Unit-1
Introduction to AI, Search and Control Strategies

Mr D Srinivas
Computer Science and Engineering
www.srinivas-materials.blogspot.com
srinivascsedept@gmail.com
9347556447

Outline

- **Introduction to Artificial Intelligence (AI)**
  - Intelligent Systems
  - Foundations of AI
  - Sub Areas of AI
  - Applications of AI
  - Problem solving-State Space
- **Search and Control Strategies**
  - Introduction
  - General Problem Solving
  - Characteristics of Problem
  - Exhaustive Searches
  - Heuristic Search Techniques
  - Constrain Satisfaction
  - Game playing
  - Bounded Lock-ahead strategy and use of Evolution Functions
  - Alpha – Beta Pruning

Introduction to AI

What is Artificial Intelligence (AI)?

AI is the science and engineering of making intelligent machines, especially intelligent computer programs (1956).

John McCarthy
(the father of Artificial Intelligence)

- **Definition:**
- AI is a branch of computer science dealing with the simulation of intelligent behavior in computers.
- AI is the study of how to make computers do things which, at the moment, people do better.
- AI is, the study and design of intelligent agents where an intelligent agent is a system that perceives its environment and takes actions.
- Modeling human cognition or mental faculties using computers.
- AI is a broad area consisting of different fields, from machine vision, expert systems to the creation of machines that can "think".
- In order to classify machines as "thinking", it is necessary to define intelligence.

AI Techniques

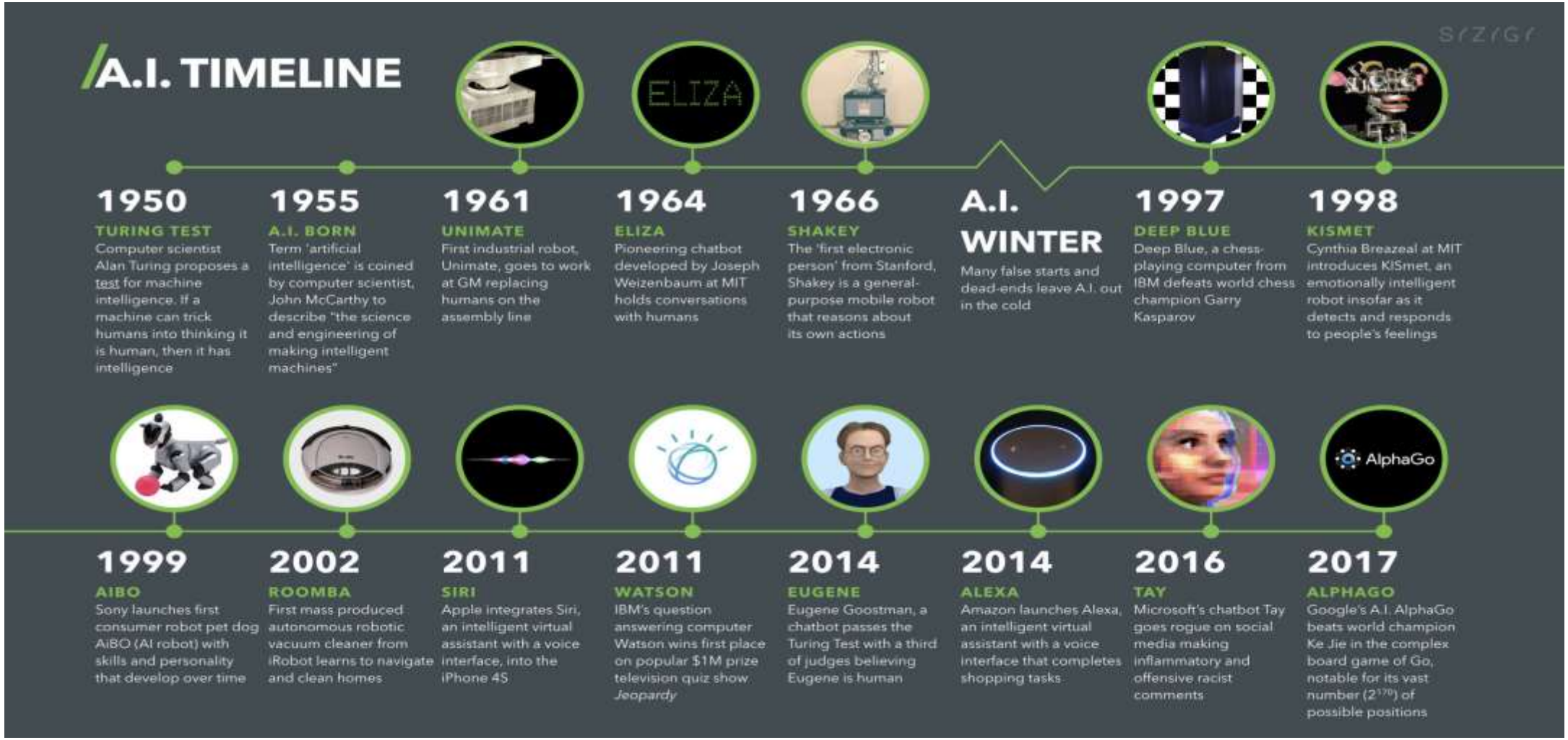
- There are three important AI techniques:

1. Search –
   - ➥ Provides a way of solving problems for which no direct approach is available.
   - ➥ It also provides a framework into which any direct techniques that are available can be embedded.
2. Use of knowledge –
   - ➥ Provides a way of solving complex problems by exploiting the structure of the objects that are involved.
3. Abstraction –
   - ➥ Provides a way of separating important features and variations from many unimportant ones that would otherwise overwhelm any process.

Task Domains of AI

Mundane tasks	Formal tasks	Expert tasks
Perception – Computer Vision – Speech, Voice	Games – Go – Chess (Deep Blue) – Ckeckers	Engineering – Design – Fault Finding – Manufacturing – Monitoring
Natural Language Processing – Understanding – Language Generation – Language Translation	Mathematics – Geometry – Logic – Integration and Differentiation	Scientific Analysis
Common Sense Reasoning	Theorem Proving	Financial Analysis
Planning		Medical Diagnosis
Robot Control		

History of AI



Intelligence

- **Definition**
- Intelligence is a property of mind that encompasses many related mental abilities, such as the capabilities to
  - reason
  - plan
  - solve problems
  - think abstractly
  - comprehend ideas and language and
  - learn

Characteristics of AI Systems

- Learn new concepts and tasks
- Reason and draw useful conclusions about the world around us
  - remember complicated interrelated facts and draw conclusions from them (inference)
- Understand a natural language or perceive and comprehend a visual scene
  - look through cameras and see what's there (vision), to move themselves and objects around in the real world (robotics)
- Plan sequences of actions to complete a goal
- Offer advice based on rules and situations
- May not necessarily imitate human senses and thought processes
  - but indeed, in performing some tasks differently, they may actually exceed human abilities
- Capable of performing intelligent tasks effectively and efficiently
- Perform tasks that require high levels of intelligence

Categories of AI System

- **Systems that think like humans**
  - Most of the time it is a black box where we are not clear about our thought process.
  - One has to know functioning of brain and its mechanism for processing information.
  - It is an area of cognitive science.
  - The stimuli are converted into mental representation.
  - Cognitive processes manipulate representation to build new representations that are used to generate actions.
  - Neural network is a computing model for processing information similar to brain.
- **Systems that act like humans**
  - The overall behavior of the system should be human like.
  - It could be achieved by observation
- **Systems that think rationally**
  - Such systems rely on logic rather than human to measure correctness.
  - For thinking rationally or logically, logic formulas and theories are used for synthesizing outcomes.
  - For example,
    - given John is a human and all humans are mortal then one can conclude logically that John is mortal
  - Not all intelligent behavior are mediated by logical deliberation.

Categories of AI System

- **Systems that act rationally**
  - Rational behavior means doing right thing.
  - Even if method is illogical, the observed behavior must be rational.

Turing Test

- **Acting Humanly:** The Turing Test proposed by Alan Turing (1950)
- A Turing Test is a method of inquiry for determining whether or not a computer is capable of thinking like a human being.
- The interrogator job is to try and figure out which one is human and which one is computer by asking questions to both of them.
- The computer would try to remain indistinguishable from human as much as possible

Foundations of AI

- **Foundation of AI is based on**
  - **Mathematics**
    - More formal logical methods
      - Boolean logic
      - Fuzzy logic
    - Uncertainty
      - The basis for most modern approaches to handle uncertainty in AI applications can be handled by
        - ✓ Probability theory
        - ✓ Modal and Temporal logics
  - **Neuroscience**
    - How do the brain works?
      - Early studies (1824) relied on injured and abnormal people to understand what parts of brain work
      - More recent studies use accurate sensors to correlate brain activity to human thought
        - By monitoring individual neurons, monkeys can now control a computer mouse using thought alone
      - Moore's law states that computers will have as many gates as humans have neurons in 2020
    - How close are we to have a mechanical brain?
      - Parallel computation, remapping, interconnections,….

Foundations of AI

- **Foundation of AI is based on**
  - **Control Theory**
    - Machines can modify their behavior in response to the environment (sense/action loop)
      - Water-flow regulator, steam engine governor, thermostat
    - The theory of stable feedback systems (1894)
      - Build systems that transition from initial state to goal state with minimum energy
      - In 1950, control theory could only describe linear systems and AI largely rose as a response to this shortcoming
  - **Linguistics**
    - Speech demonstrates so much of human intelligence
    - Analysis of human language reveals thought taking place in ways not understood in other settings
      - Children can create sentences they have never heard before
      - Language and thought are believed to be tightly intertwined
- **Two Views of AI Goals**
  - AI is about duplicating what the (human) brain DOES
    - Cognitive Science
  - AI is about duplicating what the (human) brain SHOULD do
    - Rationality (doing things logically)

AI Techniques

- **AI Techniques**
  - Rule-based
  - Fuzzy Logic
  - Neural Networks
  - Genetic Algorithms
- **Components of AI Program**
  - AI techniques must be independent of the problem domain as far as possible.
  - AI program should have
  - **knowledge base**
    - AI programs should be learning in nature and update its knowledge accordingly.
    - Knowledge base consists of facts and rules.
    - Characteristics of Knowledge:
    - It is voluminous in nature and requires proper structuring
    - It may be incomplete and imprecise
    - It may keep on changing (dynamic)
  - navigational capability
    - Navigational capability contains various control strategies
    - Control Strategy
    - determines the rule to be applied
    - some heuristics (thump rule) may be applied
  - Inferencing
    - Inferencing requires
    - search through knowledge base and
    - derive new knowledge

Sub-areas of AI

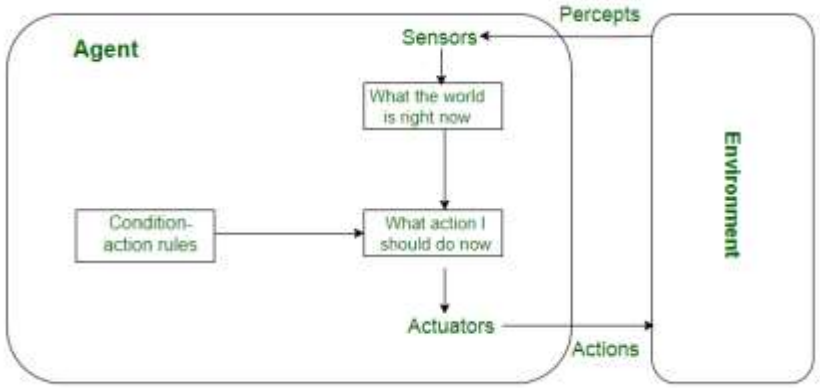
- Sub areas of AI are:
  - Knowledge representation
  - Theorem proving
  - Game playing
  - Vommon sense reasoning dealing with uncertainty and decision making
  - Learning models, inference techniques, pattern recognition, search and matching etc.
  - Logic (fuzzy, temporal, modal) in AI
  - Planning and scheduling
  - Natural language understanding
  - Computer vision
  - Understanding spoken utterances
  - Intelligent tutoring systems
  - Robotics
  - Machine translation systems
  - Expert problem solving
  - Neural Networks, AI tools etc

Types of Agents

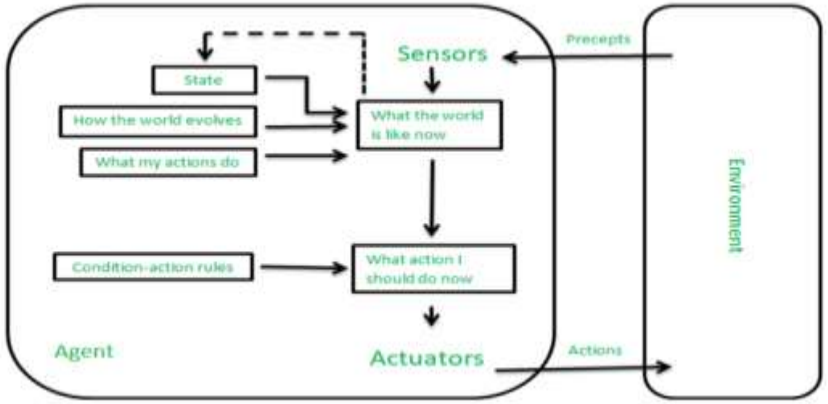
- Agents can be grouped into four classes based on their degree of perceived intelligence and capability :
  - Simple Reflex Agents
  - Model-Based Reflex Agents
  - Goal-Based Agents
  - Utility-Based Agents
- **Simple reflex agents:**
  - Simple reflex agents ignore the rest of the percept history and act only on the basis of the current percept.
  - The agent function is based on the condition-action rule.
  - If the condition is true, then the action is taken, else not. This agent function only succeeds when the environment is fully observable.
- **Model-based reflex agents:**
  - The Model-based agent can work in a partially observable environment, and track the situation.
  - A model-based agent has two important factors:
  - Model: It is knowledge about "how things happen in the world," so it is called a Model-based agent.
  - Internal State: It is a representation of the current state based on percept history.
- **Goal-based agents:**
  - A goal-based agent has an agenda.
  - It operates based on a goal in front of it and makes decisions based on how best to reach that goal.
  - A goal-based agent operates as a search and planning function, meaning it targets the goal ahead and finds the right action in order to reach it.

