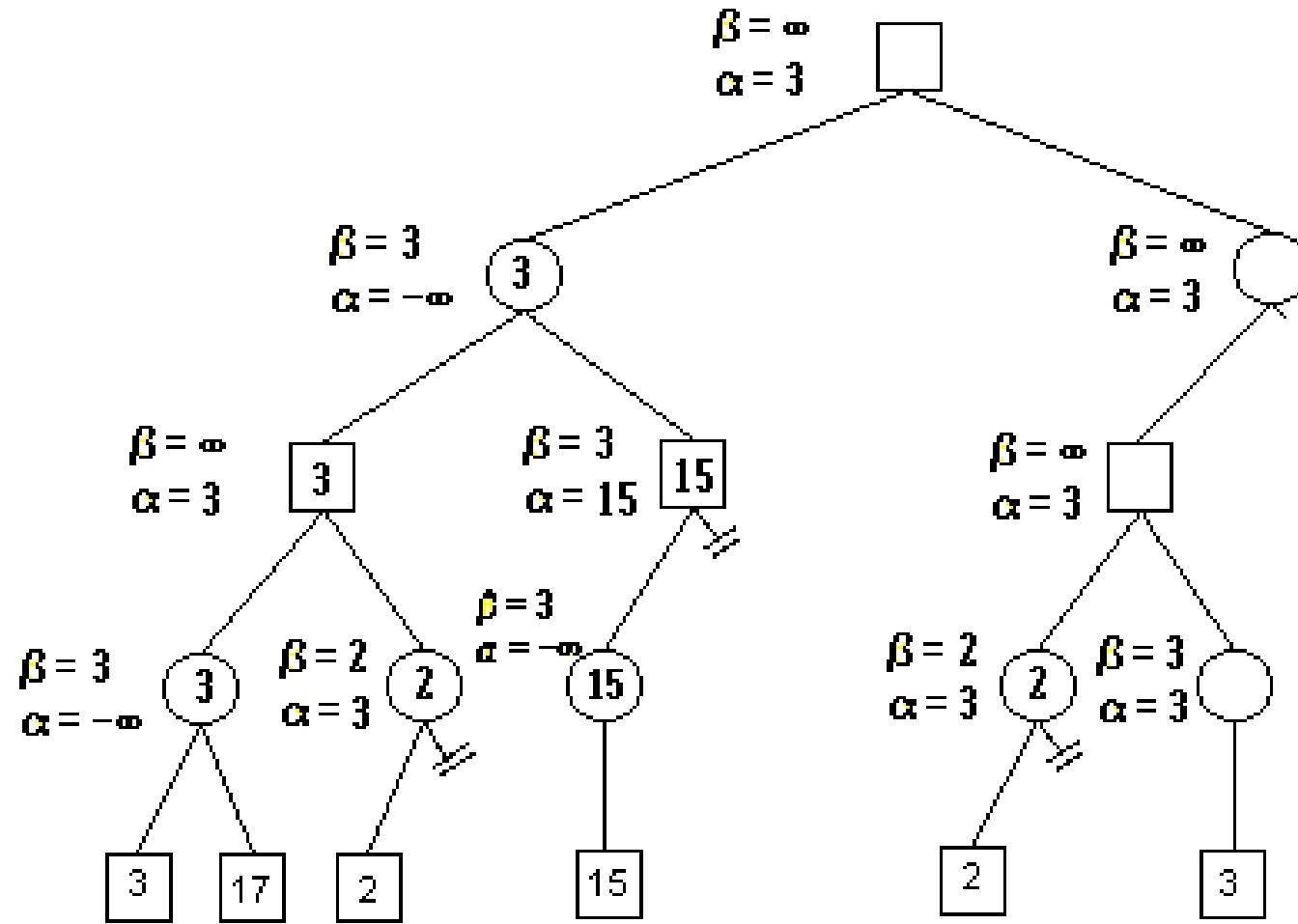


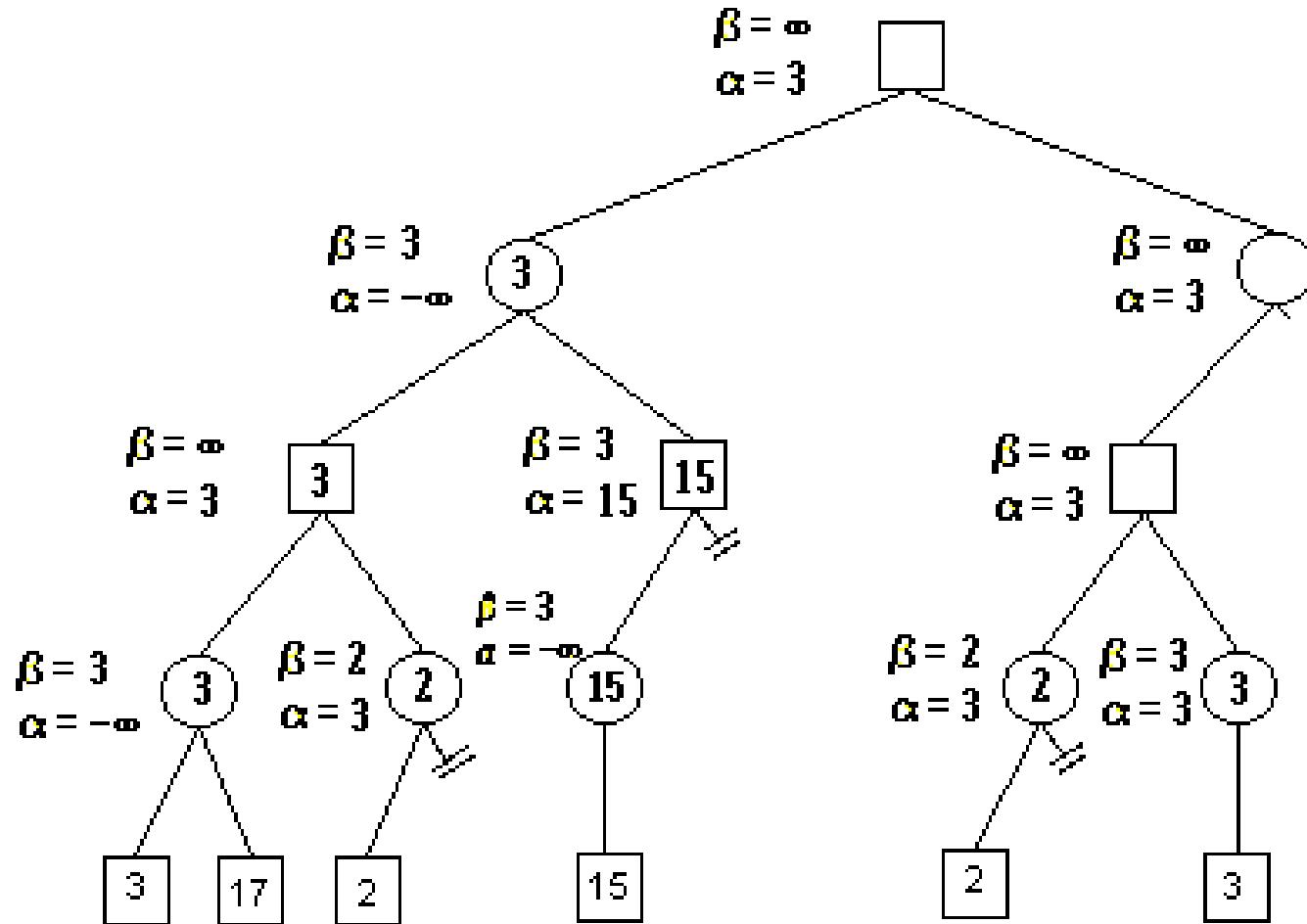


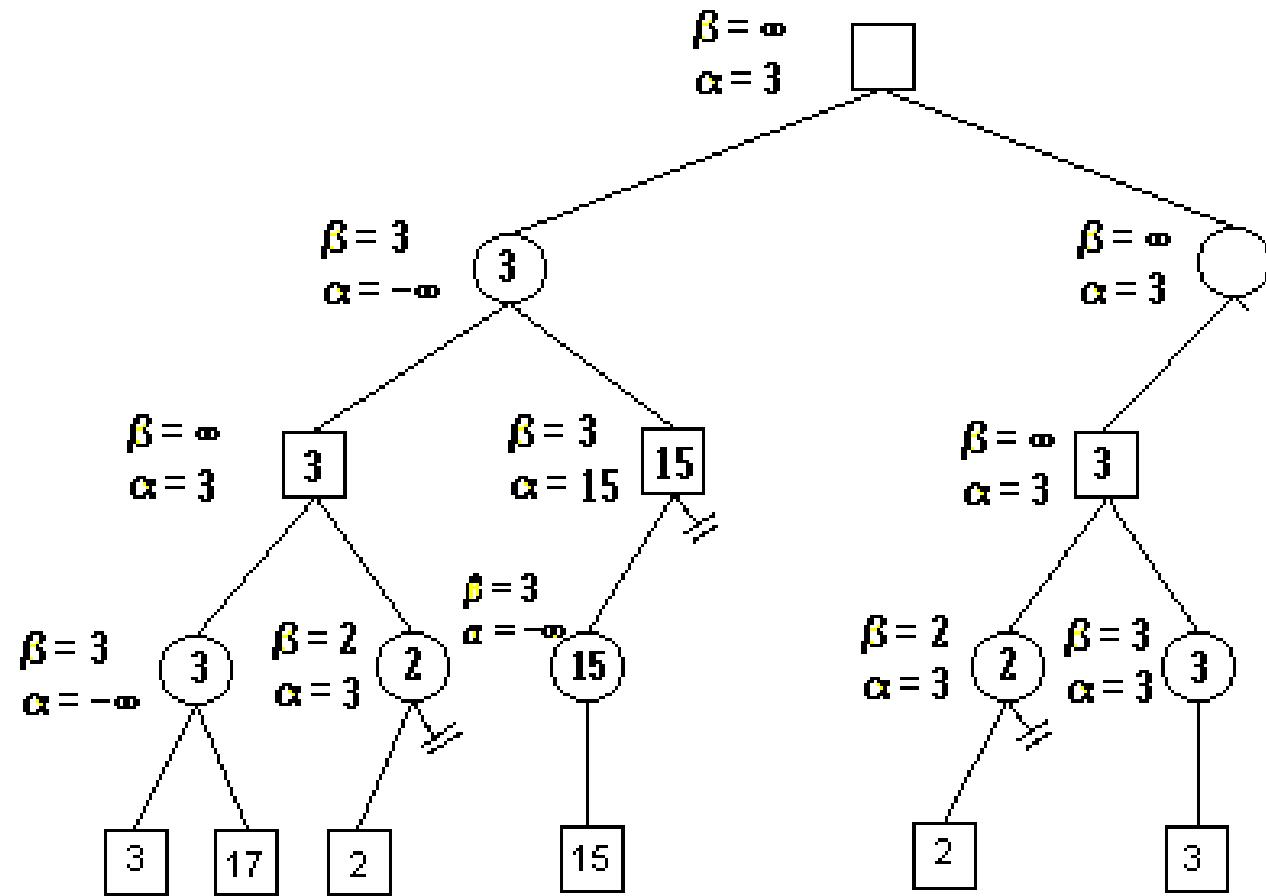


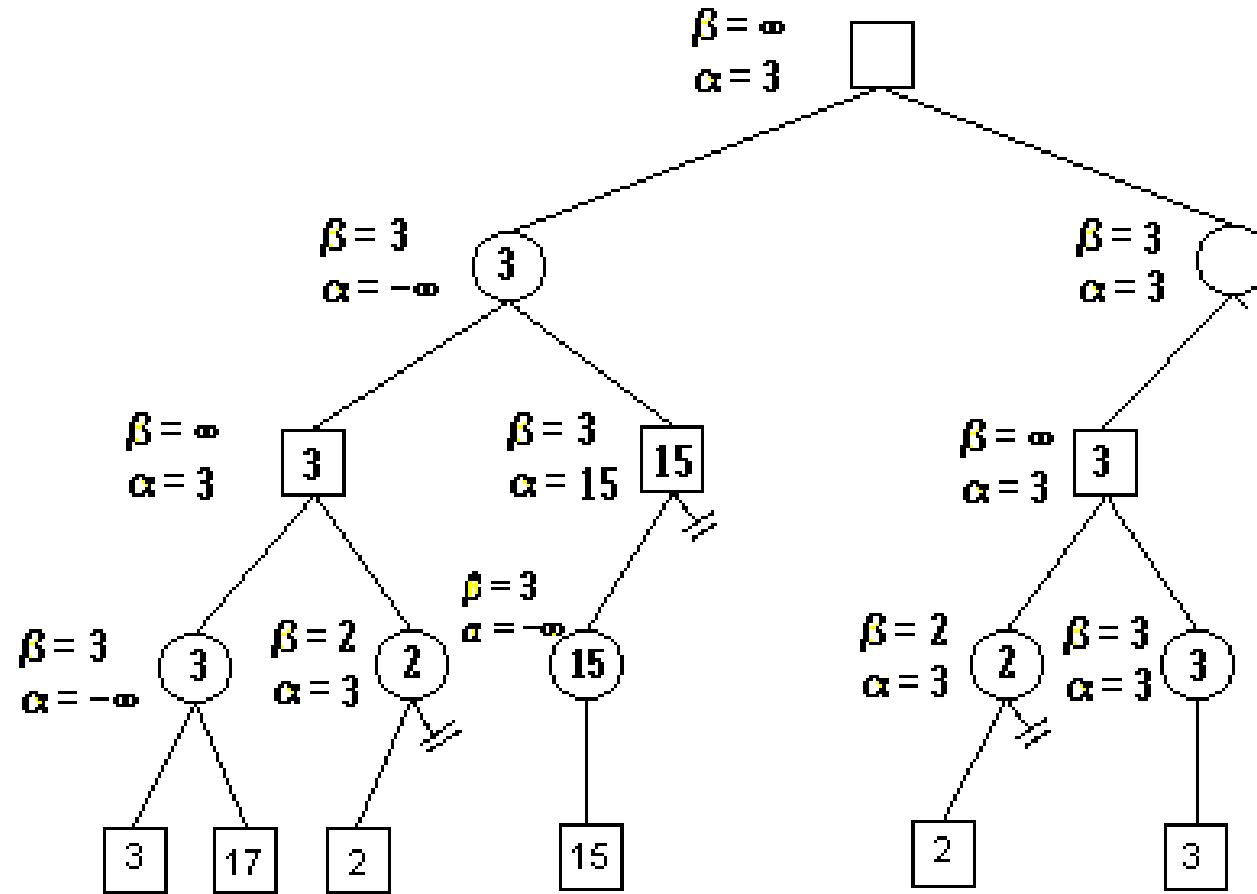










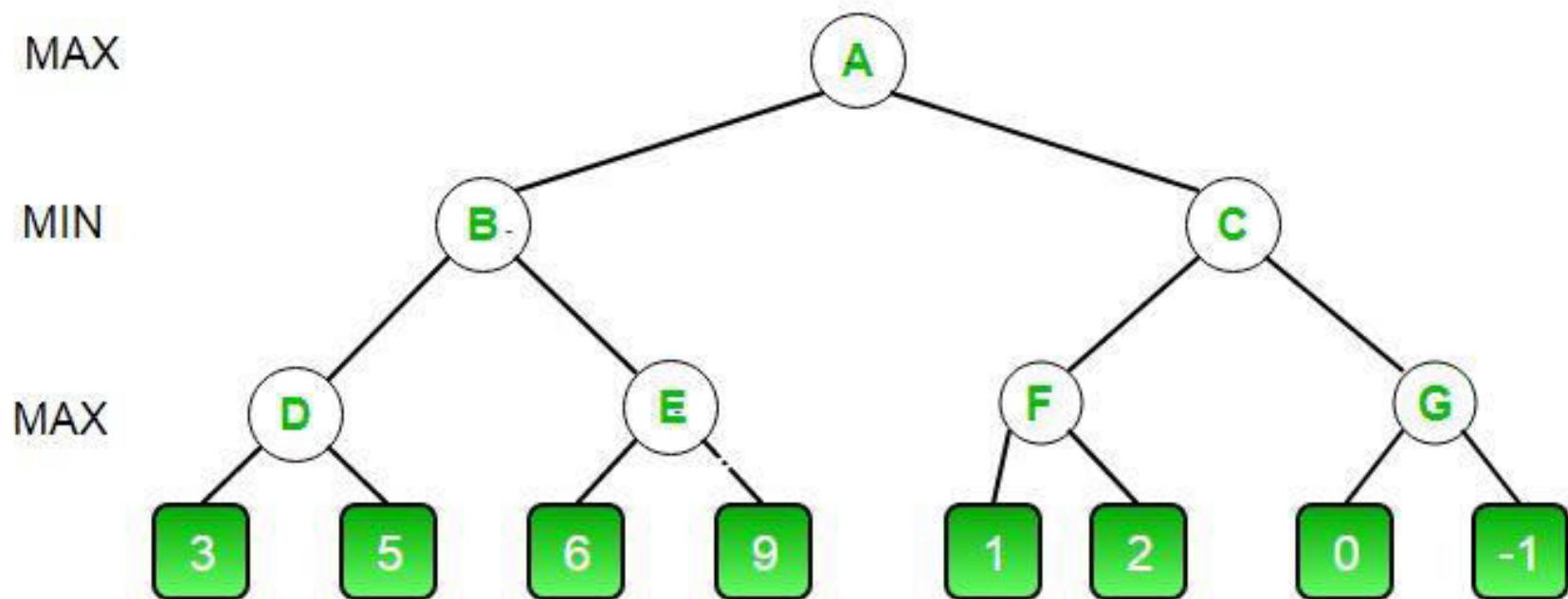


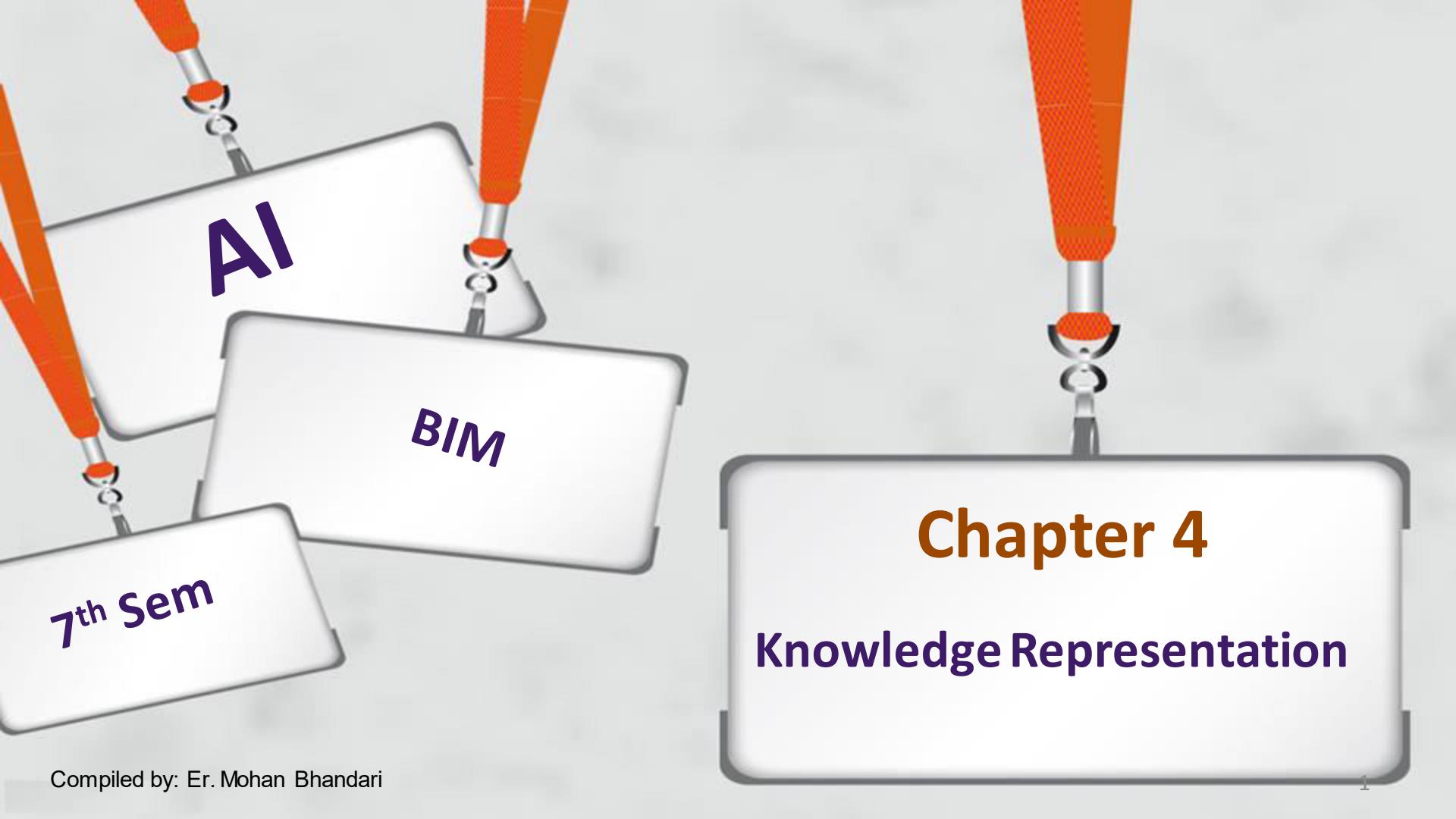
Q. What is the difference between MiniMax and AlphaBeta Prunning?

Mini max := Search game space

Alpha beta := Improve mini max by eliminating the nodes

Perform Min Max and Alphabet





Chapter 4

Knowledge Representation

Axiom

An **axiom** is a sentence or proposition that is not proved or demonstrated and is considered as self-evident or as an initial necessary consensus for a theory building or acceptance. According as requirements, the new sentences are added to the knowledge base and then new sentences are also derived from old axiom & theorems, called **inference**.

Logic is method of reasoning process in which conclusions are drawn from premises using rules of inference. The logic is knowledge representation technique that involves:

- **Syntax:** defines well-formed sentences or legal expression in the language
- **Semantics:** defines the "meaning" of sentences
- **Inference rules:** for manipulating sentences in the language

Basically, the logic can be classified as:

- Proposition (or statements or calculus) logic
- Predicate [or First Order Predicate Logic (FOPL)] logic

Propositional Logic

A proposition is a declarative sentence to which only one of the “Truth value” (i.e. TRUE or FALSE) can be assigned (but not both). Hence, the propositional logic is also called Boolean logic. When a proposition is true, we say that its truth value is T, otherwise its truth value is F.

For example:

- The square of 4 is 16 →T
- The square of 5 is 27 →F

Atomic Sentences(Simple)

The atomic sentences consist of a single proposition symbol.

Each such symbol stands for a proposition that can be true or false.

We use symbols that start with an uppercase letter and may contain other letters or subscripts, for example:
p, q, r, s etc.

For example:

p = Sun rises in West. (False sentence)

Complex Sentences (molecular or combined or compound)

The two or more statements connected together with some logical connectives such as AND (\wedge), OR (\vee), Implication (\rightarrow), etc.

Name	Representation	Meaning
Negation	$\neg p$	not p
Conjunction (true when both statement are true, otherwise false)	$p \wedge q$	p and q
Disjunction (false when both statement are false, otherwise true)	$p \vee q$	p or q (or both)
Exclusive Or (false when both statement are same)	$P \oplus q$	either p or q, but not both
Implication (false when p is true and q is false)	$p \rightarrow q$	if p then q
Bi-conditional or Bi-implication (true when both statement have same truth value)	$p \leftrightarrow q$	p if and only if q
The order of precedence in propositional logic is (from highest to lowest): Inverse, AND, OR, Implication and Double Implication.		

Converse:	If $p \rightarrow q$ is an implication, then its converse is $q \rightarrow p$
Inverse:	If $p \rightarrow q$ is an implication, then its inverse is $\neg p \rightarrow \neg q$
Contrapositive:	If $p \rightarrow q$ is an implication, then its contrapositive is $\neg q \rightarrow \neg p$

Q. Verify that $p \leftrightarrow q$ is equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$

p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$	$p \leftrightarrow q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

Q. Construct the truth table of $\neg(p \wedge q) \vee (r \wedge \neg p)$

p	q	r	$p \wedge q$	$\neg(p \wedge q)$	$\neg p$	$r \wedge \neg p$	$\neg(p \wedge q) \vee (r \wedge \neg p)$
T	T	T	T	F	F	F	F
T	T	F	T	F	F	F	F
T	F	T	F	T	F	F	T
T	F	F	F	T	F	F	T
F	T	T	F	T	T	T	T
F	T	F	F	T	T	F	T
F	F	T	F	T	T	T	T
F	F	F	F	T	T	F	T

Logical equivalence

Two proposition p and q are logically equivalent and written as if both p and q have identical truth values

$$\neg(p \wedge q) \equiv (\neg p \vee \neg q)$$

The following logical equivalences apply to any statements; the p's, q's and r's can stand for atomic statements or compound statements.

i. Double Negative Law

$$\neg(\neg p) \equiv p$$

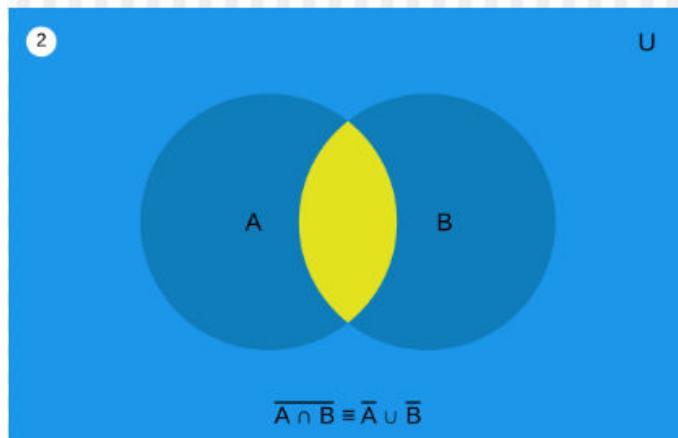
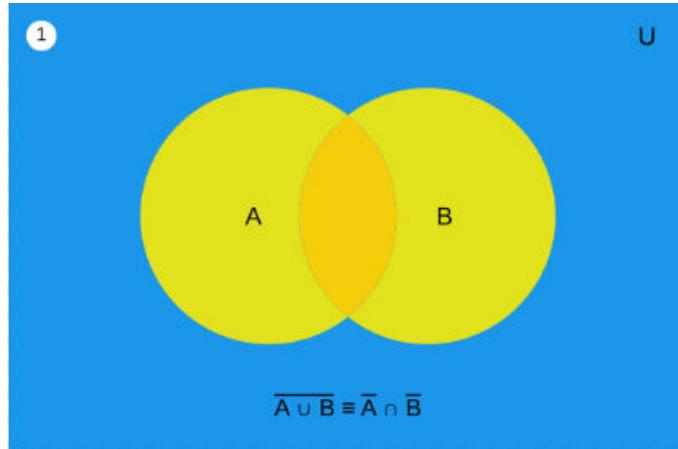
i. De Morgan's Laws

$$\neg(p \vee q) \equiv (\neg p) \wedge (\neg q)$$

$$\neg(p \wedge q) \equiv (\neg p) \vee (\neg q)$$

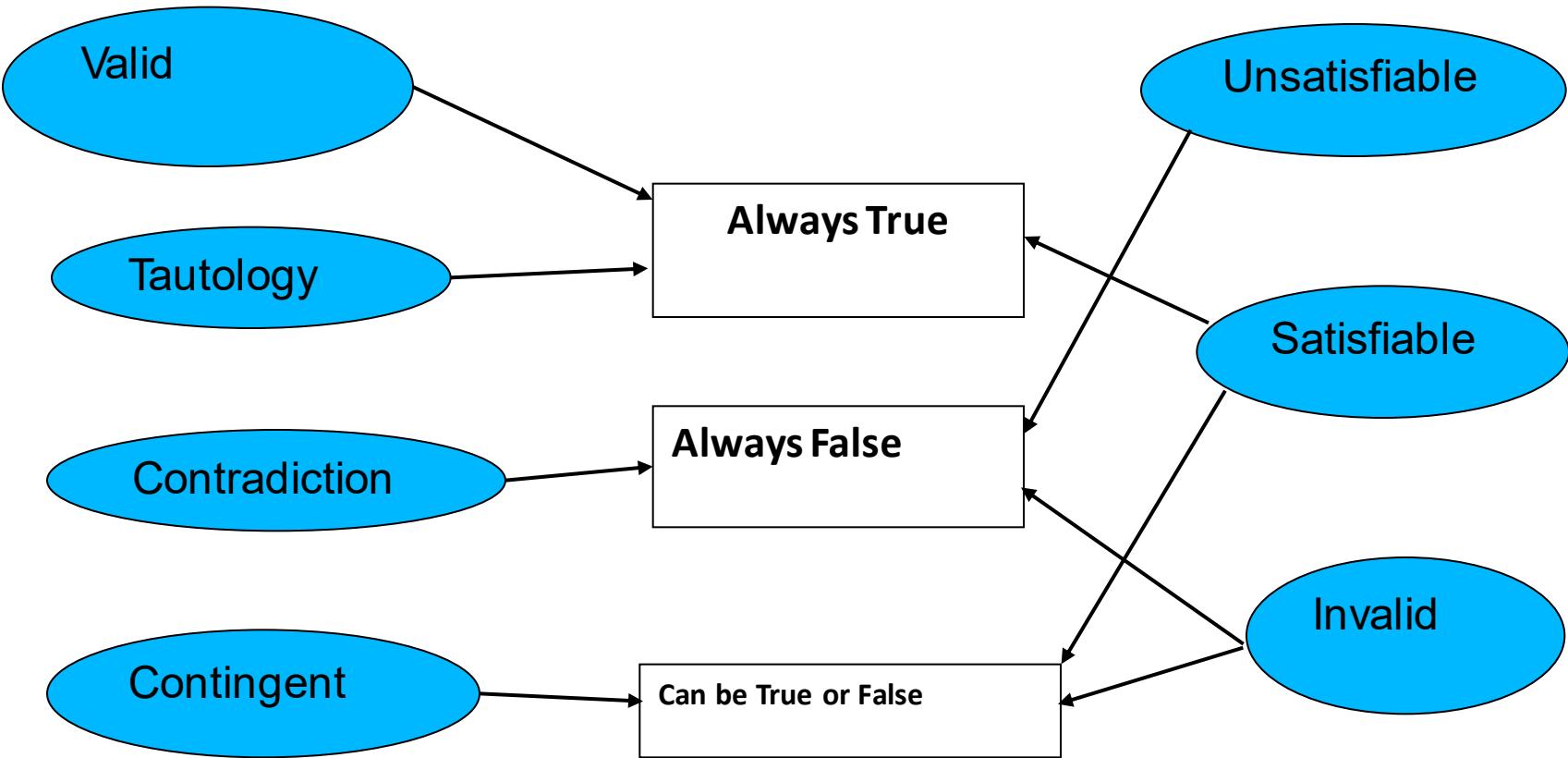
i. Distributive Laws

$$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$



Tautology	If a proposition have a truth value for every interpretation
	E.g.: i) $p \vee \neg p$ ii) $[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$
Contradiction	If a proposition have a false value for every interpretation
	E.g: $p \wedge \neg p$
Contingent	If a proposition have both true and false value
	E.g.: $p \wedge q$

- An argument in propositional logic is sequence of propositions.
- All proposition are called premises and the final proposition is called the conclusion.
- An argument is valid if the truth of all its premises implies that the conclusion is true



State whether the following sentences are valid, unsatisfiable, or neither.

- a. $\text{Smoke} = > \text{Smoke}$
- b. $\text{Smoke} = > \text{Fire}$
- c. $(\text{Smoke} = > \text{Fire}) = > (\neg \text{Smoke} = > \neg \text{Fire})$
- d. $\text{Smoke} \vee \text{Fire} \vee \neg \text{Fire}$

Q. Write implication, converse, inverse, negation, contra positive of the following integration.

“The program is readable only if it is well structured”

p = “the program is readable”

q = “the program is well structured”

➤ Implication: $(p \rightarrow q)$

If the program is readable, then it is well structured.

➤ Converse: $(q \rightarrow p)$

If the program is well structured, then it is readable.

➤ Inverse: $(\neg p \rightarrow \neg q)$

If the program is not readable, then it is not well structured.

➤ Contrapositive: $(\neg q \rightarrow \neg p)$

If the program is not well structured, then it is not readable

➤ Negation of p : $(\neg p)$

The program is not readable.

Q• There are two restaurants next to each other.

One has a sign board as: “Good food is not cheap”.

The other has a sign board as “Cheap food is not good”.

Are both the sign board saying the same thing?

G = “Food is Good”

C = “Food is Cheap”

Sentence 1:- “Good food is not cheap” can be symbolically written as $G \rightarrow \neg C$

Sentence 2:- “Cheap food is not good” can be symbolically written as $C \rightarrow \neg G$

Now, The Truth Table is:

G	C	$\neg G$	$\neg C$	$G \rightarrow \neg C$	$C \rightarrow \neg G$
T	T	F	F	F	F
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	T	T

Since, $G \rightarrow \neg C$ and $C \rightarrow \neg G$ are logically equivalent. So both are saying same thing.

Inference

- The process of drawing conclusion from given premises in an argument is called inference.
- To draw the conclusion from the given statements, we must be able to apply some well-defined steps that helps reaching the conclusion.

Let S be the set of all right answers.

- A **sound** algorithm never includes a **wrong answer** in S , but it might miss a few right answers. => not necessarily "complete".
- A **complete** algorithm should get every right answer in S : include the complete set of right answers. But it **might include a few wrong answers**. It might return a wrong answer for a single input. => not necessarily "sound".

➤ Addition rule

$$\frac{p}{\therefore p \vee q}$$

Example:

- Ram is a student of BE.

\therefore Ram is a student of BE or BCA.

Corresponding Tautology:
 $p \rightarrow (p \vee q)$

p	q	p or q	$p \rightarrow (p \vee q)$
F	F	F	T
F	T	T	T
T	F	T	T
T	T	T	T

➤ Simplification rule

$$\frac{p \wedge q}{\therefore p}$$

Corresponding Tautology:
 $(p \wedge q) \rightarrow p$

Example:

-Subodh and Shyam are the students of BE.

\therefore Subodh is the student of BE

or

Shyam is the student of BE.

➤ Modus Ponens Rule

$$p \rightarrow q$$

P

$$\therefore q$$

Corresponding Tautology:
$$(p \wedge (p \rightarrow q)) \rightarrow q$$

Example:

- If Ram is hard working, then he is intelligent.
 - Ram is hard working.
-

\therefore Ram is intelligent.

➤ Modus Tollens Rule

$$p \rightarrow q$$

$$\neg q$$

$$\therefore \neg p$$

**Corresponding
Tautology:**

$$(\neg p \wedge (p \rightarrow q)) \rightarrow \neg q$$

Example:

- If it is sunny, then we will go to swimming.
 - We will not go swimming.
-

\therefore It is not Sunny.

➤ Hypothetical Syllogism Rule

$$p \rightarrow q$$

$$q \rightarrow r$$

$$\therefore p \rightarrow r$$

Corresponding Tautology:
$$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

Example:

- If Subodh is a BE student, then he loves programming.
 - If Subodh loves programming, then he is expert in java.
-

\therefore If Subodh is a BE student, then he is expert in java.

Disjunctive Syllogism Rule

$$p \vee q$$

$$\neg p$$

$$\therefore q$$

Corresponding Tautology:
 $(\neg p \wedge (p \vee q)) \rightarrow q$

Example:

- Today is Wednesday or Thursday.

- Today is not Wednesday.

\therefore Today is Thursday.

➤ Conjunction rule

$$\begin{array}{c} p \\ q \\ \hline \therefore p \wedge q \end{array}$$

Corresponding Tautology:
 $((p) \wedge (q)) \rightarrow (p \wedge q)$

Example:

- Shyam is the student of BE.
- Hari is the student of BE.

\therefore Shyam and Hari are the students of BE.

➤ Resolution Rule

$$p \vee q$$

$$\neg q \vee r$$

$$\therefore p \vee r$$

Corresponding Tautology:

$$((p \vee q) \wedge (\neg q \vee r)) \rightarrow (p \vee r)$$

“I will study discrete math or I will study English literature.”

“I will not study English literature or I will study database.”

I will study discrete math or I will study database.”

Can we conclude that the conclusion is true if the premises are true?

- If Socrates is human, then Socrates is mortal.
 - Socrates is human.
- ∴ Socrates is mortal.

Can we conclude that the conclusion is true if the premises are true?

- If George does not have eight legs, then he is not a spider.
 - George is a spider.
- ∴ George has eight legs.

Which rule of inference is used in each argument below?

- a. Alice is a Math major. Therefore, Alice is either a Math major or a CSI major.
- b. Jerry is a Math major and a CSI major. Therefore, Jerry is a Math major.
- c. If it is rainy, then the pool will be closed. It is rainy. Therefore, the pool is closed.
- d. If it snows today, the university will close. The university is not closed today. Therefore, it did not snow today.
- e. If I go swimming, then I will stay in the sun too long. If I stay in the sun too long, then I will sunburn. Therefore, if I go swimming, then I will sunburn.
- f. I go swimming or eat an ice cream. I did not go swimming. Therefore, I eat an ice cream

If I work all night on this homework, then I can answer all the exercises. If I answer all the exercises, I will understand the material. Therefore, if I work all night on this homework, then I will understand the material.

Q. “If you send me an e-mail message then I will finish writing the program”, “If you do not send me an e-mail message then I will go to sleep early”, and “If I go to sleep early then I will wake up feeling refreshed”. Lead to the conclusion “If I do not finish writing the program then I will wake up feeling refreshed”.

Solution:

Let

- p = "You send me an e-mail message"
 q = "I will finish writing the program"
 r = "I will go to sleep early"
 s = "I will wake up feeling refreshed"

Hypothesis:

- $p \rightarrow q$
- $\neg p \rightarrow r$
- $r \rightarrow s$

Conclusion: $\neg q \rightarrow s$

Steps	Operations	Reasons
1	$p \rightarrow q$	Given hypothesis
2	$\neg q \rightarrow \neg p$	Using contra positive on 1
3	$\neg p \rightarrow r$	Given hypothesis
4	$\neg q \rightarrow r$	Using hypothetical syllogism on 2 and 3
5	$r \rightarrow s$	Given hypothesis
6	$\neg q \rightarrow s$	Using hypothetical syllogism on 4 and 5

Hence the given hypotheses lead to the conclusion $\neg q \rightarrow s$

Q. “Hari is playing in garden”, “If he is playing in garden then he is not doing homework”, “If he is not doing homework, then he is not learning” leads to the conclusion “He is not learning”.

Use rules of inference to show that the hypotheses “Ram works hard,” “If Ram works hard, then he is a dull boy,” and “If Ram is a dull boy, then he will not get the job” imply the conclusion “Ram will not get the job.”

Q. Prove the conclusion based on the given hypothesis using the rule of resolution

Hypothesis

$$p \rightarrow q$$

$$\neg p \rightarrow r$$

$$r \rightarrow s$$

Conclusion:

$$\neg q \rightarrow s$$

Q. Prove by resolution

Hypothesis

- If it rains, Joe will bring his umbrella
- If joe has an umbrella, he doesnot get wet
- If it doesnot rain, joe doesnot get wet

Hence, Joe doesnot get wet

Predicate

- Predicate is a part of declarative sentences describing the properties of an object or relation among objects. For example: “is a student” is a predicate as ‘A is a student’ and ‘B is a student’.
- A predicate logic is a formal system that uses objects/variables and quantifiers (\forall , \exists) to formulate propositions.

Assignment :

Differentiate between Propositional Logic and Predicate Logic

Quantification

- A quantifier is a symbol that permits one to declare the range or scope of variables in a logical expression.
- The process of binding propositional variable over a given domain is called quantification.
- Two common quantifier are the existential quantifier (“there exists or for some or at least one”) and universal quantifier (“for all or for each or for any or for every and or for arbitrary”).

Universal Quantifier (\forall : For All)

- It is denoted by \forall and used for universal quantification.
- The universal quantification of $p(x)$ denoted by $\forall x p(x)$ is proposition that is true for all values in universal set
- The universal quantifier is read as:
 - For all x , $p(x)$ holds
 - For each x , $p(x)$ holds
 - For every x , $p(x)$ holds

Existential Quantifier(\exists : For Some)

- It is denoted by \exists and used for existential quantification.
- The existential quantification of $p(x)$ denoted by $\exists x p(x)$ is proposition that is true for some values in universal set.
- The existential quantifier is read as:
 - There is an x , such that $p(x)$
 - There is at least one x such that $p(x)$
 - For some x , $p(x)$

Q. Assume that

P (x) denotes “x is an accountant.”

Q (x) denotes “x owns a maruti.”

Now, represent the following statement symbols.

- a) All accountants own maruti.