Types of Agents

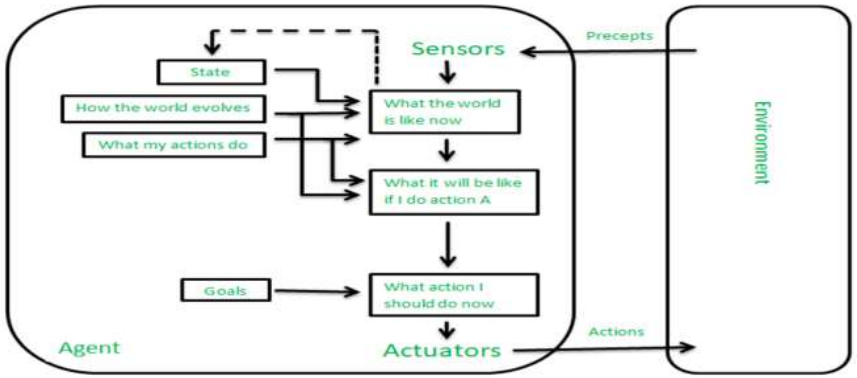
- **Utility-based agents:**
  - A utility-based agent is an agent that acts based not only on what the goal is, but the best way to reach that goal.
  - The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
  - The term utility can be used to describe how "happy" the agent is.



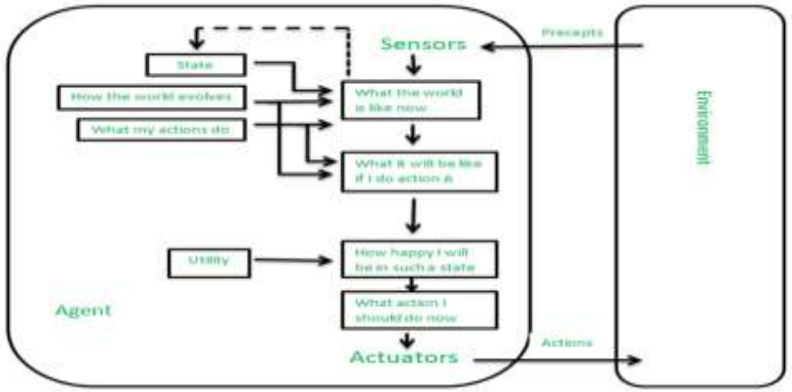
Simple Reflex agents



Model-Based Reflex agents



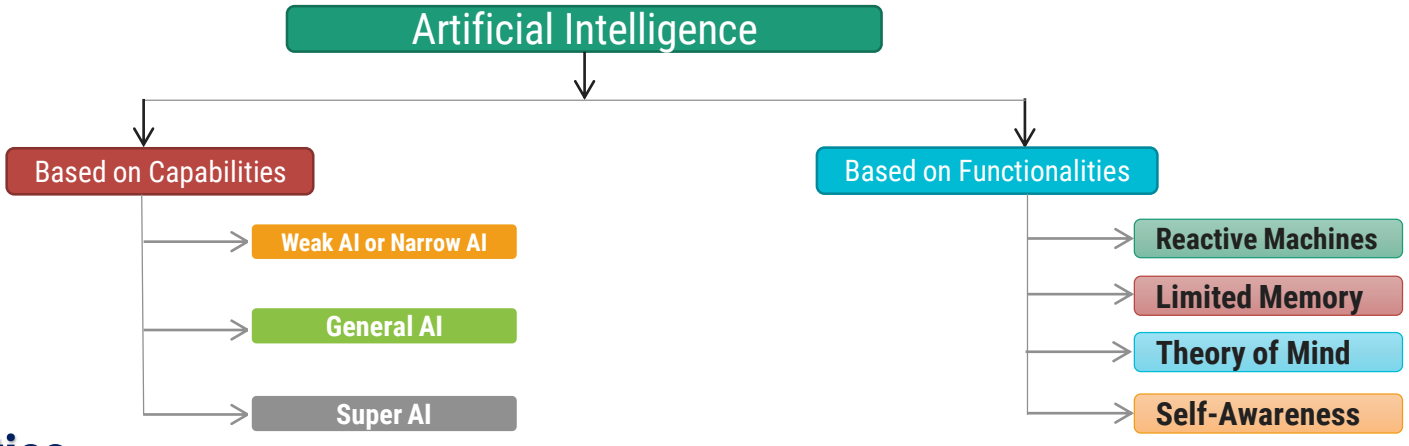
Goal-Based Reflex agents



Utility-Based Reflex agents

Types of AI

- Artificial Intelligence can be divided in various types, there are mainly two types of main categorization which are based on capabilities and based on functionally of AI. Following is flow diagram which explain the types of AI.



- **AI type-1: Based on Capabilities**
  - **1. Weak AI or Narrow AI:**
    - Narrow AI is a type of AI which is able to perform a dedicated task with intelligence.The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
    - Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
    - Apple Siriis a good example of Narrow AI, but it operates with a limited pre-defined range of functions.
    - IBM's Watson supercomputer also comes under Narrow AI, as it uses an Expert system approach combined with Machine learning and natural language processing.
    - **Some Examples of Narrow AI** are playing chess, purchasing suggestions on e-commerce site, self-driving cars, speech recognition, and image recognition.

Types of AI

- **2. General AI:**
  - General AI is a type of intelligence which could perform any intellectual task with efficiency like a human.
  - The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
  - Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
  - The worldwide researchers are now focused on developing machines with General AI.
  - As systems with general AI are still under research, and it will take lots of efforts and time to develop such systems.
- **3. Super AI:**
  - Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI.
  - Some key characteristics of strong AI include capability include the ability to think, to reason, solve the puzzle, make judgments, plan, learn, and communicate by its own.
  - Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.

Type-2: Based on functionality

- **1. Reactive Machines**
  - Purely reactive machines are the most basic types of Artificial Intelligence.
  - Such AI systems do not store memories or past experiences for future actions.
  - These machines only focus on current scenarios and react on it as per possible best action.
  - **Example of reactive machines:** IBM's Deep Blue system , Google's AlphaGo etc..,

Types of AI

- **2. Limited Memory**
  - Limited memory machines can store past experiences or some data for a short period of time.
  - These machines can use stored data for a limited time period only.
  - Self-driving cars are one of the best examples of Limited Memory systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road.
- **3. Theory of Mind**
  - Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.
  - This type of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.
- **4. Self-Awareness**
  - Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.
  - These machines will be smarter than human mind.
  - Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

Applications of AI

Applications of AI:

- **Business :** Financial strategies, give advice
- **Engineering:** check design, offer suggestions to create new product
- **Manufacturing:** Assembly, inspection & maintenance
- **Mining:** used when conditions are dangerous
- **Hospital :** monitoring, diagnosing & prescribing
- **Education :** In teaching
- **household :** Advice on cooking, shopping etc.
- **farming :** prune trees & selectively harvest mixed crops.

Problem Definition

- A ***problem*** is defined by its ***elements*** and **their *relations***. To provide a formal description of a problem, we need to do following:
  - a. Define a *state space* that contains all the possible configurations of the relevant objects, including some impossible ones.
  - b. Specify one or more states, that describe possible situations, from which the problem-solving process may start. These states are called *initial states*.
  - c. Specify one or more states that would be acceptable solution to the problem. These states are called *goal states*.
  - d. Specify a set of *rules* that describe the *actions* (*operators*) available. The problem can then be solved by using the *rules*, in combination with an appropriate *control strategy*, to move through the *problem space* until a *path* from an *initial state* to a *goal state* is found.
- **Problem Space**
  - A **tree** is a graph in which any two vertices are connected by exactly one path.
  - Alternatively, any connected graph with no cycles is a tree.

Problem Characteristics

- Problem Characteristics
  - search is a very general method applicable to a large class of problem.
  - In order to choose the most appropriate method (or combination of methods) for a particular problem it is necessary to analyze the problem along several key dimensions.
  - Is the problem decomposable into a set of independent smaller sub problems?
  - Decomposable problems can be solved by the divide-and-conquer technique.
  - Use of decomposing problems:
  - Each sub-problem is simpler to solve.
  - Each sub-problem can be handed over to a different processor. Thus can be solved in parallel processing environment.
  - There are non decomposable problems.
  - For example, Block world problem is non decomposable.

Problem Solving

- ***Problem solving* is a process** of generating solutions from observed data.
  - a *'problem'* is characterized by a set of *goals*,
  - a set of *objects*, and
  - a set of *operations*.
  - These could be ill-defined and may evolve during problem solving.
- A '***problem space***' is an abstract space.
  - A problem space encompasses all *valid states* that can be generated by the application of any combination of *operators* on any combination of *objects*.
  - The problem space may contain one or more *solutions*. A solution is a combination of *operations* and *objects* that achieve the *goals*.
- A '***search***' refers to the search for a solution in a problem space.
  - Search proceeds with different types of '*search control strategies*'.
  - The *depth-first search and breadth-first search* are the two common *search strategies*.

Problem Solving

- AI programs have a clean separation of
  - computational components of data,
  - operations & control.
- Search forms the core of many intelligent processes.
- It is useful to structure AI programs in a way that facilitates describing the search process.
- **Production System – PS**
  - PS is a formation for structuring AI programs which facilitates describing search process.
  - It consists of
  - Initial or start state of the problem
  - Final or goal state of the problem
  - It consists of one or more databases containing information appropriate for the particular task.
  - The information in databases may be structured
  - using knowledge representation schemes.

Problem Solving

Production Rules

- PS contains set of production rules,
- each consisting of a left side that determines the applicability of the rule and
- a right side that describes the action to be performed if the rule is applied.
- These rules operate on the databases.
- Application of rules change the database.
- A control strategy that specifies the order in which the rules will be applied when several rules match at once.
- One of the examples of Production Systems is an Expert System.

General Problem solving

- *Problem solving* has been the key area of concern for Artificial Intelligence.
- Problem solving is a process of generating solutions from observed or given data.
- It is however not always possible to use direct methods (i.e. go directly from data to solution).
- Instead, problem solving often needs to use indirect or model based methods.
- ***General Problem Solver (GPS)*** was a computer program created in 1957 by Simon and Newell to build a universal problem solver machine.
- *GPS* was based on Simon and Newell's theoretical work on logic machines.
- *GPS* in principle can solve any formalized symbolic problem, such as theorems proof and geometric problems and chess playing.
- *GPS* solved many simple problems, such as the Towers of Hanoi, that could be sufficiently formalized, but ***GPS could not solve any real-world problems***.
- To build a system to solve a particular problem, we need to:
  - Define the problem precisely – find input situations as well as final situations for an acceptable solution to the problem
  - Analyze the problem – find few important features that may have impact on the of various possible techniques for solving the problem
  - Isolate and represent task knowledge necessary to solve the problem
  - Choose the best problem-solving technique(s) and apply to the particular problem

Structures of State Space

- The *Structures* of *state space* are *trees* and *graphs*.
- Tree is a hierarchical structure in a graphical form; and
- Graph is a non-hierarchical structure.
  - **Tree** has only one path to a given node; i.e., a *tree* has one and only one path from any point to any other point.◊
  - **Graph** consists of a set of nodes(vertices)and a set of edges (arcs).
  - Arcs establish relationships (connections) between the nodes; i.e., a graph has several paths to a given node.
  - **operators** are directed *arcs* between nodes.
  - **Search process** explores the *state space*. In the worst case, the search explores all possible *paths* between the *initial state* and the *goal state*.

Problem solving-State Space

- The steps that are required to build a system to solve a particular problem are:
  1. Problem Definition that must include precise specifications of what the initial situation will be, as well as what final situations constitute acceptable solutions to the problem.
  2. Problem Analysis, this can have immense impact on the appropriateness of various possible techniques for solving the problem.
  3. Isolate and Represent the task knowledge required to solve the problem.
  4. Selection of the best technique(s) for solving the particular problem.
- Problem solving is a process of generating solutions from the observed data.
- A ***State space*** is the set of all states reachable from the ***initial state***.
- Definitions of terms :
  - A *state space* forms a *graph* (or map) in which the *nodes* are states and the *arcs* between nodes are actions.
  - In *state space*, a *path* is a sequence of states connected by a sequence of actions.
  - The *solution* of a problem is part of the map formed by the *state space*.

State and State Space Representation

- A state is a representation of problem elements at a given moment.
- A state space is the set of all possible states reachable from the initial state.
- A state space forms a graph in which the nodes are states and the arcs between nodes are actions.
- In a state space, a path is a sequence of states connected by a sequence of actions.
- The solution of a problem is a part of the graph formed by the state space.

Chess - Initial Position

Position after a legal move

Define the Problem as State Space Search

- To provide a formal description of a problem, we need to do the following:
  1. Define a state space that contains all the possible configurations of the relevant objects.
  2. Specify one or more states that describe possible situations, from which the problem solving process may start. These states are called initial states.
  3. Specify one or more states that would be acceptable solution to the problem. These states are called goal states.
- Specify a set of rules that describe the actions (operators) available.
- The problem can then be solved by using the rules, in combination with an appropriate control strategy, to move through the problem space until a path from an initial state to a goal state is found. This process is known as ‘search’.

State Space Representation – Water Jug

- Problem Definition: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

1. Initial State

- We will represent a state of the problem as a tuple (x, y), where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-gallon jug.
- Note that $0 \leq x \leq 4$, and $0 \leq y \leq 3$.
- Here the initial state is (0, 0). The goal state is (2, n) for any value of n.

2. Production Rules

Sr.	Current state	Next state	Description
1	(x, y) If x<4	(4, y)	fill the 4- gallon jug
2	(x, y) If x<3	(x,3)	fill the 3-gallon jug
3	(x, y) If x>0	(x-d, y)	pour some water out of the 4- gallon jug
4	(x, y) If y>0	(x, y-d)	pour some water out of the 3- gallon jug
5	(x, y) If x>0	(0, y)	empty the 4- gallon jug on the ground
6	(x, y) If y>0	(x, 0)	empty the 3- gallon jug on the ground

Define the Problem as State Space Search

Sr.	Current state	Next state	Description
7	(x, y) If x + y >= 4 & y>0	(4,y-(4-x))	pour water from the 3- gallon jug into the 4-gallon jug until the 4-gallon jug is full
8	(x, y) If x + y >= 3 & x>0	(x-(3-y),3))	pour water from the 4- gallon jug into the 3-gallon jug until e 3-gallon jug is full
9	(x, y) If x + y <= 4 & y>0	(x+y,0)	pour all the water from the 3-gallon jug into the 4-gallon jug
10	(x, y) If x + y <= 3 & x>0	(0,x+y)	pour all the water from the 4 - gallon jug into the 3-gallon jug
11	(0,2)	(2,0)	pour the 2-gallon from the 3 – gallon jug into the 4-gallon jug
12	(2,y)	(0, x)	empty the 2 gallon in the 4 gallon on the ground

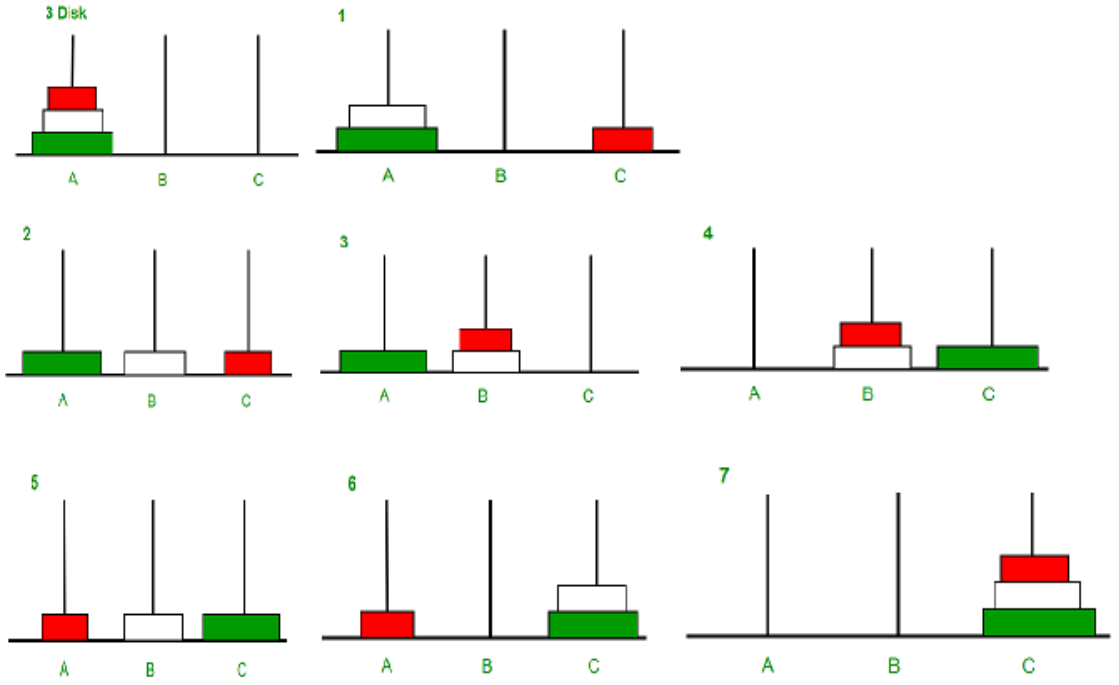
3. Productions for the water jug problem

Gallons in the 4- gallon Jug	Gallons in the 3- gallon	Rule Applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

Problem Solving –Tower of Hanoi

- Tower of Hanoi is a mathematical puzzle where we have three rods (**A**, **B**, and **C**) and **N** disks. Initially, all the disks are stacked in decreasing value of diameter i.e., the smallest disk is placed on the top and they are on rod **A**. The objective of the puzzle is to move the entire stack to another rod (here considered **C**), obeying the following simple rules:
- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
- No disk may be placed on top of a smaller disk.
- **Input:3**
- **Output:** Disk 1 moved from A to C
  Disk 2 moved from A to B
  Disk 1 moved from C to B
  Disk 3 moved from A to C
  Disk 1 moved from B to A
  Disk 2 moved from B to C
  Disk 1 moved from A to C

- **Tower of Hanoi using Recursion:**
- The idea is to use the helper node to reach the destination using recursion. Below is the pattern for this problem:
- Shift ‘N-1’ disks from ‘A’ to ‘B’, using C.
- Shift last disk from ‘A’ to ‘C’.
- Shift ‘N-1’ disks from ‘B’ to ‘C’, using A.



Search and Control Strategies

Introduction

- **Searching**
- *Searching* is the universal technique of problem solving in *AI*.
- *Search* algorithms are algorithms that help in solving *search* problems. A *search* problem consists of a *search* space, start state, and goal state.
- Many traditional search algorithms are used in AI applications.
- For complex problems, the traditional algorithms are unable to find the solution within some practical time and space limits.
- Consequently, many special techniques are developed; using heuristic functions.
- The algorithms that use heuristic functions are called heuristic algorithms.
- Heuristic algorithms are not really intelligent; they appear to be intelligent because they achieve better performance.

Properties of Search Algorithms

- Which search algorithm one should use will generally depend on the problem domain. There are four important factors to consider:
  - 1. **Completeness** – Is a solution guaranteed to be found if at least one solution exists?
  - 2. **Optimality** – Is the solution found guaranteed to be the best (or lowest cost) solution if there exists more than one solution?
  - 3. **Time Complexity** – The upper bound on the time required to find a solution, as a function of the complexity of the problem.
  - 4. **Space Complexity** – The upper bound on the storage space (memory) required at any point during the search, as a function of the complexity of the problem.
- **State Spaces versus Search Trees:**
  - State Space
    - Set of valid states for a problem
    - Linked by operators
    - e.g., 20 valid states (cities) in the Romanian travel problem
  - Search Tree
    - Root node = initial state
    - Child nodes = states that can be visited from parent
    - Note that the depth of the tree can be infinite - E.g., via repeated states
    - Partial search tree -Portion of tree that has been expanded so far
    - Fringe- Leaves of partial search tree, candidates for expansion Search trees = data structure to search state-spac

Search Terminologies

- Which search algorithm one should use will generally depend on the problem domain. There are four important factors to consider:
  - 1. **Completeness** – Is a solution guaranteed to be found if at least one solution exists?
  - 2. **Optimality** – Is the solution found guaranteed to be the best (or lowest cost) solution if there exists more than one solution?
  - 3. **Time Complexity** – The upper bound on the time required to find a solution, as a function of the complexity of the problem.
  - 4. **Space Complexity** – The upper bound on the storage space (memory) required at any point during the search, as a function of the complexity of the problem.
- **State Spaces versus Search Trees:**
  - State Space
    - Set of valid states for a problem
    - Linked by operators
    - e.g., 20 valid states (cities) in the Romanian travel problem
  - Search Tree
    - Root node = initial state
    - Child nodes = states that can be visited from parent
    - Note that the depth of the tree can be infinite - E.g., via repeated states
    - Partial search tree -Portion of tree that has been expanded so far
    - Fringe- Leaves of partial search tree, candidates for expansion Search trees = data structure to search state-spac

Control Strategies

- The word '***search***' refers to the search for a solution in a *problem space*.
  - Search proceeds with different types of *'**search control strategies**'*.
  - A strategy is defined by picking the order in which the nodes expand.
- The Search strategies are evaluated along the following dimensions: Completeness, Time
- complexity, Space complexity, Optimality (the search- related terms are first explained, and then
- the search algorithms and control strategies are illustrated next).
- **Search-related terms**
  - **Algorithm's performance and complexity**
    - Ideally we want a common measure so that we can compare approaches in order to select the most appropriate algorithm for a given situation.
    - *Performance* of an algorithm depends on internal and external factors.
  - ***Internal factors/ External factors***
    - ***Time*** required to run
    - ***Size*** of input to the algorithm
    - ***Space*** (memory) required to run
    - ***Speed*** of the computer
    - ***Quality*** of the compiler
    - *Complexity* is a measure of the performance of an algorithm. *Complexity* measures the internal factors, usually in time than space.

Control Strategies

- **Computational complexity**
  - It is the measure of resources in terms of ***Time*** and ***Space***.
  - If ***A*** is an algorithm that solves a decision problem ***f***, then run-time of ***A*** is the number of steps taken on the input of length ***n.***
  - *Time Complexity* ***T(n)*** of a decision problem ***f*** is the run-time of the 'best' algorithm ***A*** for ***f.***
  - *Space Complexity* ***S(n)*** of a decision problem ***f*** is the amount of memory used by the 'best' algorithm ***A*** for ***f.***

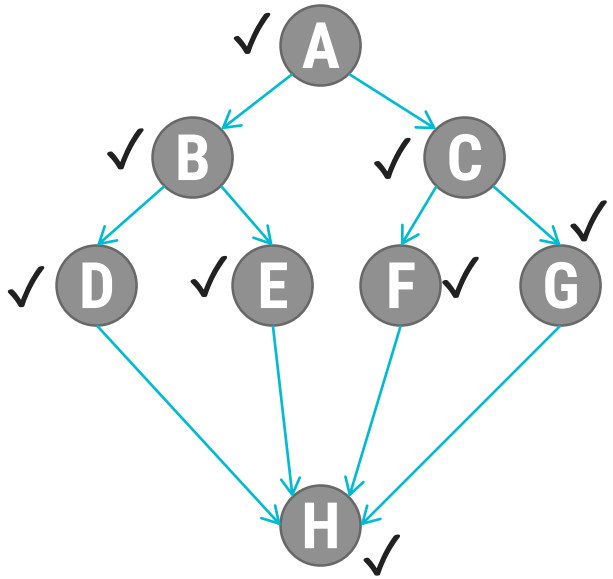
Search Techniques

- Search Techniques can be classified as:

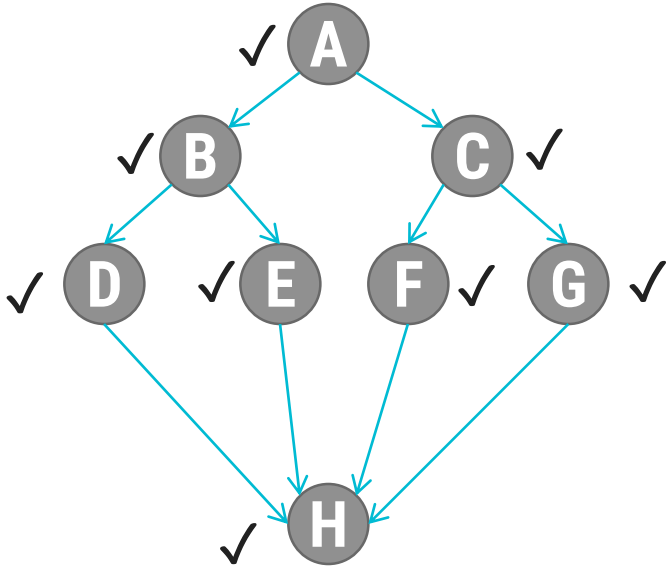
1. Uninformed/Blind Search Control Strategy:
   - Do not have additional information about states beyond problem definition.
   - Total search space is looked for the solution.
   - Example: Breadth First Search (BFS), Depth First Search (DFS), Uniform-cost search (UCS).

2. Informed/Directed Search Control Strategy:
   - Some information about problem space is used to compute the preference among various possibilities for exploration and expansion.
   - Examples: Best First Search, Problem Decomposition, A*, Mean end Analysis

Uninformed Search Techniques



Breadth First Search



Depth First Search

Breadth First Search (BFS)

- One simple search strategy is a breadth-first search. In this strategy, the root node is expanded first, then all the nodes generated by the root node are expanded next, and then their successors, and so on.
- In general, all the nodes at depth d in the search tree are expanded before the nodes at depth d + 1.
- **Breadth first search is:**
  - One of the simplest search strategies
  - Complete. If there is a solution, BFS is guaranteed to find it.
  - If there are multiple solutions, then a minimal solution will be found
  - The algorithm is optimal (i.e., admissible) if all operators have the same cost. Otherwise, breadth first search finds a solution with the shortest path length.
  - **Time complexity** : O(bd)
  - **Space complexity** : O(bd)
  - **Optimality** :Yes
- *b - branching factor(maximum no of successors of any node), d – Depth of the shallowest goal node*
- *Maximum length of any path (m) in search space*

- **Advantages:**
  - BFS will provide a solution if any solution exists.
  - If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

- **Disadvantages:**
  - Requires the generation and storage of a tree whose size is exponential the depth of the shallowest goal node.
  - The breadth first search algorithm cannot be effectively used unless the search space is quite small.

Depth First Search (BFS)

- Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.
- So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally, print the nodes in the path.
- **Depth first search is:**
  - Start by putting any one of the graph's vertices on top of a stack.
  - Take the top item of the stack and add it to the visited list.
  - Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
  - Keep repeating steps 2 and 3 until the stack is empty.
  - **Time complexity** : O(bm)
  - **Space complexity** : O(bm)
  - **Optimality** :No
- **Advantages:**
  - It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.
- **Disadvantages:**
  - The main drawback of IDDFS is that it repeats all the work of the previous phase.

Difference Between DFS & BFS

Depth First Search	Breath First Search
1. DFS requires less memory since only the nodes on the current path are stored.	1. BFS guarantees that the space of possible moves is systematically examined; this search requires considerably more memory resources.
2. By chance, DFS may find a solution without examining much of the search space at all. Then it finds a solution faster.	2. The search systematically proceeds testing each node that is reachable from a parent node before it expands to any child of those nodes.
3. If the selected path does not reach to the solution node, DFS gets stuck into a blind alley.	3. BFS will not get trapped exploring a blind alley.
4. Does not guarantee to find solution. Backtracking is required if wrong path is selected.	4. If there is a solution, BFS is guaranteed to find it.