Meaning: For all x, if x is an accountant, then x owns maruti

$$\underbrace{\forall x}_{\text{For all } x} \quad \underbrace{p(x)}_{\text{x is an accountant}} \quad \underbrace{q(x)}_{\text{x owns maruti}}$$

$$\Rightarrow \forall x (p(x) \rightarrow q(x)) |$$

- b) Some accountants own maruti

Meaning: For some x, x is an accountant and x owns maruti

$$\underbrace{\exists x}_{\text{For some } x} \quad \underbrace{p(x)}_{\text{x is an accountant}} \quad \underbrace{q(x)}_{\text{x owns maruti}}$$

$$\Rightarrow \exists x (p(x) \wedge q(x))$$

- c) All owners of maruti are accountants

Meaning: For all \underline{x} , if x is a owner of maruti, then x is an accountant.

$$\Rightarrow \forall \underline{x} (q(x) \rightarrow f(x))$$

- d) Someone who owns a maruti, is an accountant

Meaning: For some x , who owns a maruti and x is an accountant

$$\Rightarrow \exists \underline{x} (q(x) \wedge f(x))$$

Q. Convert into FOPL

- a. All men are people.
- b. Marcus was Pompeian.
- c. All Pompeian were Roman.
- d. Ram tries to assassinate Hari.
- e. All Romans were either loyal to caser or hated him.
- f. Socrates is a man. All men are mortal; therefore Socrates is mortal.
- g. Some student in this class has studied mathematics.

a. All men are people.

$$\Rightarrow \forall x \text{MAN}(x) \rightarrow \text{PEOPLE}(x)$$

b. Marcus was Pompeian.

$$\Rightarrow \text{POMPEIAN}(\text{Marcus})$$

c. All Pompeian were Roman.

$$\Rightarrow \forall x \text{POMPEIAN}(x) \rightarrow \text{ROMAN}(x)$$

d. Ram tries to assassinate Hari.

$$\Rightarrow \text{ASSASSINATE}(\text{Ram}, \text{Hari})$$

e. All Romans were either loyal to caser or hated him.

$$\Rightarrow \forall x \text{ ROMAN}(x) \rightarrow \text{LOYAL}(x, \text{caser}) \vee \text{HATES}(x, \text{caser})$$

f. Socrates is a man. All men are mortal; therefore Socrates is mortal.

$$\Rightarrow \text{MAN}(\text{Socrates}), \forall x \text{ MAN}(x) \rightarrow \text{MORTAL}(x), \text{MORTAL}(\text{Socrates})$$

g. Some student in this class has studied mathematics.

\Rightarrow Let

- $S(x) = "x \text{ is a student in this class}"$
- $M(x) = "x \text{ has studied mathematics}"$

Hence, required expression is: $\exists x [S(x) \wedge M(x)]$

Rules of Inference for Quantified Statements

- a. Universal Instantiation

$$\frac{\forall x p(x)}{\therefore p(d)}$$

Where d is in the domain of discourse D.

For example: If all balls in a box are red then any randomly drawn ball is also red

- b. Universal Generalization

$$\frac{p(d)}{\therefore \forall x p(x)}$$

Where, d is the domain of discourse D.

For example: If all the ball in a box are taken one by one in randomly manner and if all are red then we conclude that all balls in the box are red.

c. Existential Instantiations

$$\exists x p(x)$$

$$\therefore p(d)$$

For some d in the domain of discourse.

For example: If some balls in a box are red then resulting ball after experiment will also be red.

d. Existential Generalization

$$p(d)$$

$$\therefore \exists x p(x)$$

For some d in the domain of discourse.

For example: If we take only one random experiment for the ball drawn and apply the result of ball to the domain.

Combining Rules of Inference for Propositions and Quantified Statements

These inference rules are frequently used and combine propositions and quantified statements:

- **Universal Modus Ponens**

$$\frac{\forall x(P(x) \rightarrow Q(x)) \quad P(a), \text{ where } a \text{ is a particular element in the domain}}{\therefore Q(a)}$$

- **Universal Modus Tollens**

$$\frac{\forall x(P(x) \rightarrow Q(x)) \quad \neg Q(a), \text{ where } a \text{ is a particular element in the domain}}{\therefore \neg P(a)}$$

Q. Given Expression: All men are mortal. Einstein is a man. Prove that “Einstein is mortal” using FOPL.

Solution: Let

$$M(x) = \text{“}x \text{ is a man”}$$

$$N(x) = \text{“}x \text{ is mortal”}$$

Hypothesis: $\forall x [M(x) \rightarrow N(x)]$, $M(\text{Einstein})$

Conclusion: $N(\text{Einstein})$

Steps	Operations	Reasons
1.	$\forall x [M(x) \rightarrow N(x)]$	Given Hypothesis
2.	$M(\text{Einstein}) \rightarrow N(\text{Einstein})$	Using universal instantiation on 1
3.	$M(\text{Einstein})$	Given Hypothesis
4.	$N(\text{Einstein})$	Using modus pollens on 2 and 3

Hence the given hypotheses lead to the conclusion “Einstein is mortal”.

Q. Given Expression: “Lions are dangerous animals”, and “There are lions”. Prove that “There are dangerous animals” using FOPL.

Solution: Let

$$D(x) = \text{“}x \text{ is a dangerous animal”}$$

$$L(x) = \text{“}x \text{ is a lion”}$$

Hypothesis: $\forall x [L(x) \rightarrow D(x)], \exists x L(x)$

Conclusion: $\exists x D(x)$

Steps	Operations	Reasons
1.	$\forall x [L(x) \rightarrow D(x)]$	Given Hypothesis
2.	$L(a) \rightarrow D(a)$	Using universal instantiation on 1
3.	$\exists x L(x)$	Given Hypothesis
4.	$L(a)$	Using existential instantiation on 3
5.	$D(a)$	Using modus pollens on 2 and 4
6.	$\exists x D(x)$	Using existential generalization on 5

Hence the given hypotheses lead to the conclusion “There are dangerous animals”

Q. Show that the premises “Everyone in this discrete mathematics class has taken a course in computer science” and “Marla is a student in this class” imply the conclusion “Marla has taken a course in computer science.”

Show that the premises “A student in this class has not read the book,” and “Everyone in this class passed the first exam” imply the conclusion “Someone who passed the first exam has not read the book.”

Solution: Let $C(x)$ be “ x is in this class,” $B(x)$ be “ x has read the book,” and $P(x)$ be “ x passed the first exam.” The premises are $\exists x(C(x) \wedge \neg B(x))$ and $\forall x(C(x) \rightarrow P(x))$. The conclusion is $\exists x(P(x) \wedge \neg B(x))$. These steps can be used to establish the conclusion from the premises.

Step	Reason
1. $\exists x(C(x) \wedge \neg B(x))$	Premise
2. $C(a) \wedge \neg B(a)$	Existential instantiation from (1)
3. $C(a)$	Simplification from (2)
4. $\forall x(C(x) \rightarrow P(x))$	Premise
5. $C(a) \rightarrow P(a)$	Universal instantiation from (4)
6. $P(a)$	Modus ponens from (3) and (5)
7. $\neg B(a)$	Simplification from (2)
8. $P(a) \wedge \neg B(a)$	Conjunction from (6) and (7)
9. $\exists x(P(x) \wedge \neg B(x))$	Existential generalization from (8)

Nested Quantifiers

quantifiers are nested if one is within the scope of the other, such as

$$\forall x \exists y (x + y = 0).$$

Note that everything within the scope of a quantifier can be thought of as a propositional function. For example,

$$\forall x \exists y (x + y = 0)$$

is the same thing as $\forall x Q(x)$, where $Q(x)$ is $\exists y P(x, y)$, where $P(x, y)$ is $x + y = 0$.

- Brothers are siblings

$$\forall x;y \text{ Brother}(x;y) \Rightarrow \text{Sibling}(x;y)$$

- “Sibling” is symmetric

$$\forall x;y \text{ Sibling}(x;y), \text{ Sibling}(y;x)$$

- One's mother is one's female parent

$$\forall x;y \text{ Mother}(x;y), (\text{Female}(x) \wedge \text{Parent}(x;y))$$

- Every gardener likes the sun.
 $\forall x \text{ gardener}(x) \Rightarrow \text{likes}(x, \text{Sun})$
 - All purple mushrooms are poisonous.
 $\forall x (\text{mushroom}(x) \wedge \text{purple}(x)) \Rightarrow \text{poisonous}(x)$
-

Clausal Forms

- A clause is an **expression formed from a finite collection of literals** (variables or their negations)
- A clause that contains only \vee is called a **disjunctive clause** and only \wedge is called a **conjunctive clause**.
- Negation is allowed, but only directly on variables.
 - $p \vee \neg q \vee r$: a disjunctive clause
 - $\neg p \wedge q \wedge \neg r$: a conjunctive clause
 - $\neg p \wedge \neg q \vee r$: neither

CNF and DNF

If we put a bunch of disjunctive clauses together with \wedge , it is called conjunctive normal form.

- For example: $(p \vee r) \wedge (\neg q \vee \neg r) \wedge q$ is in conjunctive normal form.

Similarly, putting conjunctive clauses together with \vee , it is called disjunctive normal form.

- For example: $(p \wedge \neg q \wedge r) \vee (\neg q \wedge \neg r)$ is in disjunctive normal form.

Conversion Procedure for proposition into Normal Forms

- › We can use the definitions to get rid of \rightarrow , \leftrightarrow , and \oplus
 - $B \leftrightarrow (P \vee Q)$ as $(B \rightarrow (P \vee Q)) \wedge ((P \vee Q) \rightarrow B)$
 - $(B \rightarrow (P \vee Q))$ as $(\neg B \vee P \vee Q)$
 - $(P \oplus Q)$ as $(P \wedge \neg Q) \vee (\neg P \wedge Q)$
- › Use DE Morgan's laws.
 - $\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$ (De Morgan)
 - $\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$ (De Morgan)
- › Use double negation to get rid of any $\neg\neg$ that showed up.
- › Use the distributive rules to move things in/out of parens as we need to.
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$$

Converting to conjunctive normal form

a. $\neg((\neg p \rightarrow \neg q) \wedge \neg r)$

$$\begin{aligned} &\equiv \neg((\neg\neg p \vee \neg q) \wedge \neg r) && [\text{definition}] \\ &\equiv \neg(p \vee \neg q) \wedge \neg r && [\text{double negation}] \\ &\equiv \neg(p \vee \neg q) \vee \neg\neg r && [\text{DeMorgan's}] \\ &\equiv \neg(p \vee \neg q) \vee r && [\text{double negation}] \\ &\equiv (\neg p \wedge \neg\neg q) \vee r && [\text{DeMorgan's}] \\ &\equiv (\neg p \wedge q) \vee r && [\text{double negation}] \\ &\equiv (\neg p \vee r) \wedge (q \vee r) && [\text{distributive}] \end{aligned}$$

b. $(p \rightarrow q) \rightarrow (\neg r \wedge q)$ into DNF

$$\equiv \neg(p \rightarrow q) \vee (\neg r \wedge q) \quad [\text{definition}]$$

$$\equiv \neg(\neg p \vee q) \vee (\neg r \wedge q) \quad [\text{definition}]$$

$$\equiv (\neg\neg p \wedge \neg q) \vee (\neg r \wedge q) \quad [\text{DeMorgan's}]$$

$$\equiv (p \wedge \neg q) \vee (\neg r \wedge q) \quad [\text{double negation}]$$

$$\equiv (p \wedge \neg q) \vee (\neg r \wedge q)$$

$$\equiv (p \vee (\neg r \wedge q)) \wedge (\neg q \vee (\neg r \wedge q))$$

[distributive]

$$\equiv (p \vee (\neg r \wedge q)) \wedge (\neg q \vee \neg r) \wedge (\neg q \vee q)$$

[distributive]

$$\equiv (p \vee (\neg r \wedge q)) \wedge (\neg q \vee \neg r) \wedge \mathbf{T}$$

[negation]

$$\equiv (p \vee (\neg r \wedge q)) \wedge (\neg q \vee \neg r)$$

[identity]

$$\equiv (p \vee \neg r) \wedge (p \vee q) \wedge (\neg q \vee \neg r)$$

[distributive]

Assignment

What is DNF? How can we convert the sentence into DNF?

Horn Clause

- A Horn clause is a clause (a disjunction of literals) with at most one positive, i.e. unnegated, literal
- Any Horn clause therefore belongs to one of four categories:
 - 1 positive literal, at least 1 negative literal.
A rule has the form " $\neg P_1 \vee \neg P_2 \vee \dots \vee \neg P_k \vee Q$ ". This is logically equivalent to " $[P_1 \wedge P_2 \wedge \dots \wedge P_k] \Rightarrow Q$ "
 - 1 positive literal, 0 negative literals.
Examples: "man(socrates)", "parent(elizabeth,charles)", "ancestor(X,X)"
 - 0 positive literals, at least 1 negative literal.
 $\neg p \vee \neg q \vee \dots \vee \neg t$
 - The null clause: 0 positive and 0 negative literals.
Appears only as the end of a resolution proof.

Skolemization

Skolemization eliminates existential quantifiers by replacing each existentially quantified variable with a Skolem constant or Skolem function. (A Skolem constant is a function of no variables)

$$\forall x \exists y \forall z. P(x, y, z)$$

is not in Skolem normal form because it contains the existential quantifier. Skolemization replaces the y with the function $f(x)$ where f is a new function symbol, and removes the quantification over y

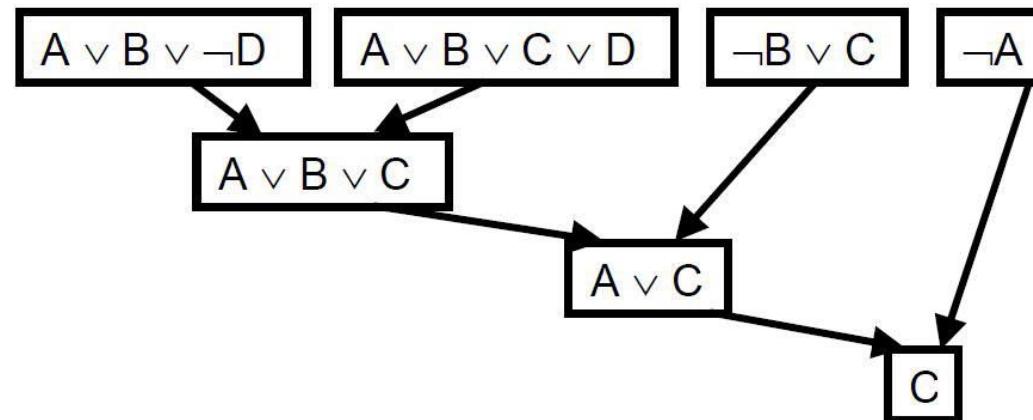
$$\forall x \forall z. P(x, f(x), z)$$

Resolution in Propositional Logic

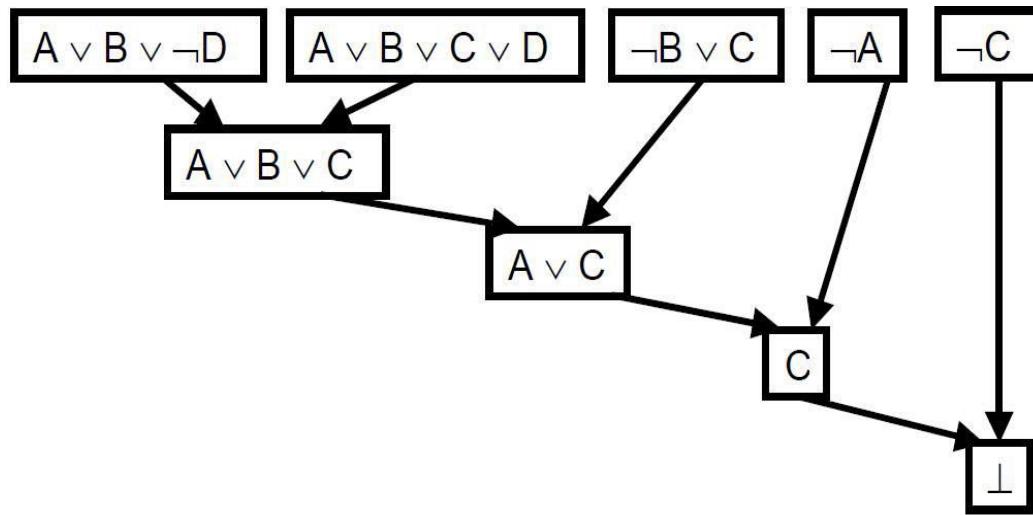
- The resolution rule in propositional logic is a single valid inference rule that produces a new clause implied by two clauses
- Resolution principle was introduced by John Alan Robinson in 1965.
- The resolution technique can be applied for sentences in propositional logic and first-order logic.
- Resolution technique can be used only for disjunctions of literals to derive new conclusion.
- The resolution rule for the propositional calculus can be stated as following:
 $(P \vee Q)$ and $(\neg Q \vee R)$, gives $(P \vee R)$.

Q. Let $P_1 = A \vee B \vee \neg D$, $P_2 = A \vee B \vee C \vee D$, $P_3 = \neg B \vee C$, $P_4 = \neg A$, $P_5 = C$ then
Show that $\{P_1, P_2, P_3, P_4\} \Rightarrow P_5$

Solution:



Q. Let $P_1 = A \vee B \vee \neg D$, $P_2 = A \vee B \vee C \vee D$, $P_3 = \neg B \vee C$, $P_4 = \neg A$, $P_5 = C$ then
 Show that $\{P_1, P_2, P_3, P_4, \neg P_5\} = \perp$



Resolution in FOP

- During resolution in propositional logic, it is easy to determine that two literals (e.g. p and $\neg p$) cannot both be true at the same time.
- In predicate logic this matching process is more complicated since the argument of the predicate must be considered.
- For example, $\text{MAN}(\text{John})$ and $\neg\text{MAN}(\text{John})$ is a contradiction, while $\text{MAN}(\text{John})$ and $\neg\text{MAN}(\text{Smith})$ is not. Thus, in order to determine contradictions, we need a matching procedure, called **unification algorithm** that compares two literals and discovers whether there exists a set of substitutions that makes them identical.

To unify two literals, the initial predicate symbol on both must be same; otherwise there is no way of unification.

For example, $Q(x, y)$ and $R(x, y)$ cannot unify but $P(x, x)$ and $P(y, z)$ can be unify by substituting z by x and y by x .

Q. Given Expression: John likes all kinds of foods. Apples are food. Chicken is food.
Prove that John likes Peanuts using resolution.

Soln:

➤FOPL

➤ $\forall x \text{ FOOD}(x) \rightarrow \text{LIKES}(\text{John}, x)$

or

➤ $\neg \text{FOOD}(x) \vee \text{LIKES}(\text{John}, x)$

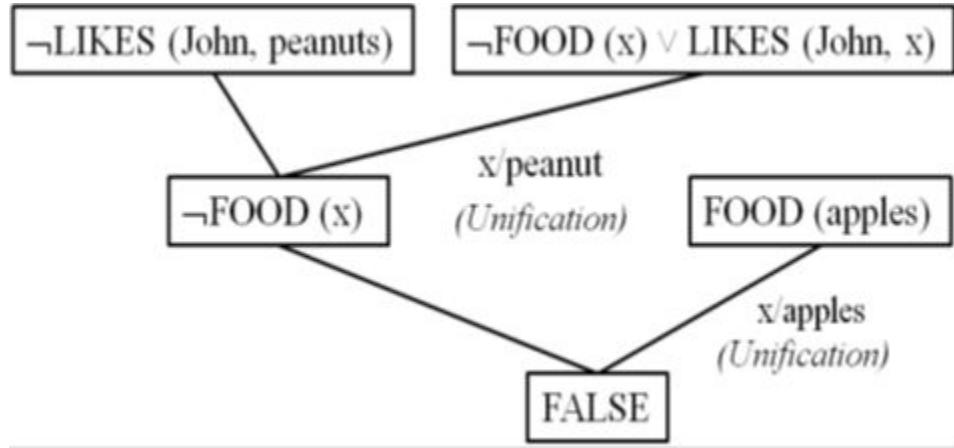
➤ $\text{FOOD}(\text{apples})$

➤ $\text{FOOD}(\text{chicken})$

Now, we have to prove: $\text{LIKES}(\text{John}, \text{peanuts})$.

To prove the statement using resolution, let's take the negation of this as:

$\neg \text{LIKES}(\text{John}, \text{peanuts})$



Since, $\neg \text{LIKES}(\text{John}, \text{peanuts})$ is not possible and hence the: $\text{LIKES}(\text{John}, \text{peanuts})$ is proved.

Q. Given Expression:

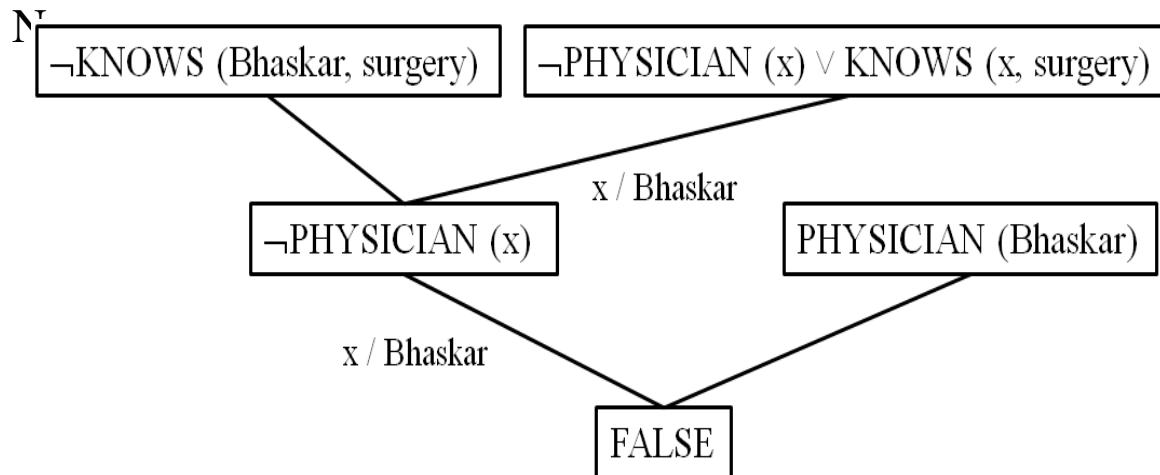
Bhaskar is a physician. All physicians know surgery.

Prove that Bhaskar knows surgery using principle of resolution.

Soln:

- FOPL is
- PHYSICIAN (Bhaskar)
- $\forall x \text{ PHYSICIAN}(x) \rightarrow \text{KNOWS}(x, \text{surgery}) \text{ or}$
 $\neg \text{PHYSICIAN}(x) \vee \text{KNOWS}(x, \text{surgery})$

Now, we have to prove that: KNOWS (Bhaskar, surgery). To prove the statement using resolution (proof by contradiction); let's take the negation of this as: $\neg \text{KNOWS}(\text{Bhaskar}, \text{surgery})$



Q. Given Expression:

All carnivorous animals have sharp teeth. Tiger is carnivorous. Fox is carnivorous.

Prove that tiger has sharp teeth.

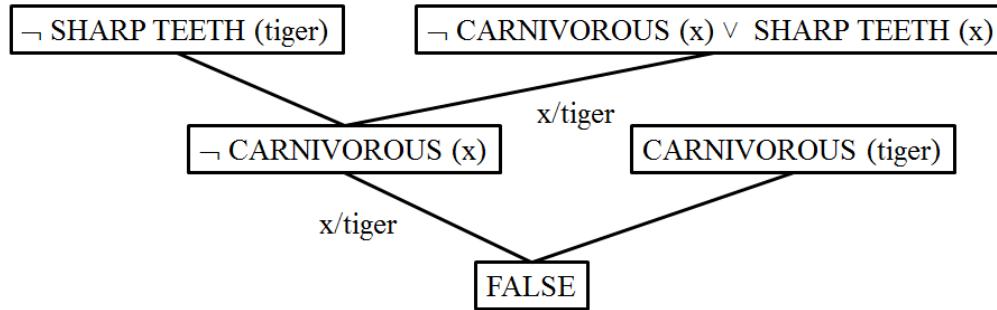
Soln:

FOPL is

- $\forall x \text{ CARNIVOROUS } (x) \rightarrow \text{SHARP TEETH } (x)$ or
 $\neg \text{CARNIVOROUS } (x) \vee \text{SHARP TEETH } (x)$
- $\text{CARNIVOROUS } (\text{tiger})$
- $\text{CARNIVOROUS } (\text{fox})$

Now, we have to prove that: $\text{SHARP TEETH } (\text{tiger})$. To prove the statement using resolution (proof by contradiction); let's take the negation of this as:
 $\neg \text{SHARPTEETH } (\text{tiger})$

Now



What is the difference between inference, reasoning and deduction?

Reason is the capacity for consciously making sense of things, applying logic, establishing and verifying facts, and changing or justifying practices, institutions, and beliefs based on new or existing information.

We use reasons or reasoning to form **inferences** which are basically conclusions drawn from propositions or assumptions that are supposed to be true.

Deduction is a specific form of reasoning and starts with a hypothesis and **examines the possibilities within that hypothesis to reach a conclusion**.

Rule Based Deduction System

Rule-based systems are used as a way to store and manipulate knowledge to interpret information in a useful way.

A classic example of a rule-based system is the domain-specific expert system that uses rules to make deductions or choices. For example, an expert system might help a doctor choose the correct diagnosis based on a cluster of symptoms

In this approach, idea is to use production rules, sometimes called IF-THEN rules.

The syntax structure is

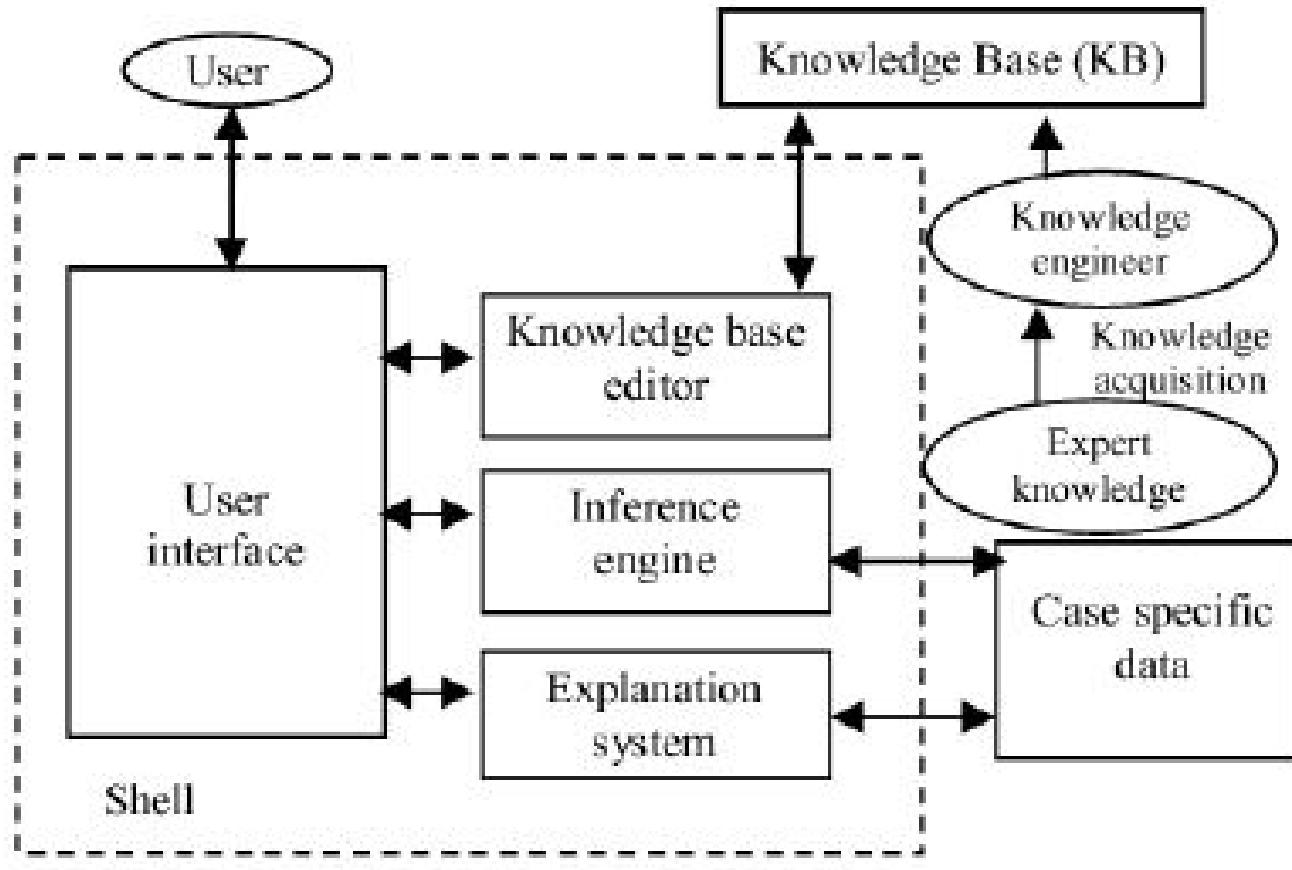
IF <premise> THEN <action>

- <premise>- is Boolean. The AND, and to a lesser degree OR and NOT, logical connectives are possible.

- <action>- a series of statements

A typical rule-based system has four basic components:

- A **list of rules** or **rule base**, which is a specific type of knowledge base.
- An **inference engine**, which infers information or takes action based on the interaction of input and the rule base.
- Temporary **working memory**.
- A **user interface** or other connection to the outside world through which input and output signals are received and sent.



Example:

“If the patient has stiff neck, high fever and a headache, check for Brain Meningitis”. Then it can be represented in rule based approach as:

IF

<fever, over, 39> and <neck, stiff, yes> and <head, pain, yes>

THEN

Add(<PATIENT,DIAGNOSE, MENINGITIS>)

Monotonic Reasoning

- A reasoning system is monotonic if the truthfulness of a conclusion does not change when new information is added to the system
- Adding knowledge base does not reduce the set of propositions that can be derived.
- Eg: We are A//B and B//C. We can conclude A//C.

Now adding a fact that A//D, does not change the result A//C

Non-Monotonic Reasoning

- A logic is non-monotonic if some conclusions can be invalidated by adding more knowledge in the knowledge base.
- Non monotonic system are harder to deal with than monotonic systems
- Non – monotonic systems require more storage space as well as more processing time than monotonic systems.
- Eg: Given “Birds Fly”. Simply means all birds fly.

But a fact “Ostrich is bird” is added, results the conclusion is false.

Assignment

Describe Fuzzy Logic

Knowledge

Knowledge **is a theoretical or practical understanding** of a subject or a domain and it is also the sum of what is currently known.

Hence, knowledge is the sum of what is known: the body of truth, information, and principles acquired by mankind.

Knowledge according to Sunasee and Sewery, 2002

“Knowledge is human proficiency stored in a person’s mind, gained through experience, and interaction with the person’s environment.”

In general, knowledge is more than just data, it consist of: facts, ideas, beliefs, heuristics, associations, rules, abstractions, relationships, customs.

Research literature classifies knowledge as follows:

- Classification-based Knowledge » Ability to classify information
- Decision-oriented Knowledge » Choosing the best option
- Descriptive knowledge » State of some world (heuristic)
- Procedural knowledge » How to do something
- Reasoning knowledge » What conclusion is valid in what situation?
- Assimilative knowledge » What its impact is?

Knowledge Representation

Knowledge representation (KR) is the **study of how knowledge about the world can be represented and what kinds of reasoning can be done with that knowledge**. Knowledge Representation is the method used to encode knowledge in Intelligent Systems.

Some issues that arise in knowledge representation from an AI perspective are:

- How do people represent knowledge?
- What is the nature of knowledge and how do we represent it?
- Should a representation scheme deal with a particular domain or should it be general purpose?
- How expressive is a representation scheme or formal language?
- Should the scheme be declarative or procedural?

The following properties/Characters should be possessed by a knowledge representation system.

- Representational Adequacy
 - the ability to represent the required knowledge;
- Inferential Adequacy
 - the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;
- Inferential Efficiency
 - the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;
- Acquisitional Efficiency
 - the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Semantics net

- Study of Meaning
- Semantic networks can
 - Show natural relationships between objects/concepts
 - Be used to represent declarative/descriptive knowledge
- Semantic Network is the graphical representation of the knowledge.

Semantic networks are constructed using nodes linked by directional lines called arcs

A node can represent a fact description

- Physical object
- Concept
- Event

An arc (or link) represents relationships between nodes. There are some 'standard' relationship types

- 'Is-a' relationship
- 'Has-a' relationship

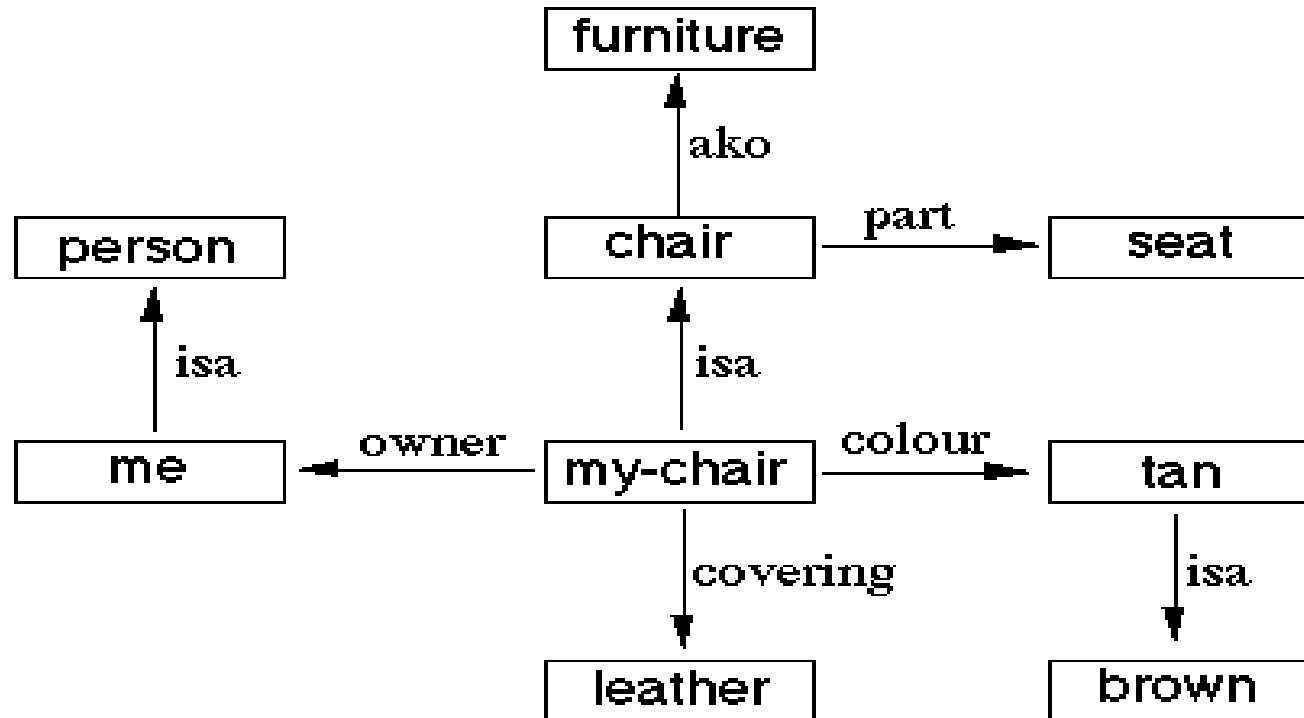
Advantages of a semantic network

- Explicit and easy to understand.
- The net is its own index – quick inference possible.
- Supports default reasoning in finite time.

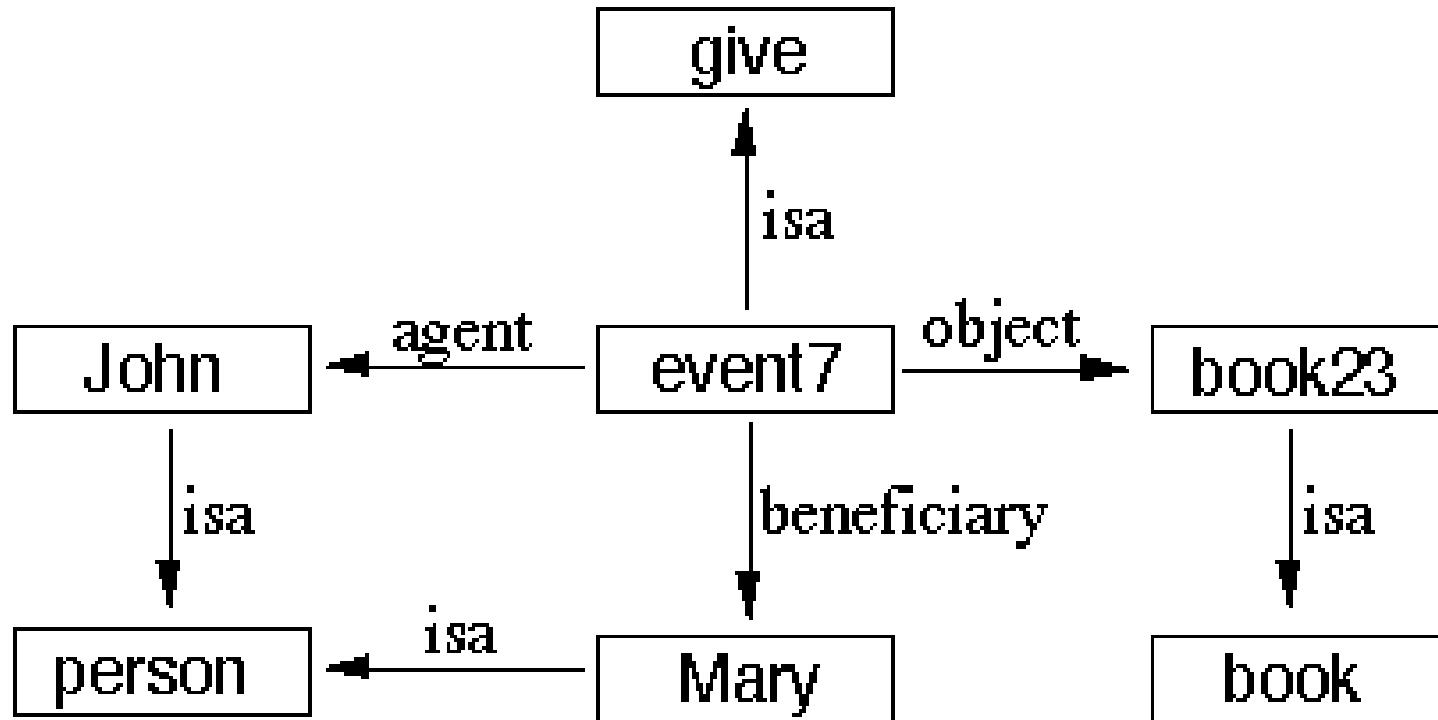
Disadvantages of a semantic network

- incomplete
- no interpretation standard
- ambiguity in node/link descriptions
- not temporal (i.e. doesn't represent time or sequence)

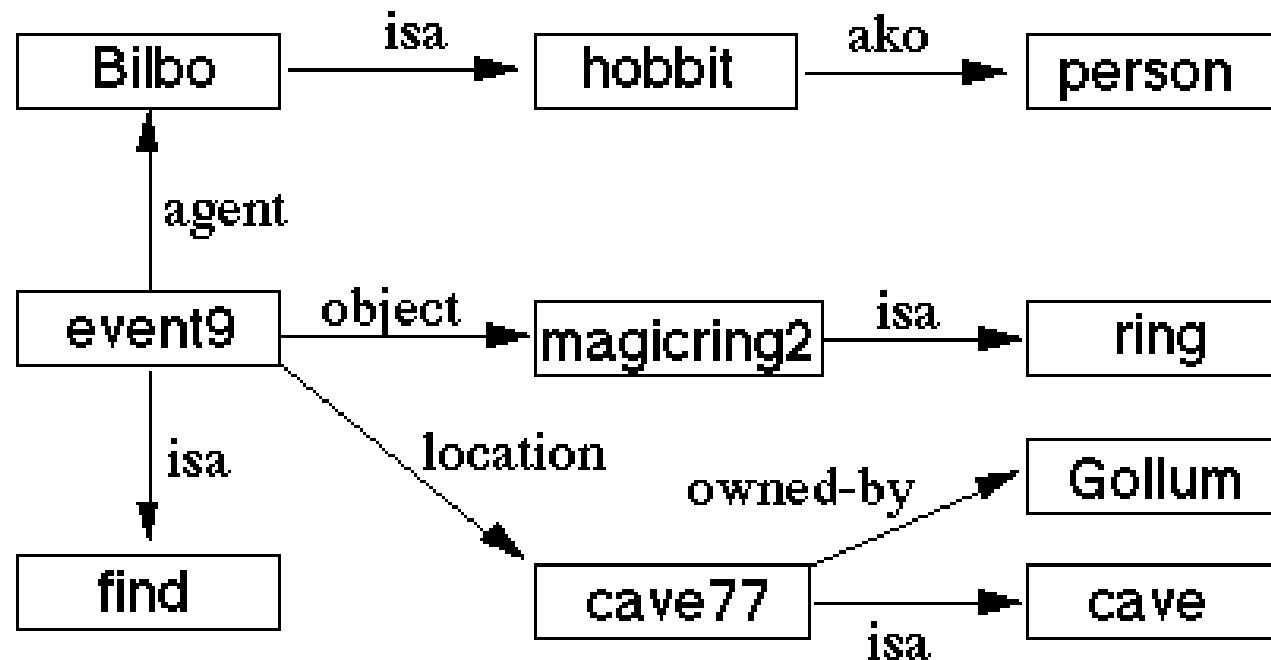
I own a tan leather chair



John gives the book to Mary



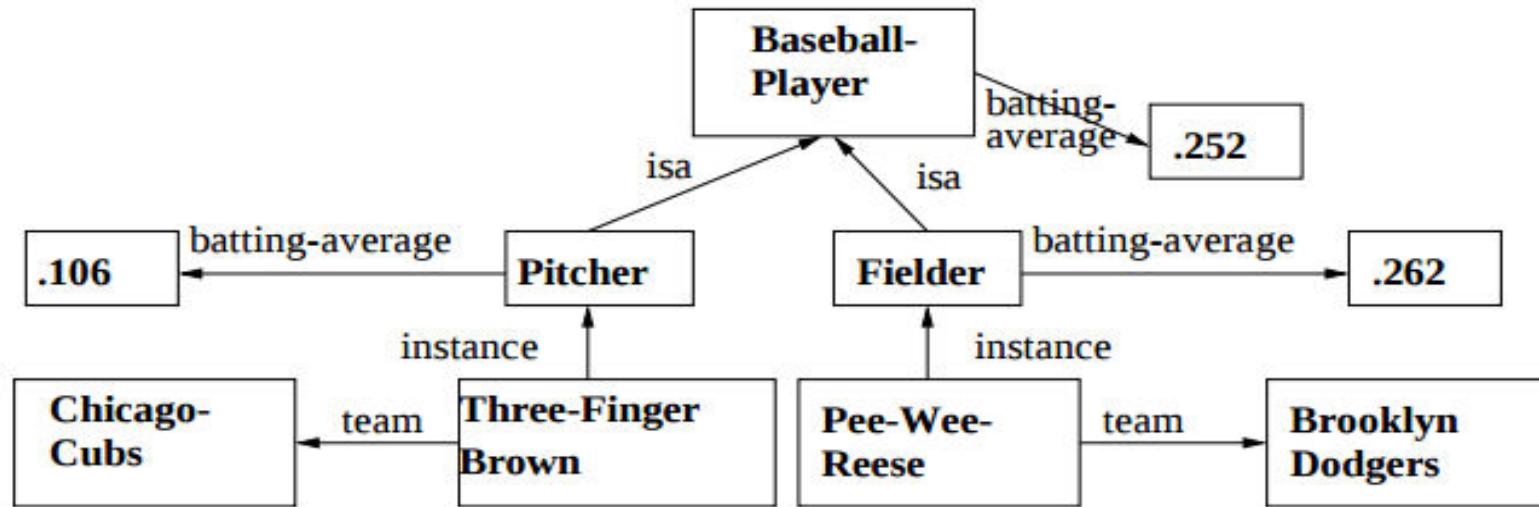
Bilbo finds the magic ring in Gollum's cave



Frames

- A *frame* is a **data structure containing typical knowledge about a concept or object** (Marvin Minsky (mid 1970s)).
- A frame represents knowledge about real world things (or entities).
- Each frame has a name and slots. Slots are the properties of the entity that has the name, and they have values or pointer to other frames (a table like data structure).
- A particular value may be:
 - a default value
 - an inherited value from a higher frame
 - a procedure, called a daemon, to find a value
 - a specific value, which might represent an exception.