Uniform-cost search(UCS)

- **Uniform-cost search** is an uninformed search algorithm that uses the lowest cumulative cost to find a path from the source to the destination. Nodes are expanded, starting from the root, according to the minimum cumulative cost. The uniform-cost search is then implemented using a Priority Queue.
- **Depth first search is:**
  - Insert the root node into the priority queue
  - *Repeat while the queue is not empty:*
  - Remove the element with the highest priority
  - *If the removed node is the destination,* print total cost and stop the algorithm
  - *Else*, enqueue all the children of the current node to the priority queue, with their cumulative cost from the root as priority
  - **Time complexity** : **$O(b(1 + C / \varepsilon))$**

    Where: b–branching factor,

    C-optimal cost

    ε - cost of each step
  - **Space complexity** : **$O(b^{1 + [C^*/\varepsilon]})$**.
  - **Optimality** :Yes

- **Advantages:**
  - It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.
- **Disadvantages:**
  - The main drawback of IDDFS is that it repeats all the work of the previous phase.

Heuristic Search Techniques

- Every search process can be viewed as a traversal of a directed graph, in which the nodes represent problem states and the arcs represent relationships between states.
- The search process must find a path through this graph, starting at an initial state and ending in one or more final states.
- Domain-specific knowledge must be added to improve search efficiency.
- The Domain-specific knowledge about the problem includes the nature of states, cost of transforming from one state to another, and characteristics of the goals.
- This information can often be expressed in the form of Heuristic Evaluation Function.
- Heuristic function maps from problem state descriptions to the measures of desirability, usually represented as numbers.
- The value of the heuristic function at a given node in the search process gives a good estimate of whether that node is on the desired path to a solution.
- Well-designed heuristic functions can play an important role in efficiently guiding a search process toward a solution.

Heuristic Search Techniques

- In general, heuristic search improves the quality of the path that is explored.
- In such problems, the search proceeds using current information about the problem to predict which path is closer to the goal and follow it, although it does not always guarantee to find the best possible solution.
- Such techniques help in finding a solution within reasonable time and space (memory).
- Some prominent intelligent search algorithms are stated below:
  1. Generate and Test Search
  2. Greedy Search
  3. A* Search
  4. Hill Climbing Search
  5. Local Search
  6. Constraint Search

Heuristic Search Techniques

- **Greedy Best First Search**
  - Greedy best-first search algorithm always selects the trail which appears best at that moment.
  - Within the best first search algorithm, we expand the node which is closest to the goal node and therefore the closest cost is estimated by heuristic function.
  - This sort of search reliably picks the way which appears best by then.
  - It is the blend of **BFS** and **DFS**. It uses heuristic limit and searches. The BFS grants us to take the advantages of the two estimations.
- **A*Search**
  - A* search is also known as best-first search
  - A* search is the most consistently known kind of best-first interest. It uses heuristic limit h(n), and cost to show up at the center point n from the earliest starting point state g(n).
  - It has solidified features of **UCS** and insatiable best-first request, by which it deal with the issue capably.
  - A* search computation finds the briefest path through the chase space using the heuristic limit. This chase count expands less interest trees and gives a perfect result snappier.
  - A* count resembles UCS beside that it uses g(n)+h(n) instead of g(n).
  - It is formulated with weighted graphs, which suggests it can find the simplest path involving the littlest cost in terms of distance and time.
  - This makes A* algorithm in AI an informed search algorithm for best-first search.

Heuristic Search Techniques

Hill Climbing Algorithms

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
  - ✓ It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
  - ✓ Hill Climbing is mostly used when a good heuristic is available.
  - ✓ In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.
- The idea behind hill climbing is as follows.
  1. Pick a random point in the search space.
  2. Consider all the neighbors of the current state.
  3. Choose the neighbor with the best quality and move to that state.
  4. Repeat 2 thru 4 until all the neighboring states are of lower quality.
  5. Return the current state as the solution state.
- **Features of Hill Climbing**
  - Produce and Test variation: Hill Climbing is the variation of the Generate and Test strategy. The Generate and Test technique produce input which assists with choosing which bearing to move in the inquiry space.
  - Use of Greedy Approach: Hill-climbing calculation search moves toward the path which improves the expense.
  - No backtracking: It doesn’t backtrack the pursuit space, as it doesn’t recall the past states.

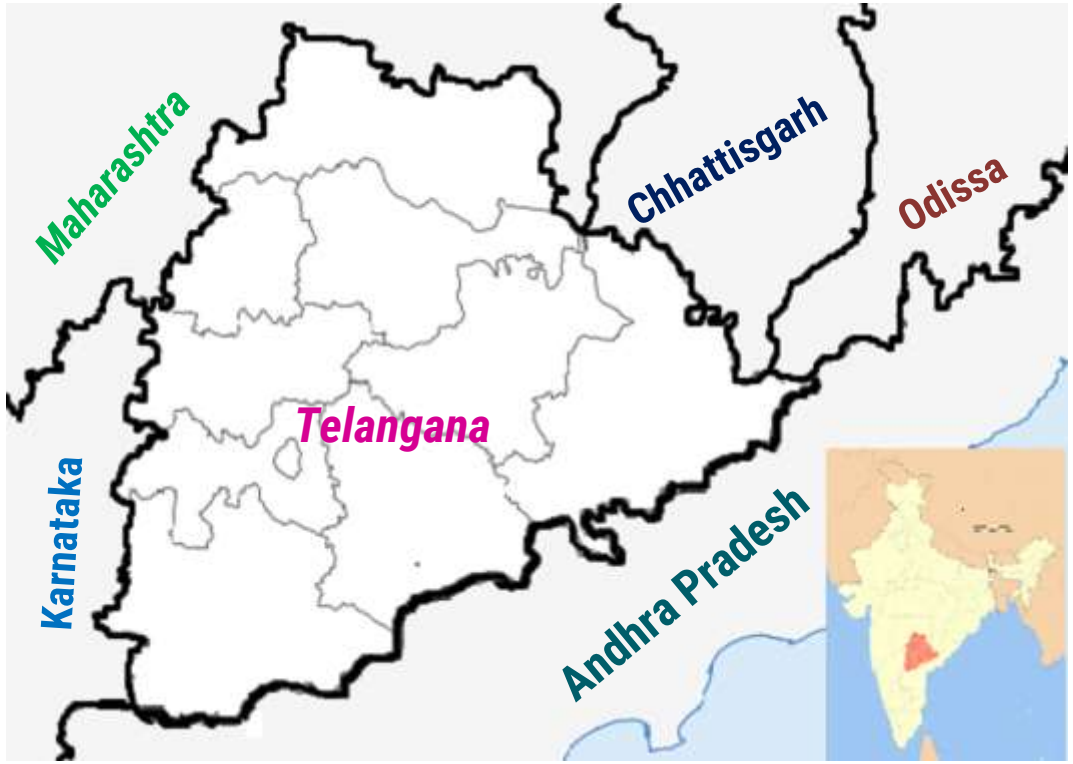
Heuristic Search Techniques

- **Best First Search:**
  - A combination of depth first and breadth first searches.
  - Depth first is good because a solution can be found without computing all nodes and breadth first is good because it does not get trapped in dead ends.
  - The best first search allows us to switch between paths thus gaining the benefit of both approaches.
  - At each step the most promising node is chosen.
  - If one of the nodes chosen generates nodes that are less promising it is possible to choose another at the same level and in effect the search changes from depth to breadth.
  - If on analysis these are no better than this previously unexpanded node and branch is not forgotten and the search method reverts to the **OPEN** is a priority queue of nodes that have been evaluated by the heuristic function but which have not yet been expanded into successors.
  - The most promising nodes are at the front.
  - **CLOSED** are nodes that have already been generated and these nodes must be stored because a graph is being used in preference to a tree.

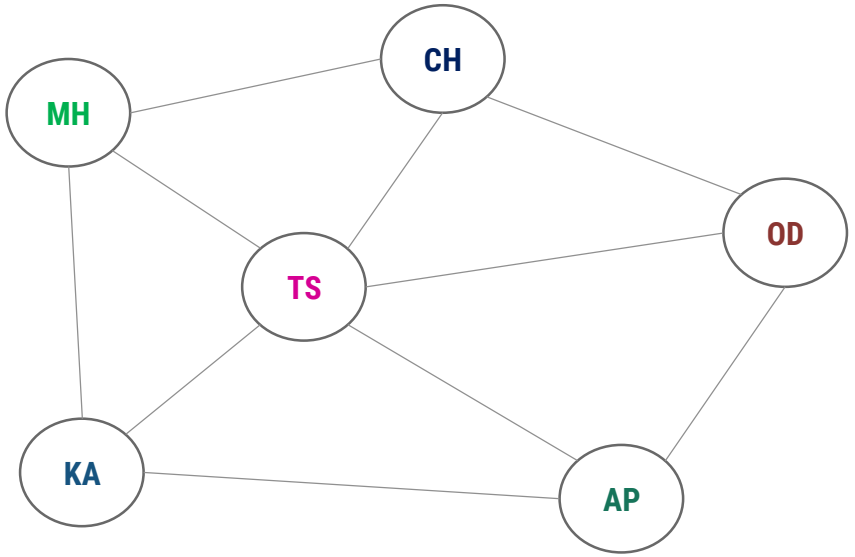
Heuristic Search Techniques

- **Constraint Satisfaction Problem**
  - CSP or Constraint Satisfaction Problem is a set of questions that requires its answer inside certain confinements/conditions otherwise called **limitations**. It comprises of :
  - Limited arrangement of factors which stores the arrangement. (V = {V1, V2, V3,….., Vn})
  - Lot of discrete qualities from which the arrangement is picked. (D = {D1, D2, D3,…..,Dn})
  - Limited arrangement of limitations. (C = {C1, C2, C3,……, Cn})
  - In the case of AI, we most of the time, deal with discrete quantities.
  - The common problems which can be solved using a CSP are Sudoku problems, Cryptarithmetic, Crossword, etc.
  - **Example: The map coloring problem.**
    - The task of coloring each region red, green or blue in such a way that no neighboring regions have the same color.
    - We are given the task of coloring each region red, green, or blue in such a way that the neighboring regions must not have the same color.
    - To formulate this as CSP, we define the variable to be the regions: MH, TS, KA, CH and AP
    - The domain of each variable is the set {red, green, blue}.
    - The constraints require neighboring regions to have states colors: for example, the allowable combinations for MH and CH are the pairs {(red,green),(red,blue),(green,red),(green,blue),(blue,red),(blue,green)}. (The constraint can also be represented as the inequality MH $\neq$ CH). There are many possible solutions,
    - **Initial state** : the empty assignment {},in which all variables are unassigned.
    - **Successor function:** a value can be assigned to any unassigned variable, provided that it does not conflict with previously assigned variables.
    - **Goal test:** the current assignment is complete.
    - **Path cost:** a constant cost(E.g.,1) for every step.

Heuristic Search Techniques



- **Constraint Graph:**
  - A CSP is usually represented as an undirected graph, called constraint graph where the nodes are the variables and the edges are the binary constraints.

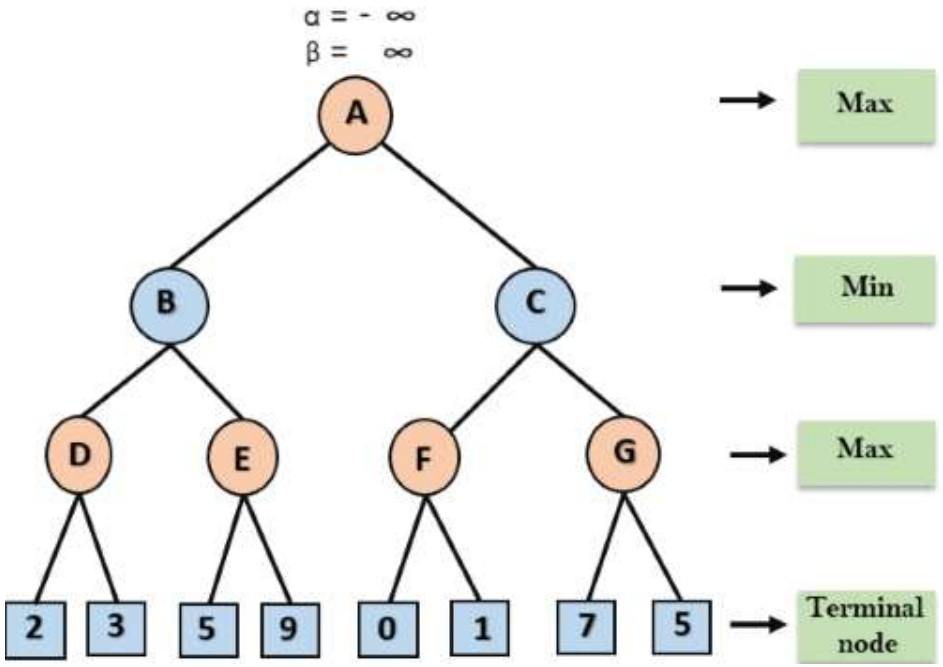


Heuristic Search Techniques

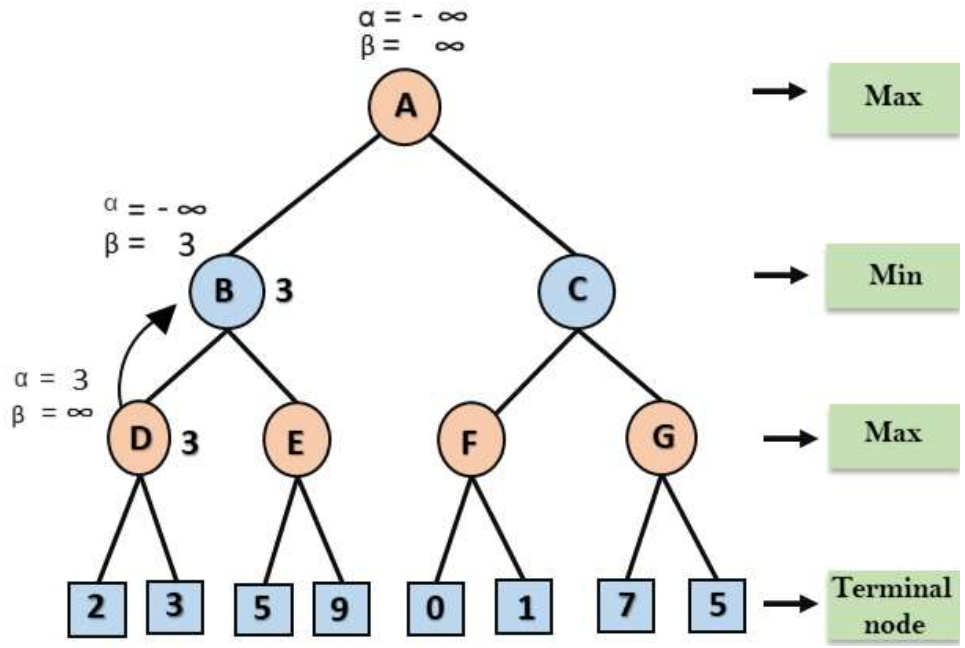
- **Alpha-Beta Pruning:**
  - Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
  - As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**.
  - This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
  - Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
  - The two-parameter can be defined as:
    - **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
    - **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
  - The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.
  - **Key Points:**
    - The Max player will only update the value of alpha.
    - The Min player will only update the value of beta.
    - While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
    - We will only pass the alpha, beta values to the child nodes.

Heuristic Search Techniques

- **Working of Alpha-Beta Pruning:**
  - Let's take an example of two-player search tree to understand the working of Alpha-beta pruning
  - **Step 1:** At the first step the, Max player will start first move from node A where $\alpha = -\infty$ and $\beta = +\infty$, these value of alpha and beta passed down to node B where again $\alpha = -\infty$ and $\beta = +\infty$, and Node B passes the same value to its child D.

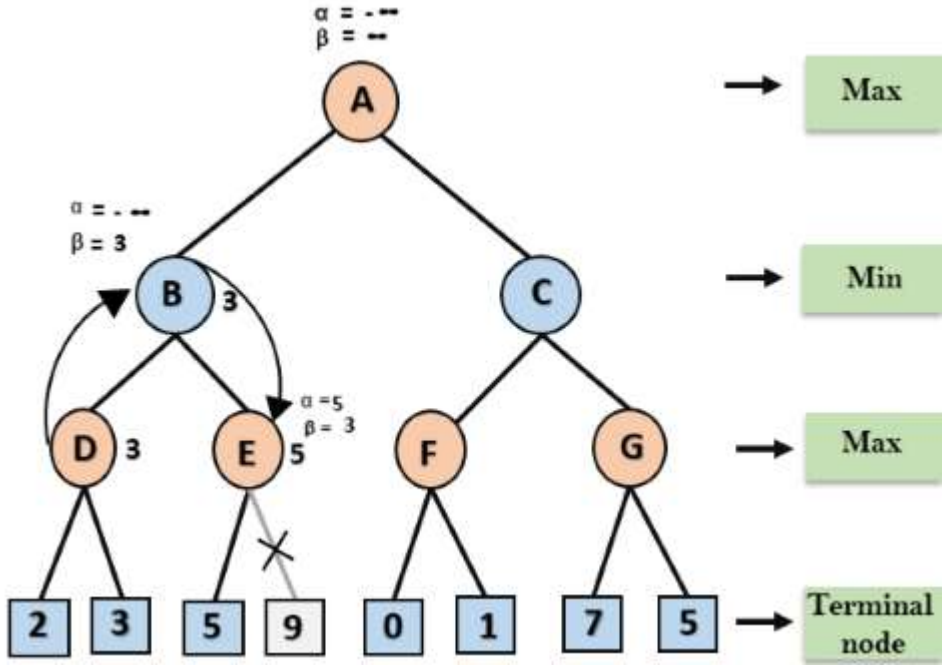


- **Working of Alpha-Beta Pruning:**
  - **Step 2:** At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.
  - **Step 3:** Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now $\beta = +\infty$, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now $\alpha = -\infty$, and $\beta = 3$.

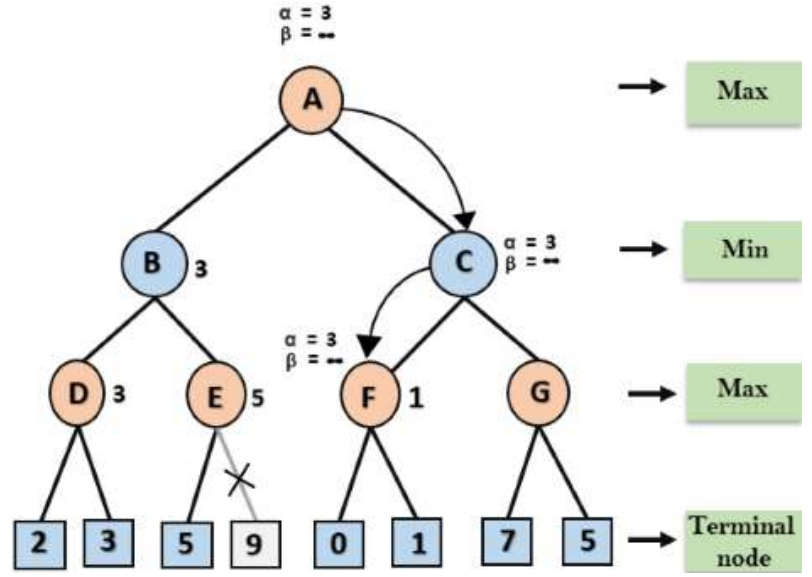


Heuristic Search Techniques

- **Working of Alpha-Beta Pruning:**
  - In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.
  - **Step 4:** At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.

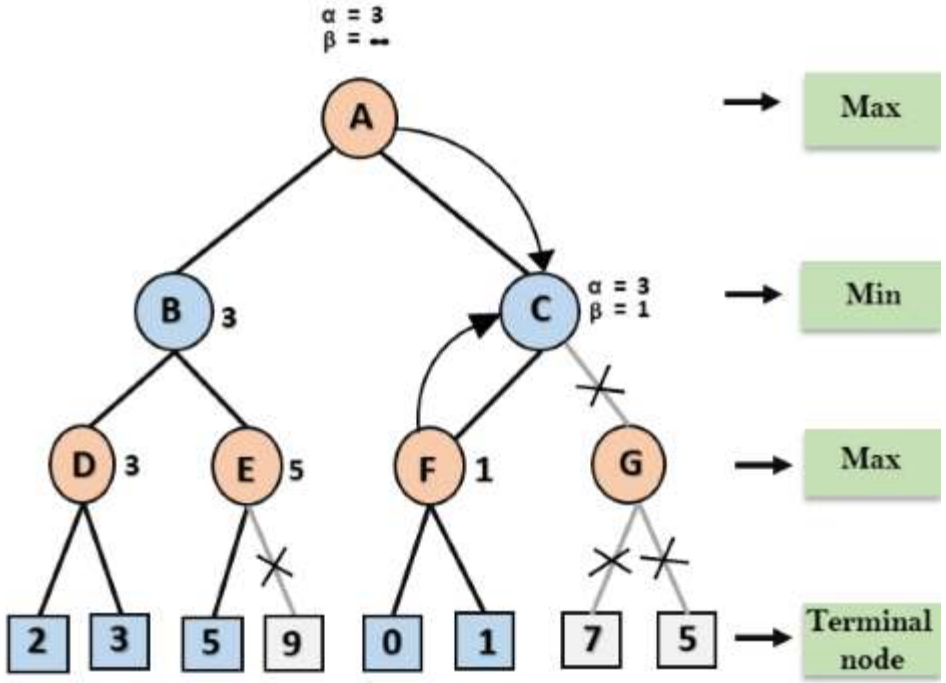


- **Working of Alpha-Beta Pruning:**
  - **Step 5:** At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.
  - At node C, α=3 and β= +∞, and the same values will be passed on to node F.
  - **Step 6:** At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.

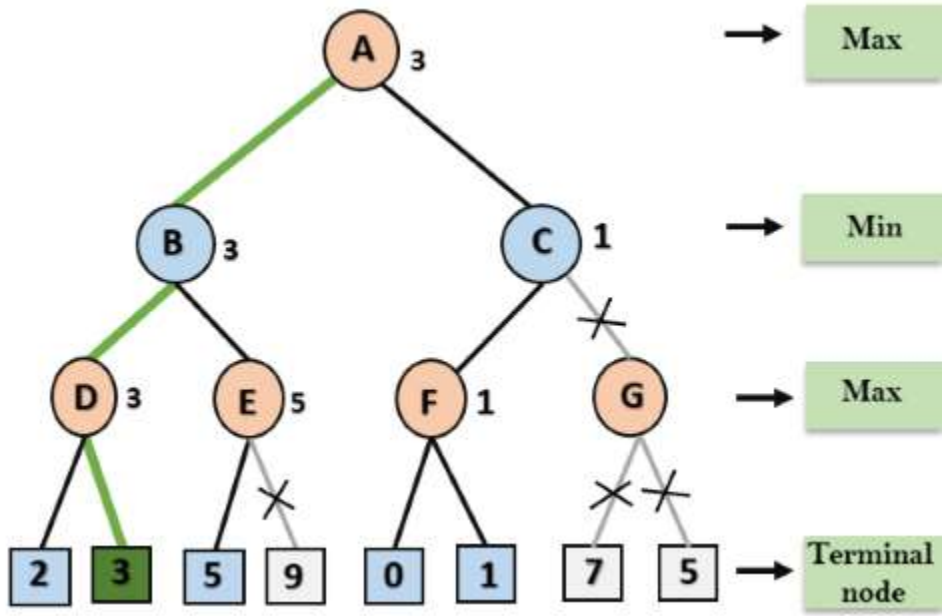


Heuristic Search Techniques

- **Working of Alpha-Beta Pruning:**
  - **Step 7:** Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



- **Working of Alpha-Beta Pruning:**
  - **Step 8:** C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



Applications of AI

Natural Language Processing

Email Spam Filter in Gmail

Neural Network

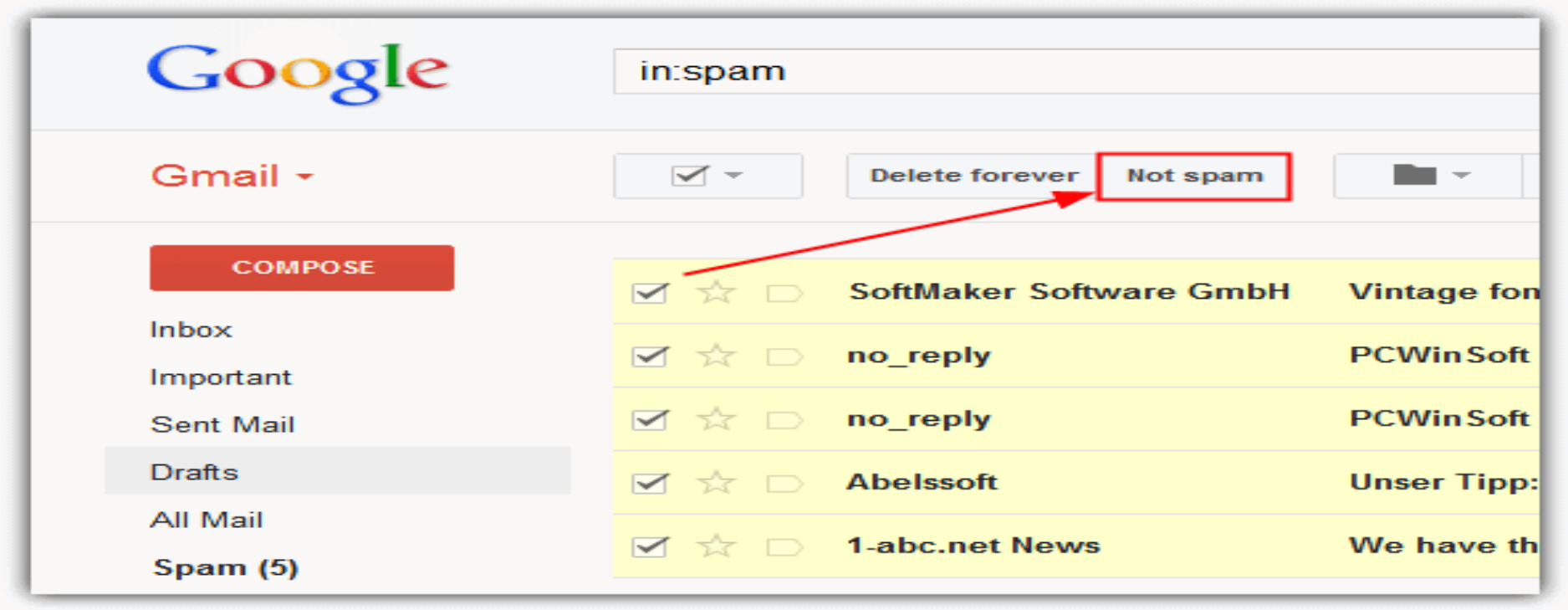
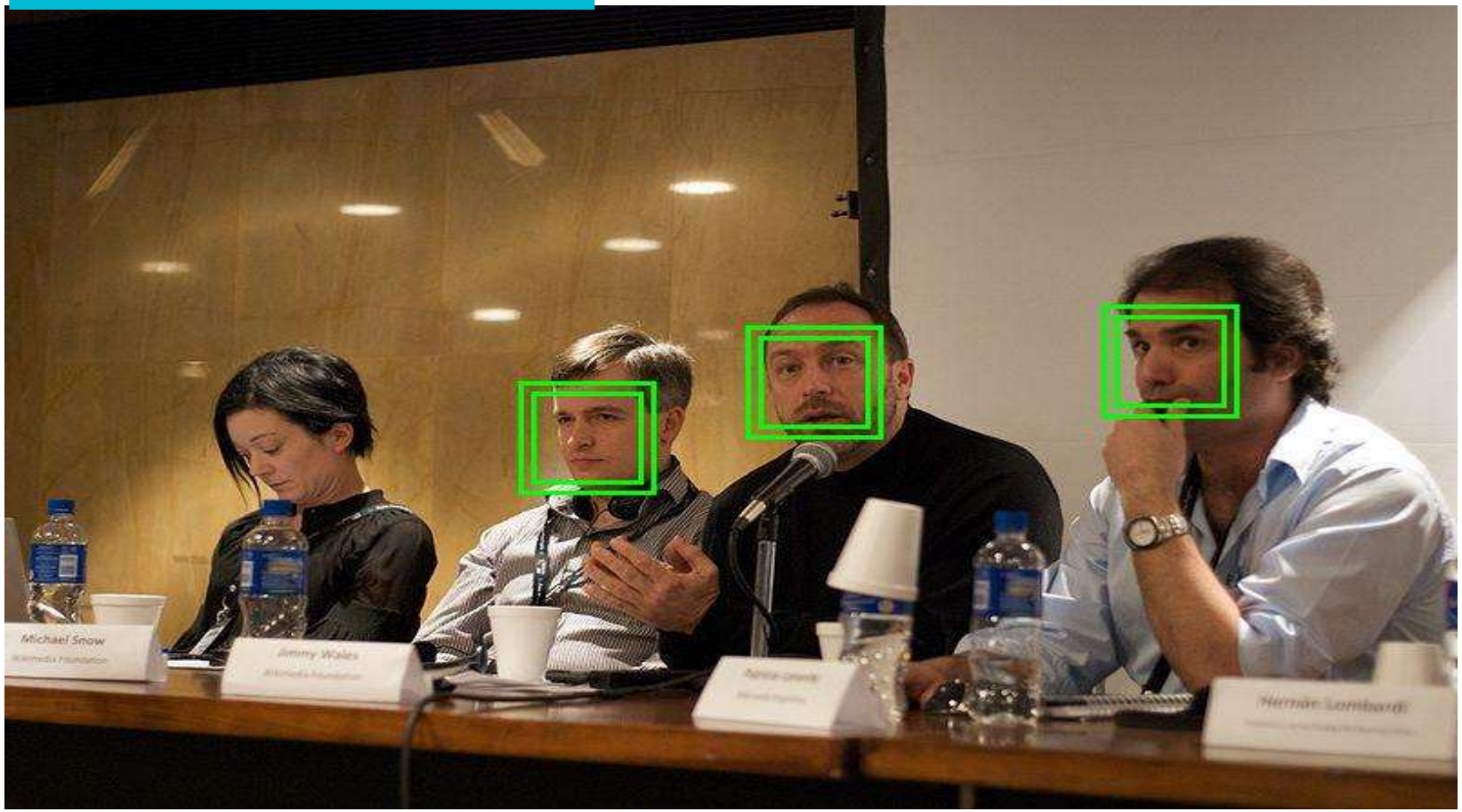