Converting Semantic Net into Frame



Baseball Player*is-a:* Adult Male*batting average:* .252*bats:* equal to handed*team:*

:

Fielder*is-a:* Baseball player*batting average:* .262**Pee-Wee-Reese***instance:* Baseball player*team:* Brooklyn Dodgers

Conceptual Dependency

- CD theory was developed by Schank in 1973 to 1975 to represent the meaning of NL sentences.
 - It helps in drawing inferences
 - It is independent of the language
- CD representation of a sentence is not built using words in the sentence rather built using conceptual primitives which give the intended meanings of words.
- CD provides **structures** and specific **set of primitives** from which representation can be built.

Conceptual dependency theory of four primitive conceptualizations

ACT Actions {one of the CD primitives}

PP Objects {picture producers}

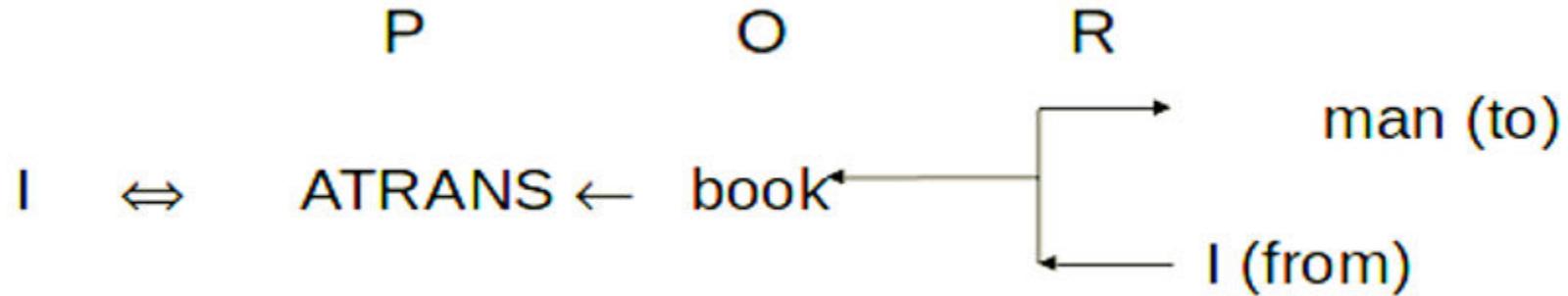
AA Modifiers of actions {action aiders}

PA Modifiers of PP's {picture aiders}

Primitive Acts of CD theory

ATRANS	Transfer of an abstract relationship (i.e. give)
PTRANS	Transfer of the physical location of an object (e.g., go)
PROPEL	Application of physical force to an object (e.g. push)
MOVE	Movement of a body part by its owner (e.g. kick)
GRASP	Grasping of an object by an action (e.g. throw)
INGEST	Ingesting of an object by an animal (e.g. eat)
EXPTEL	Expulsion of something from the body of an animal (e.g. cry)
MTRANS	Transfer of mental information (e.g. tell)
MBUILD	Building new information out of old (e.g. decide)
SPEAK	Producing of sounds (e.g. say)
ATTEND	Focusing of a sense organ toward a stimulus (e.g. listen)

I gave a book to the man



It should be noted that this representation is same for different saying with same meaning.

For example

I gave the man a book,

The man got book from me,

The book was given to man by me etc.

- Arrows indicate directions of dependency
- Double arrow indicates two way link between actor and action.
 - O – for the object case relation
 - R – for the recipient case relation
 - P – for past tense
 - D - destination

Assignment

Write the rules for Conceptual Dependency.

Scripts

- Scripts were introduced by Schank and Abelson introduced in 1977 that used CD framework.
- The scripts are useful in describing certain conventional situations.
- It consists of set of slots containing default values along with some information about the type of values similar to frames.

Script Components

Each script contains the following main components.

Entry Conditions: Must be satisfied before events in the script can occur.

Results: Conditions that will be true after events in script occur.

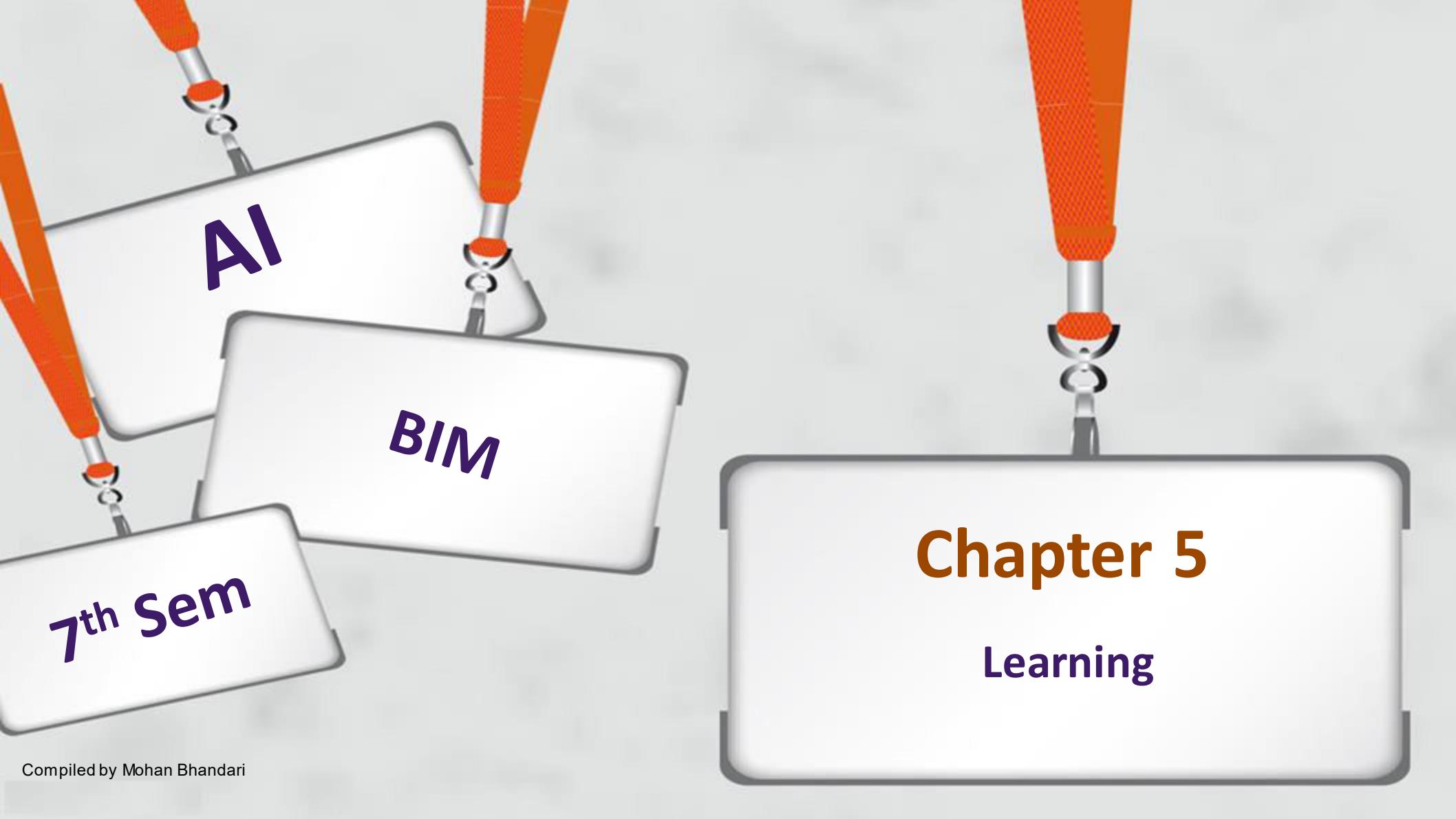
Props: Slots representing objects involved in the events.

Roles: Persons involved in the events.

Track: Specific variation on more general pattern in the script. Different tracks may share many components of the same script but not all.

Scenes: The sequence of *events* that occur. Events are represented in conceptual dependency form.

Script : Play in theater	Various Scenes
Track: Play in Theater Props: <ul style="list-style-type: none"> • Tickets • Seat • Play 	Scene 1: Going to theater <ul style="list-style-type: none"> • P PTRANS P into theater • P ATTEND eyes to ticket counter
Roles: <ul style="list-style-type: none"> • Person (who wants to see a play) – P • Ticket distributor – TD • Ticket checker – TC 	Scene 2: Buying ticket <ul style="list-style-type: none"> • P PTRANS P to ticket counter • P MTRANS (need a ticket) to TD • TD ATRANS ticket to P
Entry Conditions: <ul style="list-style-type: none"> • P wants to see a play • P has a money Results: <ul style="list-style-type: none"> • P saw a play • P has less money • P is happy (optional if he liked the play) 	Scene 3: Going inside hall of theater and sitting on a seat <ul style="list-style-type: none"> • P PTRANS P into Hall of theater • TC ATTEND eyes on ticket POSS_by P • TC MTRANS (showed seat) to P • P PTRANS P to seat • P MOVES P to sitting position
	Scene 4: Watching a play <ul style="list-style-type: none"> • P ATTEND eyes on play • P MBUILD (good moments) from play Scene 5: Exiting <ul style="list-style-type: none"> • P PTRANS P out of Hall and theater



Chapter 5

Learning

7th Sem

BIM

AI

Learning

- Learning is one of those everyday terms which is broadly and vaguely used in the English language
 - Learning is **making useful changes** in our minds
 - Learning is **constructing or modifying representations** of what is being experienced
 - Learning is **the phenomenon of knowledge acquisition in the absence of explicit programming**

Herbert Simon, 1983

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 efficiently and more effectively next time.

- Learning involves the recognition of patterns in data or experience and then using that information to improve performance on another task.

Learning

- Learning involves 3 factors:

changes → Learning changes the learner: for machine learning the problem is **determining the nature of these changes and how to best represent them**

generalization → Learning leads to generalization: **performance must improve not only on the same task but on similar tasks**

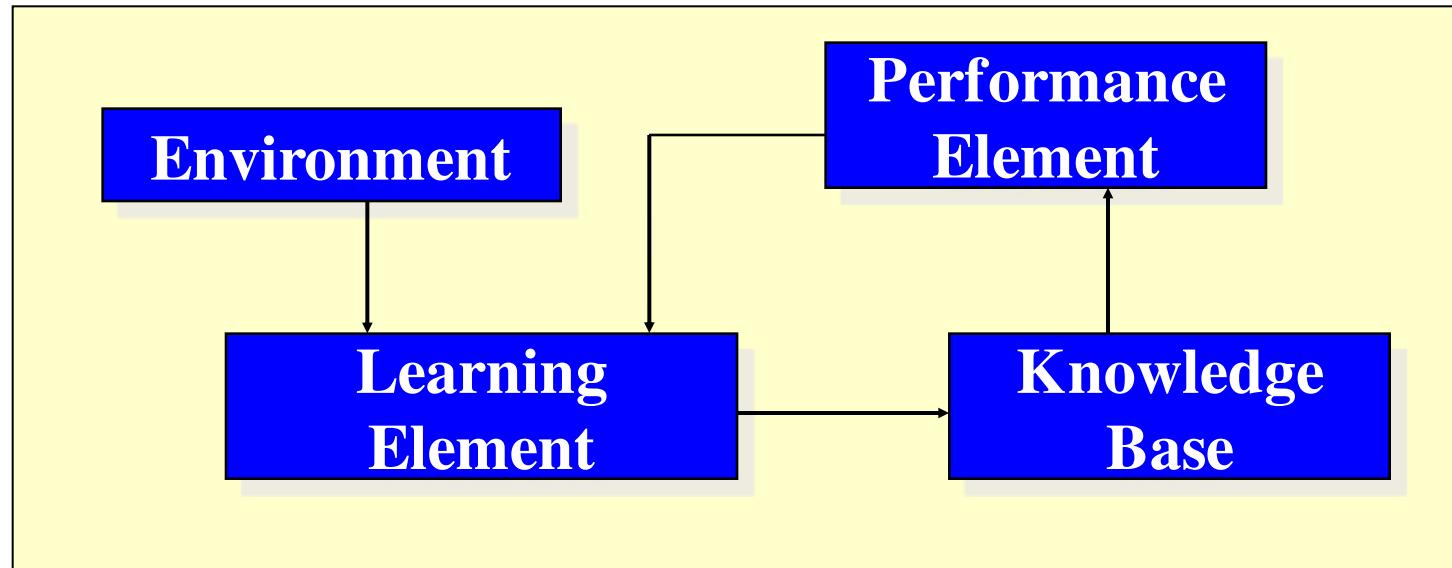
improvement → Learning leads to improvements: **machine learning must address the possibility that changes may degrade performance and find ways to prevent it.**

Learning Methods

- There are two different kinds of information processing which must be considered in a machine learning system
 - ***Inductive learning*** is concerned with determining general patterns, organizational schemes, rules, and laws from raw data, experience or examples.
 - ***Deductive learning*** is concerned with determination of specific facts using general rules or the determination of new general rules from old general rules.

Learning Framework

- There are four major components in a learning system:



Learning Framework: The Environment

- The environment refers the nature and quality of information given to the learning element
- The nature of information depends on its level
 - high level information is abstract, it deals with a broad class of problems
 - low level information is detailed, it deals with a single problem.
- The quality of information involves
 - noise free
 - reliable
 - ordered

Learning Framework: Learning Elements

- Four learning situations
 - Rote Learning
 - environment provides information at the required level
 - Learning by being told
 - information is too abstract, the learning element must hypothesize missing data
 - Learning by example
 - information is too specific, the learning element must hypothesize more general rules
 - Learning by analogy
 - information provided is relevant only to an analogous task, the learning element must discover the analogy

Learning Framework: Knowledge Base

- **Expressive**
 - the representation contains the relevant knowledge in an easy to get to fashion
- **Modifiable**
 - it must be easy to change the data in the knowledge base
- **Extendibility**
 - the knowledge base must contain meta-knowledge (knowledge on how the data base is structured) so the system can change its structure

Learning Framework: Performance Element

- Complexity
 - for learning, the simplest task is classification based on a single rule while the most complex task requires the application of multiple rules in sequence
- Feedback
 - the performance element must send information to the learning system to be used to evaluate the overall performance
- Transparency
 - the learning element should have access to all the internal actions of the performance element

Machine Learning

- Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data.
- The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension -- as is the case in data mining applications -- machine learning uses that data to detect patterns in data and adjust program actions accordingly.

Example

Facebook's News Feed uses machine learning to personalize each member's feed. If a member frequently stops scrolling in order to read or "like" a particular friend's posts, the News Feed will start to show more of that friend's activity earlier in the feed. Behind the scenes, the software is simply using statistical analysis and predictive analytics to identify patterns in the user's data and use those patterns to populate the News Feed.

Why Machine Learning?

- Complexity of task / amount of data

Other techniques fail or are computationally expensive

- Problems that cannot be defined

Discovery of patterns / data mining

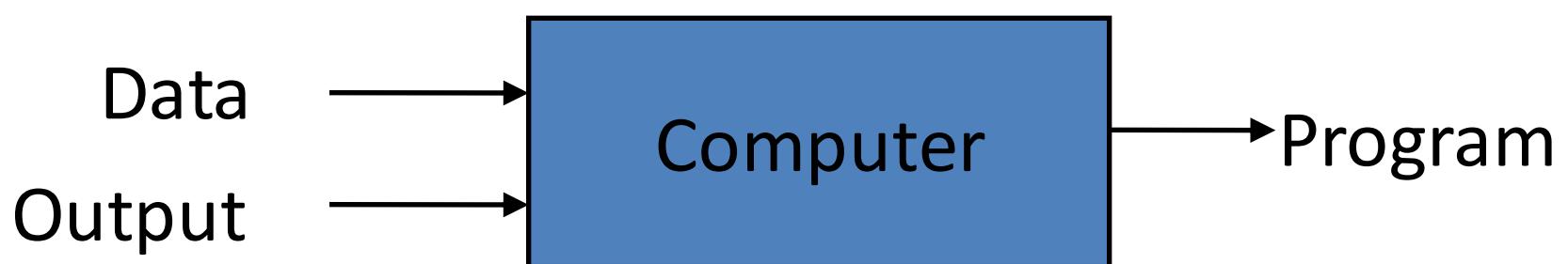
- Knowledge Engineering Bottleneck

'Cost and difficulty of building expert systems using traditional techniques'

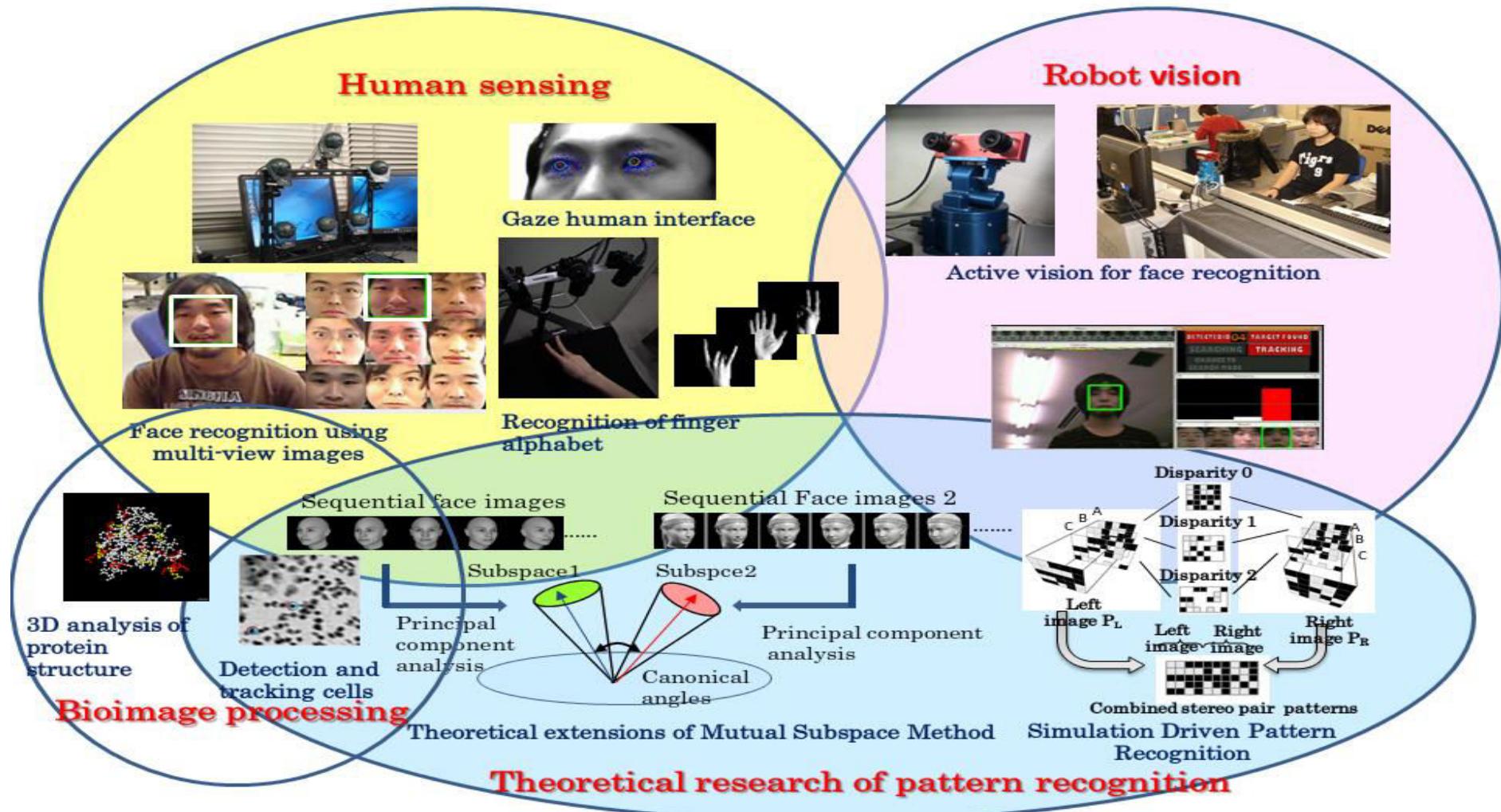
Traditional Programming



Machine Learning



Applications of ML



Examples of Machine Learning Problems

- Some of the examples of problems where Machine Learning can be used are:
 - **optical character recognition:** categorize images of handwritten characters by the letters represented
 - **face detection:** find faces in images (or indicate if a face is present)
 - **spam filtering:** identify email messages as spam or non-spam
 - **topic spotting:** categorize news articles (say) as to whether they are about politics, sports, entertainment, etc.
 - **spoken language understanding:** within the context of a limited domain, determine the meaning of something uttered by a speaker to the extent that it can be classified into one of a fixed set of categories

Explanation Based Learning (EBL)

- Humans appear to learn quite a lot from one example.
- Uses existing declarative domain knowledge to "explain" individual examples and uses this explanation to drive a knowledge-based generalization of the example.
- Produces a description of a concept that enables instances of the concept to be recognized efficiently.
- Capable of learning a very general concept from very few training examples.

Explanation Based Learning (EBL)

4 basic steps of EBL

- **A training example**
what the learning sees in the world.
- **A goal concept**
a high level description of what the program is supposed to learn.
- **A operational criterion**
a description of which concepts are usable.
- **A domain theory**
a set of rules that describe relationships between objects and actions in a domain.

Advantages/Disadvantages of EBL

- Advantages
 - needs just a small number of training examples
- Disadvantages
 - loss of coverage (EBL grammar is a subset of the original grammar)

Learning may be classified into three categories:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Supervised Learning

- **Supervised learning** is where you have input variables (X) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (Y) for that data.

$$Y = f(X)$$

- The term supervised learning refers to the fact that we use a data set in which the correct answers were given. The process of an algorithm learning from the training data set can be thought of as a teacher supervising the learning process. The algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

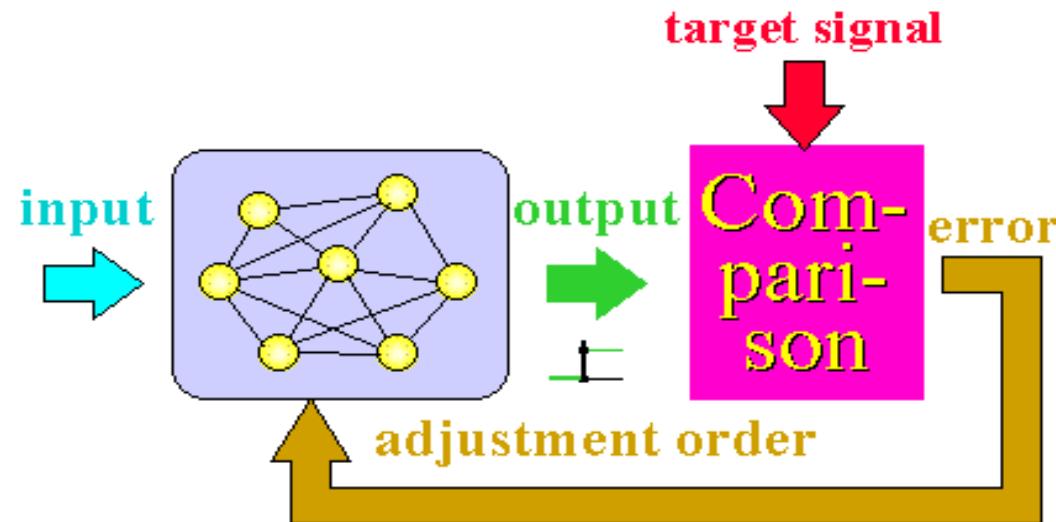
Supervised Learning

- Supervised learning problems can be further grouped into regression and classification problems.
 - **Classification:** A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
 - **Regression:** A regression problem is when the output variable is a real value, such as “dollars” or “weight”.
- One example is when you try to predict whether a tumour is malignant or benign according to the tumour size using a data set with tumour sizes and the type of tumour given to us.

Back propagation

- It is a **supervised learning** method
- It **requires** a teacher that knows, or can calculate, the desired output for any given input. It is most useful for **feed-forward networks**.
- The term is an abbreviation for "backwards propagation of errors".
- Back propagation networks are necessarily multilayer perceptrons (usually with one input, one hidden, and one output layer).

As the algorithm's name implies, the errors (and therefore the learning) propagate backwards from the output nodes to the inner nodes. So technically speaking, back propagation is used to calculate the gradient of the error of the network with respect to the network's modifiable weights.



Back propagation technique

- Randomly choose the initial weights
- While error is too large
 - For each training pattern (presented in random order)
 - ✗ Apply the inputs to the network
 - ✗ Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer
 - ✗ Calculate the error at the outputs
 - ✗ Use the output error to compute error signals for pre-output layers
 - ✗ Use the error signals to compute weight adjustments
 - ✗ Apply the weight adjustments
 - Periodically evaluate the network performance

Least Mean Square Error Supervised Training

The Least Mean Square (LMS) error algorithm is an example of supervised training, in which the learning rule is provided with a set of examples of desired network behavior:

$$\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_i, t_i\}.$$

Here, p_i is an input to the network, and t_i is the corresponding target output. As each input is applied to the network, the network output is compared to the target. The error is calculated as the difference between the target output and the network output. The aim is to minimize the average of the sum of these errors

$$MSE = \frac{1}{i} \cdot \sum_{i=1}^i (t(i) - a(i))^2$$

Unsupervised Learning

- **Unsupervised learning** is where **we only have input data (X)** and no corresponding output data.

The term unsupervised learning refers to the fact that the algorithm was provided with a data set in which we only specify the features of the instances while the class label information is not provided. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. Algorithms are left to their own devices to discover and present the interesting structure in the data.

- One way to do it is to divide the given instances in the data sets into clusters. When a new instance is given the learning algorithm can predict which cluster this new instance belongs to. This is known as clustering algorithm.

Unsupervised Learning

- Unsupervised learning problems can be further grouped into clustering and association problems.
 - **Clustering:** A clustering problem is where you want to **discover the inherent groupings in the data, such as grouping customers by purchasing behavior.**
 - **Association:** An association rule learning problem is **where we want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.**
- For example: Google news collects many new stories on the web and groups them together into different categories using this algorithm. This is done so that all stories about a single incident are grouped together.

[All](#) [Images](#) [News](#) [Videos](#) [Books](#) [More](#)[Settings](#) [Tools](#)

About 178,000,000 results (0.65 seconds)

[Hotels in Obama - Book Now with Expedia and Save](#)

www.expedia.com/Hotels/Obama_Fukui ▾

Find and Compare Exclusive Deals on Hotels in Obama and Save Big! New Expedia Rewards. Packages: Save up to 20% 600,000+ Hotels Worldwide. 11+ Million Hotel Reviews. Expedia's Best Prices. 24/7 Customer Support. Limited Time Offers. Save up to 50% on Hotels. Compare & Save. Secure Booking. Types: Boutique Hotels, Luxury Hotels, Airport Hotels, Hostels, Vacation Rentals.

[Expedia Rewards](#)

[Expedia® Daily Deals™](#)

[Book Hotel+Flight & Save](#)

[Hotels in Obama - Book Now with Expedia and Save](#)

www.expedia.com/Hotels/Obama_Fukui ▾

Find and Compare Exclusive Deals on Hotels in Obama and Save Big! New Expedia Rewards. Packages: Save up to 20% 600,000+ Hotels Worldwide. 11+ Million Hotel Reviews. Expedia's Best Prices. 24/7 Customer Support. Limited Time Offers. Save up to 50% on Hotels. Compare & Save. Secure Booking. Types: Boutique Hotels, Luxury Hotels, Airport Hotels, Hostels, Vacation Rentals.

[Expedia Rewards](#)

[Expedia® Daily Deals™](#)

[Book Hotel+Flight & Save](#)

[The Office of Barack and Michelle Obama](#)

<https://barackobama.com/> ▾

Welcome to the Office of Barack and Michelle Obama. We Love You Back. Play video. The Office of Barack and Michelle Obama · Obama Foundation. [obama](#).

[Barack Obama - Wikipedia](#)

https://en.wikipedia.org/wiki/Barack_Obama ▾

Barack Obama

44th U.S. President



barackobama.com

Barack Hussein Obama II is an American attorney and politician who served as the 44th president of the United States from 2009 to 2017. A member of the Democratic Party, he was the first African American to be elected to the presidency. He previously served as a U.S. senator from Illinois from 2005 to 2008. [Wikipedia](#)

Born: August 4, 1961 (age 57 years), Kapiolani Medical Center for Women & Children, Honolulu, Hawaii, United States

Height: 1.85 m

Party: Democratic Party

Education: Harvard Law School (1988–1991), [MORE](#)

Parents: Ann Dunham, Barack Obama Sr.

Books

[View 40+ more](#)



Hebbian Learning:

- Hebb's postulate of learning is :-

"When an axon of cell A is near enough to excite a cell B and repeatedly taking part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B is increased"

- **The weight between two neurons increases if the two neurons activate simultaneously—and reduces if they activate separately.**
- Nodes that tend to be either both positive or both negative at the same time have strong positive weights, while those that tend to be opposite have strong negative weights.

Hebb's Algorithm:

Step 0: initialize all weights to 0

Step 1: Given a training input, s , with its target output, t , set the activations of the input units: $x_i = s_i$

Step 2: Set the activation of the output unit to the target value: $y = t$

Step 3: Adjust the weights: $w_i(\text{new}) = w_i(\text{old}) + x_i y$

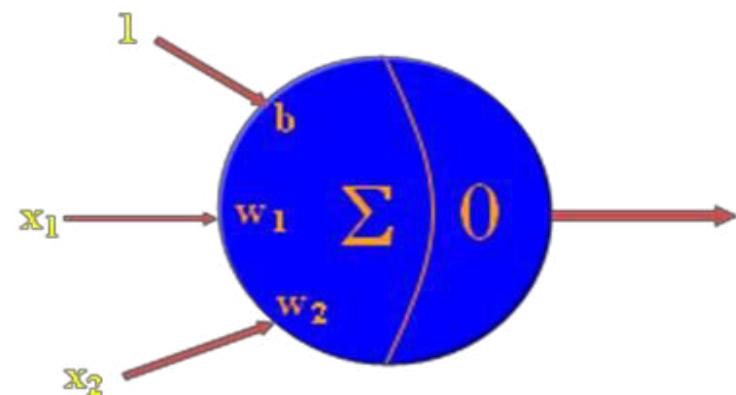
Step 4: Adjust the bias (just like the weights): $b(\text{new}) = b(\text{old}) + y$

(bias: the difference between the expectation of sample estimator and true value)

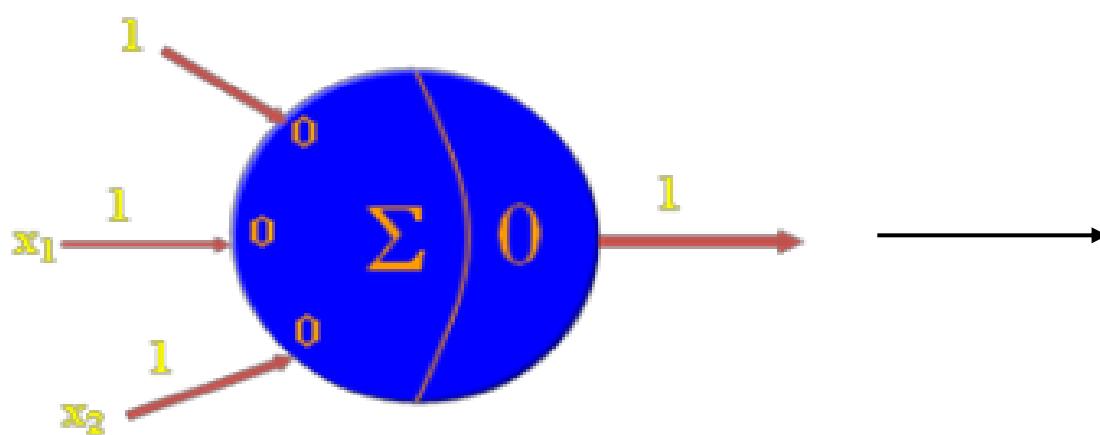
Construct a Hebb Net which performs like an AND function, that is, only when both features are “active” will the data be in the target class.

TRAINING SET (with the bias input always at 1):

x1	x2	bias	Target
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1



- › Initialize the weights to 0
- › Present the first input (1 1 1) with target of 1



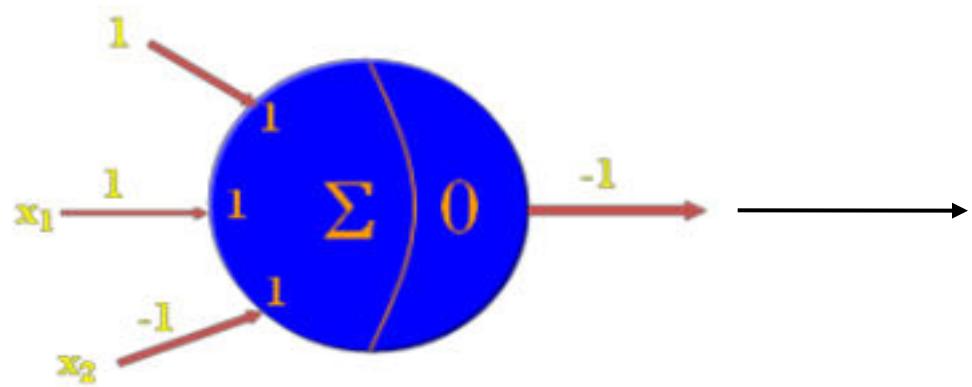
Update the weights:

$$w_1(\text{new}) = w_1(\text{old}) + x_1 t \\ = 0 + 1 = 1$$

$$w_2(\text{new}) = w_2(\text{old}) + x_2 t \\ = 0 + 1 = 1$$

$$b(\text{new}) = b(\text{old}) + t \\ = 0 + 1 = 1$$

- Present the second input $(1 \ -1 \ 1)$ with target of -1

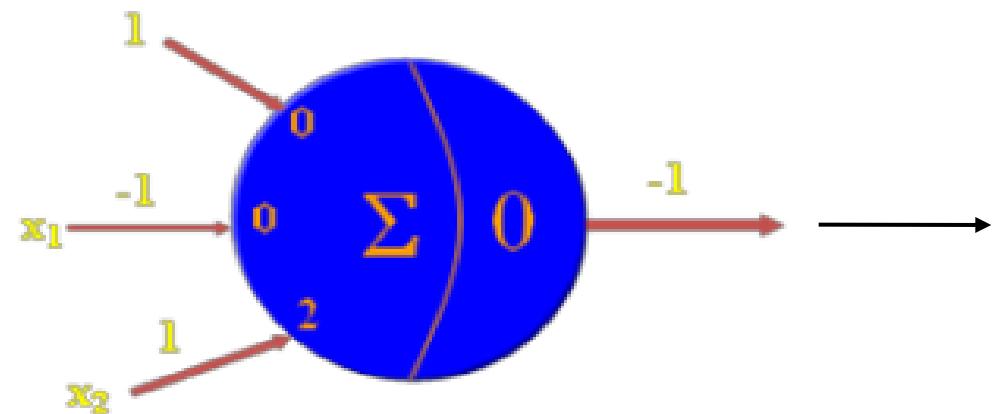


Update the weights:

$$\begin{aligned}
 w_1(\text{new}) &= w_1(\text{old}) + x_1 t \\
 &= 1 + 1(-1) = 0 \\
 w_2(\text{new}) &= w_2(\text{old}) + x_2 t \\
 &= 1 + (-1)(-1) = 2 \\
 b(\text{new}) &= b(\text{old}) + t \\
 &= 1 + (-1) = 0
 \end{aligned}$$

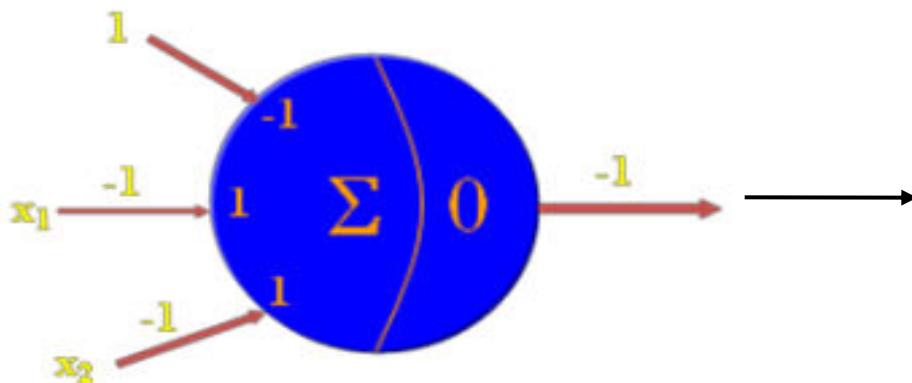
- Present the third input $(-1 \ 1 \ 1)$ with target of -1

Update the weights:



$$\begin{aligned}
 w_1(\text{new}) &= w_1(\text{old}) + x_1 t \\
 &= 0 + (-1)(-1) = 1 \\
 w_2(\text{new}) &= w_2(\text{old}) + x_2 t \\
 &= 2 + 1(-1) = 1 \\
 b(\text{new}) &= b(\text{old}) + t \\
 &= 0 + (-1) = -1
 \end{aligned}$$

- Present the Fourth input $(-1 \ -1 \ 1)$ with target of -1



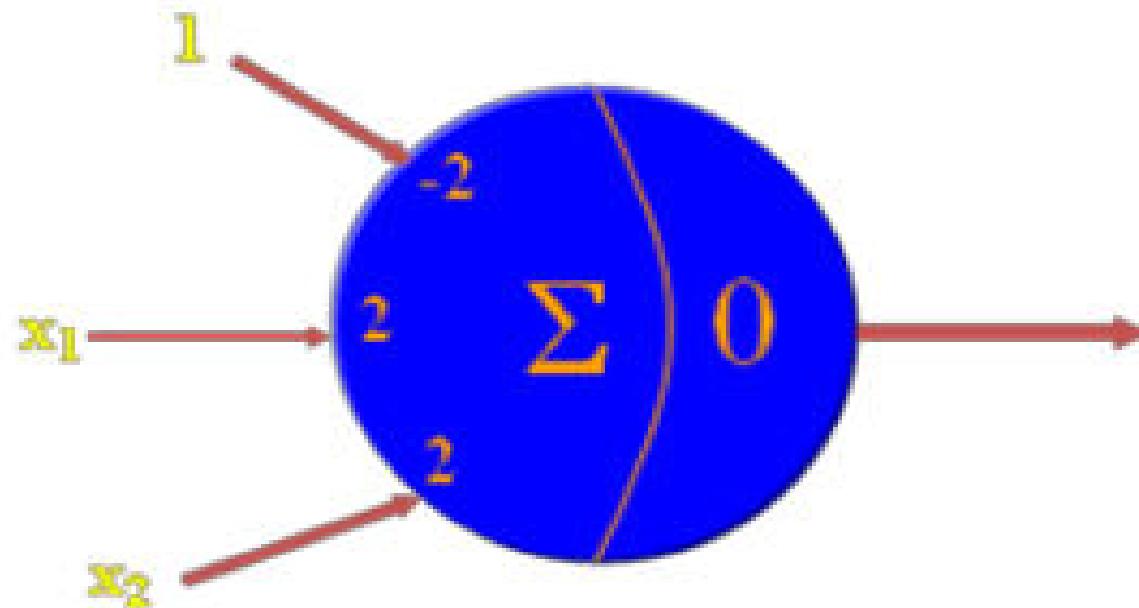
Update the weights:

$$\begin{aligned}w_1(\text{new}) &= w_1(\text{old}) + x_1 t \\&= 1 + (-1)(-1) = 2\end{aligned}$$

$$\begin{aligned}w_2(\text{new}) &= w_2(\text{old}) + x_2 t \\&= 1 + (-1)(-1) = 2\end{aligned}$$

$$\begin{aligned}b(\text{new}) &= b(\text{old}) + t \\&= -1 + (-1) = -2\end{aligned}$$

The Final Neuron

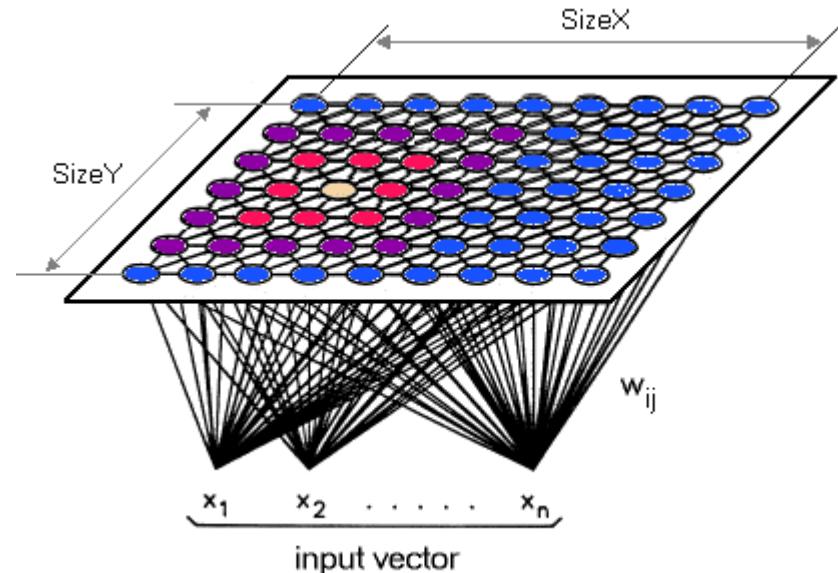


Competitive Learning

1. In competitive learning, neurons compete among themselves to be activated.
2. While in Hebbian learning, several output neurons can be activated simultaneously, in competitive learning, only a single output neuron is active at any time.
3. The output neuron that wins the “competition” is called the winner-takes-all neuron.
4. Self-organising feature maps are based on competitive learning.