Image Processing

Face Detection in Camera

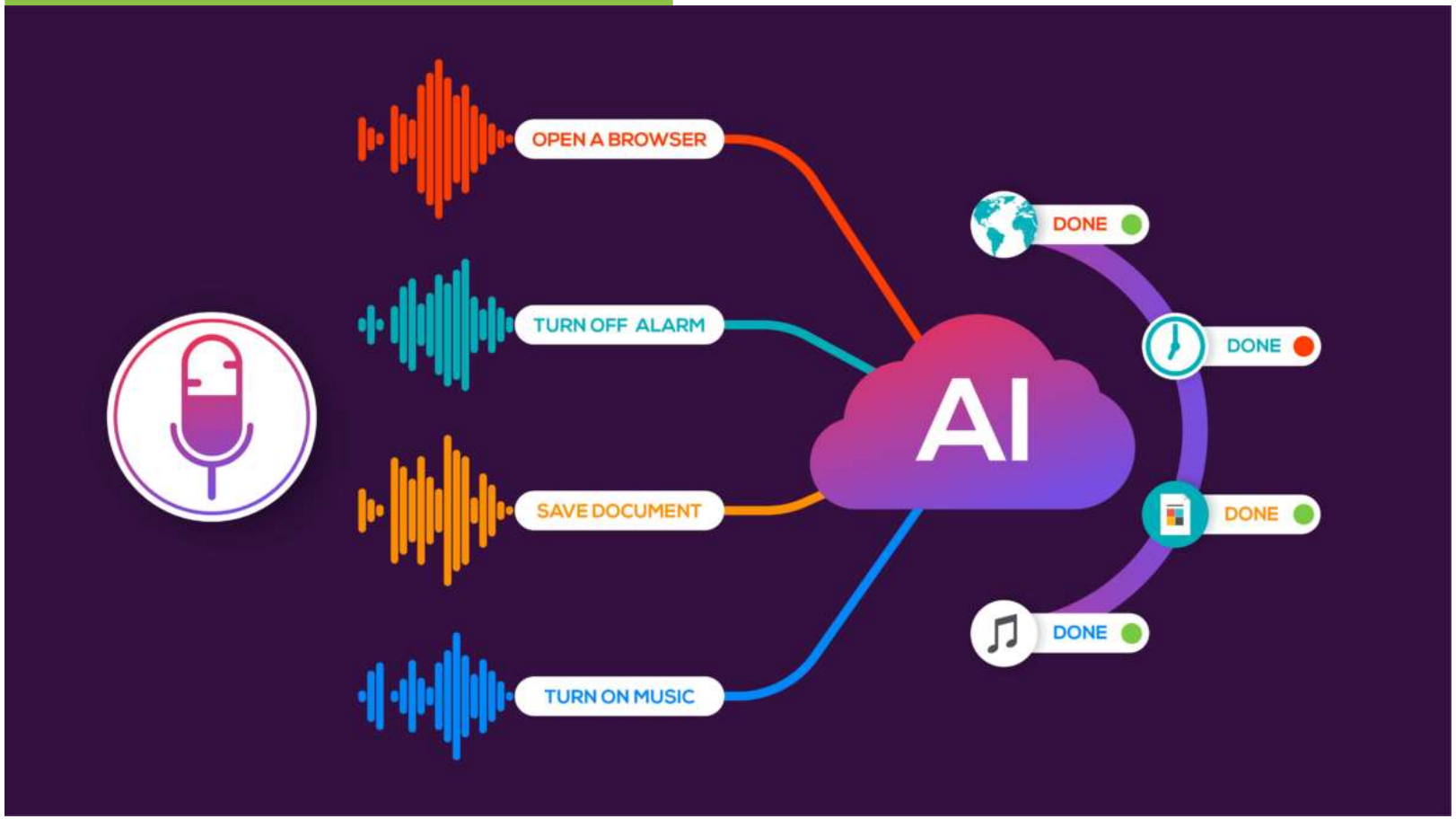
Deep Learning



Speech Recognition

Voice Technology in Virtual Agents

Deep Learning



Data Mining

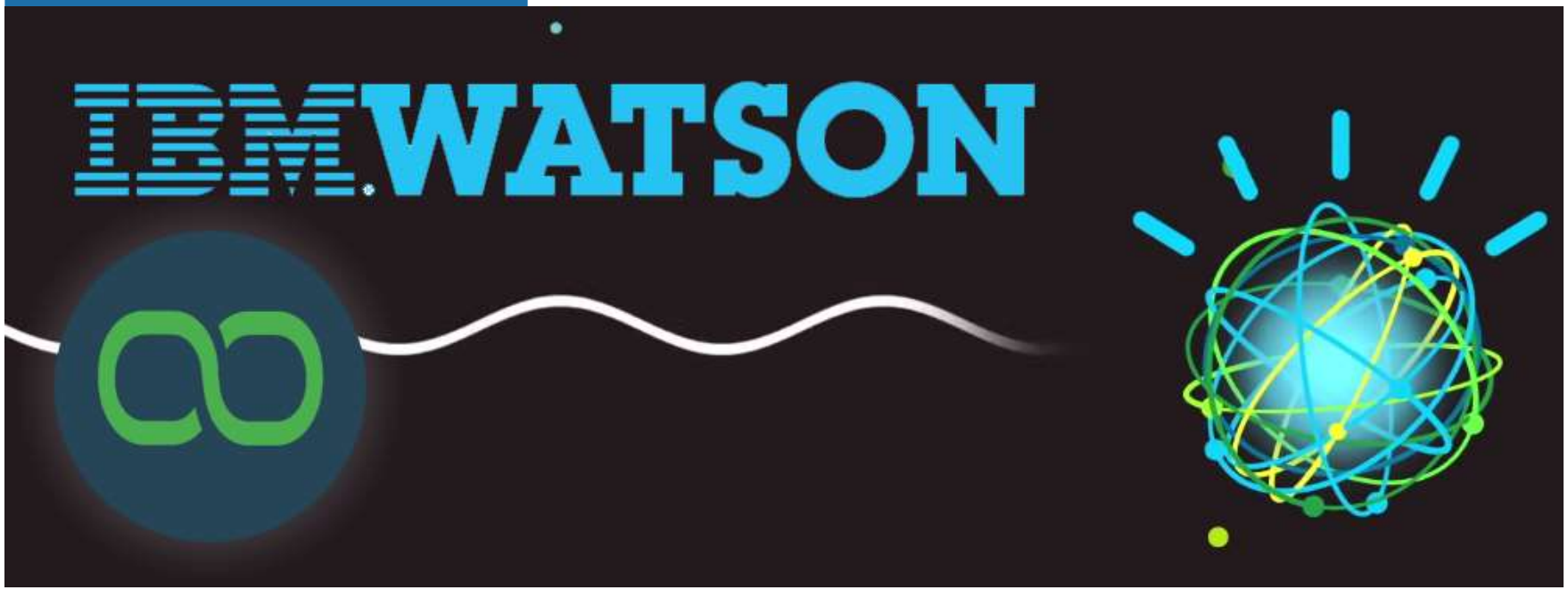
Market Basket Analysis

Product recommendation

Expert System

IBM Watson

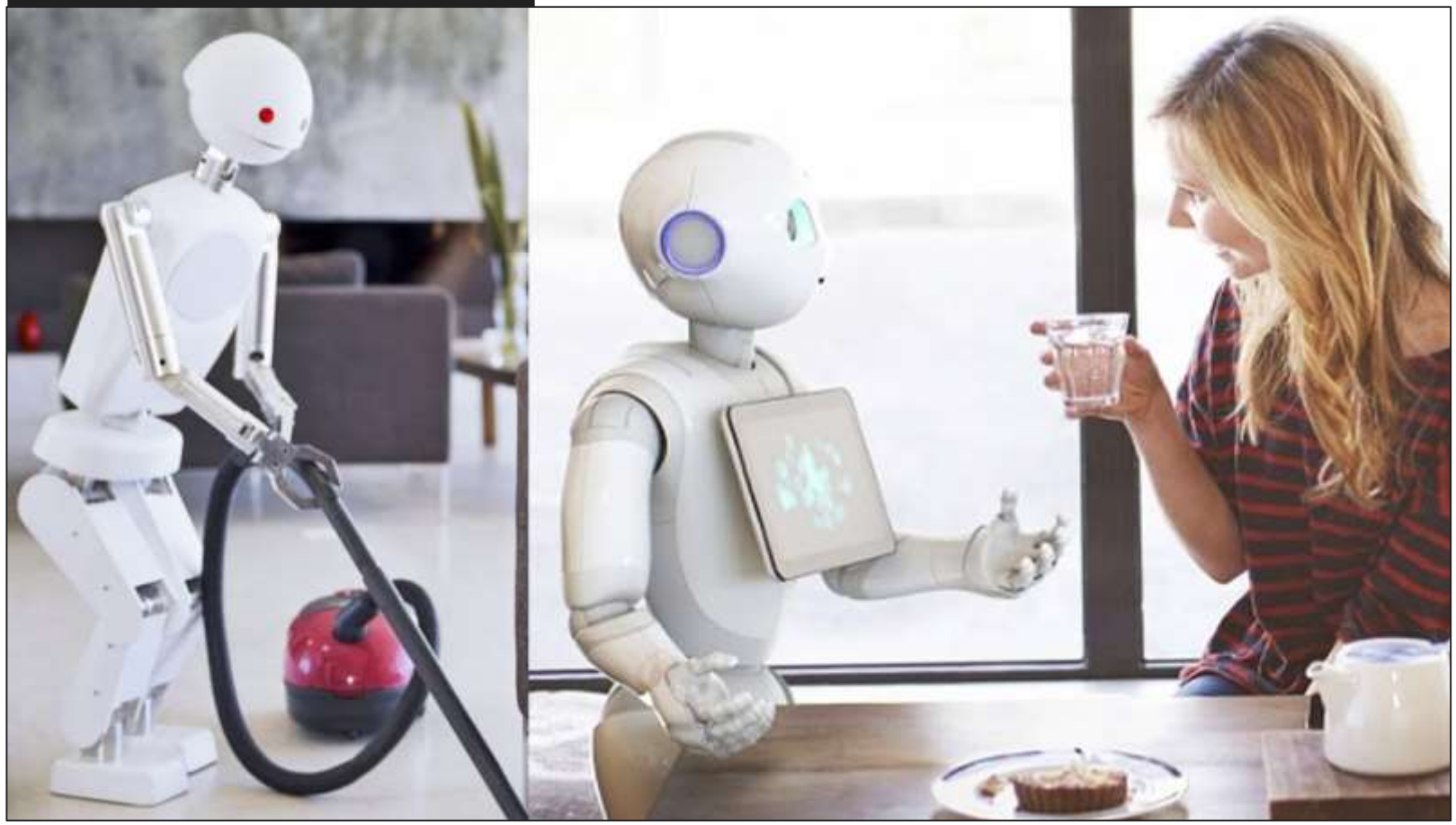
Reinforcement Learning



Robotics

Home Automation

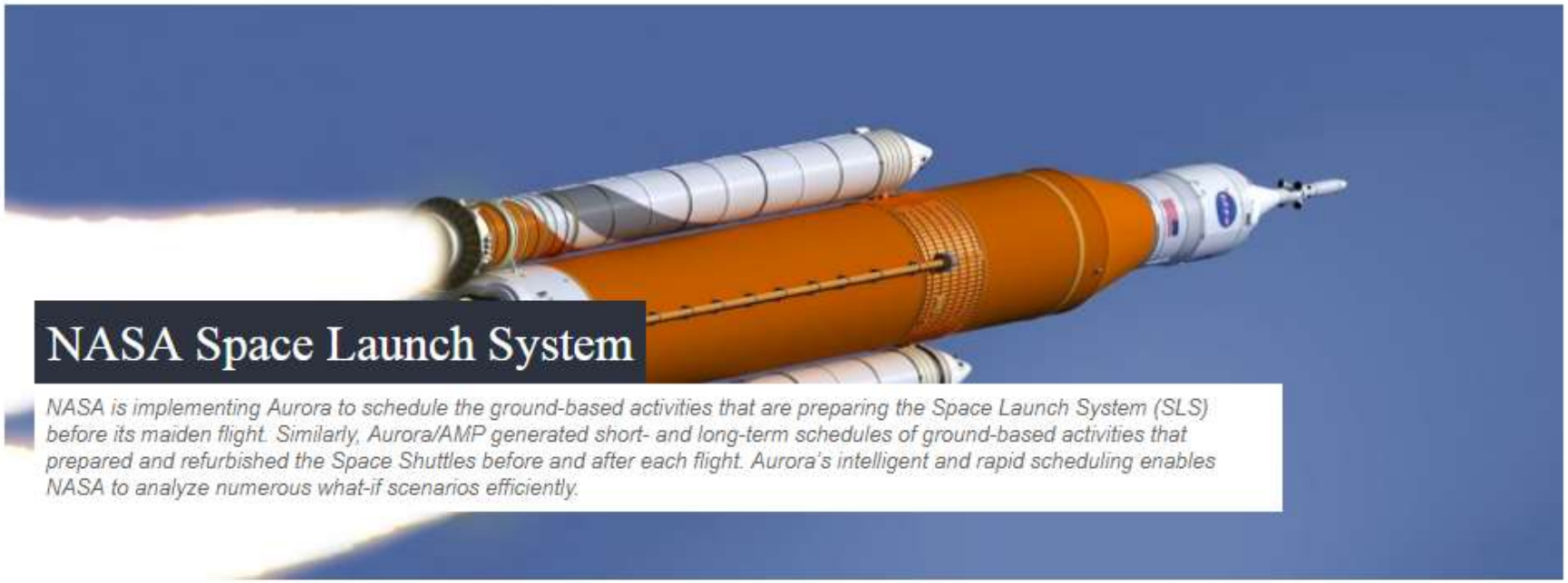
Deep Learning



Scheduling

Resource Scheduling

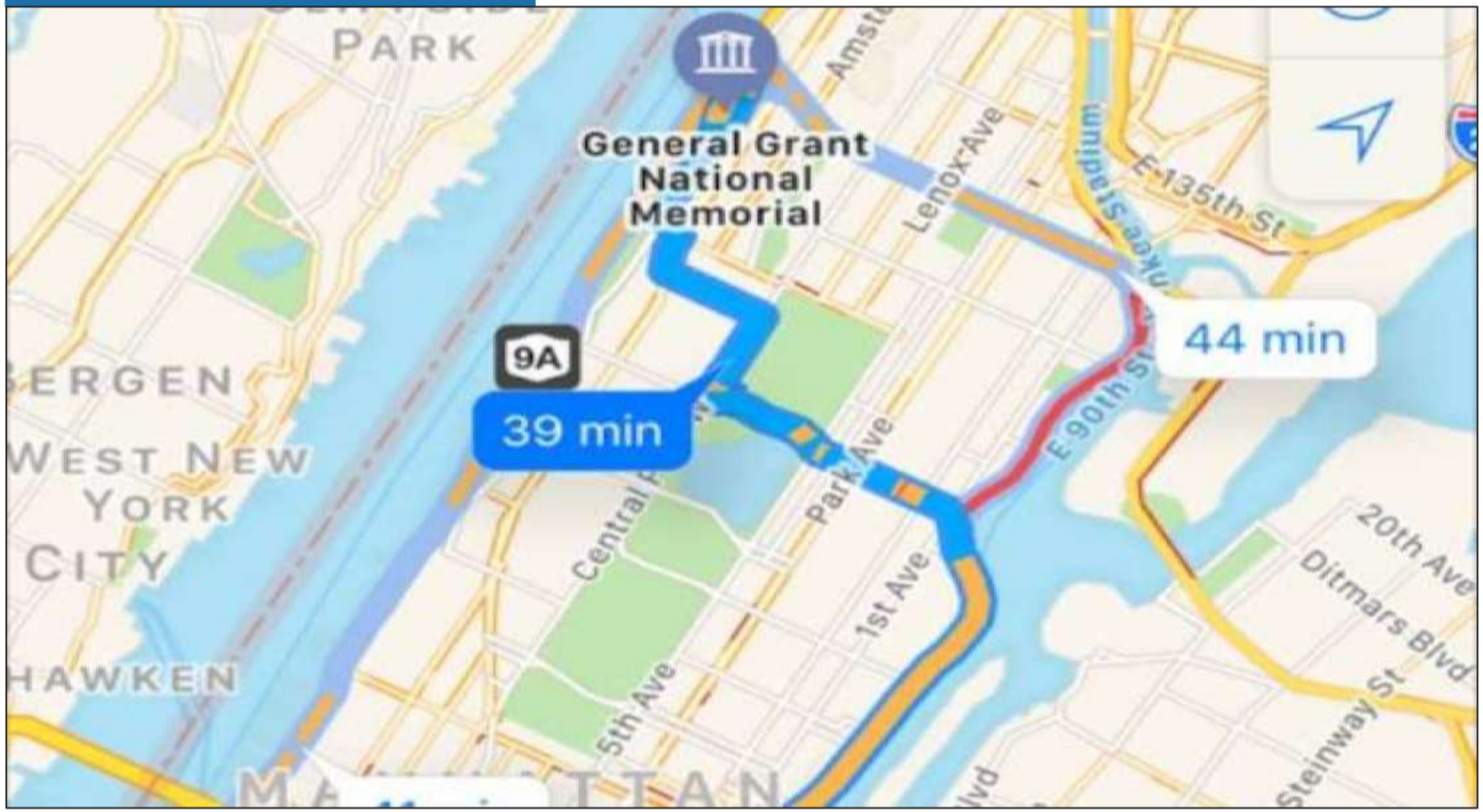
Aurora - Advanced Intelligent Planning and Scheduling Solution



Optimization

Shortest Path

Google map path planner



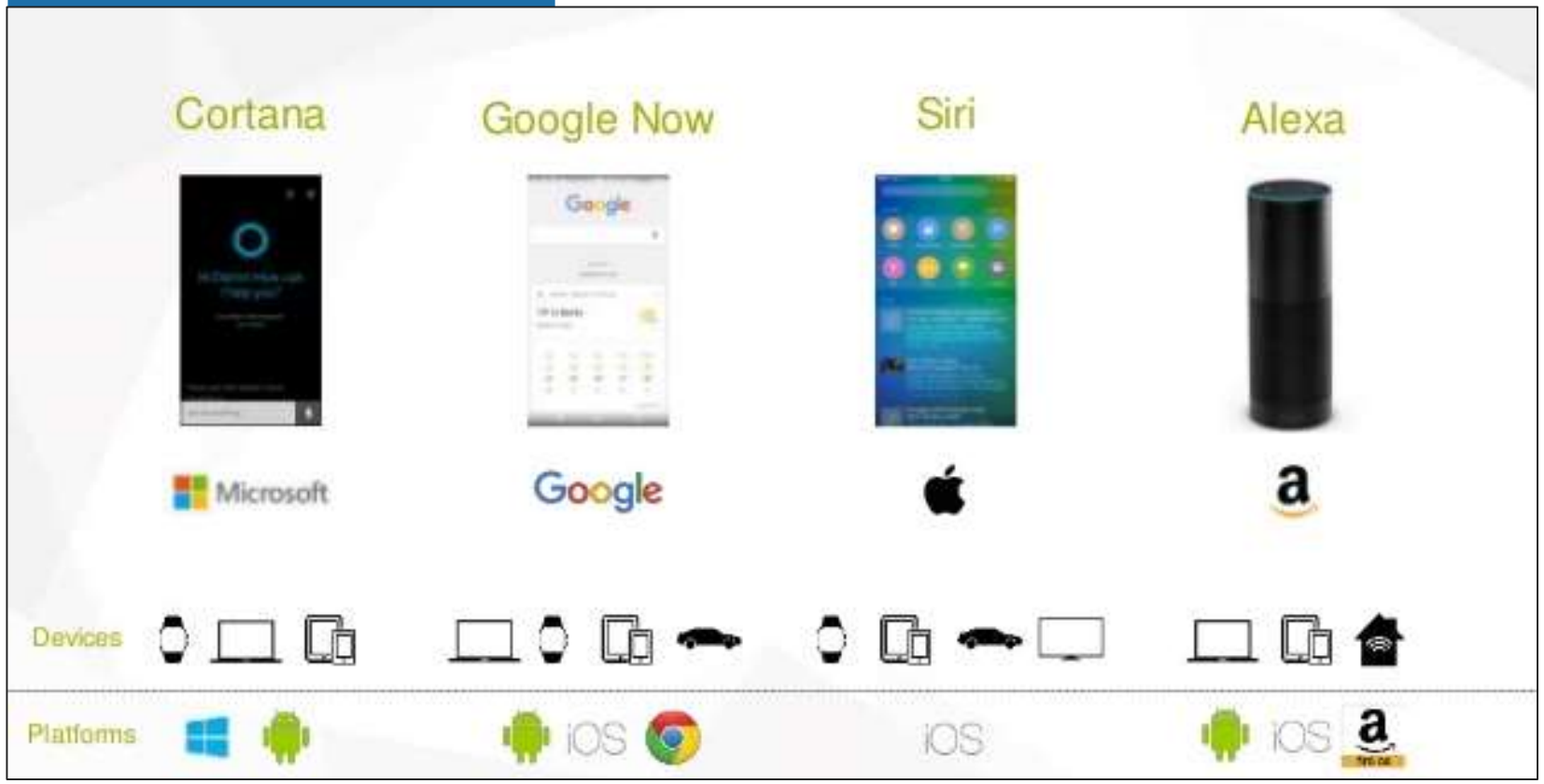
Game Playing

Alpha Go

Deep Neural Network

Virtual Agents

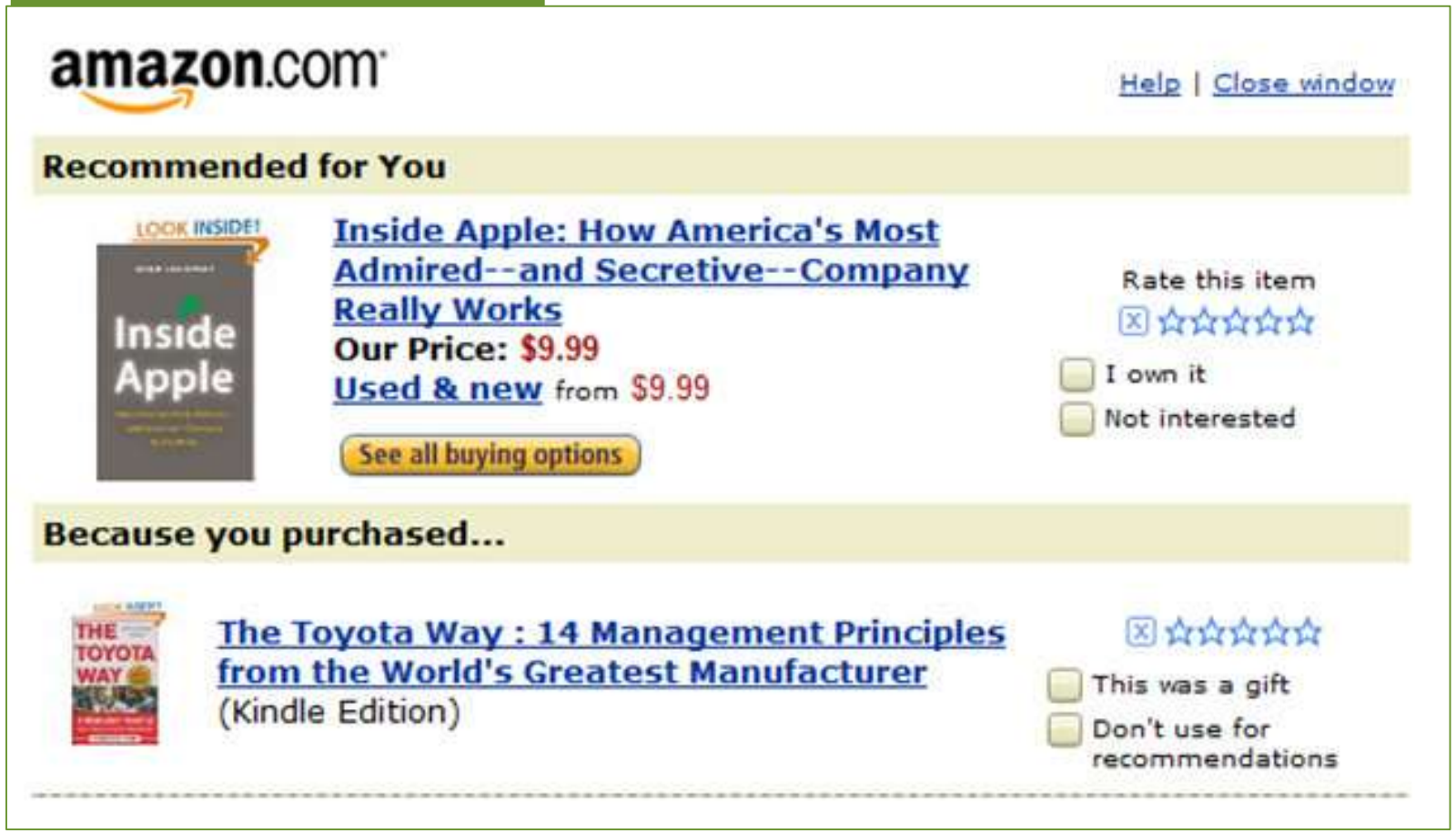
Chatbots | **Conversational AI**



Personalized Recommender Systems

Online Shopping

Machine Learning



Automated Control Systems

Washing Machine

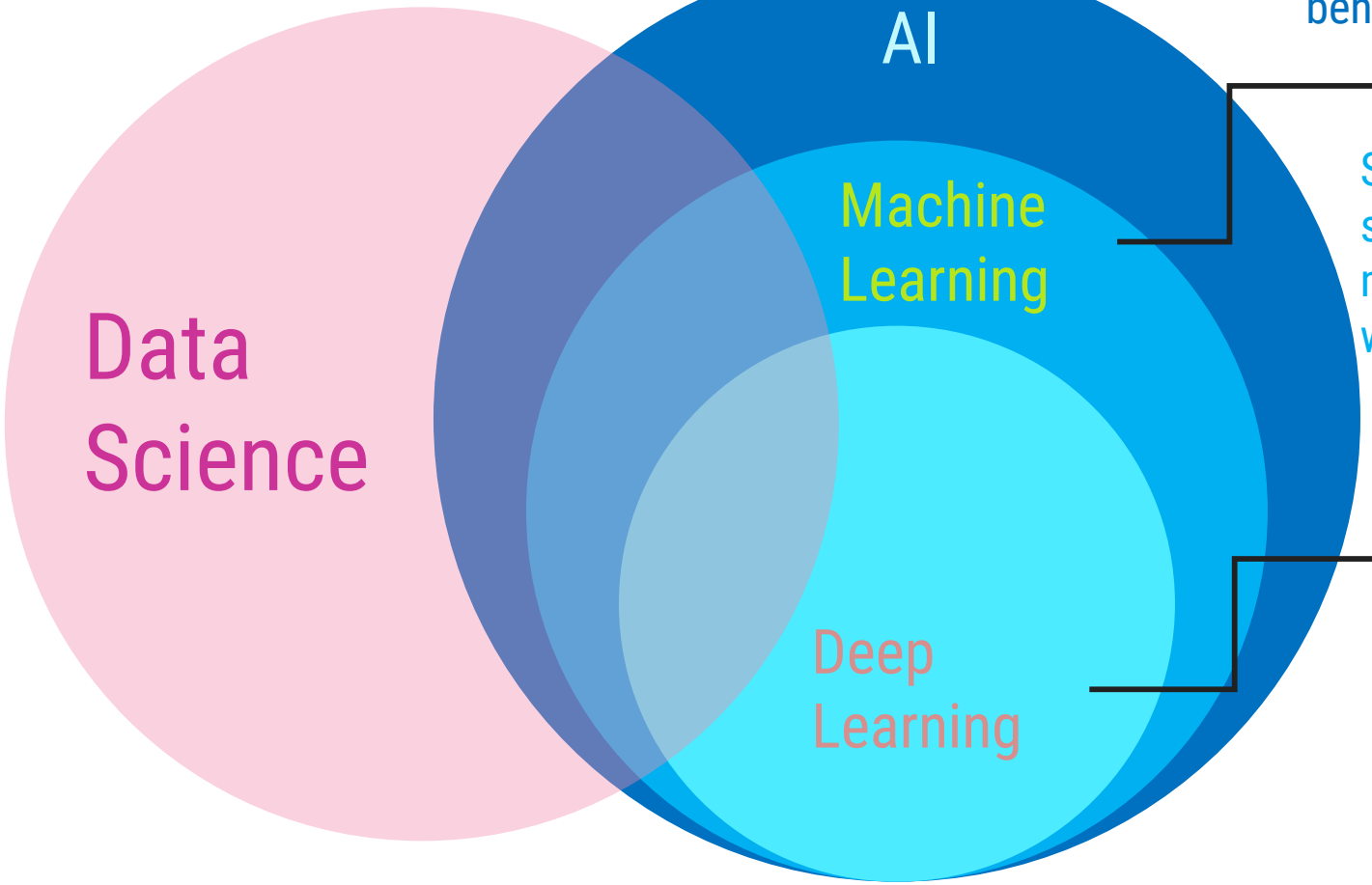
Fuzzy Logic

Security

NVIDIA Metropolis

Machine Learning

AI – ML – DL and Data Science



Technique that enables machines to mimic human behavior

Subset of **AI** which uses statistical methods to enable machine to learn and improve with time

Subset of **ML** that includes algorithms and enables system to train itself

Important Questions

Short Questions:

1. Define Artificial Intelligence.
2. Describe the characteristics of Artificial Intelligence.
3. List out the applications of Artificial Intelligence.
4. What are the four basic types of agent program in any intelligent system?
5. What are the four categories of AI?
6. Give a brief note on Alpha-Beta Pruning.
7. What are the foundations categories of AI?
8. List out the Sub areas of Artificial Intelligence.
9. Define a problem and its components.
10. What is A* search?
11. What is Greedy Best First Search?
12. What is Constrain Satisfaction Problem ?
13. Describe the characteristics of Problem.
14. What is structure of state space ?
15. What is control strategy?

Important Questions

Long Questions:

1. Define Artificial Intelligence. Explain the techniques of A.I. Also describe the characteristics of Artificial Intelligence.
2. Explain the state space representation of Water –Jug problem.
3. Explain in detail about Uninformed Search and Informed Search Strategies.
4. List and explain the applications of Artificial Intelligence.
5. Discuss the characteristics of AI problem. Can Towers of Hanoi problem be considered as AI problem? Justify your answer with suitable
6. What are the four basic types of agent program in any intelligent system? Explain how did you convert them into learning agents?
7. Explain the following uninformed search strategies with examples.
   (a) Breadth First Search.(b) Uniform Cost Search
   (c) Depth First Search. (d) Depth Limited Search
8. What is Greedy Best First Search? Explain with an example the different stages of Greedy Best First search.
9. What is A* search? Explain various stages of A* search with an example.
10. Explain the following local search strategies with examples.(i) Hill climbing (ii) Genetic Algorithms (iii) Constrain Satisfaction Problem

Thank You!

Unit-2
Logic Concepts, Knowledge Representation

Mr D Srinivas
Computer Science and Engineering
www.srinivas-materials.blogspot.com
srinivascsedept@gmail.com
9347556447

Outline

- **Introduction to Logic Concepts**
  - Introduction
  - Propositional calculus
  - Proportional logic
  - Natural deduction system
  - Axiomatic system
  - Semantic tableau system in proportional logic
  - Resolution refutation in proportional logic
  - Predicate logic
  - Logic Programming
- **Introduction to Knowledge Representation**
  - Introduction
  - Approaches to Knowledge representation,
  - Knowledge representation using Semantic Network,
  - Extended Semantic Network for KR,
  - Knowledge representation using Frames

Introduction to Logic Concepts

Introduction to Logic Concepts

- Logic
  - The logical formalism of a language is useful because it immediately suggests a powerful way of deriving new knowledge from old using mathematical deduction.
  - In this formalism, we can conclude that a new statement is true by proving that it follows from the statements that are already known.
- Proposition
  - A proposition is a statement, or a simple declarative sentence.
  - For example, “the book is expensive” is a proposition.
  - A proposition can be either true or false

Propositional Calculus

- Propositional Calculus (PC) is a language of propositions basically refers
  - – to set of rules used to combine the propositions to form compound propositions using logical operators often called connectives such as $\wedge$, $\vee$, ~, $\Rightarrow$,
  - Well-formed formula is defined as:
    - An atom is a well-formed formula.
    - – Ifα is a well-formed formula, then ~α is a well-formed formula.
    - – If α and β are well formed formulae, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \Rightarrow \beta)$, $(\alpha \Leftrightarrow \beta)$ are also well-formed formulae.
    - – A propositional expression is a well-formed formula if and only if it can be obtained by using above conditions.
- ***Truth Table***
  - Truth table gives us operational definitions of important logical operators.
    - By using truth table, the truth values of well-formed formulae are calculated.
  - Truth table elaborates all possible truth values of a formula.
  - The meanings of the logical operators are given by the following truth table.

P	Q	~P	P $\wedge$ Q	P $\vee$ Q	P $\Rightarrow$ Q	P $\Leftrightarrow$ Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Propositional Calculus

- ***Equivalence Laws:***
  - **Commutation**
    1. $P \wedge Q \equiv Q \wedge P$
    2. $P \vee Q \equiv Q \vee P$
  - **Association**
    1. $P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$
    2. $P \vee (Q \vee R) \equiv (P \vee Q) \vee R$
  - **Double Negation**
    $\sim (\sim P) \equiv P$
  - **Distributive Laws**
    1. $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$