Reducing Data Dimensions

Unlike other learning technique, training a SOM requires no target vector. A SOM learns to classify the training data without any external supervision.



Learning occurs in several steps and over many iterations:

1. Each node's weights are initialized.
2. A vector is chosen at random from the set of training data.
3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).
4. Then the neighbourhood of the BMU is calculated. The amount of neighbors decreases over time.
5. The winning weight is rewarded with becoming more like the sample vector.
6. Repeat step 2 for N iterations.

Reinforcement Learning

- Reinforcement learning (RL) provide the learning agent with a reward function and let it figure out the best strategy for obtaining large rewards.
- **Reinforcement learning** is training by rewards and punishments. Here we train a computer as if we train a dog. If the dog obeys and acts according to our instructions we encourage it by giving biscuits or we punish it by beating or by scolding.
- Similarly, if the system works well then the teacher (environment) gives positive value (i.e. reward) or the teacher gives negative value (i.e. punishment). The learning system which gets the punishment has to improve itself.

Reinforcement Learning

- Thus it is a trial and error process.
- The *reinforcement learning algorithms* selectively retain the outputs that maximize the received reward over time.
- To accumulate a lot of rewards, the learning system must prefer the best experienced actions; however, it has to try new actions in order to discover better action selections for the future.
- Examples are learning to play games, robot control, elevator control, network routing, and animal learning.

Steps for Reinforcement Learning

- The agent observes an input state
- An action is determined by a decision making function (policy)
- The action is performed
- The agent receives a scalar reward or reinforcement from the environment
- Information about reward given for that state/action pair is recorded

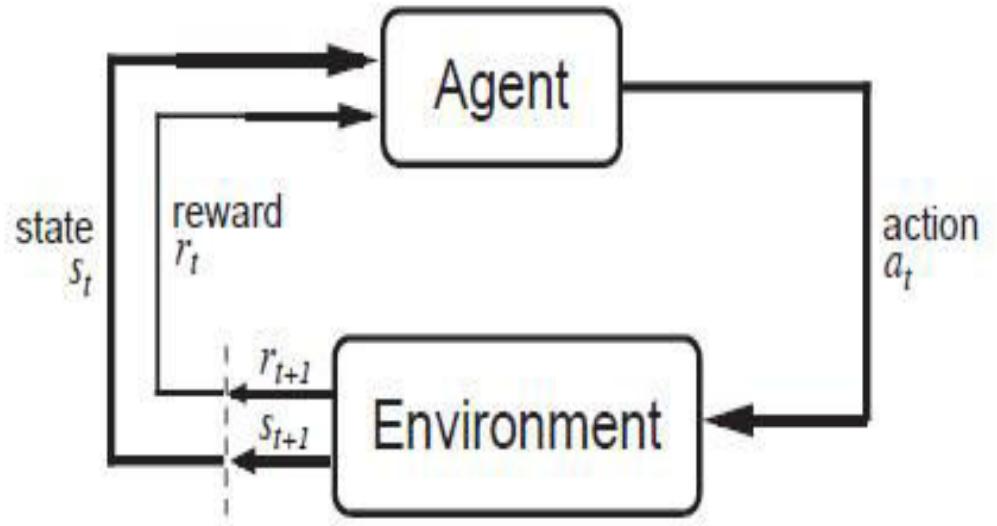


Figure: Agent Environment interface in Reinforcement Learning

Example of Reinforcement Learning

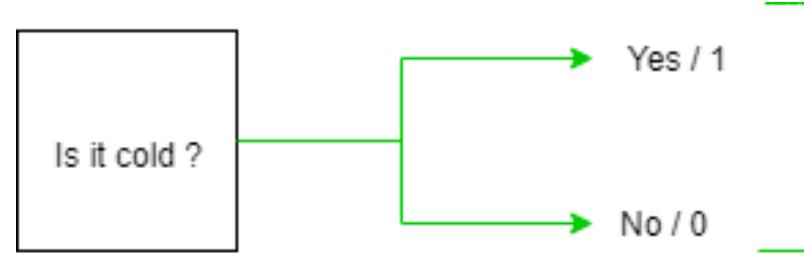
- **Personalization Shopping Support System** is one software that is designed to provide product information as per the user's interests.
 - It applies the reinforcement learning to analyze and learn customer behaviors and list out the products that the customers wish to buy.
 - If the system selects the right item that the customer wish to buy then it is given reward by assigning a particular value for the state and if the system selects an item which the user does not wish to buy then it is given the penalty.
 - This way the system learns the personal interests. In this process, the system acquires the knowledge of the user behavior and interest which makes it decide which information should be given to a particular user.
 - This results in greater customer satisfaction and increase in the success rate of product promotion.

Fuzzy Learning

The term **fuzzy** refers to things **which are not clear or are vague**. In the real world many times we encounter a situation when we can't determine whether the state is true or false, their fuzzy logic provides a very valuable flexibility for reasoning.

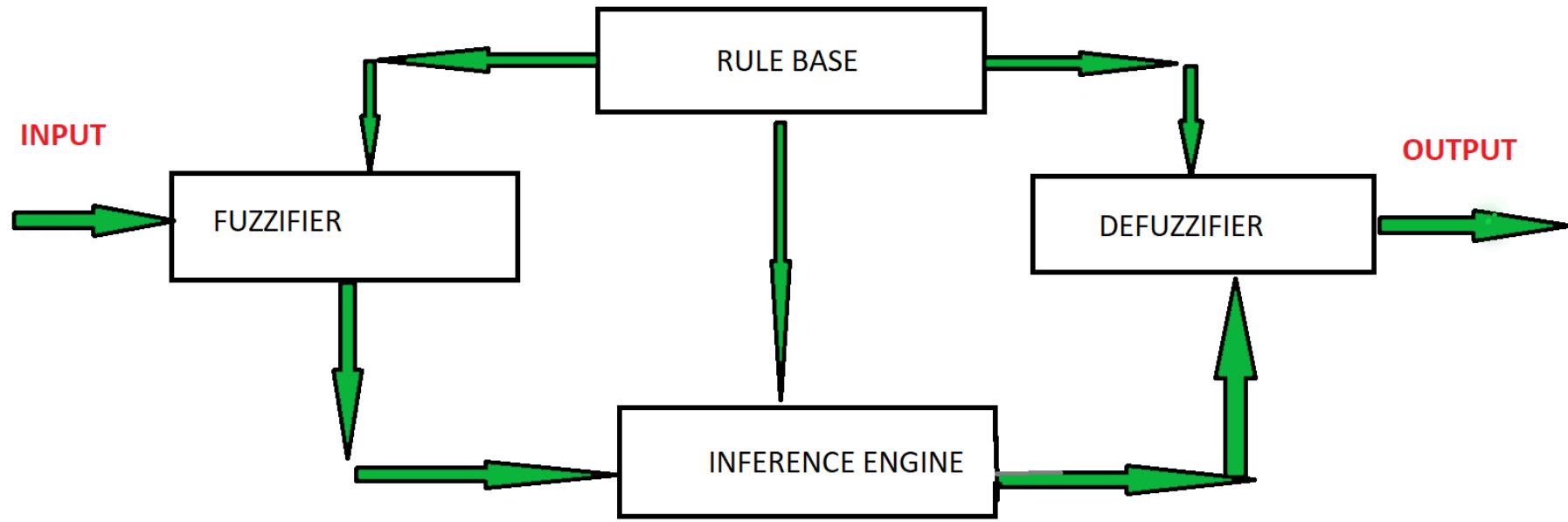
In this way, we can consider the inaccuracies and uncertainties of any situation.

In Boolean system truth value, 1.0 represents absolute truth value and 0.0 represents absolute false value.



In the fuzzy system, there is no logic for absolute truth and absolute false value. But in fuzzy logic, there is intermediate value too present which is partially true and partially false.





FUZZY LOGIC ARCHITECTURE

RULE BASE:

It contains the set of rules and the IF-THEN conditions provided by the experts to govern the decision making system, on the basis of linguistic information.

FUZZIFICATION:

It is used to convert inputs i.e. crisp numbers into fuzzy sets. Crisp inputs are basically the exact inputs measured by sensors and passed into the control system for processing, such as temperature, pressure, rpm's, etc.

INFERENCE ENGINE:

It determines the matching degree of the current fuzzy input with respect to each rule and decides which rules are to be fired according to the input field.

DEFUZZIFICATION:

It is used to convert the fuzzy sets obtained by inference engine into a crisp value.

Application

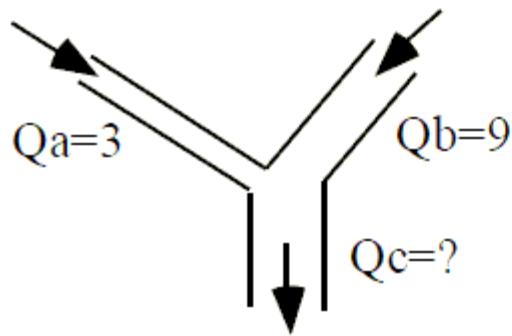
- ❑ It is used in the aerospace field for altitude control of spacecraft and satellite.
- ❑ It has used in the automotive system for speed control, traffic control.
- ❑ It is used for decision making support systems and personal evaluation in the large company business.

LEARNING FROM EXAMPLES

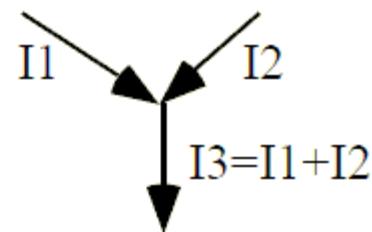
- This involves the process of **inductive learning**-- where a system tries to induce a general rule from a set of observed instances.
- This involves **classification** -- assigning, to a particular input, the name of a class to which it belongs. Classification is important to many problem solving tasks.
- A learning system has to be capable of evolving its own class descriptions:
 - Initial class definitions may not be adequate.
 - The world may not be well understood or rapidly changing.
- The task of constructing class definitions is called **induction** or **concept learning**
- Example: ID3 Algorithm

Learning by analogy

Learning by analogy means acquiring new knowledge about an input entity by transferring it from a known similar entity



Simple Hydraulics Problem



Kirchoff's First Law

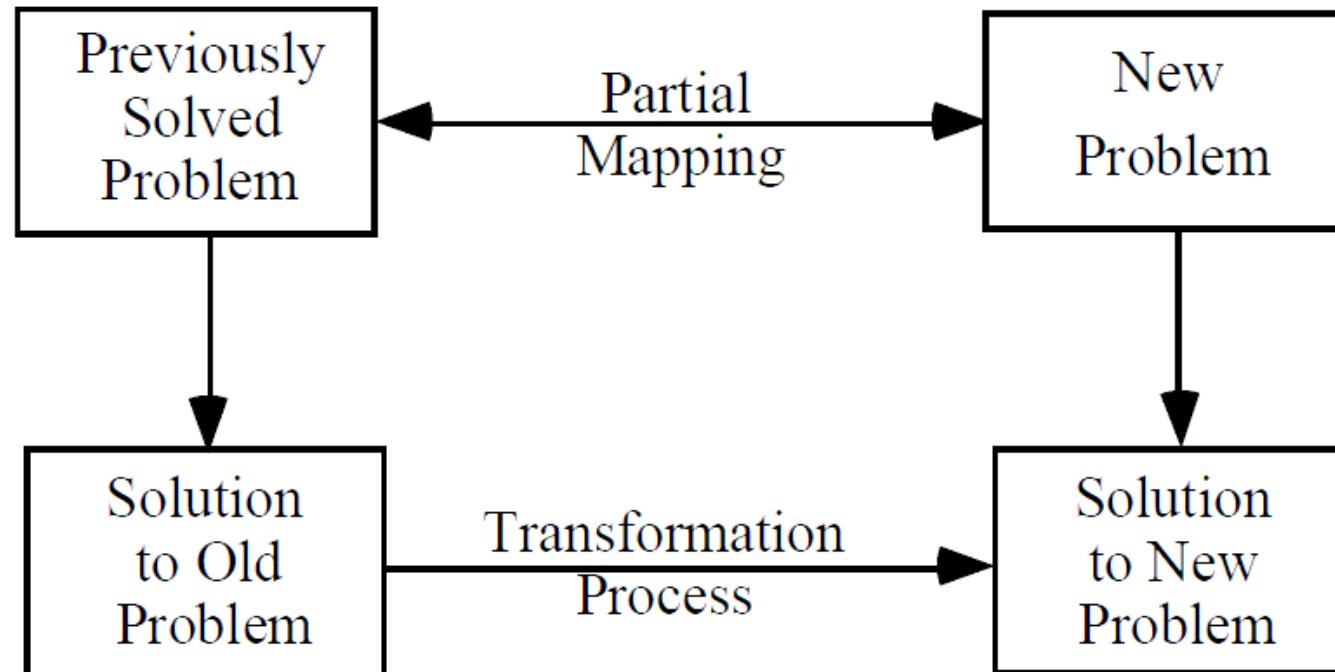
One may infer, by analogy, that hydraulics laws are similar to Kirchhoff's laws

Problem solving by analogy

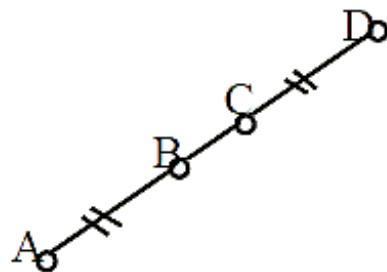
Analogy means deriving new knowledge about an input entity by transferring it from a known similar entity.

How could we define problem solving by analogy?

Problem solving by analogy is the process of transferring knowledge from past problem-solving episodes to new problems that share significant aspects with corresponding past experience and using the transferred knowledge to construct solutions to the new problems.



GIVEN: $AB = CD$
PROVE: $AC = BD$



$$AB = CD$$

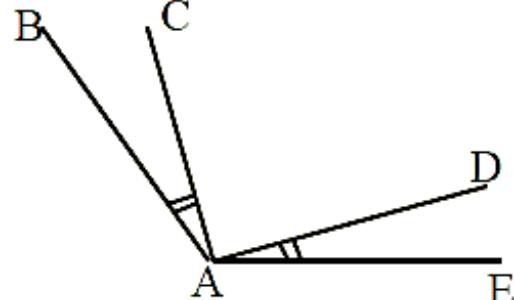
$$BC = BC$$

$$AB + BC = BC + CD$$

$$AC = BD$$

$$\sigma = (AB <- \angle BAC
CD <- \angle DAE
AC <- \angle BAD
BD <- \angle CAE)$$

GIVEN: $\angle BAC = \angle DAE$
PROVE: $\angle BAD = \angle CAE$



$$\angle BAC = \angle DAE$$

$$\angle CAD = \angle CAE$$

$$\angle BAC + \angle CAD = \angle CAD + \angle DAE$$

$$\angle BAD = \angle CAE$$

Learning By Simulated Evolution

- Evolving a solution
- Begin with **population of individuals**
 - Individuals = candidate solutions ~chromosomes
- Produce **offspring** with variation
 - Mutation: change features
 - Crossover: exchange features between individuals
- Apply **natural selection**
 - Select “best” individuals to go on to next generation Continue until satisfied with solution

Genetic Algorithm

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection.

It is frequently used to solve optimization problems, in research, and in machine learning.

Optimization is the process of making something better.

Optimization refers to finding the values of inputs in such a way that we get the “best” output values. The definition of “best” varies from problem to problem, but in mathematical terms, it refers to maximizing or minimizing one or more objective functions, by varying the input parameters.

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution.

This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

The process of natural selection starts with the **selection of fittest individuals** from a population.

They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a **generation with the fittest individuals will be found**.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Five phases are considered in a genetic algorithm.

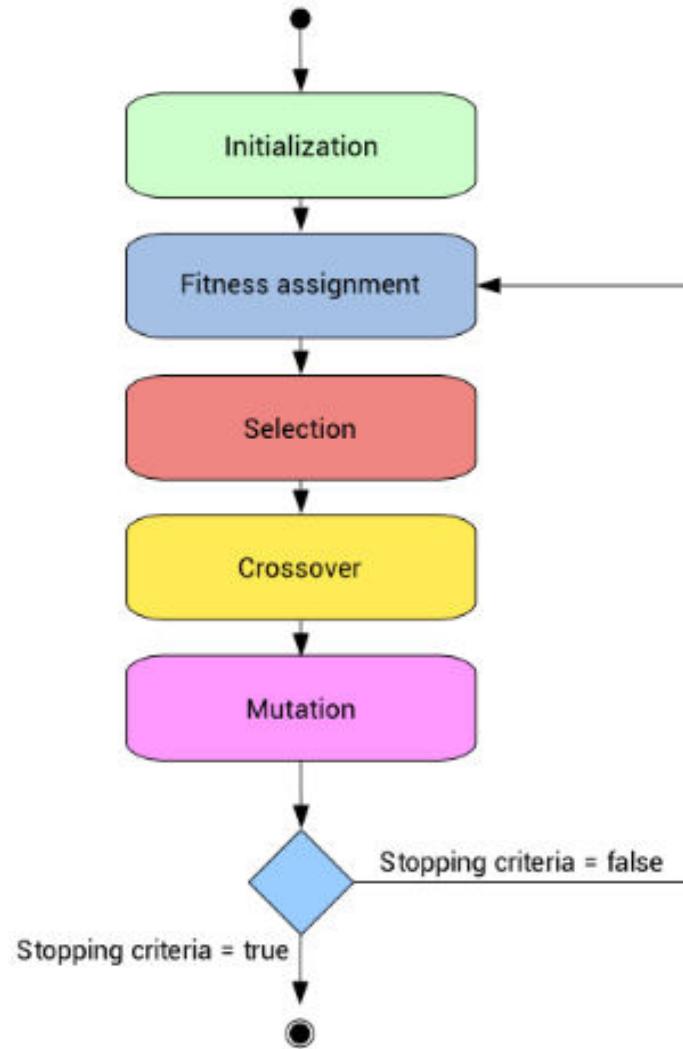
1. Initial population

2. Fitness function

3. Selection

4. Crossover

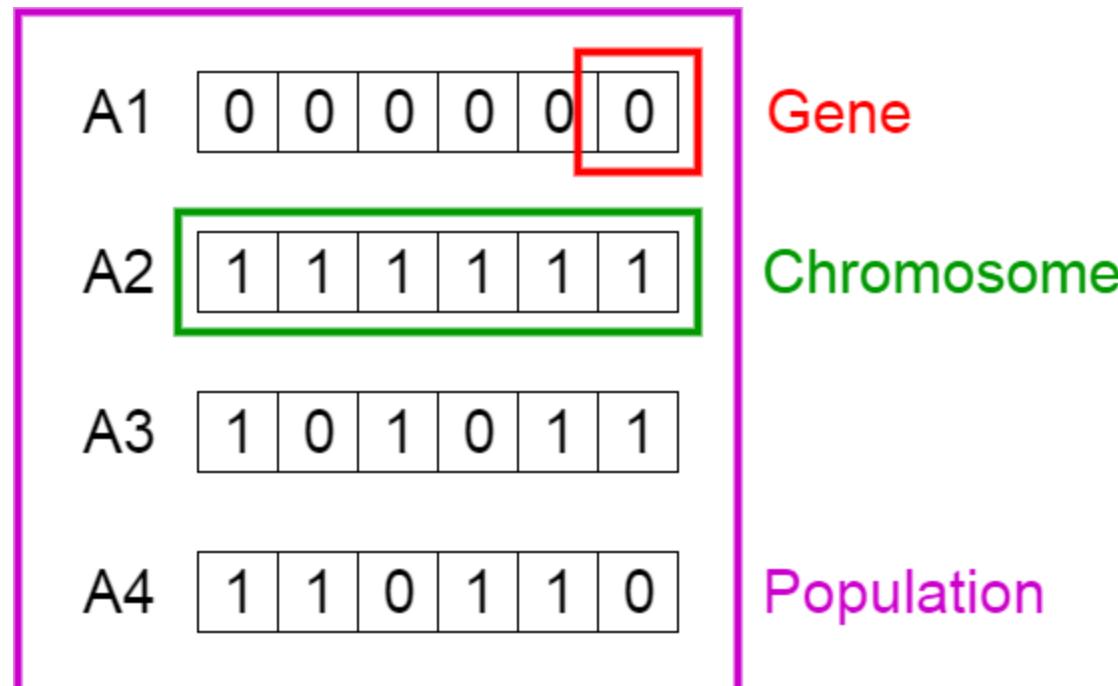
5. Mutation



Initial Population

The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem we want to solve.

An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution).



Fitness Function

The fitness function determines **how fit an individual is** (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

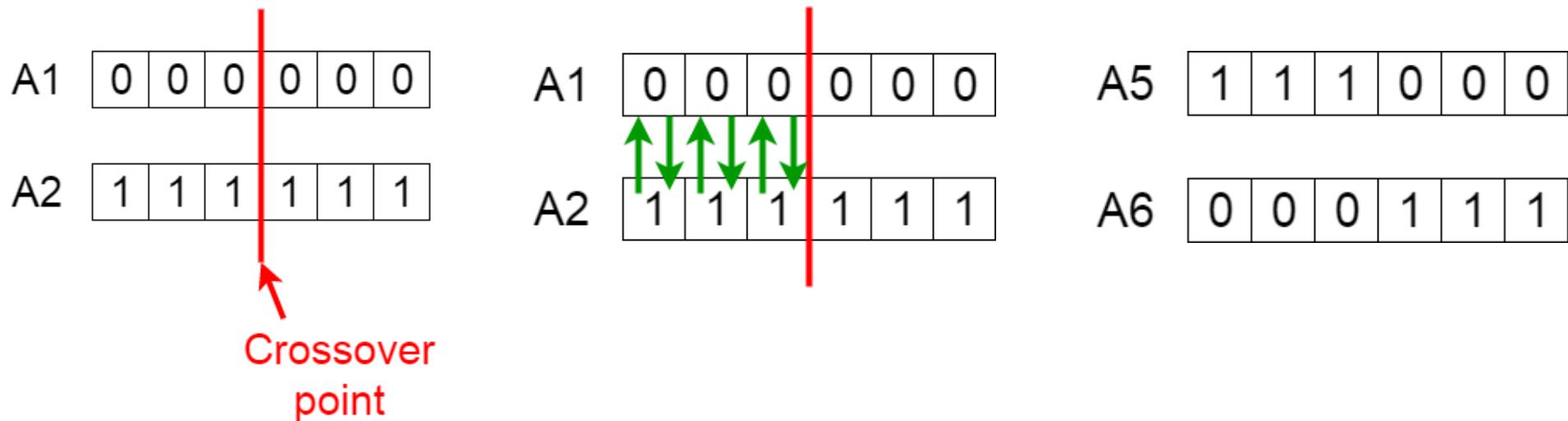
Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation.

Two pairs of individuals (parents) are selected based on their fitness scores.
Individuals with high fitness have more chance to be selected for reproduction.

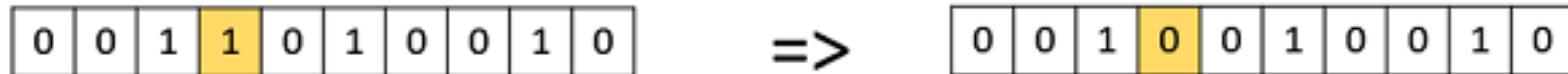
Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.



Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.



Termination

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation).

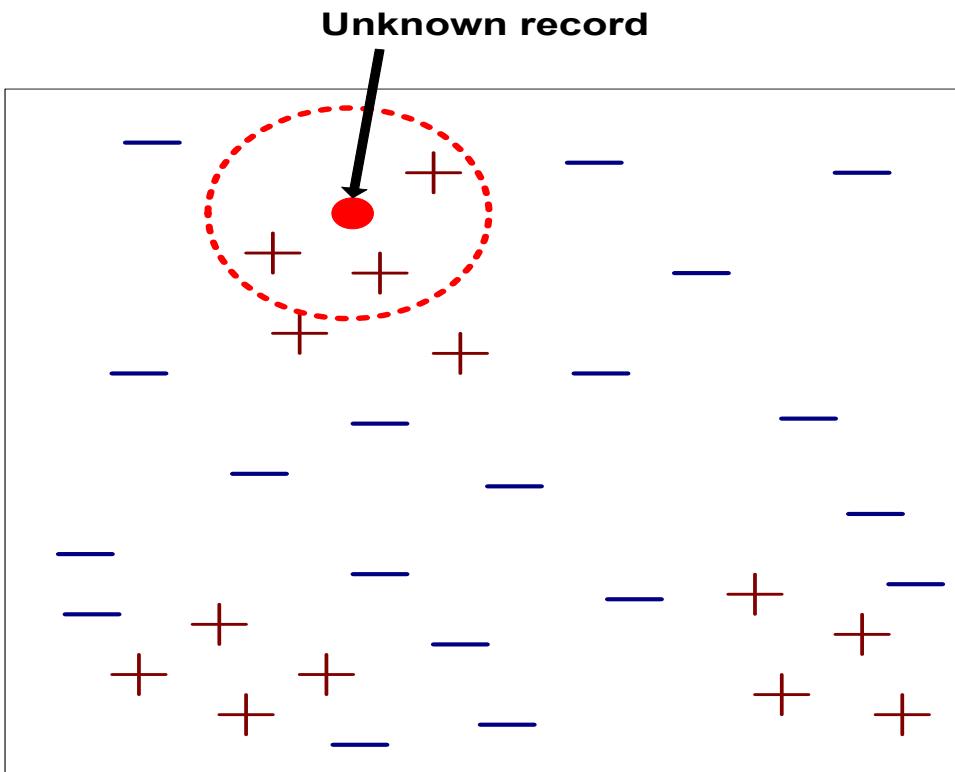
Then it is said that the genetic algorithm has provided a set of solutions to our problem.

If you want to **READ** extra

K-Nearest Neighbor (KNN)

- KNN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (usually Euclidean Distance)
- It can be used both for classification and regression
- KNN is a “**non parametric**” “**lazy learning**” algorithm
 - Non parametric means that it does not make any assumptions on the underlying data distribution
 - Lazy algorithm because there is zero effort at training time and all the effort at prediction time.

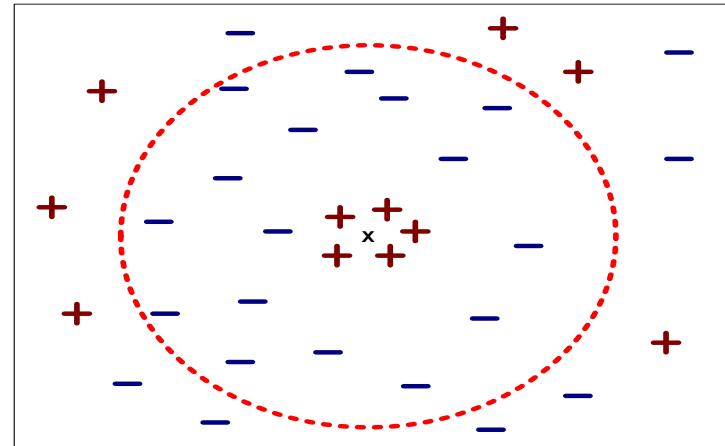
K-Nearest Neighbor Classifiers

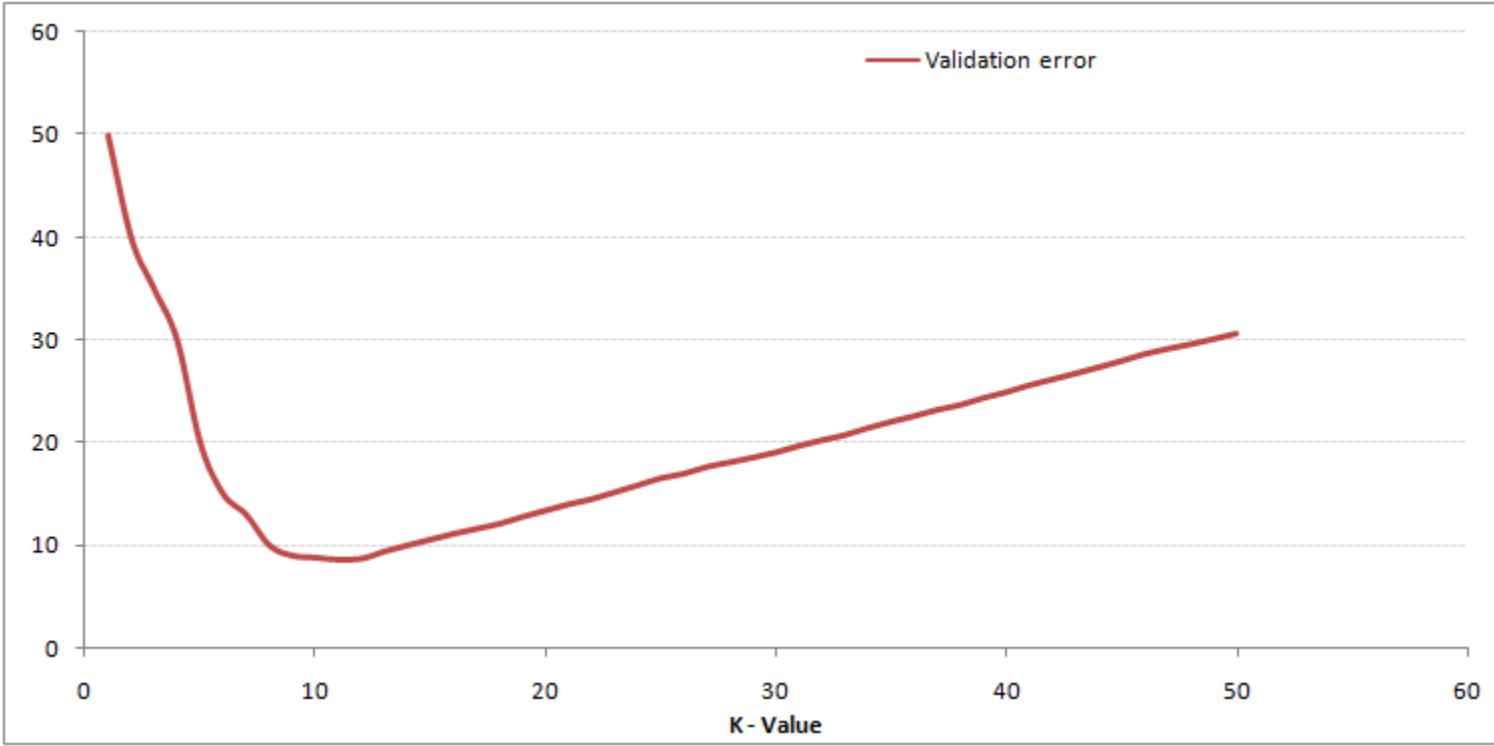


- Requires **three** things
 - The set of stored records
 - Distance Metric to compute distance between records
 - The value of k , the number of nearest neighbors to retrieve
 - To classify an unknown record:
 - Compute distance (Euclidean distance) to other training records
 - Identify k nearest neighbors
 - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

K-Nearest Neighbor Classification

- Choosing the value of k:
 - If k is too small, sensitive to noise points
 - If k is too large, neighborhood may include points from other classes





To get the optimal value of K, we can separate the training and validation from the initial data set and plot the validation error curve to get the optimal value of K

Suppose we have height, weight and T-shirt size of some customers.

We need to predict the T-shirt size of a *new customer* given only height (161cm) and weight(61kg)

Euclidean :

$$d(x, y) = \sqrt{\sum_{i=1}^m (x_i - y_i)^2}$$

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Let k be 5.

Then the algorithm searches for the 5 customers closest to *new customer*, i.e. most similar to new customer in terms of attributes, and see what categories those 5 customers were in. If 4 of them had ‘Medium T shirt sizes’ and 1 had ‘Large T shirt size’ then your best guess for new customer is ‘Medium T shirt.

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	158	58	M	4.2	
3	158	59	M	3.6	
4	158	63	M	3.6	
5	160	59	M	2.2	3
6	160	60	M	1.4	1
7	163	60	M	2.2	3
8	163	61	M	2.0	2
9	160	64	L	3.2	5
10	163	64	L	3.6	
11	165	61	L	4.0	
12	165	62	L	4.1	
13	165	65	L	5.7	
14	168	62	L	7.1	
15	168	63	L	7.3	
16	168	66	L	8.6	
17	170	63	L	9.2	
18	170	64	L	9.5	
19	170	68	L	11.4	
20					
21	161	61			

Naive Bayes

- Naive Bayes is a **probabilistic machine learning algorithm based on the Bayes Theorem**, used in a wide variety of classification tasks.
- Typical applications include filtering spam, classifying documents, sentiment prediction etc.
- Naive (showing a lack of experience, wisdom, or judgement) is named so because **it assumes the features that go into the model is independent of each other**. That is changing the value of one feature, does not directly influence or change the value of any of the other features used in the algorithm.

There are two parts to this algorithm:

- **Naïve**

- independence

- **Bayes**

- Bayesian Theorem

What is Bayes Theorem?

In Statistics and probability theory, Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event. It serves as a way to figure out conditional probability.

Given a Hypothesis H and evidence E , Bayes' Theorem states that the relationship between the probability of Hypothesis before getting the evidence $P(H)$ and the probability of the hypothesis after getting the evidence $P(H|E)$ is :

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

This relates the probability of the hypothesis before getting the evidence $P(H)$, to the probability of the hypothesis after getting the evidence, $P(H|E)$. For this reason, $P(H)$ is called the **prior probability**, while $P(H|E)$ is called the **posterior probability**. The factor that relates the two, $P(H|E) / P(H)$, is called the **likelihood ratio**. Using these terms, Bayes' theorem can be rephrased as:

“The posterior probability equals the prior probability times the likelihood ratio.”

Game Prediction using Bayes' Theorem

we have our Data, which comprises of the Day, Outlook, Humidity, Wind Conditions and the final column being Play, which we have to predict.

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

First, we will create a **frequency** table using each attribute of the dataset.

Frequency Table		Play	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	3	2

Frequency Table		Play	
		Yes	No
Humidity	High	3	4
	Normal	6	1

Frequency Table		Play	
		Yes	No
Wind	Strong	6	2
	Weak	3	3

For each frequency table, we will generate
a li

Likelihood Table		Play		5/14
		Yes	No	
Outlook	Sunny	3/10	2/4	5/14
	Overcast	4/10	0/4	4/14
	Rainy	3/10	2/4	5/14
		10/14	4/14	

$P(\text{Sunny} | \text{Yes}) = 3/10 = 0.3$

$P(\text{Sunny}) = 5/14 = 0.36$

$P(\text{Yes}) = 10/14 = 0.71$

Likelihood of ‘Yes’ given ‘Sunny’ is:

$$P(\text{Yes} | \text{Sunny}) = P(\text{Sunny} | \text{Yes}) * P(\text{Yes}) / P(\text{Sunny}) = (0.3 \times 0.71) \\ / 0.36 = 0.591$$

Similarly Likelihood of ‘No’ given ‘Sunny’ is:

$$P(\text{No} | \text{Sunny}) = P(\text{Sunny} | \text{No}) * P(\text{No}) / P(\text{Sunny}) = (0.4 \times 0.36) \\ / 0.36 = 0.40$$

Likelihood table for Humidity

Likelihood Table		Play		
		Yes	No	
Humidity	High	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

$$P(\text{Yes}|\text{High}) = 0.33 \times 0.6 / 0.5 = 0.42$$

$$P(\text{No}|\text{High}) = 0.8 \times 0.36 / 0.5 = 0.58$$

Likelihood table for Wind

Likelihood Table		Play		
		Yes	No	
Wind	Weak	6/9	2/5	8/14
	Strong	3/9	3/5	6/14
		9/14	5/14	

$$P(\text{Yes}|\text{Weak}) = 0.67 \times 0.64 / 0.57 = 0.75$$

$$P(\text{No}|\text{Weak}) = 0.4 \times 0.36 / 0.57 = 0.25$$

Suppose we have a **Day** with the following values :

- **Outlook** = Rain
- **Humidity** = High
- **Wind** = Weak
- **Play** =?

So, with the data, we have to predict whether “we can play on that day or not”.

$$\begin{aligned}\text{Likelihood of 'Yes' on that Day} &= P(\text{Outlook} = \text{Rain} | \text{Yes}) * P(\text{Humidity} = \text{High} | \text{Yes}) * P(\text{Wind} = \\ &\quad \text{Weak} | \text{Yes}) * P(\text{Yes}) \\ &= 2/9 * 3/9 * 6/9 * 9/14 = \mathbf{0.0199}\end{aligned}$$

$$\begin{aligned}\text{Likelihood of 'No' on that Day} &= P(\text{Outlook} = \text{Rain} | \text{No}) * P(\text{Humidity} = \text{High} | \text{No}) * P(\text{Wind} = \\ &\quad \text{Weak} | \text{No}) * P(\text{No}) \\ &= 2/5 * 4/5 * 2/5 * 5/14 = \mathbf{0.0166}\end{aligned}$$

Now we normalize the values, then

$$\begin{aligned}P(\text{Yes}) &= 0.0199 / (0.0199 + 0.0166) = \mathbf{0.55} \\ P(\text{No}) &= 0.0166 / (0.0199 + 0.0166) = \mathbf{0.45}\end{aligned}$$

Our model predicts that there is a **55%** chance there will be a Game tomorrow.

Naive Bayes in the Industry

- News Categorization
- Spam Filtering
- Weather Forecasting
- Medical Diagnosis

Logistic regression

(Regression is a measure of the relation between the mean value of one variable (e.g. output) and corresponding values of other variables)

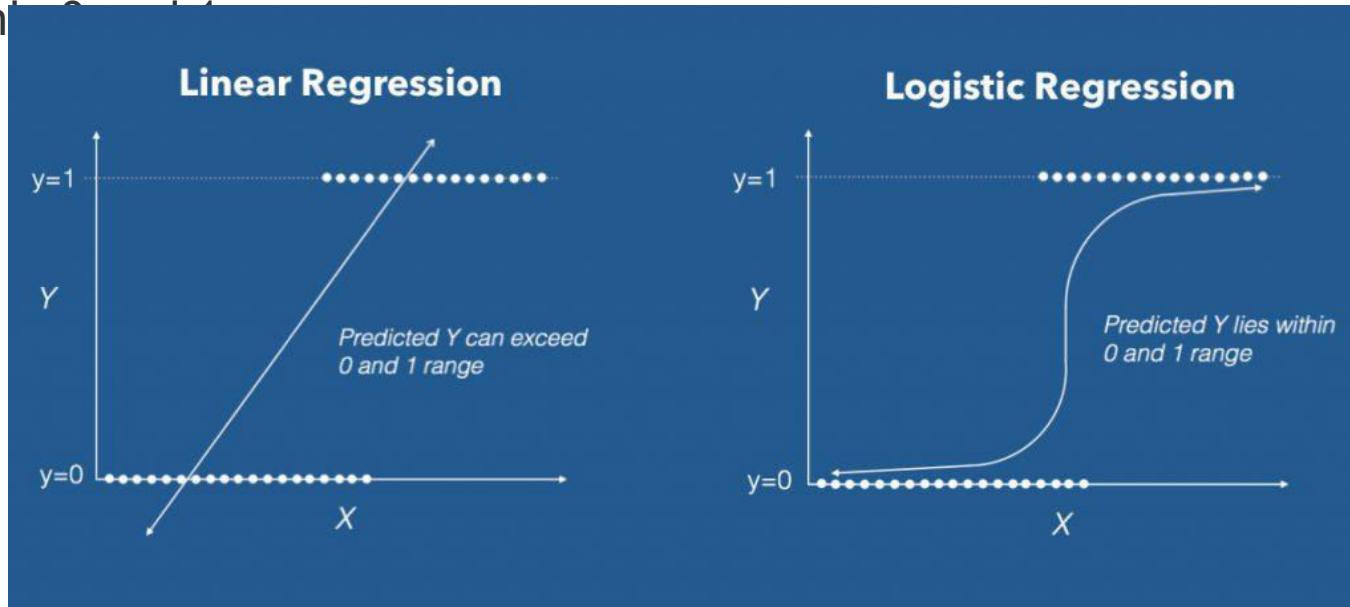
Logistic regression is a **predictive modelling algorithm** that is used when the Y variable is binary categorical. That is, it can take only two values like 1 or 0.

The goal is to determine a mathematical equation that can be used to predict the probability of event 1. Once the equation is established, it can be used to predict the Y when only the X's are known.

Why not linear regression?

When the response variable has only 2 possible values, it is desirable to have a model that predicts the value either as 0 or 1 or as a probability score that ranges between 0 and 1.

Linear regression does *not* have this capability. Because, If we use linear regression to model a binary response variable, the resulting model may not restrict the predicted Y values within 0 and 1.



Earlier we saw what is linear regression and how to use it to predict continuous Y variables.

In linear regression the Y variable is always a continuous variable. If suppose, the Y variable was categorical, you cannot use linear regression model it.

So what would you do when the Y is a categorical variable with 2 classes?

Logistic regression can be used to model and solve such problems, also called as binary classification problems.

A key point to note here is that Y can have 2 classes only and not more than that. If Y has more than 2 classes, it would become a multi class classification and you can no longer use the vanilla logistic regression for that.

Yet, Logistic regression is a classic predictive modelling technique and still remains a popular choice for modelling binary categorical variables.

Another advantage of logistic regression is that it computes a prediction probability score of an event. More on that when you actually start building the models.

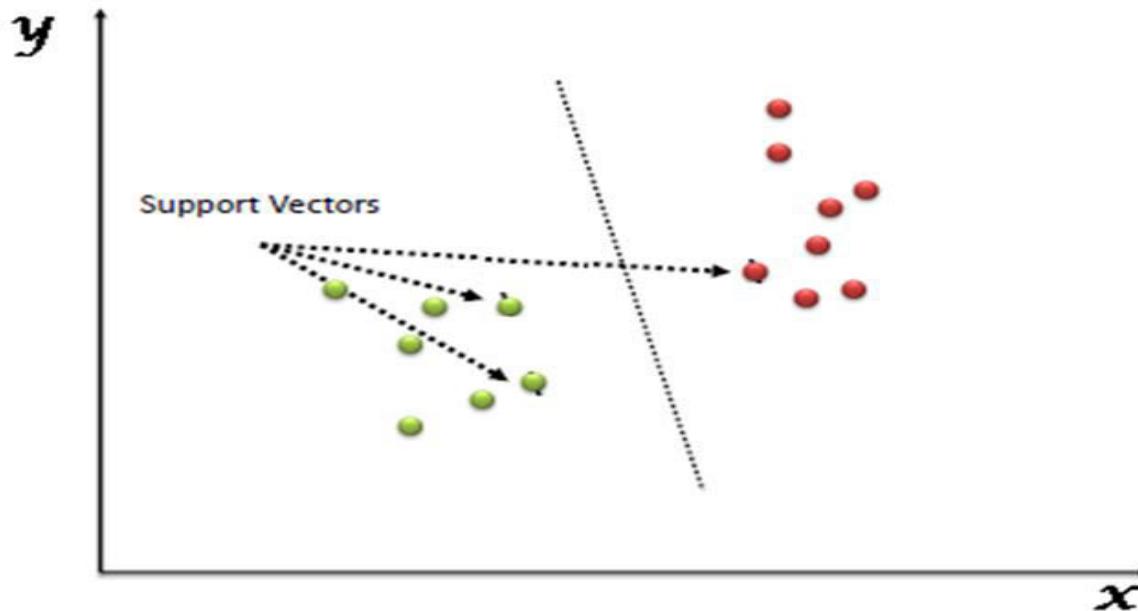
Here are some examples of binary classification problems:

- **Spam Detection** : Predicting if an **email** is **Spam** or not
- **Credit Card Fraud** : Predicting if a given credit card **transaction** is **fraud** or not
- **Health** : Predicting if a given mass of tissue **is benign** or **malignant**
- **Marketing** : Predicting if a given **user will buy an insurance product** or not

Support Vector Machines (SVM)

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems.

In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well

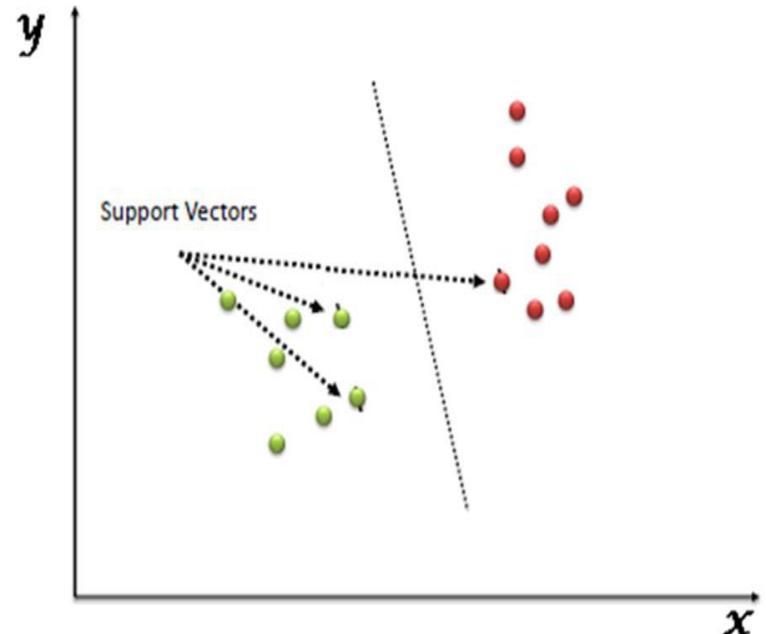


Support Vectors

Support vectors are the **data points nearest to the hyperplane**, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.

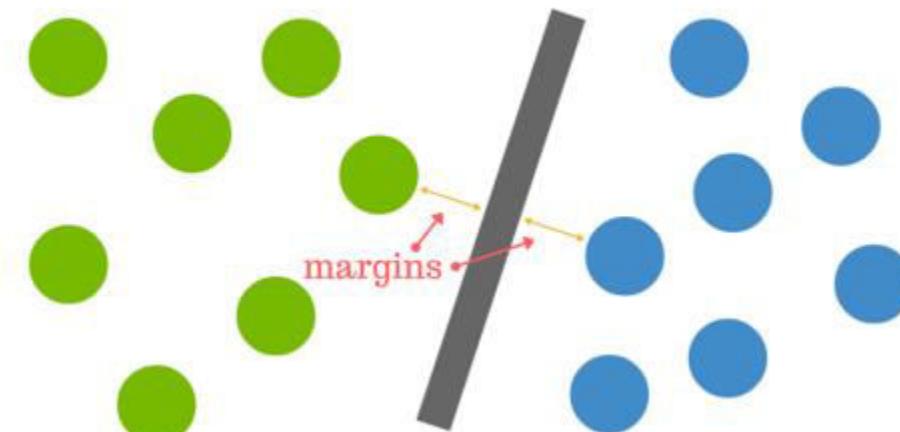
What is a hyperplane?

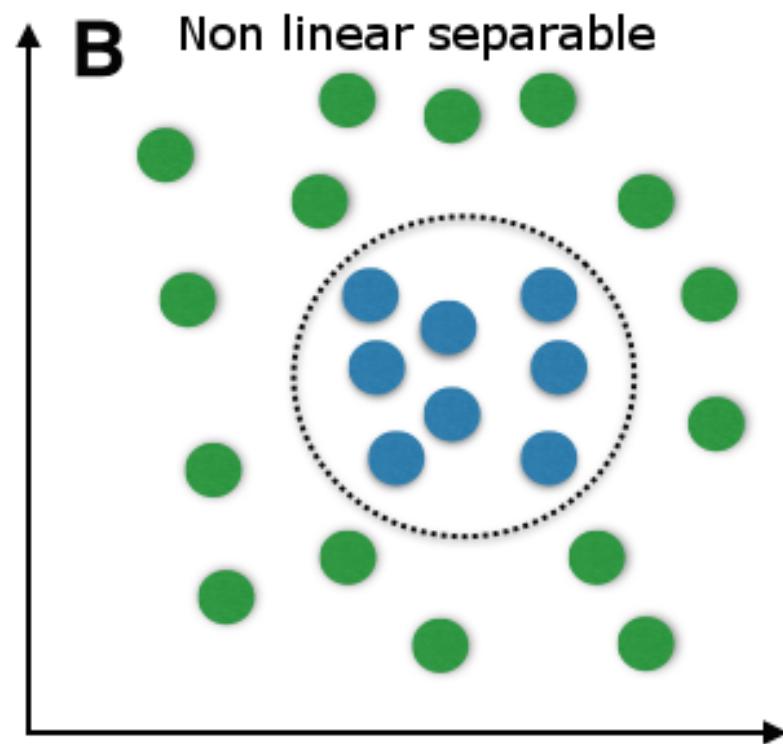
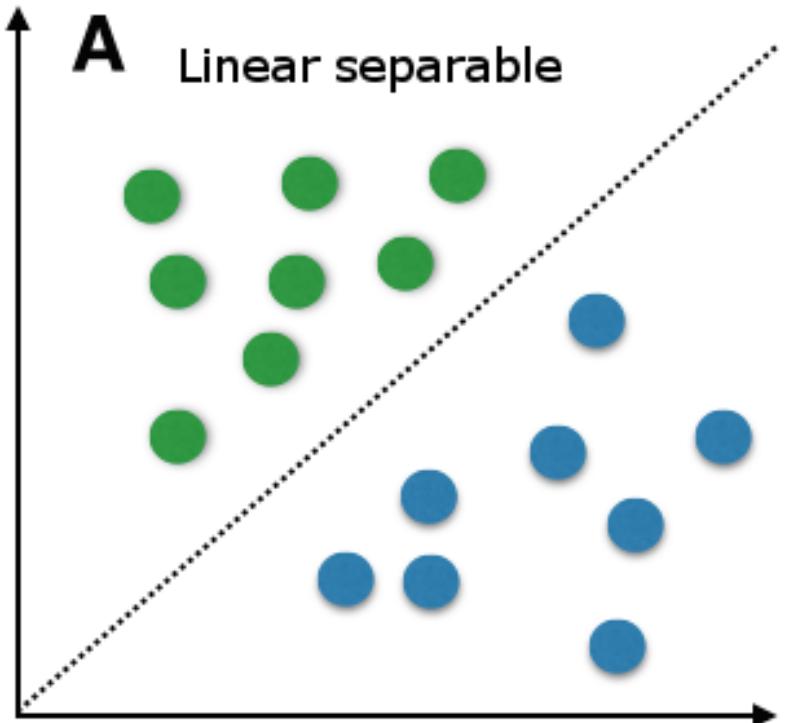
As a simple example, for a classification task with only two features (like the image), we can think of a hyperplane as a line that linearly separates and classifies a set of data.



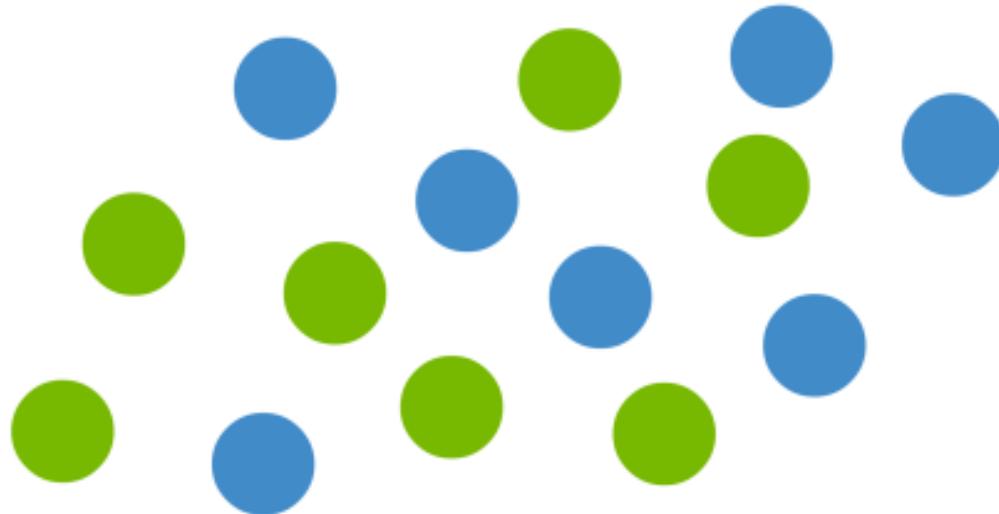
How do we find the right hyperplane?

- ❑ Or, in other words, how do we best segregate the two classes within the data?
- ❑ The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.

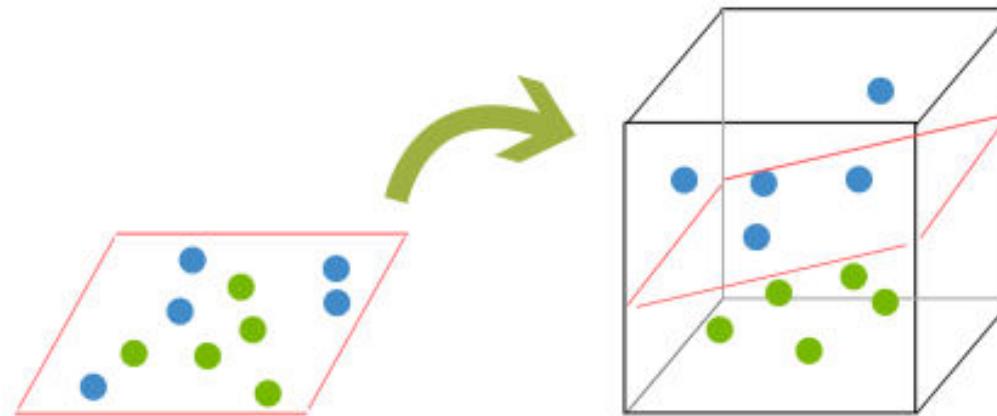




But what happens when there is no clear hyperplane?



In order to classify a dataset like the one above it's necessary to move away from a 2d view of the data to a 3d view. Explaining this is easiest with another simplified example. Imagine that our two sets of colored balls above are sitting on a sheet and this sheet is lifted suddenly, launching the balls into the air. While the balls are up in the air, you use the sheet to separate them. This 'lifting' of the balls represents the mapping of data into a higher dimension. This is known as kernelling



Because we are now in three dimensions, our hyperplane can no longer be a line. It must now be a plane as shown in the example above. The idea is that the data will continue to be mapped into higher and higher dimensions until a hyperplane can be formed to segregate it.

Pros

- Accuracy
- Works well on smaller cleaner datasets
- It can be more efficient because it uses a subset of training points

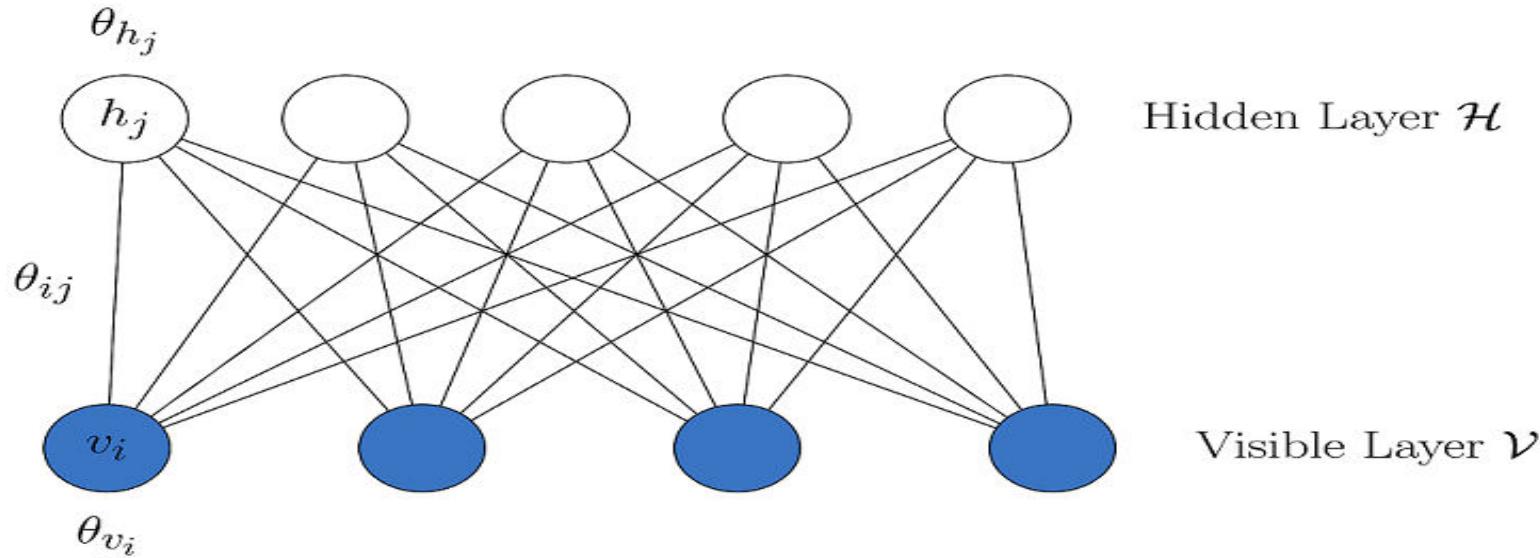
Cons

- Isn't suited to larger datasets as the training time with SVMs can be high
- Less effective on noisier datasets with overlapping classes

Boltzmann Machines

- A Hopfield Net consisting of Binary Stochastic Neuron with hidden units is called Boltzmann Machine.
- A Boltzmann Machine is a network of symmetrically connected, neuron like units that make stochastic decisions about whether to be on or off.

Structure of Boltzmann Machine



- The stochastic neurons of Boltzmann machine are in two groups: visible and hidden.
- **Visible neurons provide an interface between the net and its environment.**

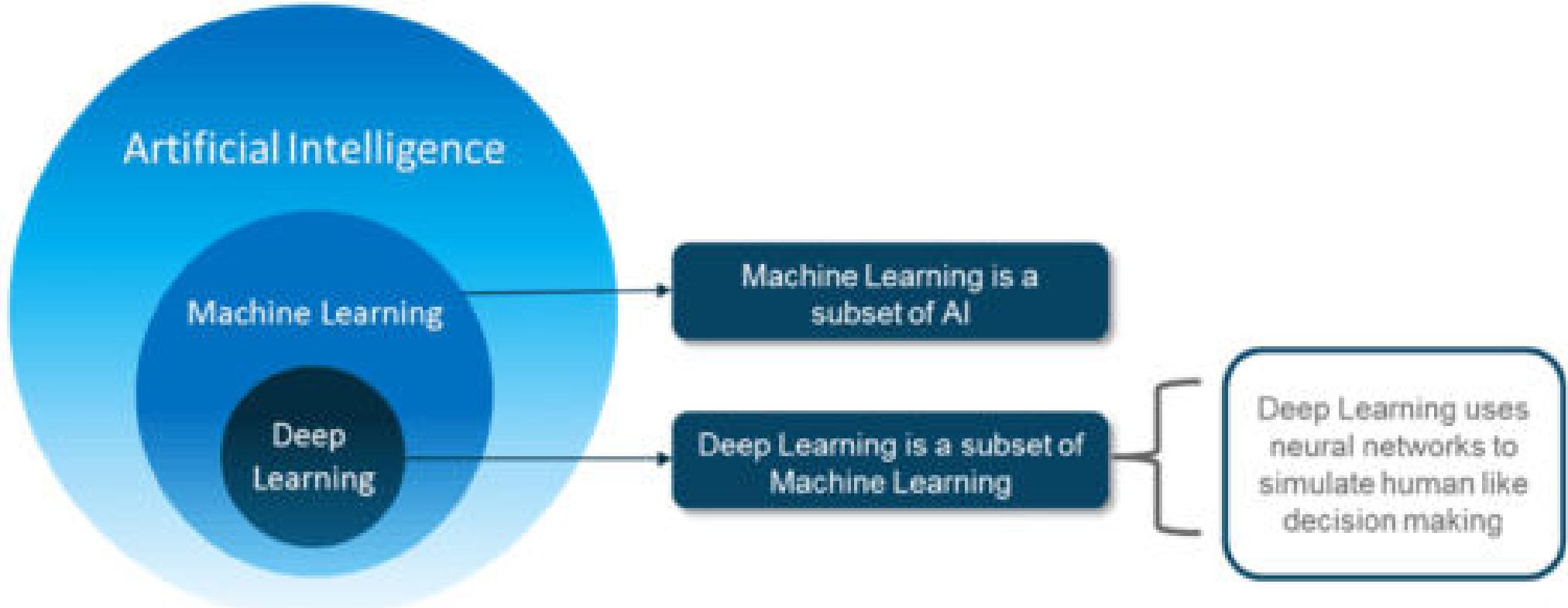
Structure of Boltzmann Machine

- During the training phase, the visible neurons are clamped; the hidden neurons always operate freely, they are used to explain underlying constraints in the environmental input vectors.
- The **hidden units explain underlying constraints by capturing higher-order correlations between the clamping vectors.**
- Boltzmann machine learning may be viewed as an unsupervised learning procedure for modeling a distribution that is specified by the clamping patterns.
- In Boltzmann Machine, everything is defined in terms of the energies of joint configuration of the visible and hidden units.

Applications of Boltzmann Machine

- Stock market trend prediction
- Character recognition
- Face recognition
- Internet application
- Cancer detection
- Loan application
- Decision making

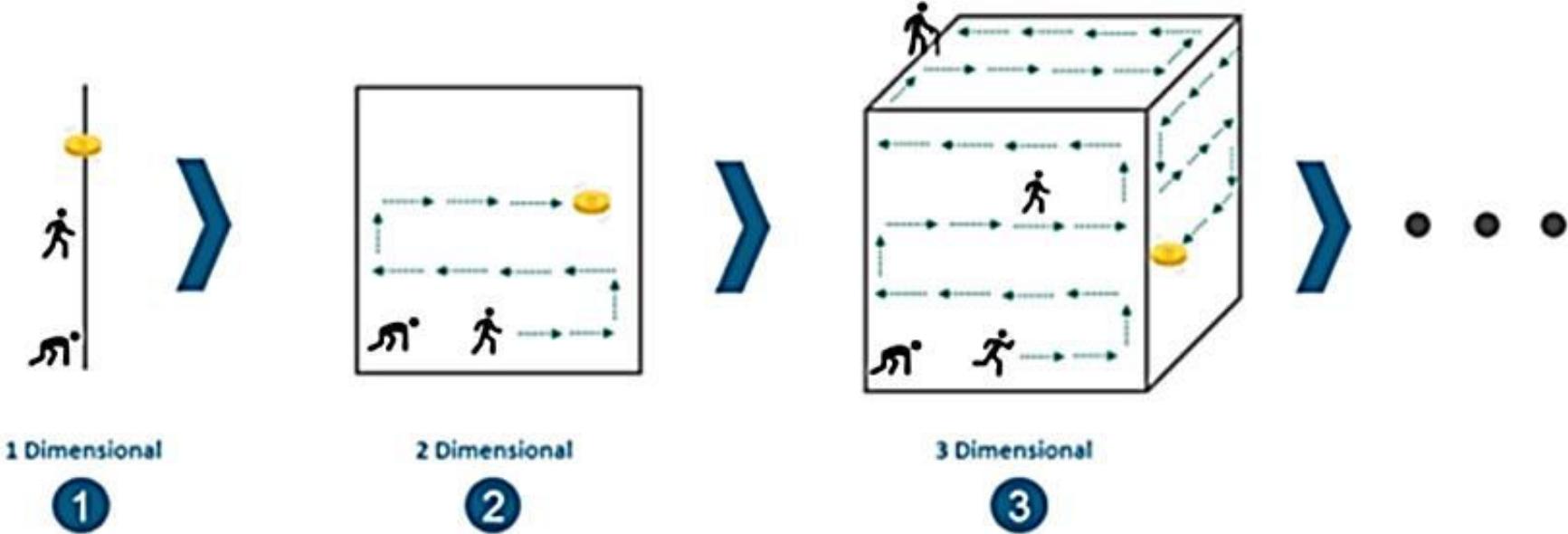
Deep Learning



Deep Learning

Machine Learning is not capable of handling high dimensional data that is where input & output is quite large. Handling and processing such type of data becomes very complex and resource exhaustive. This is termed as **Curse of Dimensionality**

Deep Learning



You have dropped a coin somewhere on the line. Now, it's quite convenient for you to find the coin

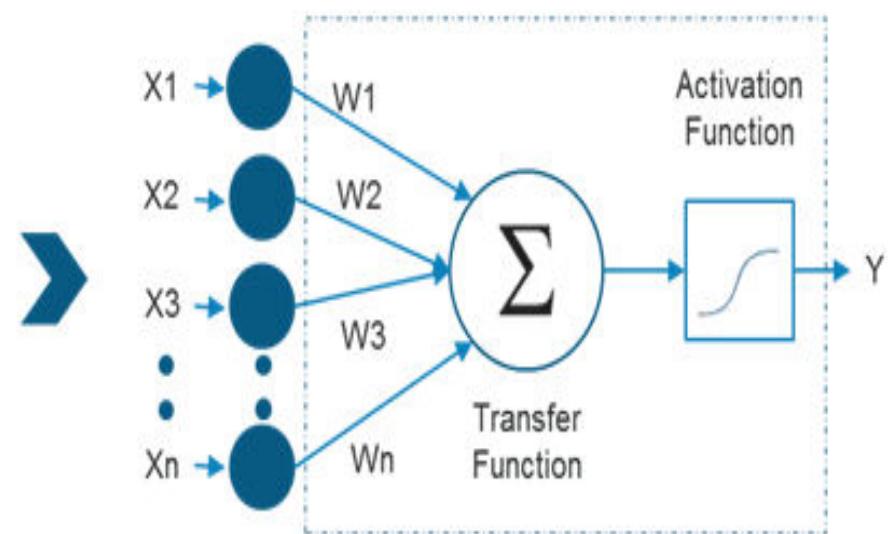
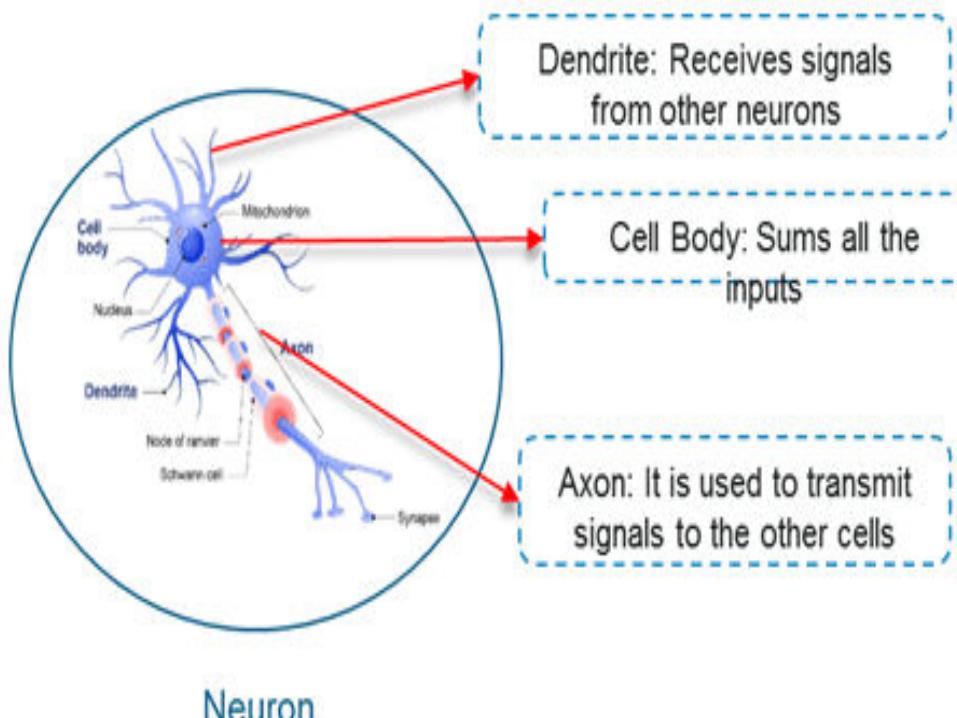
Deep Learning

Hence, we can observe the **complexity is increasing as the dimensions are increasing**. And in real-life, the high dimensional data that we were talking about has thousands of dimensions that makes it very complex to handle and process. The high dimensional data can easily be found in use-cases like Image processing, NLP, Image Translation etc.

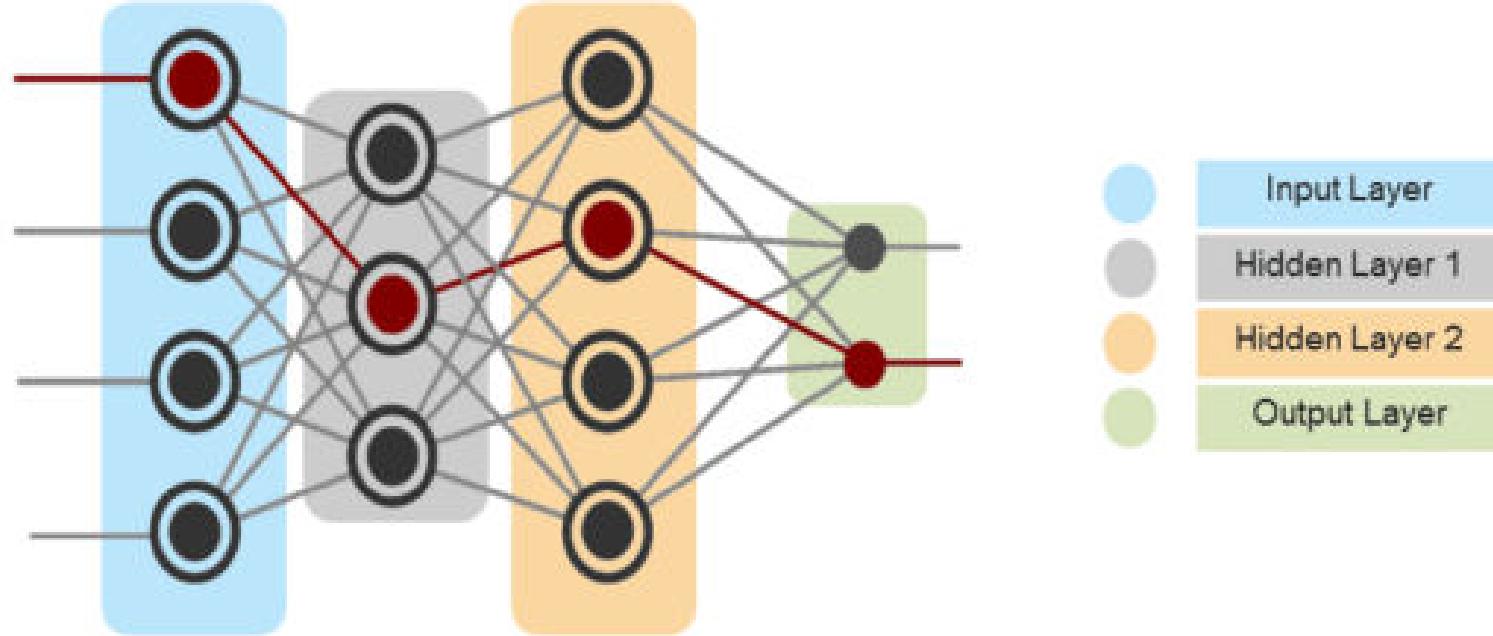
Machine learning was not capable of solving these use-cases and hence, Deep learning came to the rescue. Deep learning is capable of handling the high dimensional data and is also efficient in focusing on the right features on its own. This process is called feature extraction

How Deep Learning Works?

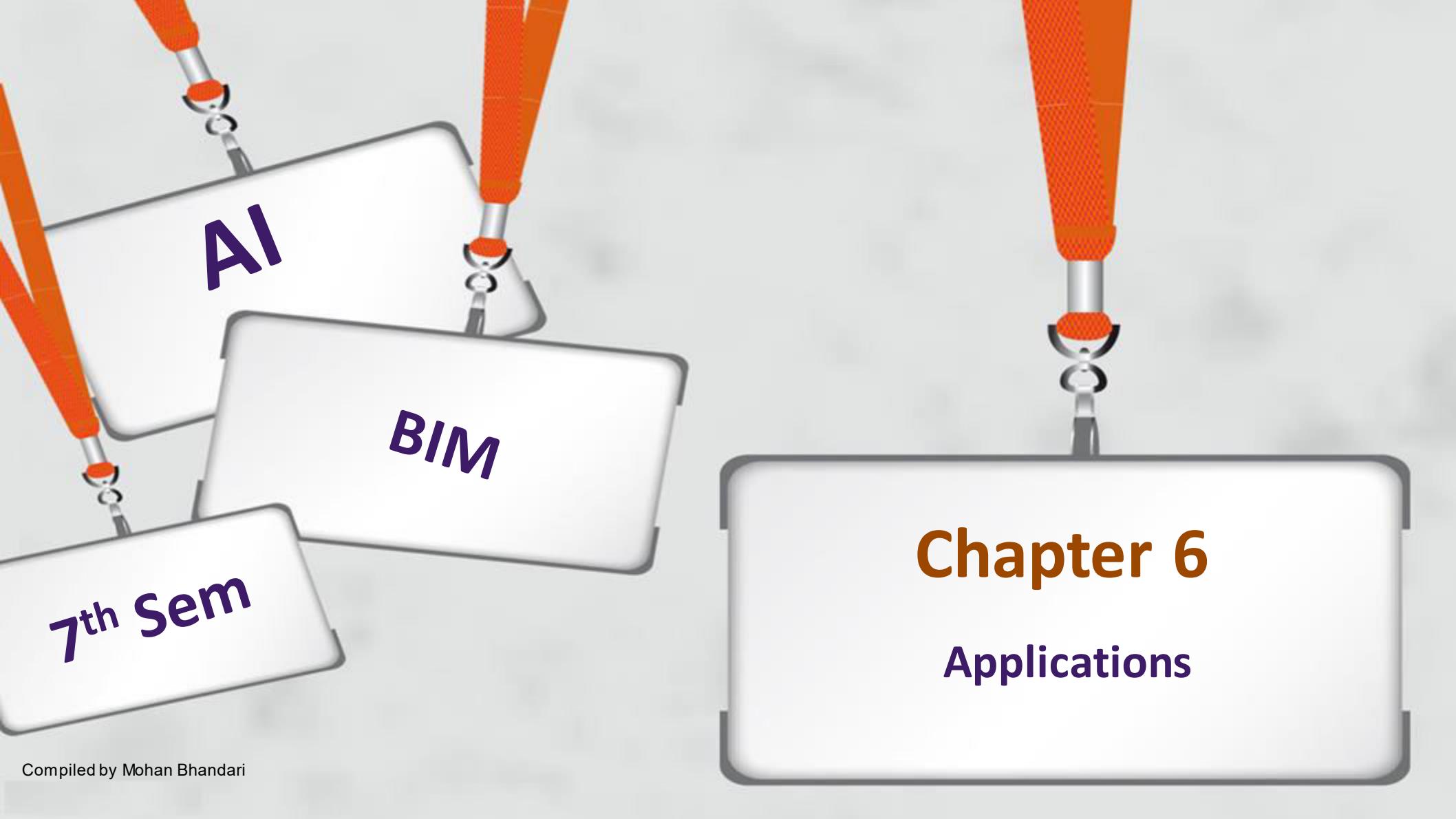
In an attempt to re-engineer a human brain,



Schematic for a neuron in a neural net



First layer is the **input layer** which receives all the inputs and the last layer is the **output layer** which provides the desired output. All the layers in between these layers are called **hidden layers**. There can be n number of hidden layers



Chapter 6

Applications

7th Sem

BIM

AI

Neural Networks

A neuron is a cell in brain whose principle function is the collection, Processing, and dissemination of electrical signals.

Brains Information processing capacity comes from networks of such neurons.

Due to this reason some earliest AI work aimed to create such artificial networks/ connectionism / Parallel distributed processing/ Neural computing

Neural Network

- An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information.
- It is composed of a large number of highly interconnected processing elements (neurones) working to solve specific problems.
- ANNs, like people, learn by example.
- An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.

- A neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain.
- *A neural network usually involves a large number of processors operating in parallel and arranged in tiers. The first tier receives the raw input information -- analogous to optic nerves in human visual processing. Each successive tier receives the output from the tier preceding it, rather than from the raw input -- in the same way neurons further from the optic nerve receive signals from those closer to it. The last tier produces the output of the system.*

Misconceptions

- Neural Networks are model of human brain
- Bigger size neural Networks works better
- Many Training data exists in neural networks

Why Neural Networks

- **Pattern Extraction** : can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.
- **Expert Systems**: A trained neural network can be thought of as an expert in the category of information it has been given to analyze.
- **Adaptive learning**: An ability to learn how to do tasks based on the data given for training or initial experience.
- **Fault Tolerance via Redundant Information Coding**: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage

How Do Neural Networks Differ From Conventional Computing?

- ANNs are not sequential.
- There are no complex central processors, rather there are many simple ones which generally do nothing more than take the weighted sum of their inputs from other processors.
- ANNs do not execute programed instructions; they respond in parallel (either simulated or actual) to the pattern of inputs presented to it.
- There are also no separate memory addresses for storing data. Instead, information is contained in the overall activation 'state' of the network.

Units of Neural Network

- **Nodes (units)**

Nodes represent a cell of neural network.

- **Links**

Links are directed arrows that show propagation of information from one node to another node.

- **Activation**

Activation are inputs to or outputs from a unit.

- **Weight**

Each link has weight associated with it which determines strength and sign of the connection.

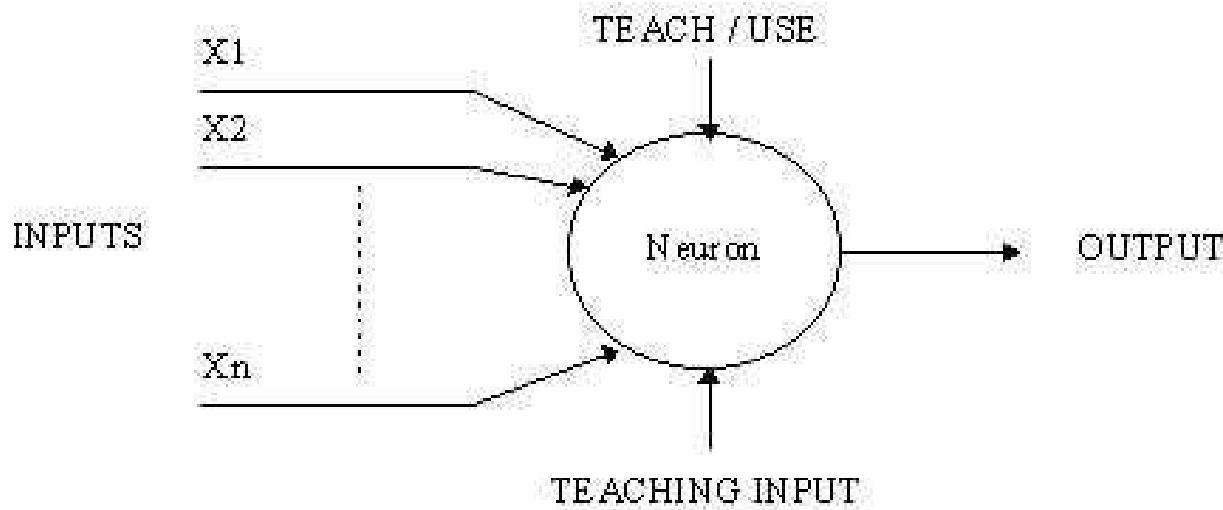
- **Activation function:**

A function which is used to derive output activation from the input activations to a given node is called activation function.

- **Bias:**

Bias is a feature of a statistical technique or of its results whereby the expected value of the results differs from the true underlying quantitative parameter being estimated.

A simple neuron



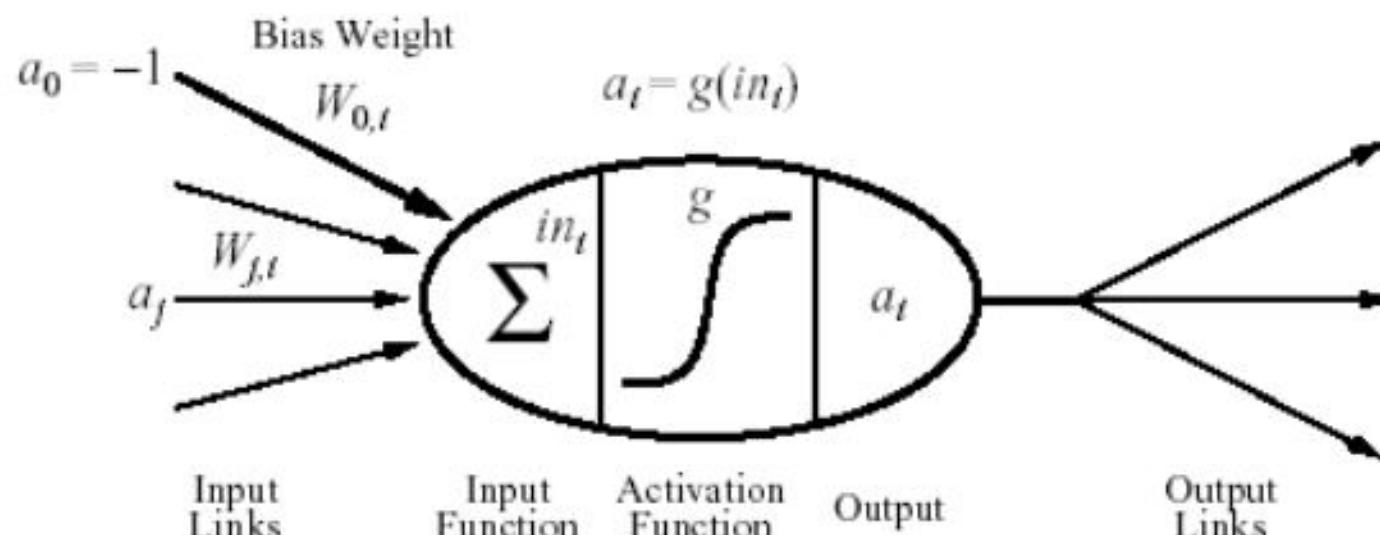
An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output.

If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

Simple Model of Neural Network

A simple mathematical model of neuron is devised by McCulloch and Pitt is given in the figure given below:

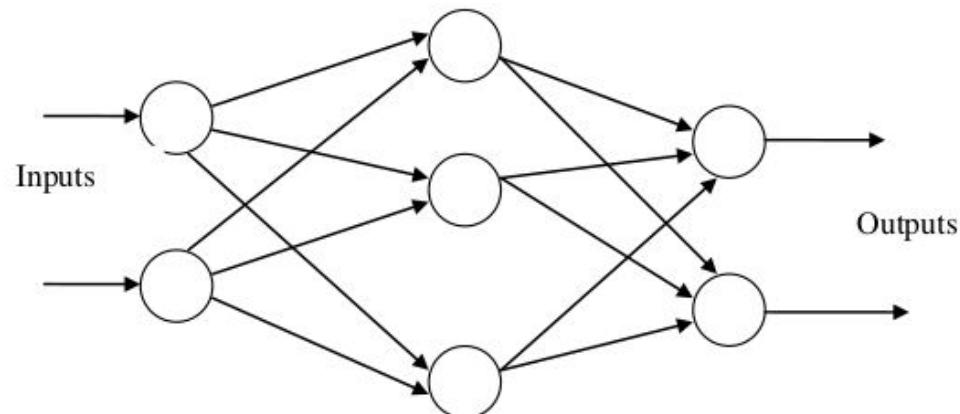
$$g(in_i) = g \left(\sum w_{ij} a_j \right)$$

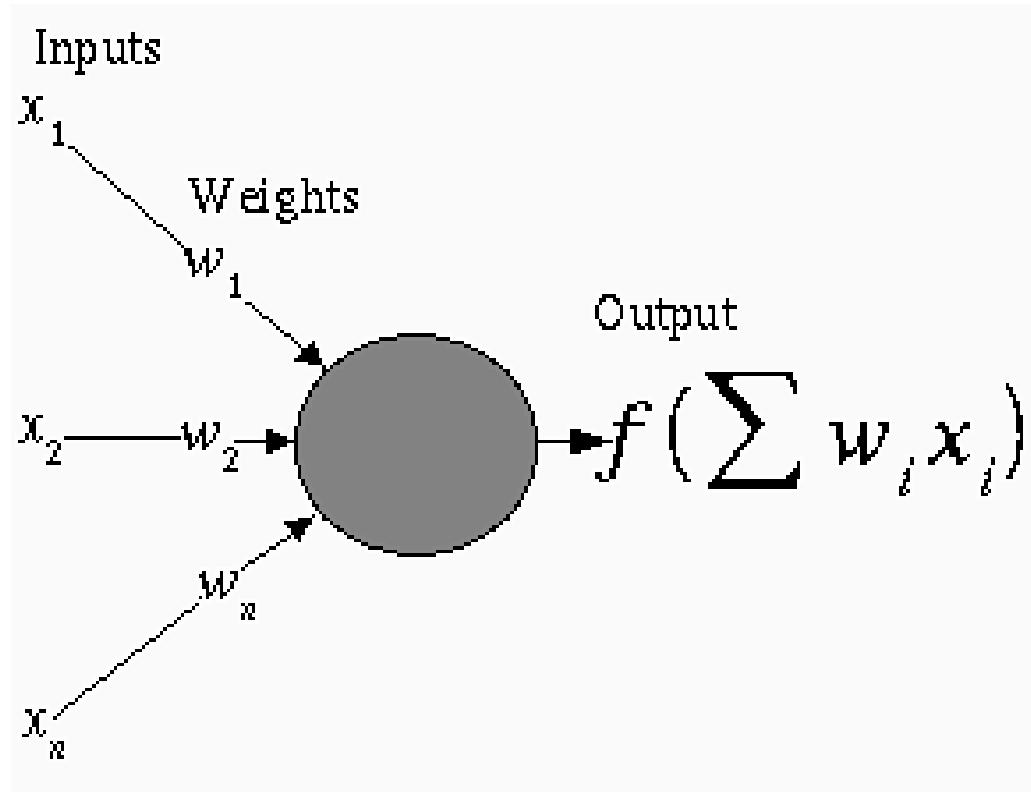


Architecture of Neural Networks

Feed-forward networks

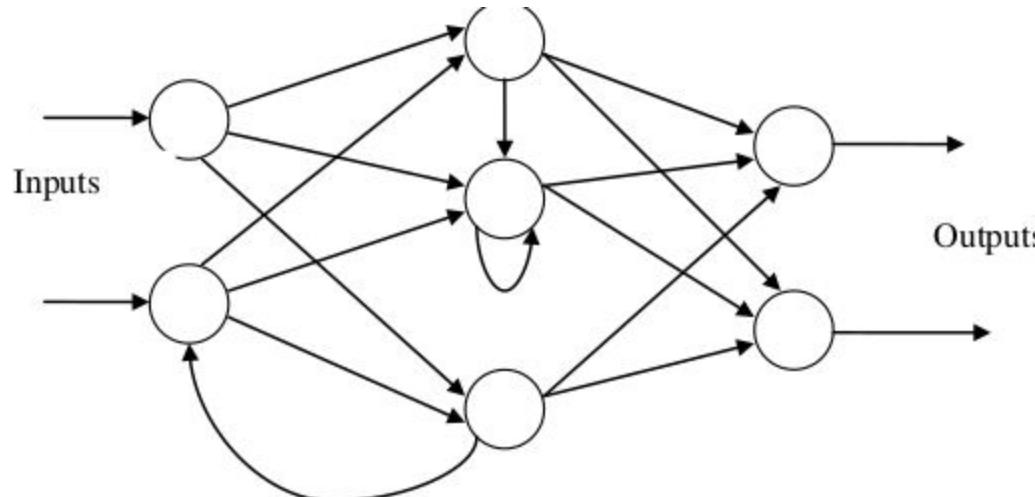
- Feed-forward ANNs allow signals to travel one way only; from input to output.
- There is no feedback (loops) i.e. the output of any layer does not affect that same layer.
- Feed forward ANNs tend to be straight forward networks that associate inputs with outputs.
- They are extensively used in pattern recognition.
- This type of organization is also referred to as bottom-up or top-down.





Feedback networks

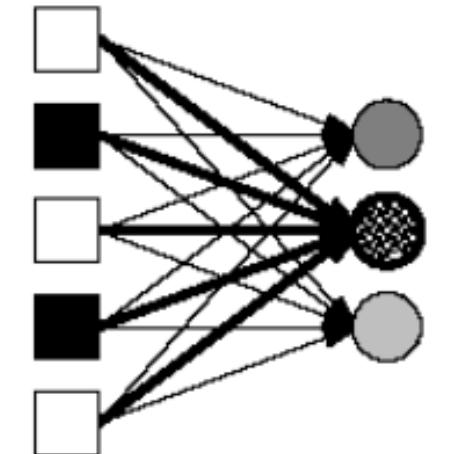
- › Feedback networks can have signals traveling in both directions by introducing loops in the network.
- › Feedback networks are very powerful and can get extremely complicated.
- › Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found.



Types of Feed Forward Neural Network

Single-layer neural networks (perceptrons)

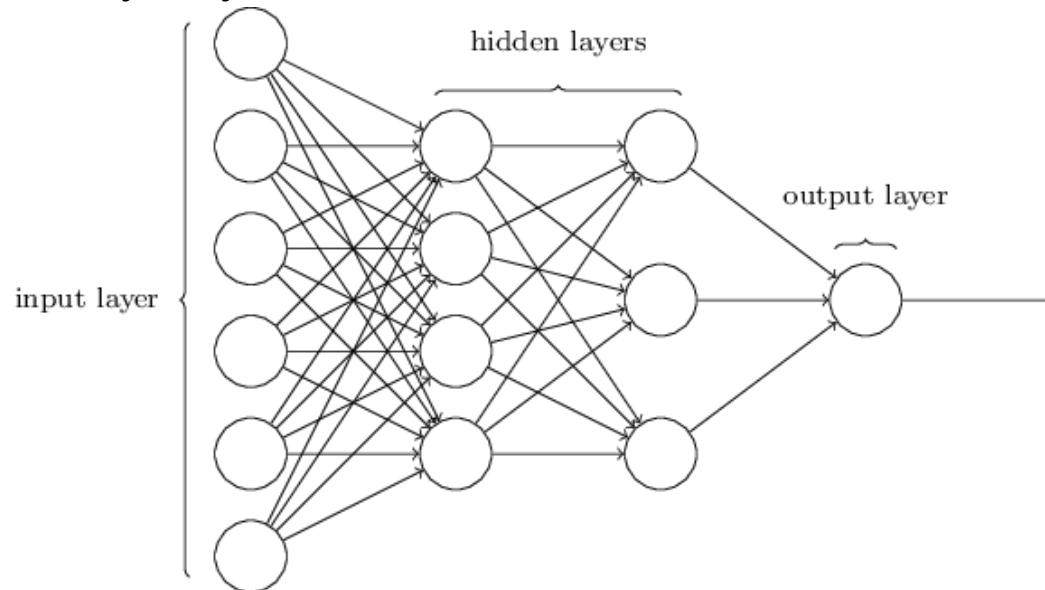
- A neural network in which all the inputs connected directly to the outputs is called a single- layer neural network, or a perceptron network.
- Since each output unit is independent of the others each weight affects only one of the outputs.



*Input
Units* $W_{j,i}$ *Output
Units*

Multilayer neural networks (perceptrons)

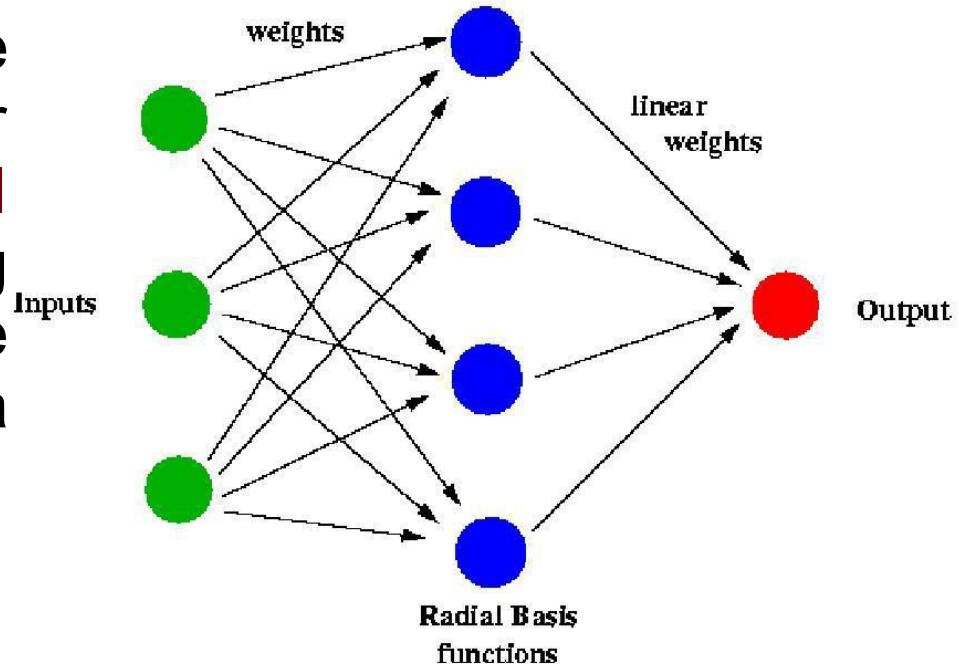
- The neural network which contains input layers, output layers and some hidden layers also is called multilayer neural network.
- The advantage of adding hidden layers is that it enlarges the space of hypothesis. Layers of the network are normally fully connected.



Other Special Types of Neural Networks

a. Radial Basis Function (RBF) Neural Network

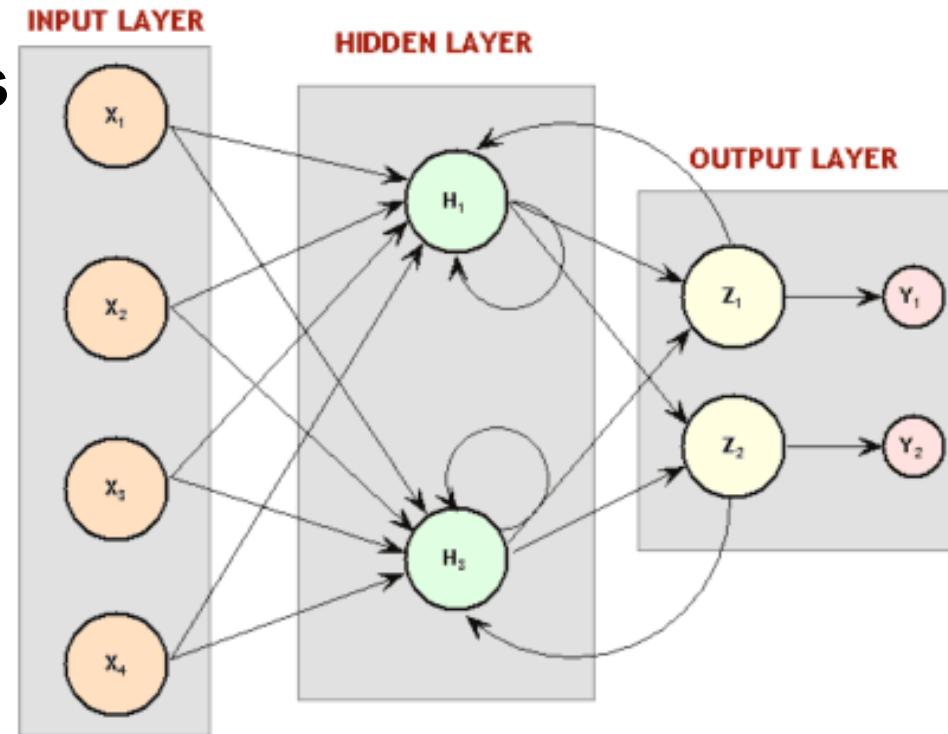
Radial basis functions are powerful techniques for **interpolation in multidimensional space.** (method of constructing new data points within the range of a discrete set of known data points)



Other Special Types of Neural Networks

b. Recurrent Neural Networks

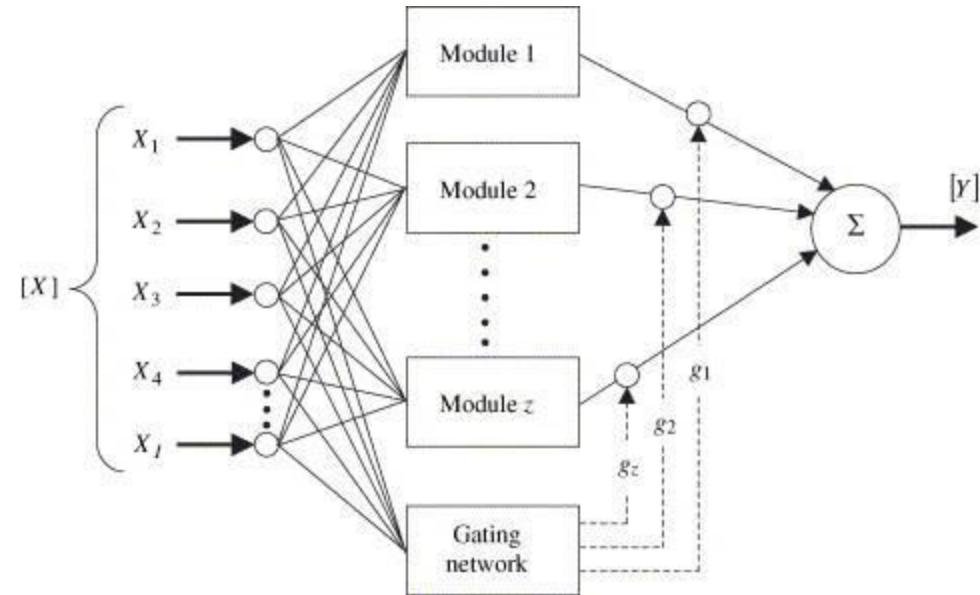
- Recurrent neural networks (RNNs) are models with bi-directional data flow.
- Recurrent neural networks can be used as **general** sequence processors



Other Special Types of Neural Networks

c. Modular Neural Network

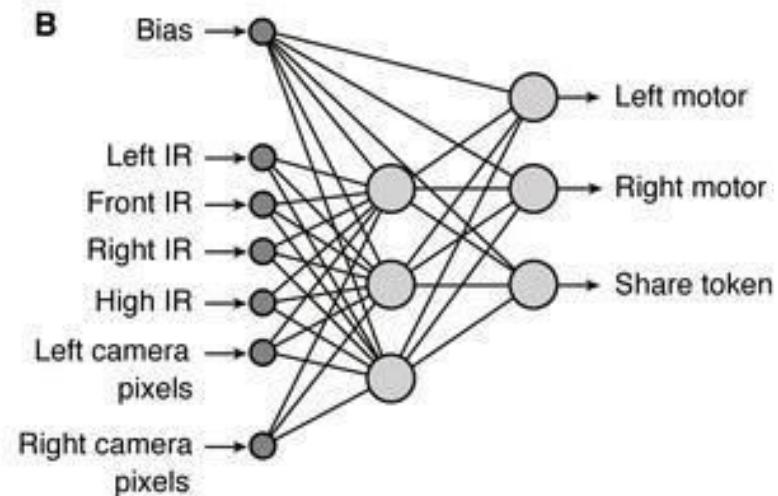
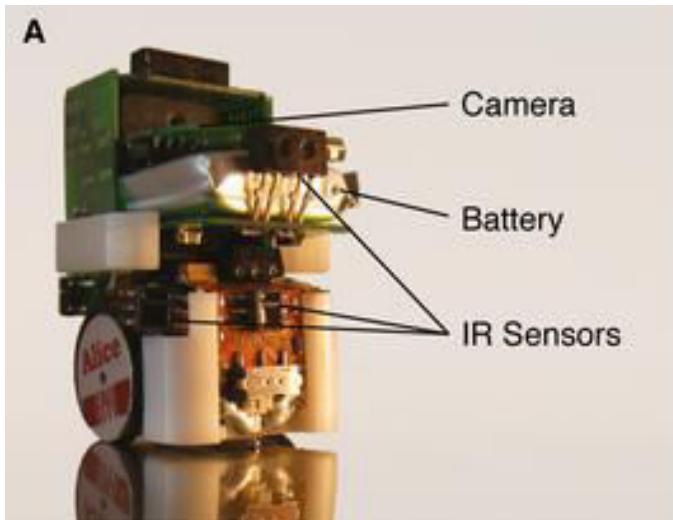
- Biological studies have shown that the **human brain functions** not as a single massive network, but as a collection of small networks.
- This realization gave birth to the concept of **modular neural networks**, in which several small networks cooperate or compete to solve problems.



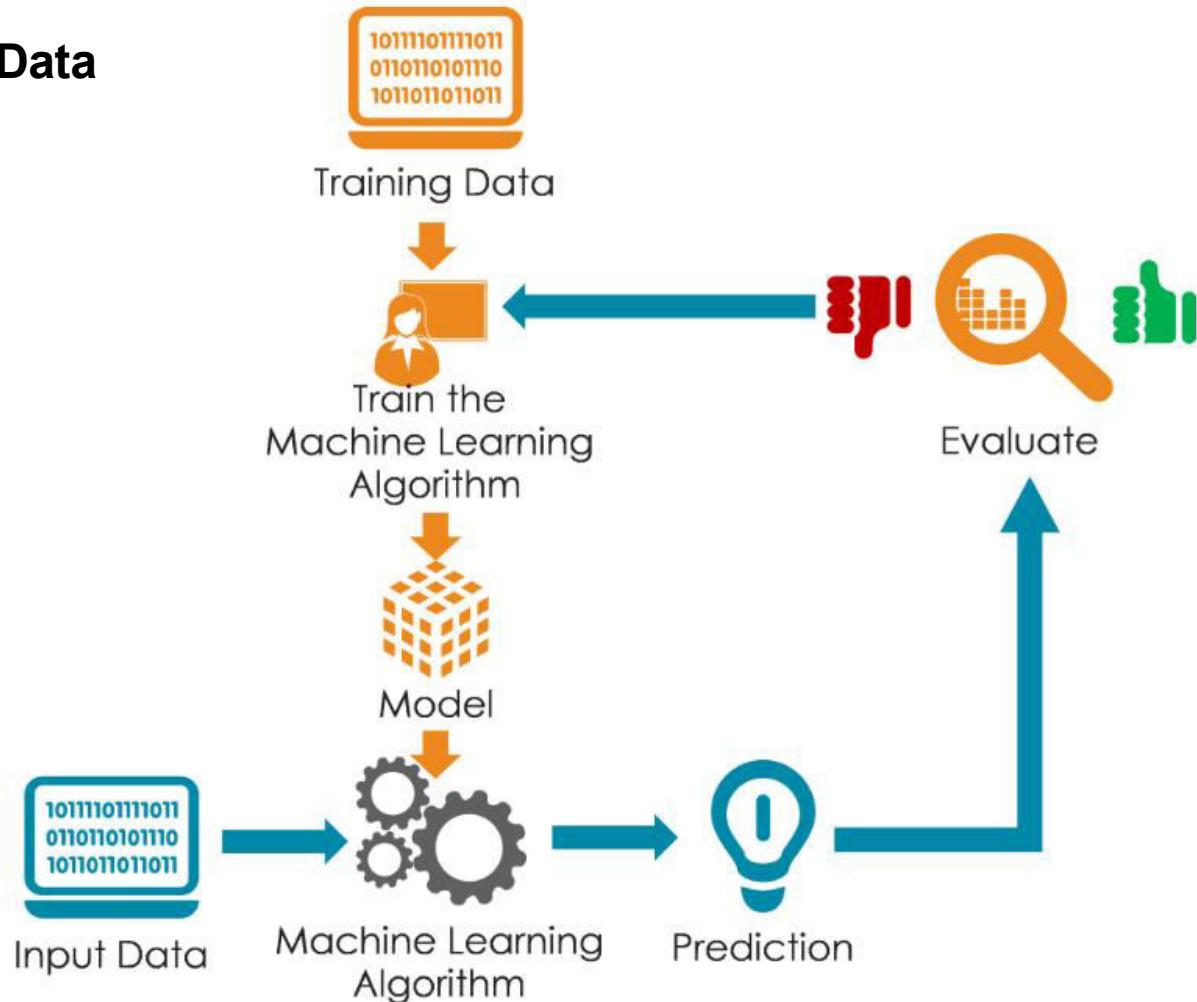
Other Special Types of Neural Networks

d. Physical Neural Network

A physical neural network includes electrically adjustable resistance material to simulate artificial synapses



Training and Test Data



➤ Training Data

The observations in the **training set** form the experience that the algorithm uses to learn.

In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

➤ Test Data

The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set. If the test set does contain examples from the training set, it will be difficult to assess whether the algorithm has learned to generalize from the training set or has simply memorized it.

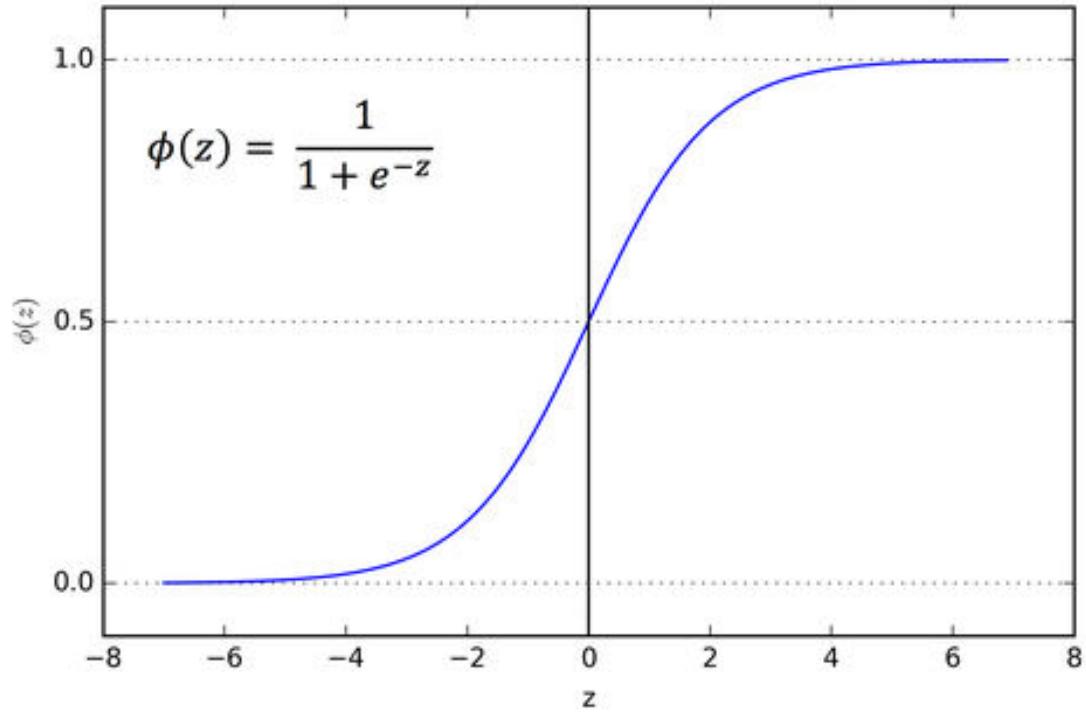
Activation Function

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs

Types of Activation Function

□ Sigmoid or Logistic Activation Function

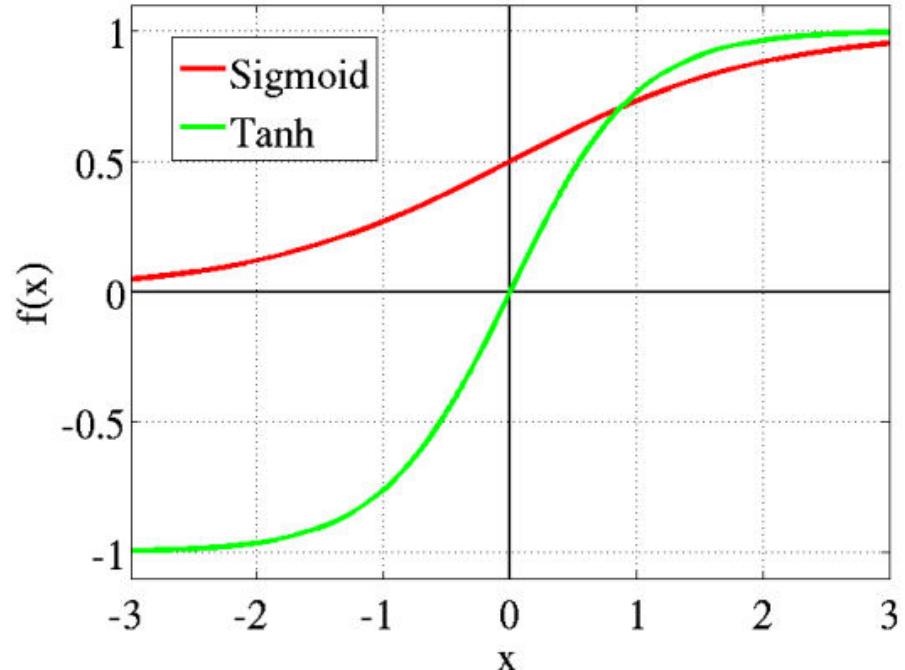
- ✓ The Sigmoid Function curve looks like a S-shape
- ✓ The main reason why we use sigmoid function is because it exists between **(0 to 1)**. Therefore, it is especially used for models where we have to **predict the probability** as output.
- ✓ Since probability of anything exists only between the range of **0 and 1**, sigmoid is the right choice.



Types of Activation Function

□ Tanh or hyperbolic tangent Activation Function

- ✓ tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).
- ✓ The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.



Types of Activation Function

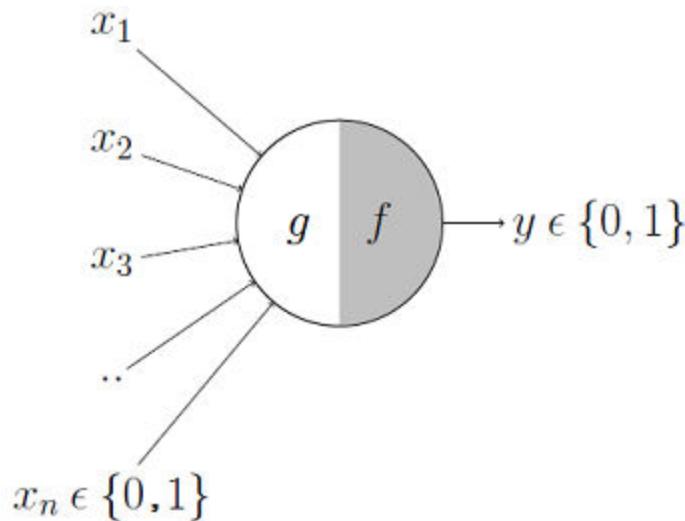
- unit (unary and binary)
- Ramp
- piecewise linear

Assignment:

Explain the above activation function.

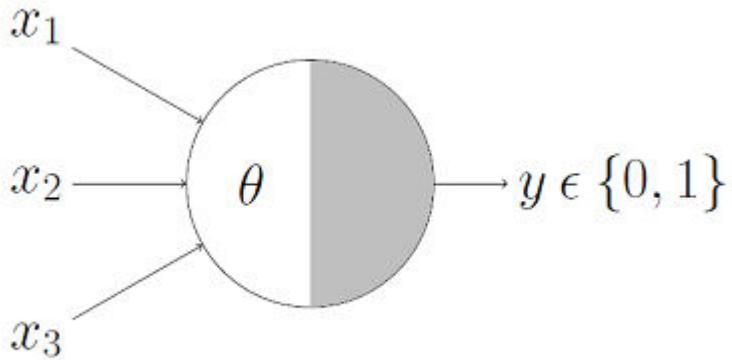
McCulloch-Pitts Neuron

The first computational model of a neuron was proposed by Warren McCulloch (neuroscientist) and Walter Pitts (logician) in 1943.

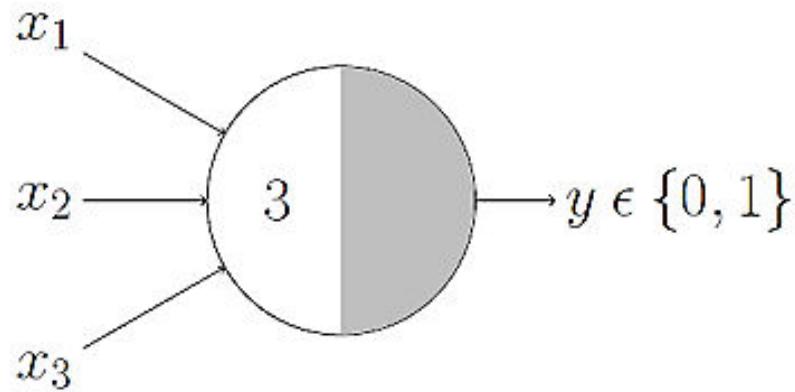


It may be divided into 2 parts. The first part, **g** takes an input, performs an aggregation and based on the aggregated value the second part, **f** makes a decision.

M-P Neuron: A Concise Representation



This representation just denotes that, for the boolean inputs x_1 , x_2 and x_3 if the $g(x)$ i.e., sum $\geq \theta$, the neuron will fire otherwise, it won't.



AND Function

An AND function neuron would only fire when ALL the inputs are ON i.e., $g(x) \geq 3$ here

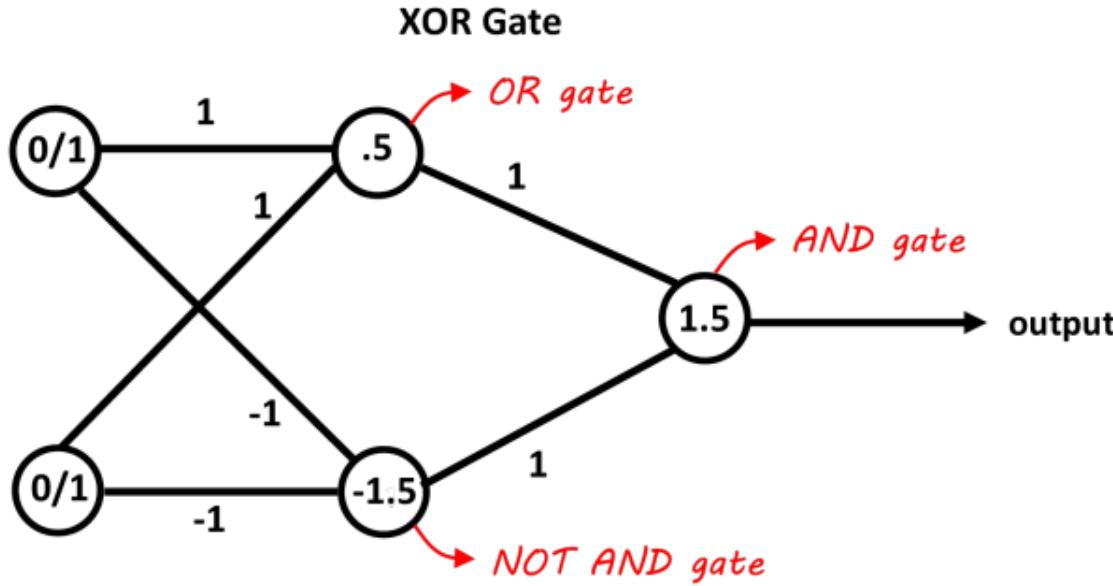
OR Function

An OR function neuron would fire if ANY of the inputs is ON i.e., $g(x) \geq 3$



NOT Function

For a NOT neuron, 1 outputs 0 and 0 outputs 1. So we take the input as an inhibitory input and set the thresholding parameter to 0.



XOR Function

$$x_1 \text{ XOR } x_2 == (x_1 \text{ AND NOT } x_2) \text{ OR } (x_2 \text{ AND NOT } x_1)$$

Neural Networks in Practice (Applications)

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management
- target marketing

Perceptron

- The perceptron is an algorithm for supervised learning
- Functions that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not

Perceptron Learning Theory

- Perceptron (an element of artificial Neural Network consisting of one or more layer of artificial neuron)
- The term "Perceptrons" was coined by Frank RosenBlatt in 1962 and is used to describe the connection of simple neurons into networks.
- These networks are simplified versions of the real nervous system where some properties are exaggerated and others are ignored.

- We need to train the neural network so they can do things that are useful.
- To do this we must allow the neuron to learn from its mistakes.
- There is in fact a learning paradigm that achieves this, it is known as supervised learning and works in the following manner.
 1. Set the weight and thresholds of the neuron to random values.
 2. Present an input.
 3. Calculate the output of the neuron.
 4. Alter the weights to reinforce correct decisions and discourage wrong decisions, hence reducing the error. So for the network to learn we shall increase the weights on the active inputs when we want the output to be active, and to decrease them when we want the output to be inactive.
 5. Now present the next input and repeat steps 3.

Delta Rule/ Widrow – Hoff Rule

- Called as the Least Mean Square (LMS) method Delta rule is one of the most commonly used learning rules.
- It is a special case of the more general back propagation algorithm.
- For a given input vector, the output vector is compared to the correct answer, If the difference is zero, no learning takes place; otherwise, the weights are adjusted to reduce this difference.

For a neuron j with activation function $g(x)$, the delta rule for j 's i^{th} weight w_{ji} is given by

$$\Delta w_{ji} = \alpha(t_j - y_j)g'(h_j)x_i$$

where

α is a small constant called *learning rate*

$g(x)$ is the neuron's activation function

t_j is the target output

h_j is the weighted sum of the neuron's inputs

y_j is the actual output

x_i is the i^{th} input.

It holds that $h_j = \sum x_i w_{ji}$ and $y_j = g(h_j)$.

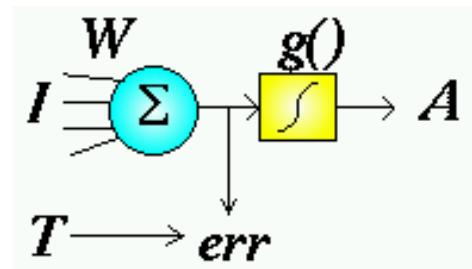
Adaline learning

ADALINE is an acronym for ADAptive LINear Element (or ADAptive LInear NEuron). It was developed by Bernard Widrow and Marcian Hoff (1960).

The adaline learning rule is a training rule that minimises the output error using (approximate) gradient descent.

After each training pattern I^p is presented, the correction to apply to the weights is proportional to the error.

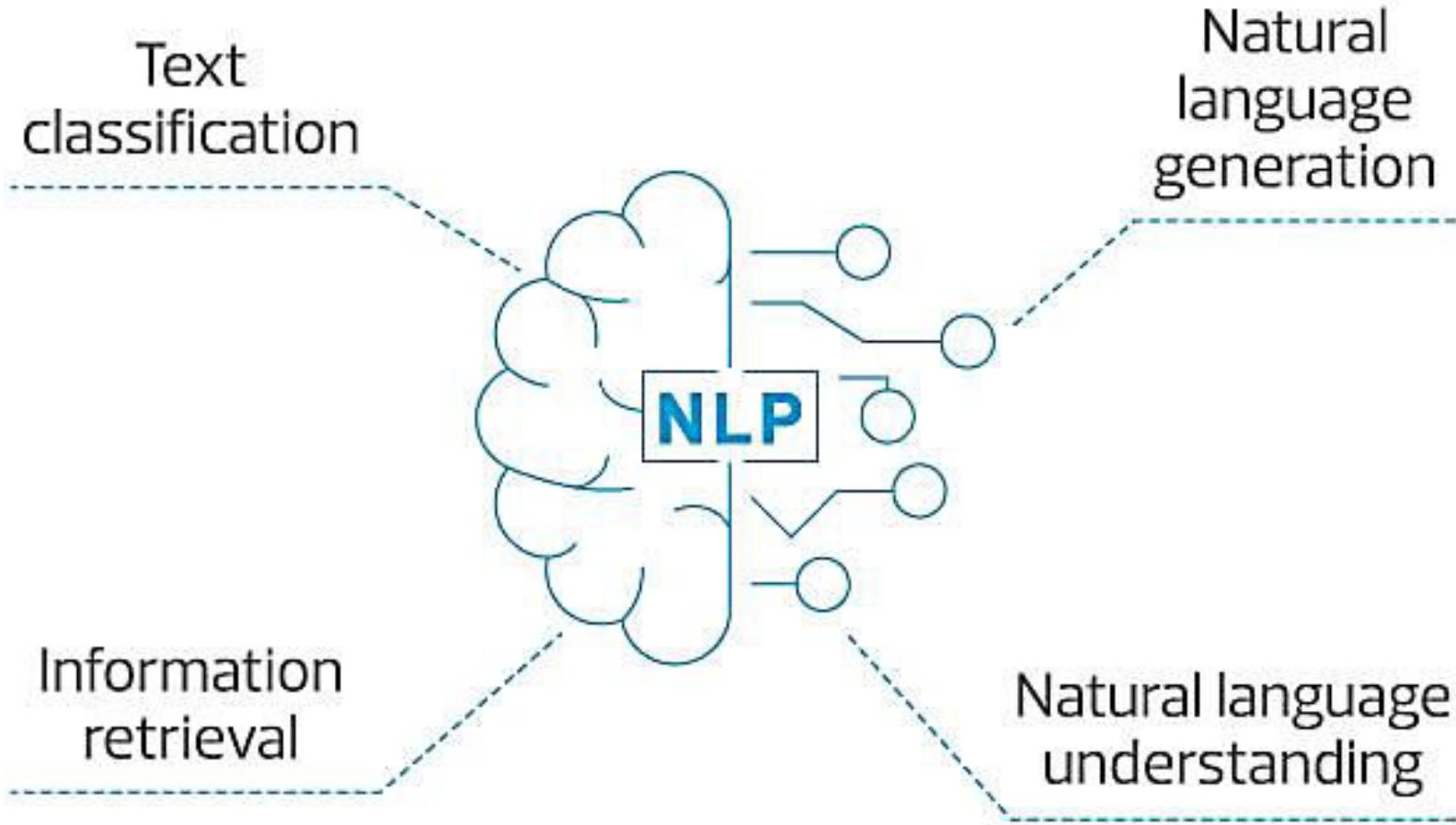
The correction is calculated before the thresholding step, using err_{ij}
 $p = T^p - W_{ij} I^p$:



Thus, the weights are adjusted by

$$W_{ij} (t+1) = W_{ij} (t) + \eta (T^p - W_{ij} I^p) (I^p) \quad \text{where } 0 \leq \eta < 1$$

Natural Language Processing



Natural Language

- In philosophy of language, a natural language or ordinary language is any language that has evolved naturally in humans through use and repetition without conscious planning or premeditation.
- Natural languages can take different forms, such as speech or signing.
- They are distinguished from constructed and formal languages such as those used to program computers or to study logic.
- In computing, natural language refers to a human language such as English, Russian, German, or Japanese as distinct from the typically artificial command or programming language with which one usually talks to a computer.

- Humans perceive and communicate through their five basic senses of sight, hearing, touch, smell and taste, and their ability to generate meaningful utterances.
- Developing programs that understand a natural language is a difficult problem.

- Developing programs to understand natural language is important in AI because a natural form of communication with systems is essential.
- AI programs must be able to communicate with their human counterparts in a natural way, and natural language is one of the most important mediums for that purpose.
- So, **Natural Language Processing (NLP)** is the field that deals with the computer processing of natural languages, mainly evolved by people working in the field of Artificial Intelligence.

What makes NLP challenging?

Natural Language contain infinity of different sentences. No matter how many sentences a person has heard or seen, new ones can always be produced. Also, there is much ambiguity in a natural language. Many words have several meanings and sentences can have different meanings in different contexts. This makes the creation of programs that understand a natural language, one of the most challenging tasks in AI.

Some of the better-known applications of NLP include:

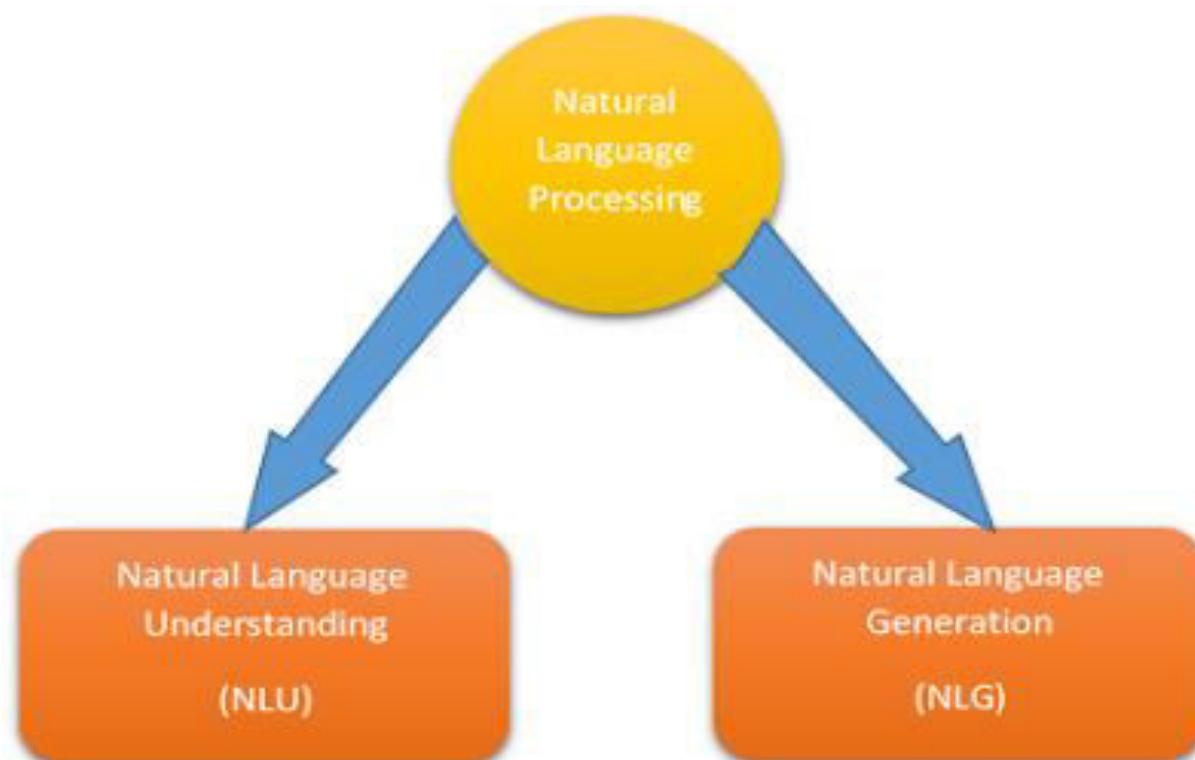
Voice recognition software: Translates speech into input for word processors or other applications;

Text-to-speech synthesizers: Read text aloud for users such as the hearing-impaired;

Grammar and style checkers: Analyze text in an attempt to highlight errors of grammar or usage;

Machine translation: Systems which automatically render a document such as a web page in another language.

Components in NLP



Natural Language Generation

- NLG also referred to as text generation, is a subfield of natural language processing.
- Natural Language Generation (NLG) is the natural language processing task of generating natural language from a machine representation system such as a knowledge base or a logical form.
- NLG system is like a translator that converts a computer based representation into a natural language representation.

NLG includes

Text planning : - Retrieving the relevant content from database. Here database can be includes vocabulary, sentences, knowledge, sample data and many more.

Sentence planning : - We get our content using text planning now next step to do is choosing required words and forming meaningful sentence setting the words in right grammatical way.

Text realization :- We have all the thing need to create actual text in humans language.

Natural Language Understanding

- It deals with machine reading comprehension : ability to read text, to do process on it, and understand its meaning.
- It is involved with mapping the given inputs in useful representations and analyzing the different aspects of the language.
- Many words have several meaning such as can, bear, fly, bank etc, and sentences have different meanings in different contexts.
- Understanding the language is not only the transmission of words. It also requires inference about the speakers' goal, knowledge as well as the context of the interaction

Difficulties in NLU

- NL has an extremely rich form and structure.
- It is very ambiguous.
- Different levels of ambiguity -
 - ✖ **Lexical ambiguity** – It is at very primitive level such as word-level.
 - ✖ **Syntax Level ambiguity** – A sentence can be parsed in different ways.
 - ✖ **Referential ambiguity** – Referring to something using pronouns.
 - ✖ One input can mean different meanings.
 - ✖ Many inputs can mean the same thing.

Levels of knowledge used in Language Understanding

- › A language understanding knowledge must have considerable knowledge about the structures of the language including what the words are and how they combine into phrases and sentences.
- › It must also know the meanings of the words and how they contribute to the meanings of the sentence and to the context within which they are being used.

The component forms of knowledge needed for an understanding of natural languages are sometimes classified according to the following levels.

- **Phonological:**

Relates sound to the words we recognize. A phoneme is the smallest unit of the sound.

- **Morphological:**

This is lexical (Linguistic unit) knowledge which **relates to the word construction from basic units called morphemes.** A morpheme is the smallest unit of meaning.

Eg:- friend + ly = friendly.

➤ **Syntactic:**

This knowledge **relates to how words are put together** or structured together to form grammatically correct sentences in the language.

➤ **Semantic :**

This knowledge is **concerned with the meanings of words and phrases** and how they combine to form sentence meaning

➤ **Pragmatic:**

This is high – level knowledge which **relates to the use of sentences in different contexts** and how the context affects the meaning of the Sentence in real world.

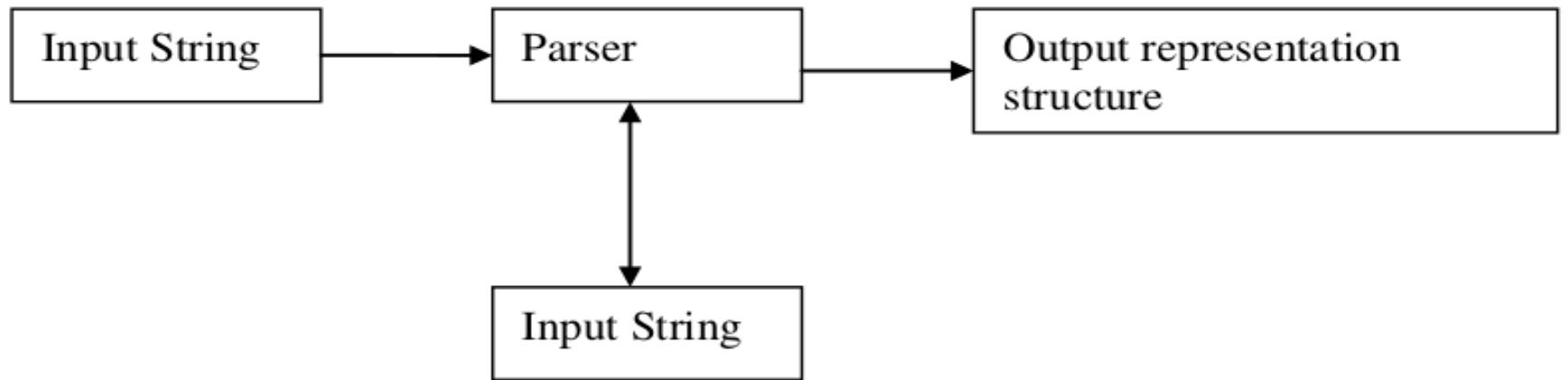
➤ **World :**

Includes the knowledge of the physical world, the world of human social interaction, and the roles of goals and intentions in communication.

Basic Parsing Techniques

- Before the meaning of a sentence can be determined, the meanings of its constituent parts must be established.
- This requires knowledge of the structure of the sentence, the meaning of the individual words and how the words modify each other.
- The process of determining the syntactical structure of a sentence is known as parsing

- Parsing is the process of analyzing a sentence by taking it apart word – by – word and determining its structure from its constituent parts and sub parts.
- The structure of a sentence can be represented with a syntactic tree. When given an input string, the lexical parts or terms (root words), must first be identified by type and then the role they play in a sentence must be determined.
- These parts can be combined successively into larger units until a complete tree has been computed.



A. Top-Down parsing:

- ✗ Using Top-Down technique, parser searches for a parse tree by trying to build from the **root node S down to the leaves**.
- ✗ The algorithm starts by assuming the input can be derived by the designated start symbol S.
- ✗ The next step is to find the tops of all the trees which can start with S, by looking on the grammar rules with S on left hand side, all the possible trees are generated.
- ✗ Top-Down parsing is goal directed search technique.

B. Bottom – Up

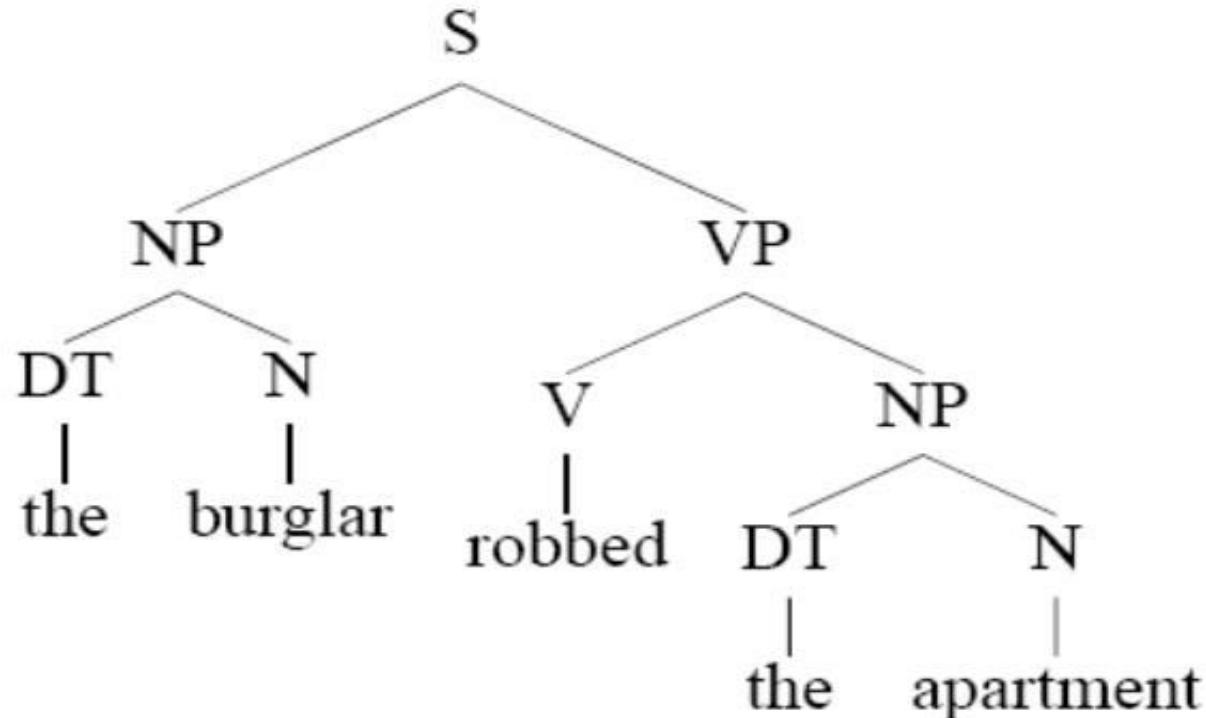
- Bottom – Up parsing starts with the words of input and tries to build trees from the words up, again by applying rules from the grammar one at a time.
- The parse is successful if the parse succeeds in building a tree rooted in the start symbol S that covers all of the input.
- Bottom up parsing is a data directed search.

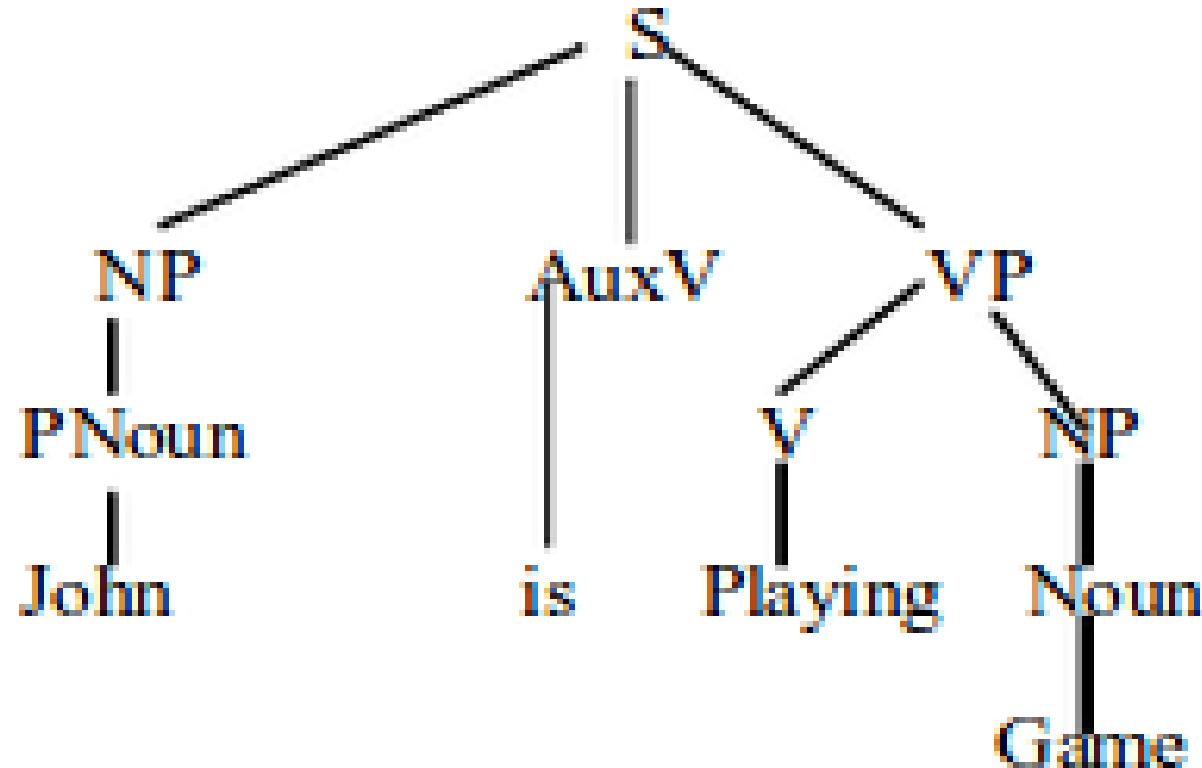
N = NOUN

V = VERB

DT =

DETERMINER

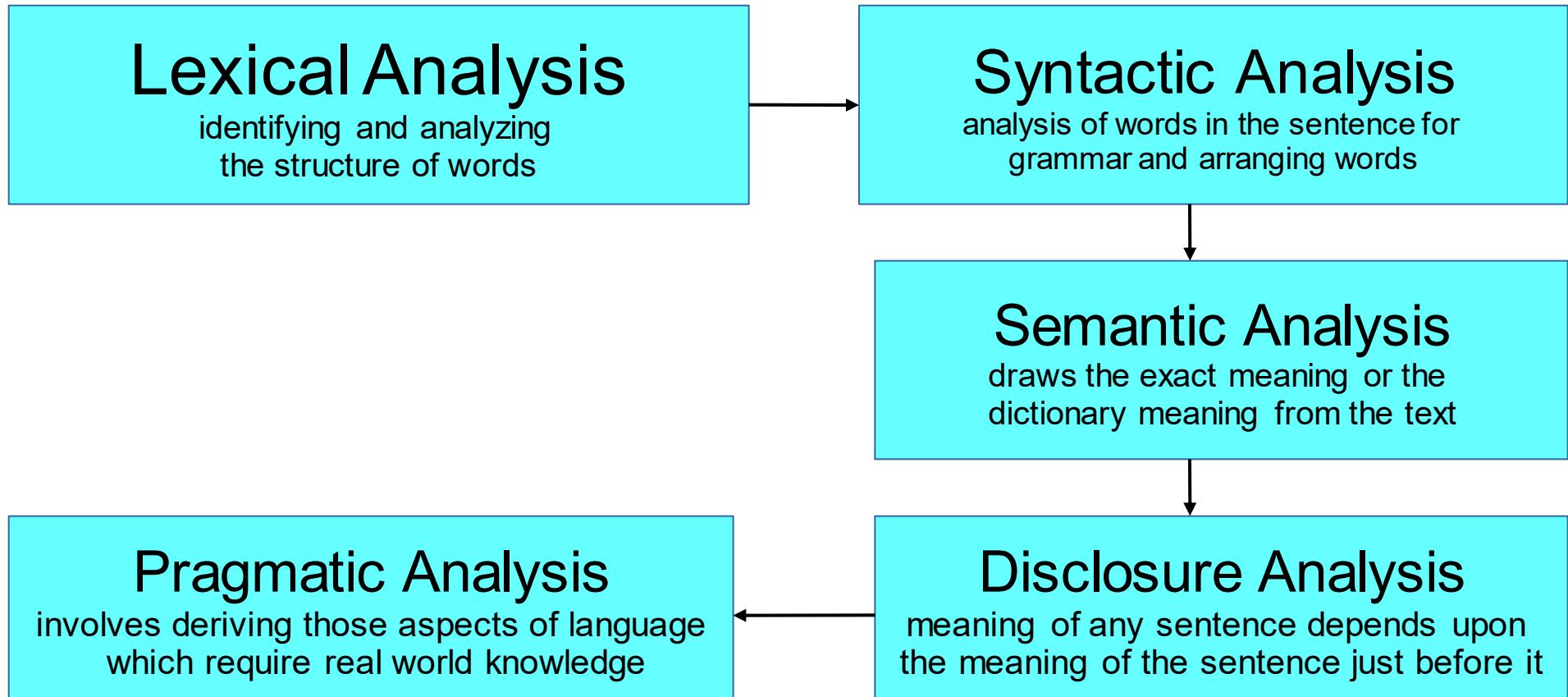




Assignment

Explain context free grammar

Steps used in NLP



Advantages of NLP

- Relieves burden of Learning syntax
- No training needed for learning Languages

Disadvantages of NLP

- Requires clarification of dialogue
- May not show context
- Unpredictable

Parameters in Natural Language Processing:

- To be exact, the parameters always (regardless of the research field) depend on the task.
- Different parameters may include
 - ✖ Auditory Inputs
 - ✖ Segmentation
 - ✖ Syntax Structure
 - ✖ Semantic Structure
 - ✖ Pragmatic Analysis

a. Auditory Inputs

- Three of our five senses – **sight, hearing and touch** – are used as **major inputs**.
- These are usually referred to as the visual, auditory and tactile inputs respectively.
- They are **sometimes called input channels**.
- In some cases, an audio output device can be used as an input device, in order to capture produced sound.
 - ✖ Microphone
 - ✖ MIDI keyboard or other digital musical instrument

b. Text segmentation

- Segmentation: Determining the positions at which topics change in a stream of text or Speech
- Text segmentation is the process of dividing written text into meaningful units, such as words, sentences, or topics.
- The term applies both to mental processes used by humans when reading text, and to artificial processes implemented in computers, which are the subject of natural language processing.

Problems in Text Segmentation

- **Word segmentation** is the problem of dividing a string of written language into its component words.
For example, ice box = ice-box = icebox
- **Intent segmentation** is the problem of dividing written words into key phrases
[All things are made of *atoms*]
- **Sentence segmentation** is the problem of dividing a string of written language into its component sentences
- **Topic Segmentation** consists of two main tasks:
 - ✗ topic identification
 - ✗ text segmentation

c. Syntax Structure

- Syntactic analysis takes an input sentence and produces a representation of its grammatical structure.
- A grammar describes the valid parts of speech of a language and how to combine them into phrases.
- The English Grammar is nearly context free.
- A computer Grammar specifies which sentences are in a language and their parse trees. A parse tree is a hierarchical structure that shows how the grammar applies to the input. Each level of the tree corresponds to the application of one grammar rule.

Consider the sentence

“John saw Mary with a telescope”

Two different readings based on the groupings.

- John saw (Mary with a telescope).
- John (saw Mary with a telescope).

d. Semantic Structure

- Semantic analysis is a process of converting the syntactic representations into meaning representation.
- This involves the following tasks:
 - Word sense determination
 - Sentence level analysis
 - Knowledge representation

- **Word sense**

Words have different meanings in different contexts.

Example :

- Mary had a bat in her office.
 - bat = `a baseball thing'
 - bat = `a flying mammal'

- **Sentence level analysis**

Once the words are understood, the sentence must be assigned some meaning

- I saw an astronomer with a telescope.

- **Knowledge Representation**

Understanding language requires lots of knowledge.

e. Pragmatic Analysis

- Pragmatics is a subfield of linguistics that studies the ways in which context contributes to meaning
- It is the study of the ways in which language is used and its effect on the listener
- Pragmatic analysis refers to a set of linguistic and logical tools with which analysts develop systematic accounts of logical interactions.

Example

“When is the next flight to Sydney?”

“Does *it* have any seat left?”

Here, “it”, refers to a particular flight to Sydney, not Sydney itself.

Pragmatics – Ambiguity in Language

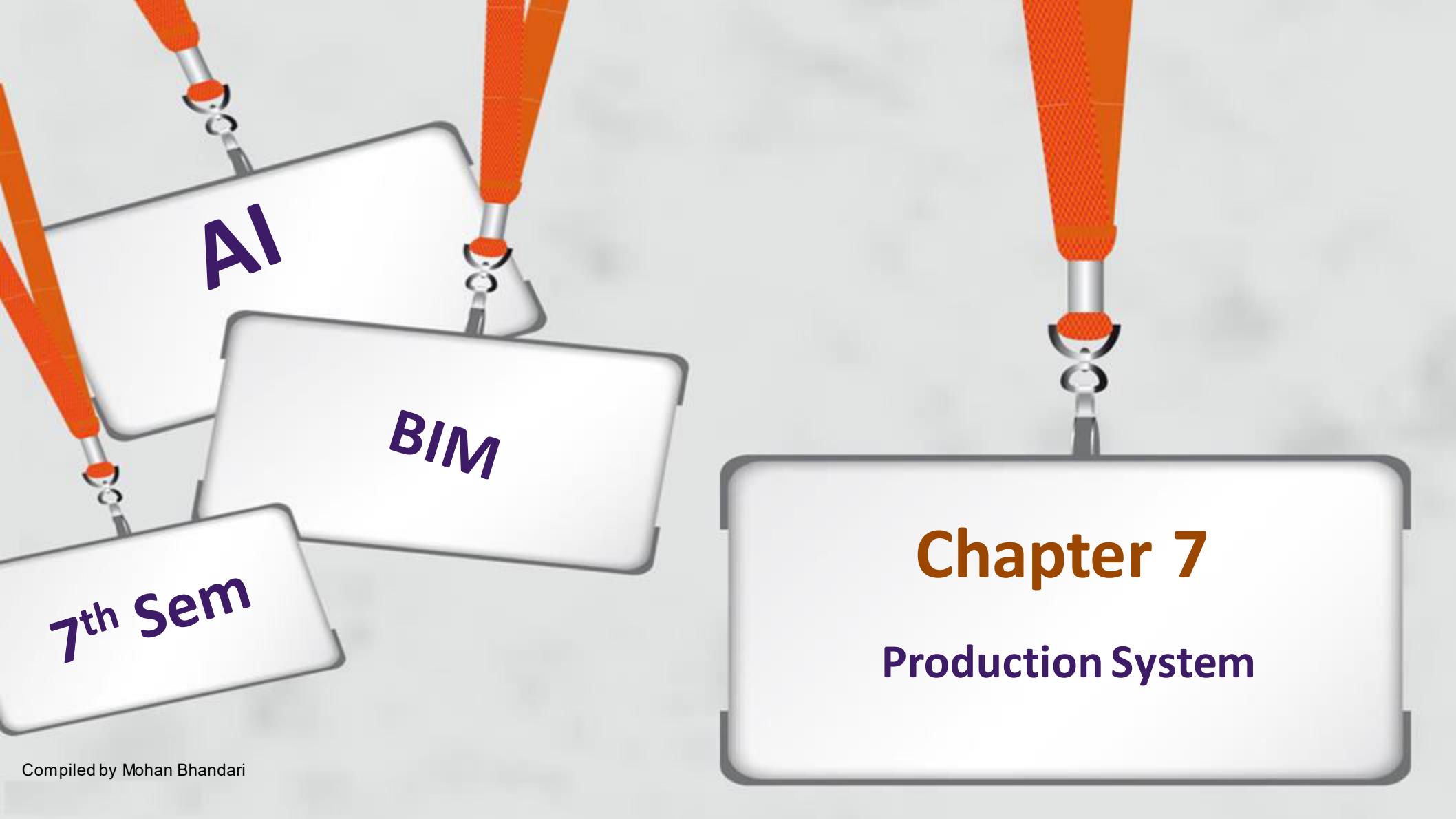
“I saw an astronomer with a telescope.”

Above sentence **has a prepositional phrase** “with a telescope” which may be attached with either with verb to make phrase “saw something with telescope” or to object noun phrase to make phrase “a astronomer with a telescope”. If we do first, then it can be interpreted as “I saw an astronomer who is having a telescope”, and if we do second, it can be interpreted as “Using a telescope I saw an astronomer”

Now, to remove such ambiguity, one possible idea is that we have to consider the context. If the knowledge base (KB) can prove that whether the telescope is with astronomer or not, then the problem is solved.

If A says the same sentence “I saw an astronomer with a telescope.” To B, then in practical, it is more probable that, B (listener) realizes that “A has seen astronomer who is having a telescope”. It is because, normally, the word “telescope” belongs to “astronomer”, so it is obvious that B realizes so.

If A has says that “I saw a lady with a telescope.” In this case, B realizes that “A has seen the lady using a telescope”, because the word “telescope” has not any practical relationship with “lady” like “astronomer”.



Chapter 7

Production System

Production System

Knowledge representation formalism consists of collections of condition-action rules. A system that uses this rule of knowledge representation is called a production system.

A production system consists of rules and factors. Knowledge is encoded in a declarative form which comprises of a set of rules of the form

Situation ----- Action

→ SITUATION that implies ACTION.

Example:-

IF the initial state is a goal state THEN quit.

In simpler terms, it is said to contain a number of rules which are defined by their Left Hand Side (LHS) and Right Hand Side (RHS).

The LHS contains the necessary conditions to be met if the rule is to be applied, while the RHS denotes the outcome of each applicability of the rule. This form of operation is analogous to the popular IF-THEN structure of programming, wherein the IF and THEN correspond to the LHS and RHS respectively.

Productions consist of two parts:

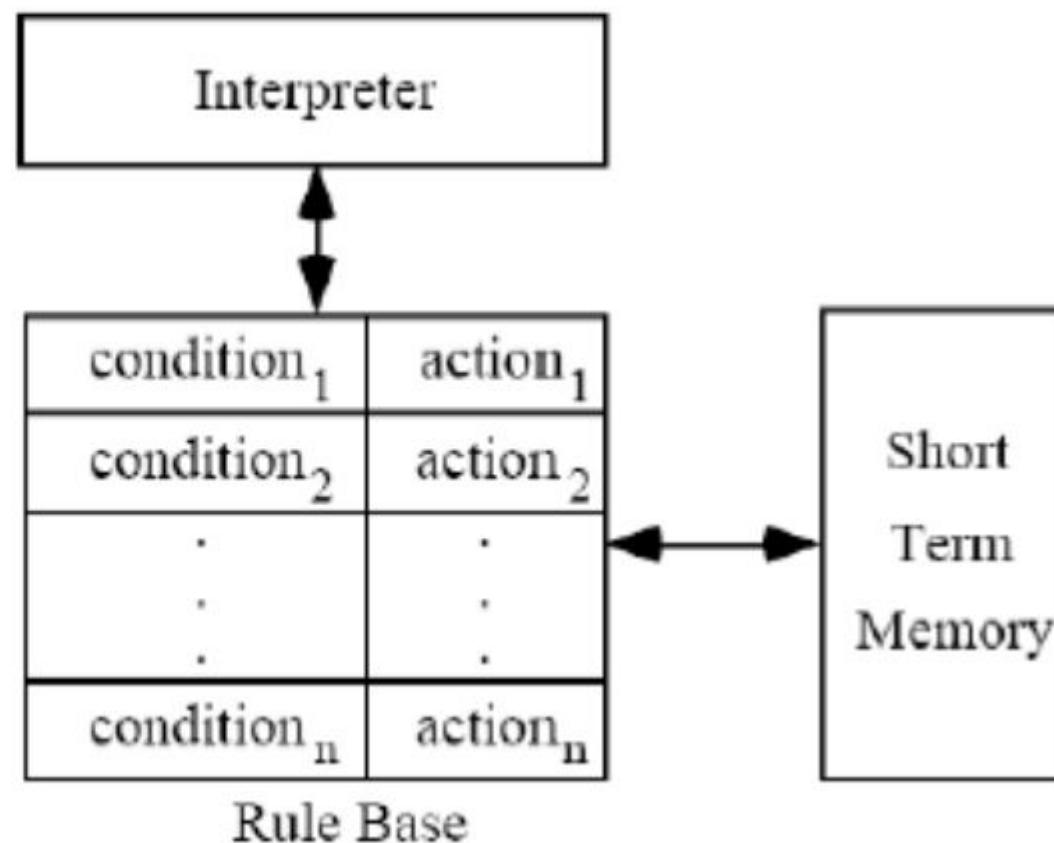
- a sensory precondition (or "IF" statement)
- an action (or "THEN").

If a production's precondition matches the current state of the world, then the production is said to be triggered. If a production's action is executed, it is said to have fired.

Assignment

Explain Different Types of Production Systems.

Architecture of Production System



A production system consists of four basic components:

1. **A set of rules** of the form $C_i \rightarrow A_i$ where C_i is the condition part and A_i is the action part. The condition determines when a given rule is applied, and the action determines what happens when it is applied.
2. One or more **knowledge databases** that contain whatever information is relevant for the given problem. Some parts of the database may be permanent, while others may temporary and only exist during the solution of the current problem. The information in the databases may be structured in any appropriate manner.
3. **A control strategy/ rule interpreter** that determines the order in which the rules are applied to the database, and provides a way of resolving any conflicts that can arise when several rules match at once.
4. **A rule applier** which is the computational system that implements the control strategy and applies the rules.

- Strong methods are those **designed to address a specific type of problem**. They depend on a system being given a great deal of knowledge about its world and the problems that it might encounter
- Weak methods are **general approaches that may be applied to many types of problems**. They use systems such as logic, automated reasoning, and other general structures that can be applied to a wide range of problem

Advantages of production systems

1. Production systems provide an excellent tool for structuring AI programs.
2. Production Systems are highly modular because the individual rules can be added, removed or modified independently.
3. The production rules are expressed in a natural form, so the statements contained in the knowledge base should be a recording of an expert thinking out loud.

Disadvantages of production system

1. **Opacity:** This problem is generated by the combination of production rules. The opacity is generated because of less prioritization of rules. More priority to a rule has the less opacity.
2. **Absence of learning:** Rule based production systems do not store the result of the problem for future use. Hence, it does not exhibit any type of learning capabilities. So for each time for a particular problem, some new solutions may come.
3. **Conflict resolution:** The rules in a production system should not have any type of conflict operations. When a new rule is added to a database, it should ensure that it does not have any conflicts with the existing rules.
4. It may be very difficult analyze the flow of control within a production system because the individual rules don't call each other.

Inference Methods

1. Deductive reasoning
2. Inductive reasoning
3. Abductive reasoning

Deductive Reasoning

Deductive reasoning starts with the assertion of a general rule and proceeds from there to a guaranteed specific conclusion. Deductive reasoning moves from the general rule to the specific application: In deductive reasoning, if the original assertions are true, then the conclusion must also be true.

Abductive Reasoning:

Abductive reasoning is to abduces (or take away) a logical assumption (or explanation, inference, conclusion, hypothesis, best guess) from an observation or set of observations. Because the conclusion is merely a best guess, the conclusion that is drawn may or may not be true.

Inductive reasoning:

The term "inductive reasoning" refers to reasoning that takes specific information and **makes a broader generalization** that is considered probable, allowing for the fact that the conclusion may not be accurate.

Recognize-Act Cycle (refer chapter 7)

Conflict resolution strategies:

Conflict resolution strategies are used in production systems in artificial intelligence, such as in rule-based expert systems, to **help in choosing which production rule to fire**. The need for such a strategy arises when the conditions of two or more rules are satisfied by the currently known facts.

Conflict resolution strategies fall into several main categories:

1. **Specificity** - If all of the conditions of two or more rules are satisfied, choose the rule according to how specific its conditions are. It is possible to favor either the more general or the more specific case.[1]The most specific may be identified roughly as the one having the greatest number of preconditions.
2. **Recency** - When two or more rules could be chosen, favor the one that matches the most recently added facts, as these are most likely to describe the current situation.
3. **Not previously used** - If a rule's conditions are satisfied, but previously the same rule has been satisfied by the same facts, ignore the rule. This helps to prevent the system from entering infinite loops.
4. **Order:** -
 - First in First Serve
The rule applied will be the first rule that is matched. Pick the first applicable rule in order of presentation. This is the strategy that Prolog interpreters use by default.
 - Last in First Serve
The rule applied will be the last rule that is matched.

- 5. Prioritization** - The rule to apply will be selected based on priorities set on rules, with priority information usually provided by an expert or knowledge engineer.
- 6. Arbitrary choice** - Pick a rule at random. This has the merit of being simple to compute.

Forward Chaining, Backward Chaining

See Chapter 7

Assignment

List the advantages and disadvantages of forward and backward chaining

Production System example:

Problem: Sorting a string composed of letters a, b & c.

Short Term Memory: cbaca

Production Set:

$$1. \text{ ba} \longrightarrow \text{ab}$$

$$2. \text{ ca} \longrightarrow \text{ac}$$

$$3. \text{ cb} \longrightarrow \text{bc}$$

Interpreter: Choose one rule according to some strategy

Iteration #	Memory	Conflict Set	Rule fired
0	cbaca	1, 2, 3	1
1	cabca	2	2
2	acbca	2, 3	2
3	acbac	1, 3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	0	halt

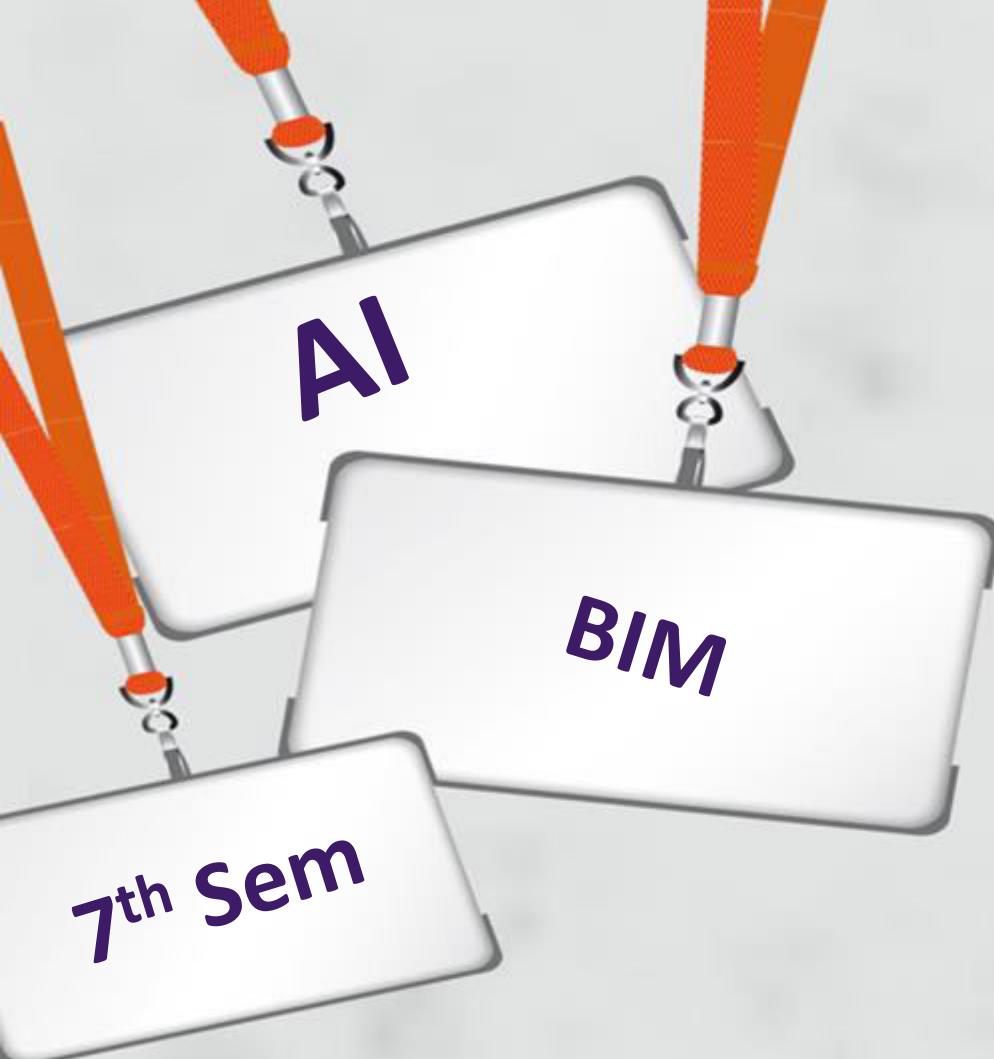
Q. Consider the following production system characterized by

Initial short term memory: c5c1c3

Production Rule:

1. $c1c2 \longrightarrow c4$
2. $c3 \longrightarrow c2$
3. $c1c3 \longrightarrow c6$
4. $c4 \longrightarrow c6$
5. $c5 \longrightarrow c1$

Show a possible sequence of act cycles. Which will be the new content of short term memory?



Chapter 8

Uncertainty in AI

Uncertainty refers to the situations involving imperfect or unknown information.

It applies to predictions of future events, to physical measurements that are already made, or to the unknown.

Uncertainty arises in partially observable and/or stochastic environments, as well as due to ignorance, idleness, or both.

Fuzzy Logic:

Fuzzy logic is an **approach to computing based on “degree of truth” rather than the usual “true or false” (1 or 0) Boolean logic** on which the modern computer is based.

Fuzzy Logic (FL) is a method of reasoning that resembles human reasoning. **The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO.**

The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human's YES or NO. The inventor of fuzzy logic, observed that unlike computers, the human decision making includes a range of possibilities between YES and NO, such as:

CERTAINLY YES
POSSIBLY YES
CANNOT SAY
POSSIBLY NO
CERTAINLY NO

Why Fuzzy Logic?

Unlike two-valued Boolean logic, fuzzy logic is multi-valued. It deals with degrees of membership and degrees of truth. Fuzzy logic uses the range of logical values between 0 (completely false) and 1 (completely true). Instead of just black and white, it employs the spectrum of colors, accepting that things can be partly true and partly false at the same time. As can be seen in Figure below, fuzzy logic adds a range of logical values to Boolean logic.

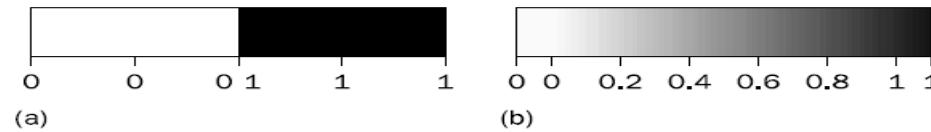


Fig: Range of logical values in Boolean and fuzzy logic: (a) Boolean logic; (b) multi-valued logic

Fuzzy logic is useful for commercial and practical purposes.

- It can control machines and consumer products.
- It may not give accurate reasoning, but acceptable reasoning.
- Fuzzy logic helps to deal with the uncertainty in engineering.

fuzzy set

The basic idea of the fuzzy set theory is that an element belongs to a fuzzy set with a certain degree of membership.

Thus, a proposition is not either true or false, but may be partly true (or partly false) to any degree. This degree is usually taken as a real number in the interval [0, 1].

A fuzzy set can be simply **defined as a set with fuzzy boundaries**.

Let X be the universe of discourse and its elements be denoted as x . In classical set theory, classical set A of X is defined as function $f_A(x)$ called the **characteristic function of A**:

$$f_A(x) : X \rightarrow \{0, 1\},$$

where

$$f_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$$

This set maps universe X to a set of two elements. For any element x of universe X , characteristic function $f_A(x)$ is equal to 1 if x is an element of set A , and is equal to 0 if x is not an element of A .

In the fuzzy theory, fuzzy set A of universe X is defined by function $\mu_A(x)$ called the membership function of set A

$$\mu_A(x) : X \rightarrow [0, 1],$$

where

$$\mu_A(x) = 1 \text{ if } x \text{ is totally in } A;$$

$$\mu_A(x) = 0 \text{ if } x \text{ is not in } A;$$

$$0 < \mu_A(x) < 1 \text{ if } x \text{ is partly in } A.$$

Membership Functions

Membership functions allow us to **quantify linguistic term** and represent a fuzzy set graphically.

A membership function for a fuzzy set A on the universe of discourse X is defined as $\mu_A: X \rightarrow [0, 1]$.

Here, each element of X is mapped to a value between 0 and 1. It is called membership value or degree of membership.

It quantifies the degree of membership of the element in X to the fuzzy set A.

- x axis represents the universe of discourse.
- y axis represents the degrees of membership in the [0, 1] interval.

Operations of fuzzy sets

The classical set theory developed in the late 19th century by Georg Cantor describes how crisp sets can interact. These interactions are called operations.

We look at three of them: complement, intersection and union.

Complement

Crisp sets: Who does not belong to the set?

Fuzzy sets: How much do elements not belong to the set?

The complement of a set is an opposite of this set.

For example, if we have the set of tall men, its complement is the set of NOT tall men. When we remove the tall men set from the universe of discourse, we obtain the complement. If A is the fuzzy set, its complement $\sim A$ can be found as follows:

$$\mu_{\sim A}(x) = 1 - \mu_A(x)$$

For example, if we have a fuzzy set of *tall men*, we can easily obtain the fuzzy set of NOT *tall men*:

$$\textit{tall men} = (0/180, 0.25/182.5, 0.5/185, 0.75/187.5, 1/190)$$

$$\text{NOT } \textit{tall men} = (1/180, 0.75/182.5, 0.5/185, 0.25/187.5, 0/190)$$

Intersection

Crisp sets: Which element belongs to both sets?

Fuzzy sets: How much of the element is in both sets?

In classical set theory, an intersection between two sets contains the elements shared by these sets.

If we have, for example, the set of tall men and the set of fat men, the intersection is the area where these sets overlap, i.e. Tom is in the intersection only if he is tall AND fat. In fuzzy sets, however, an element may partly belong to both sets with different memberships.

Thus, a fuzzy intersection is the lower membership in both sets of each element. The fuzzy operation for creating the intersection of two fuzzy sets A and B on universe of discourse X can be obtained as:

$$\mu_{A \cap B}(x) = \min [\mu_A(x), \mu_B(x)] = \mu_A(x) \cap \mu_B(x), \quad \text{where } x \in X$$

Consider, for example, the fuzzy sets of *tall* and *average men*:

$$\textit{tall men} = (0/165, 0/175, 0.0/180, 0.25/182.5, 0.5/185, 1/190)$$

$$\textit{average men} = (0/165, 1/175, 0.5/180, 0.25/182.5, 0.0/185, 0/190)$$

According to above equations, the intersection of these two sets is

$$\textit{tall men} \cap \textit{average men} = (0/165, 0/175, 0/180, 0.25/182.5, 0/185, 0/190)$$

Union

Crisp sets: Which element belongs to either set?

Fuzzy sets: How much of the element is in either set?

The union of two crisp sets consists of every element that falls into either set.

For example, the union of tall men and fat men contains all men who are tall OR fat, i.e. Tom is in the union since he is tall, and it does not matter whether he is fat or not.

In fuzzy sets, the union is the reverse of the intersection. That is, the union is the largest membership value of the element in either set. The fuzzy operation for forming the union of two fuzzy sets A and B on universe X can be given as:

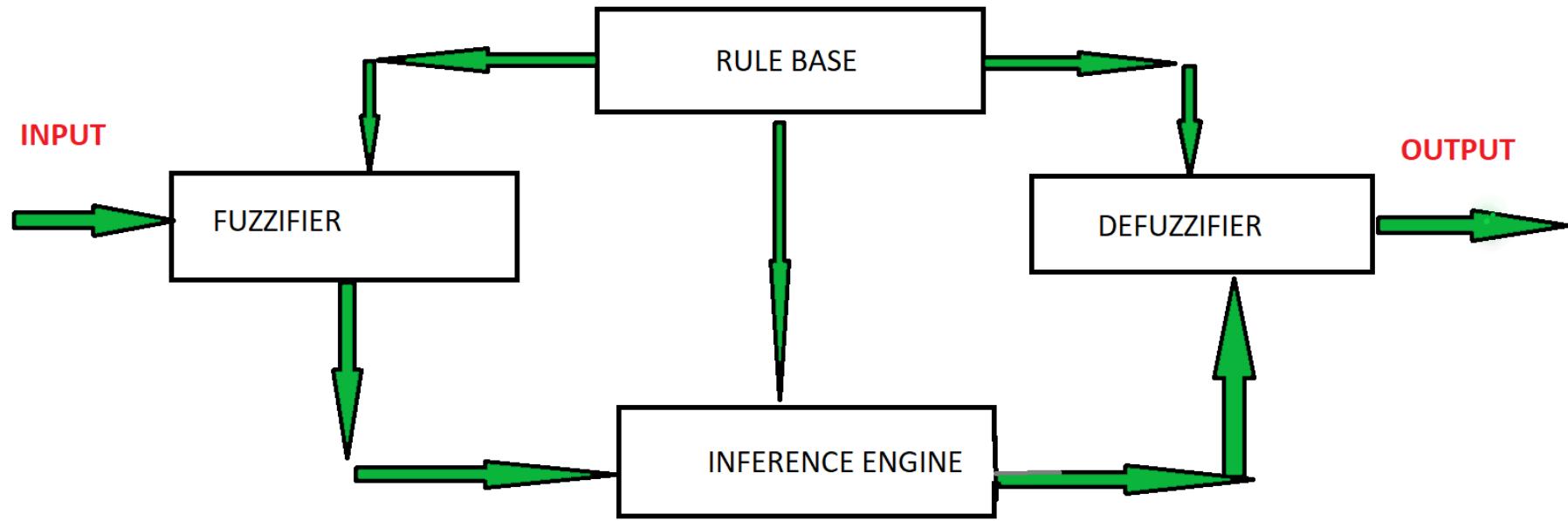
$$\mu_{A \cup B}(x) = \max [\mu_A(x), \mu_B(x)] = \mu_A(x) \cup \mu_B(x), \quad \text{where } x \in X$$

Consider again the fuzzy sets of *tall* and *average* men:

$$\textit{tall men} = (0/165, 0/175, 0.0/180, 0.25/182.5, 0.5/185, 1/190)$$

$$\textit{average men} = (0/165, 1/175, 0.5/180, 0.25/182.5, 0.0/185, 0/190)$$

$$\textit{tall men} \cup \textit{average men} = (0/165, 1/175, 0.5/180, 0.25/182.5, 0.5/185, 1/190)$$



FUZZY LOGIC ARCHITECTURE

RULE BASE:

It contains the set of rules and the IF-THEN conditions provided by the experts to govern the decision making system, on the basis of linguistic information.

FUZZIFICATION:

It is used to convert inputs i.e. crisp numbers into fuzzy sets. Crisp inputs are basically the exact inputs measured by sensors and passed into the control system for processing, such as temperature, pressure, rpm's, etc.

INFERENCE ENGINE:

It determines the matching degree of the current fuzzy input with respect to each rule and decides which rules are to be fired according to the input field.

DEFUZZIFICATION:

It is used to convert the fuzzy sets obtained by inference engine into a crisp value.

If-Then Rules

Fuzzy sets and fuzzy operators are the subjects and verbs of fuzzy logic. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic.

A single fuzzy if-then rule assumes the form

if x is A then y is B

where, A and B are linguistic values defined by fuzzy sets on the ranges (universes of discourse) X and Y, respectively. The if-part of the rule "x is A" is called the antecedent or premise, while the then-part of the rule "y is B" is called the consequent or conclusion. An example of such a rule might be

If service is good then tip is average

Note that good is represented as a number between 0 and 1, and so the antecedent is an interpretation that returns a single number between 0 and 1.

Interpreting if-then rules is a three-part process.

- **Fuzzify inputs:**

Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1.

If there is only one part to the antecedent, this is the **degree of support for the rule**.

- **Apply fuzzy operator to multiple part antecedents:**

If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 and 1. This is the degree of support for the rule.

- **Apply implication method:**

Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output.

Fuzzy Inference Methods

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic.

There are two types of fuzzy inference methods:

- Mamdani inference method
- Sugeno fuzzy inference method

Mamdani inference method:

Mamdani fuzzy inference is the most commonly seen inference method. This method was introduced by Mamdani and Assilian (1975).

Mamdani Fuzzy model

1. Determining a set of fuzzy rules
2. Fuzzifying the inputs using the input membership functions
3. Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength (Fuzzy Operations)
4. Finding the consequence of the rule by combining the rule strength and the output membership function (implication)
5. Combining the consequences to get an output distribution (aggregation)
6. Defuzzifying the output distribution (this step is only if a crisp output (class) is needed).

Takagi-Sugeno Fuzzy Model (TS Method)

This model was proposed by Takagi, Sugeno and Kang in 1985. Format of this rule is given as –

$$\text{IF } x \text{ is } A \text{ and } y \text{ is } B \text{ THEN } Z = f(x,y)$$

Here, A, B are fuzzy sets in antecedents and $Z = f(x,y)$ is a crisp function in the consequent.

Fuzzy Inference Process

The fuzzy inference process under Takagi-Sugeno Fuzzy Model (TS Method) works in the following way –

- **Step 1: Fuzzifying the inputs** – Here, the inputs of the system are made fuzzy.
- **Step 2: Applying the fuzzy operator** – In this step, the fuzzy operators must be applied to get the output.

Assignment

Compare Takagi-Sugeno Fuzzy Model VS Mamdani Fuzzy Inference System

Conditional Probability and Bayes Theorem

Let A and B are two dependent events then the probability of the event A when the event B has already happened is called the conditional probability.
Conditional probability is the probability of one event occurring with some relationship to one or more other events.

For example:

Event A is that it is raining outside, and it has a 0.3 (30%) chance of raining today.
Event B is that you will need to go outside, and that has a probability of 0.5 (50%)

A conditional probability would look at these two events in relationship with one another, such as the probability that it is both raining *and* you will need to go outside.

The formula for conditional probability is:

$$P(B | A) = P(A \cap B) / P(A)$$

Example 1

In a group of 100 sports car buyers, 40 bought alarm systems, 30 purchased bucket seats, and 20 purchased an alarm system and bucket seats. If a car buyer chosen at random bought an alarm system, what is the probability they also bought bucket seats?

Step 1:

Figure out $P(A)$. It's given in the question as 40%, or 0.4.

Step 2:

Figure out $P(A \cap B)$. This is the intersection of A and B: both happening together.

It's given in the question 20 out of 100 buyers, or 0.2.

Step 3:

$$\begin{aligned}P(B|A) &= P(A \cap B) / P(A) \\&= 0.2 / 0.4 = 0.5.\end{aligned}$$

Bayes rule

Let A and B are two dependent events then the probability of the event A when the event B has already happened is called the conditional probability. It is denoted by $P(A|B)$ and is given by:

$$P(A|B) = P(A \cap B)/P(B), \text{ where } P(B) \neq 0, \Rightarrow P(A \cap B) = P(A|B) \cdot P(B) \quad \text{--- (i)}$$

Similarly,

$$P(B|A) = P(A \cap B)/P(A), \text{ where } P(A) \neq 0, \Rightarrow P(A \cap B) = P(B|A) \cdot P(A) \quad \text{--- (ii)}$$

From equation (i) and (ii), we have

$$P(B|A) \cdot P(A) = P(A|B) \cdot P(B)$$

Bayes rule is useful for those cases where $P(A|B)$ can be estimated but $P(B|A)$ is hard to find experimentally

- In a task such as medical diagnosis, we often have conditional probabilities on causal relationships and want to derive a diagnosis.
- A doctor knows the probability of symptoms condition to disease $P(S|D)$, and the patient knows his own feeling or symptoms $P(S)$. Also the doctor knows about probability of disease $P(D)$, then the probability of disease condition to symptoms can be defined as:

For example:

Let,

- S = Symptoms on patients such as stiff neck whose probability $P(S)$ is $= 1/20$
- D = Disease known by doctor whose probability $P(D)$ is $= 1/50000$
- Given $P(S|D) = 0.5$ or 50%

Now, the probability of disease condition to symptoms,

$P(D|S)$

$$= [P(S|D) \cdot P(D)] / P(S)$$

$$= 0.0002$$

Q. At a certain University, 4% of men are over 6 feet tall and 1% of women are 6 Feet tall. The total student population is divided in the ratio 3:2. If a student is selected at random from among all those over fix feet tall, what is the probability that the selected student is woman.

Let

m = men student

w = women student

t = student over 6 feet tall

We know

probability of male student, $p(m) = 2/5 = 0.4$

probability of women student, $p(w) = 3/5 = 0.6$

Also, $p(t/m)= 4\% = 0.04$

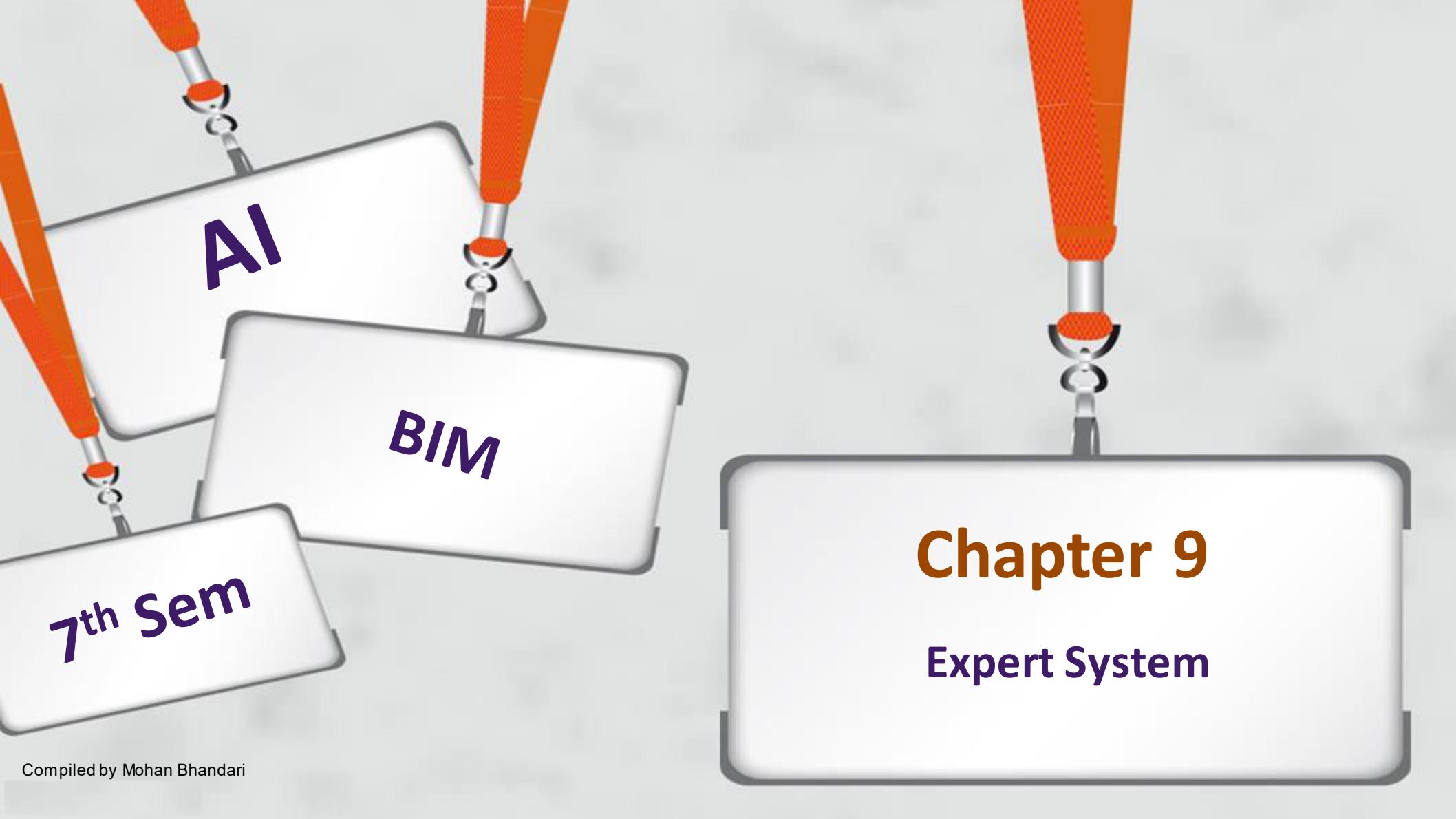
$p(t/w)= 1\% = 0.01$

Using Bayes theorem

$$\begin{aligned} p(w/t) &= (p(t/w).p(w)) / (p(t/w).p(w) + p(t/m).p(m)) \\ &= 3/11 \end{aligned}$$

Application of Bayes Theorem

- In Manufacturing Process, Selecting best products among two manufacturers
- In bio-chemistry, deciding the diseases based on various blood sample tests. In fact those results are based on probability, so it never be 100% true.
- For project managers : All project managers want to know whether the projects they're working on will finish on time. So, as our example, we'll assume that a project manager asks the question: what's the probability that my project will finish on time? There are only two possibilities here: either the project finishes on (or before) time or it doesn't on the basis of various factors like tools, number of workers, efficiency of workers..



Chapter 9

Expert System

7th Sem

BIM

AI



Just A Real Virtual Intelligent System

It recognize voice commands to do something, even for locking door and turn off the lamp.

MYCIN would attempt to diagnose patients based on reported symptoms and medical test results.

The program could request further information concerning the patient, as well as suggest additional laboratory tests, to arrive at a probable diagnosis, after which it would recommend a course of treatment.

If requested, MYCIN would explain the reasoning that led to its diagnosis and recommendation.

CALEX is user friendly computer program developed in California, USA, that simulates human problem solving behaviour. This computer program combines database management, regression models, simulation models and rule bases (a rule base is an "advisor").

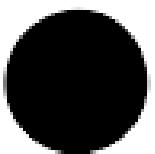
It is a blackboard based integrated system for agricultural management, developed at University of California.

CALEX can be used by growers, pest control advisors, consultants and other managers.

DENDRAL's primary aim was to help organic chemists in identifying unknown organic molecules, by analyzing their mass spectra and using knowledge of chemistry.

Emycin is an *expert system*, a framework for building programs that record the knowledge of *domain experts* and use that knowledge to help non-expert users solve problems.

It provides an interface that helps experts define data types and rules, a backwards-chaining reasoning algorithm, a mechanism for dealing with uncertainty that permit users to learn what the system knows and what it is doing.



The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.

An expert system is a computer system **whose performance is guided by specific, expert knowledge in solving problems**. It is a computer system that simulates the decision-making process of a human expert in a specific domain.

Expert system is one of the early (large-scale) successes of artificial intelligence. An expert system is an “intelligent” program that solves problems in a narrow problem area by using high-quality, specific knowledge rather than an algorithm. Expert systems are used by most of the large or medium sized organization as a major tool for improving productivity and quality. An expert system’s knowledge is obtained from expert sources and code in a form suitable for the system to use in its process.

For example, there are expert systems that can diagnose human illnesses, make financial forecasts, and schedule routes for delivery vehicles. The expert system is a branch of AI designed to work within a particular domain.

Characteristics of Expert Systems

- High performance
- Reliable
- Highly responsive

Capabilities of Expert Systems

Capabilities	Incapabilities
Advising	Possessing human capabilities
Instructing and assisting human in decision making	Substituting human decision makers
Deriving a solution	Producing accurate output for inadequate knowledge base
Predicting results	Refining their own knowledge

Human Expert Vs Expert System

Factor	Human Expert	Expert System
Time (can be obtained)	Working days only	Anytime
Geography	Local	Anywhere
Safety	Cannot be replaced	Can be replaced
Damages	Yes	No
Speed and Efficiency	Changes	Consistent
Cost	High	Intermediate

Components of Expert Systems

The components of ES include -

- Knowledge Base
- Inference Engine
- User Interface

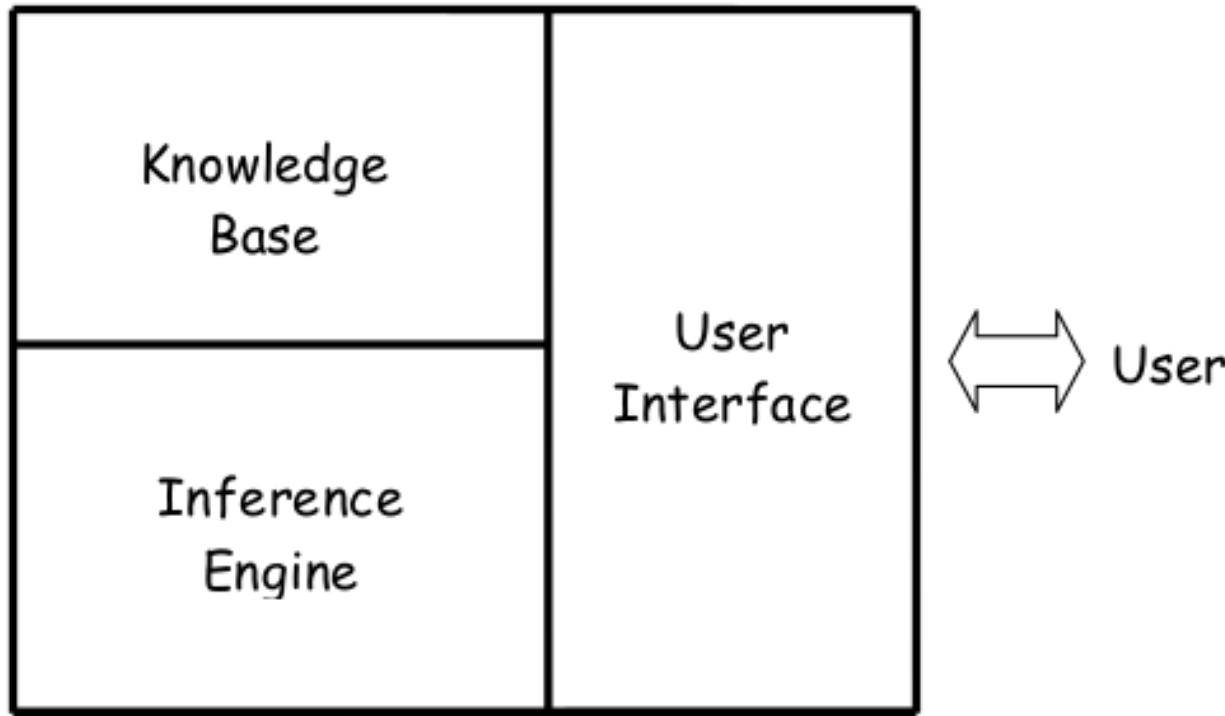


Fig: Block Diagram of expert system

Knowledge Base

- Knowledge is required to exhibit intelligence.
- The success of any ES depends upon the collection of highly accurate and precise knowledge.
- **Components of Knowledge Base**
 - The knowledge base of an ES is a store of both, factual and heuristic knowledge.
- **Factual Knowledge** – It is the information widely accepted by the Knowledge Engineers and scholars in the task domain.
- **Heuristic Knowledge** – It is about practice, accurate judgment, one's ability of evaluation, and guessing.
- **Episodic Knowledge** - Experiential learning is the process of learning through experience, and is more specifically defined as "learning through reflection on doing"
- **Meta-knowledge** – Knowledge about knowledge

Sources of Knowledge

- **Expert**: primary source.
- **End Users**.
 - Usually have a good overview of the problem domain.
 - May provide valuable insight during initial investigations.
- **Secondary / Tertiary Experts**
 - Can provide specialized knowledge on sub-problems.
 - May give rise to conflicting advice.
- **Literature**
 - Reports, Guidelines, Books, Manuals etc.
 - Provide background and insight in early stages

Inference Engine

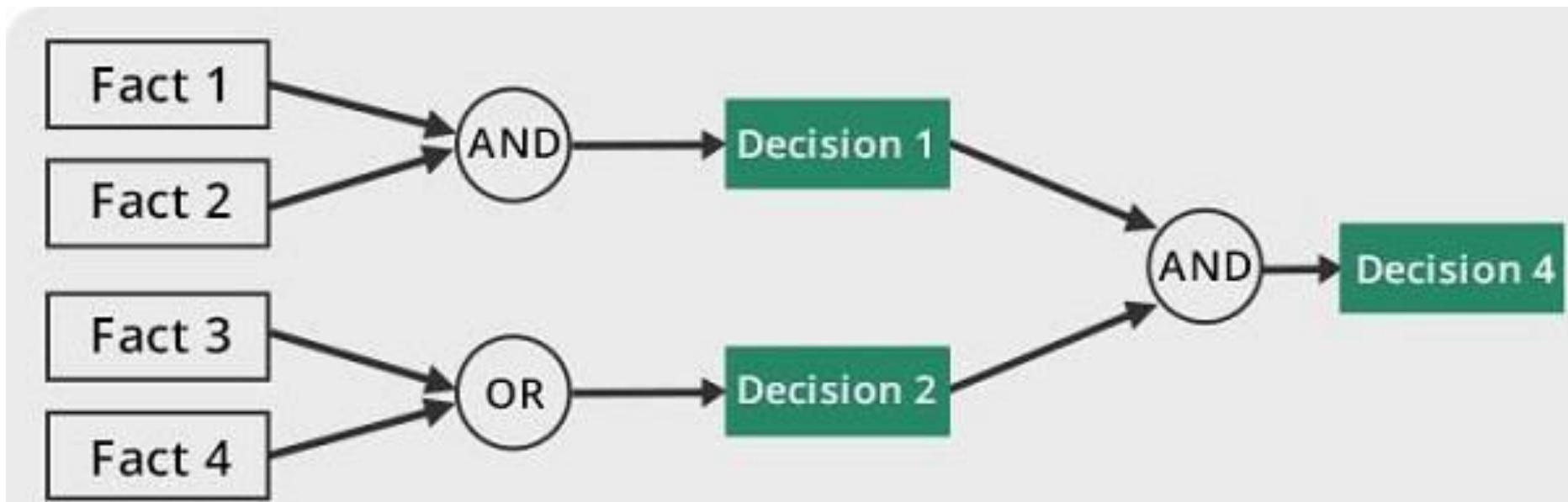
In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.

In case of rule based ES, it –

- Applies rules repeatedly to the facts, which are obtained from earlier rule application.
- Adds new knowledge into the knowledge base if required.
- Resolves rules conflict when multiple rules are applicable to a particular case.
- To recommend a solution, the Inference Engine uses the following strategies –
 - Forward Chaining
 - Backward Chaining

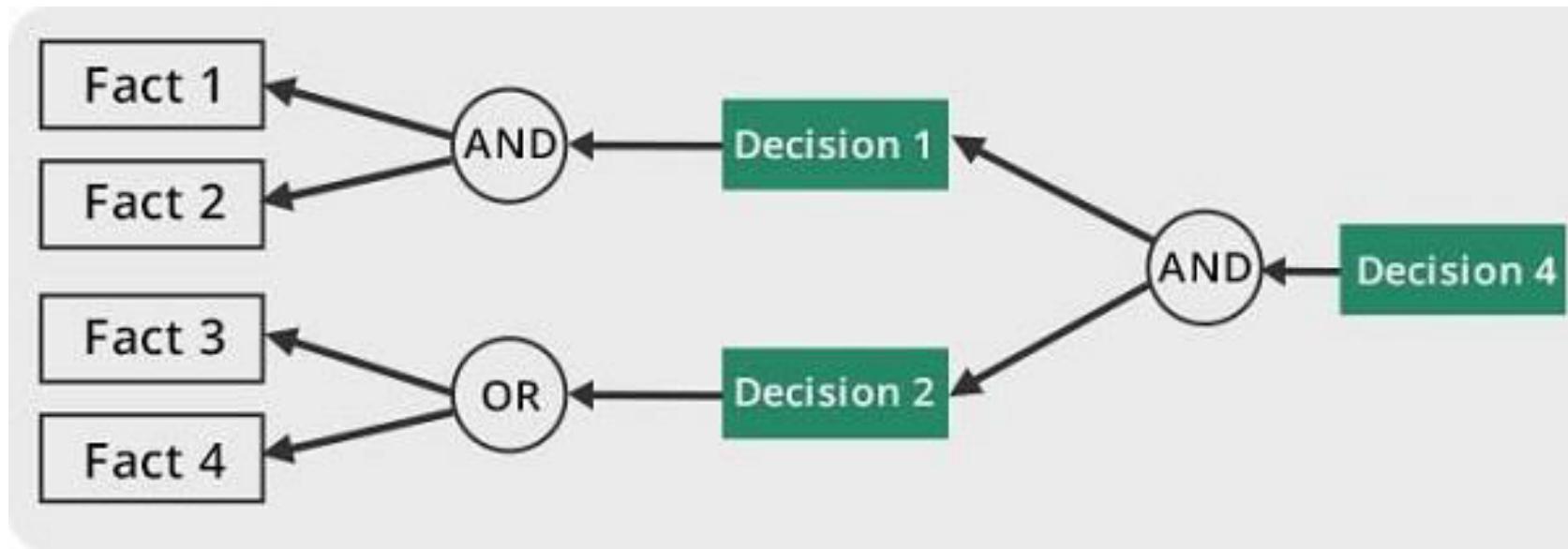
Forward Chaining

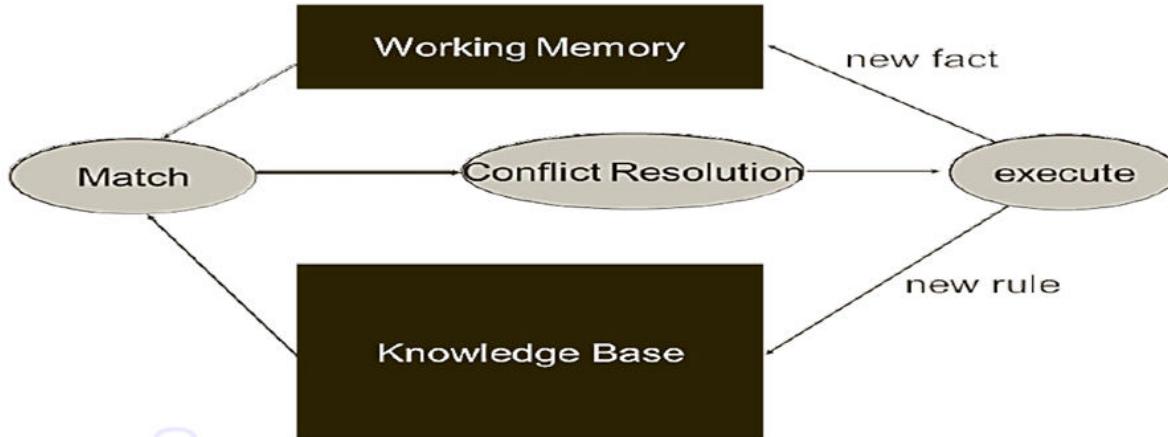
It is a strategy of an expert system to answer the question, “What can happen next?”



Backward Chaining

With this strategy, an expert system finds out the answer to the question, “Why this happened?”





Recognize-Select-Act Cycle

It consists of 3 steps:

1. Match: Rules are compared to working memory to determine matches
2. Conflict Resolution: Select or enable a single rule for execution
3. Execute: Fire the selected rule

User Interface

- User interface provides interaction between user of the ES and the ES itself.
- It is generally Natural Language Processing so as to be used by the user who is well-verses in the task domain.
- The user of the ES need not be necessarily an expert in Artificial Intelligence.

Expert Systems Limitations

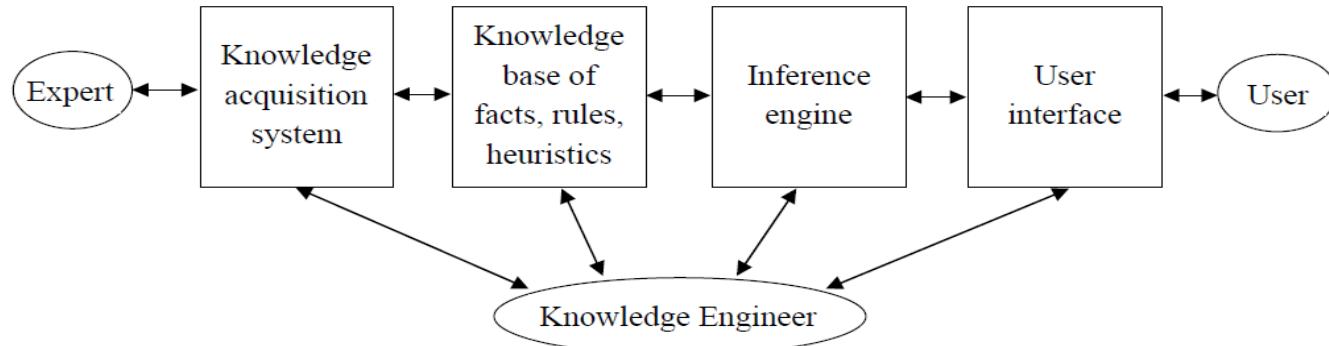
- Limitations of the technology
- Difficult knowledge acquisition
- ES are difficult to maintain
- High development costs

Applications of Expert System

Application	Description
Design Domain	Camera lens design, automobile design.
Medical Domain	Diagnosis Systems to deduce cause of disease from observed data, conduction medical operations on humans.
Monitoring Systems	Comparing data continuously with observed system or with prescribed behavior such as leakage monitoring in long petroleum pipeline.
Process Control Systems	Controlling a physical process based on monitoring.
Knowledge Domain	Finding out faults in vehicles, computers.
Finance/Commerce	Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

The Architecture of Expert Systems

The process of building expert systems is often called knowledge engineering. The knowledge engineer is involved with all components of an expert system:



Building expert systems is generally an iterative process. The components and their interaction will be refined over the course of numerous meetings of the knowledge engineer with the experts and users.

Knowledge Acquisition

The knowledge acquisition component allows the expert to enter their knowledge or expertise into the expert system, and to refine it later as and when required.

Historically, the knowledge engineer played a major role in this process, but automated systems that allow the expert to interact directly with the system are becoming increasingly common.

The knowledge acquisition process is usually comprised of three principal stages:

1. Knowledge elicitation is the interaction between the expert and the knowledge engineer/program to elicit the expert knowledge in some systematic way.
2. The knowledge thus obtained is usually stored in some form of human friendly intermediate representation.
3. The intermediate representation of the knowledge is then compiled into an executable form (e.g. production rules) that the inference engine can process.

In practice, many iterations through these three stages are usually required.

Knowledge Elicitation

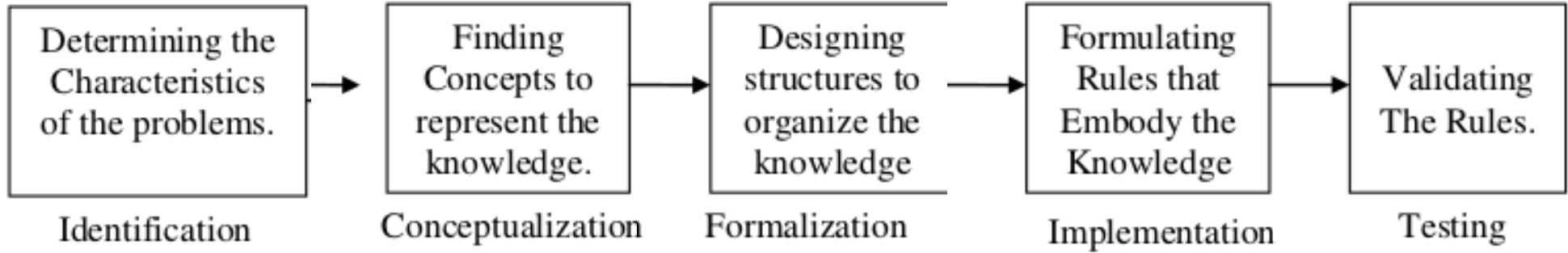
The knowledge elicitation process itself usually consists of several stages:

1. Find as much as possible about the problem and domain from books, manuals, etc.
2. Try to characterize the types of reasoning and problem solving tasks that the system will be required to perform.
3. Find an expert (or set of experts) that is willing to collaborate on the project.
4. Interview the expert (usually many times during the course of building the system). Find out how they solve the problems your system will be expected to solve. Have them check and refine your intermediate knowledge representation.

This is a time intensive process, and automated knowledge elicitation and machine learning techniques are increasingly common modern alternatives.

Stages of Expert System Development:

- An expert system typically is developed and refined over a period of several years.
- We can divide the process of expert system development into five distinct stages. In practice, it may not be possible to break down the expert system development cycle precisely.
- However, an examination of these five stages may serve to provide us with some insight into the ways in which expert systems are developed.



Identification

- The problem must be suitable for an expert system to solve it.
- Find the experts in task domain for the ES project.
- Establish cost-effectiveness of the system.

Conceptualization

- Identify the ES Technology
- Know and establish the degree of integration with the other systems and databases.
- Realize how the concepts can represent the domain knowledge best.

Formalization

- The various techniques of knowledge representation and heuristic search used in expert systems.
- The expert system tools that can greatly assist the development process.
- Other expert systems that may solve similar problems and thus may be adequate to the problem at hand.

Implementation

During the implementation stage, the formalized concepts are programmed onto the computer that has been chosen for system development, using the predetermined techniques and tools to implement a first pass prototype of the expert system.

Testing

- Testing provides opportunities to identify the weakness in the structure and implementation of the system and to make the appropriate corrections.
- Depending on the types of problems encountered, the testing procedure may indicate that the system was

Benefits of Expert Systems

- **Availability** – They are easily available due to mass production of software.
- **Speed** – They offer great speed. They reduce the amount of work an individual puts in.
- **Less Error Rate** – Error rate is low as compared to human errors.
- **Reducing Risk** – They can work in the environment dangerous to humans.
- **Steady response** – They work steadily without getting motional, tensed or fatigued.

Case-Base Reasoning:

- › AI programs solve problems by reasoning from first principles. They can explain their reasoning by reporting the string of deductions that led from the input data to the conclusion, with the Human Experts.
- › An expert encountering a new problem is usually reminded of similar cases seen in the past, remembering the result of those cases and perhaps the reasoning behind those results.

Example: Medical expertise follow this pattern.

- › Computer systems that solve new problems by analogy with old ones are often called Case Base Reasoning (CBR).

A successful CBR systems must answer the following questions.

1. How are cases organized in memory ?
2. How are relevant cases retrieved from the memory ?
3. How can previous cases be adopted to new problems ?
4. How are cases originally acquired ?