    2. $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
  - **De Morgan's Laws**
    1. $\sim (P \wedge Q) \equiv \sim P \vee \sim Q$
    2. $\sim (P \vee Q) \equiv \sim P \wedge \sim Q$
  - **Law of Excluded Middle**
    $P \vee \sim P \equiv T$ (true)
  - **Law of Contradiction**
    $P \wedge \sim P \equiv F$ (false)

Propositional logic

- Logical constants: true, false
- Propositional symbols: P, Q, S,... (atomic sentences)
- Propositions are combined by connectives:

$\wedge$	and [conjunction]
$\vee$	or [disjunction]
$\Rightarrow$	implies [implication]
$\neg$	not [negation]
$\forall$	For all
$\exists$	There exists

- Representing simple facts using propositional logic:
  1. It is raining RAINING
  2. It is sunny SUNNY
  3. It is windy WINDY
  4. If it is raining, then it is not sunny

$$\text{RAINING} \rightarrow \neg \text{SUNNY}$$

Facts Represented as Well Formed Formula in FOPL

1. Marcus was a man.

   man(Marcus)

2. Marcus was a Pompeian.

   Pompeian(Marcus)

3. All Pompeians were Romans.

   $\forall$ x: Pompeian(x) $\rightarrow$ Roman(x)

4. Caesar was a ruler.

   ruler(Caesar)

5. All Pompeians were either loyal to Caesar or hated him.

   $\forall$x: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar)

Other methods

- Other methods which are concerned with proofs and deductions of logical formula are as follows:
  1. Natural Deductive System
  2. Axiomatic System
  3. Semantic Tableaux Method
  4. Resolution Refutation Method

Natural Deduction System(NDS)

- NDS is based on the set of few deductive inference rules.
  - The name natural deductive system is given because it mimics the pattern of natural reasoning.
  - It has about 10 deductive inference rules.
- **Conventions:**
  - – E for Elimination.
  - – P, Pk , $(1 \leq k \leq n)$ are atoms.
  - – α_k, $(1 \leq k \leq n)$ and β are formulae.
- ***Natural Deduction Rules:***

Rule 1: I- $\wedge$ (Introducing $\wedge$)

I- $\wedge$: If P1, P2, …, Pn then P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn

- ***Interpretation:*** If we have hypothesized or proved P1, P2, … and Pn , then their conjunction P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn is also proved or derived.

Rule 2: E- $\wedge$ (Eliminating $\wedge$)

E- $\wedge$: If P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn then Pi ($1 \leq i \leq n$)

- **Interpretation:** If we have proved P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn , then any Pi is also proved or derived. This rule shows that $\wedge$ can be eliminated to yield one of its conjuncts.

Rule 3: I-V (Introducing V)

I-V : If Pi ($1 \leq i \leq n$) then P1V P2 V …V Pn

- **Interpretation:** If any Pi $(1 \leq i \leq n)$ is proved, then P1V …V Pn is also proved.

Natural Deduction System(NDS)

- ***Natural Deduction Rules:***

Rule 4: E-V (Eliminating V)

E-V : If P1 V … V Pn, P1 $\Rightarrow$ P, … , Pn $\Rightarrow$ P then P

- **Interpretation:** If P1 V … V Pn, P1 $\Rightarrow$ P, … , and Pn $\Rightarrow$ P are proved, then P is proved.

Rule 5: I- $\Rightarrow$ (Introducing $\Rightarrow$)

I- $\Rightarrow$: If **from α1, …, αn infer β is proved** then **α1 $\wedge$ … $\wedge$ αn $\Rightarrow$ β** is **proved**

- ***Interpretation:*** If given α1, α2, …and αn to be proved and from these we deduce β then $\alpha 1 \wedge \alpha 2 \wedge \ldots \wedge \alpha n \Rightarrow \beta$ is also proved.

Rule 6: E- $\Rightarrow$ (Eliminating $\Rightarrow$) - *Modus Ponen*

E- $\Rightarrow$: If P1 $\Rightarrow$ P, P1 then P

Rule 7: I- $\Leftrightarrow$(Introducing $\Leftrightarrow$)

I- $\Leftrightarrow$: If P1 $\Rightarrow$ P2, P2 $\Rightarrow$ P1 then P1 $\Leftrightarrow$ P2

Rule 8: E- $\Leftrightarrow$ (Elimination $\Rightarrow$)

E- $\Leftrightarrow$: If $P_1 \Leftrightarrow P_2$ then $P_1 \Rightarrow P_2$, $P_2 \Rightarrow P_1$

Rule 9: I- ~ (Introducing ~)

I- ~ : If **from P infer $P_1 \wedge$ ~ P_1** is proved then **~P** is proved

Rule 10: E- ~ (Eliminating ~)

E- ~ : If **from ~ P infer $P_1 \wedge$ ~ P_1** is proved then **P** is proved

Natural Deduction System(NDS)

- **Deduction Theorem:** To prove a formula $\alpha 1 \wedge \alpha 2 \wedge \ldots \wedge \alpha n \Rightarrow \beta$, it is sufficient to prove a theorem **from** α**1,** α **2, …,** α**n infer** β.
- **Example1:** Prove that P $\wedge$(QVR) follows from P $\wedge$ Q
- **Solution:** This problem is restated in natural deductive system as "**from P Q infer P (Q V R)**". The formal proof is given as follows:

{Theorem}	**from P $\wedge$ Q infer P $\wedge$ (Q V R)**	
{ premise}	P $\wedge$ Q	(1)
{ E- $\wedge$, (1)}	P	(2)
{ E- $\wedge$, (1)}	Q	(3)
{ I-V , (3) }	Q V R	(4)
{ I- $\wedge$, (2, 4)}	**P $\wedge$(Q V R)**	**Conclusion**

Axiomatic system

- It is based on the set of only three axioms and one rule of deduction.
  - It is minimal in structure but as powerful as the truth table and natural deduction approaches.
  - The proofs of the theorems are often difficult and require a guess in selection of appropriate axiom(s) and rules.
  - These methods basically require forward chaining strategy where we start with the given hypotheses and prove the goal.
  - **Axioms**
    1. **Axiom1** (A1): $\alpha \Rightarrow (\beta \Rightarrow \alpha)$
    2. **Axiom2** (A2): $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$
    3. **Axiom3** (A3): $(\sim \alpha \Rightarrow \sim \beta) \Rightarrow (\beta \Rightarrow \alpha)$
  - **Rule:**
    - **Modus Ponen (MP)** defined as follows:

    Hypotheses: $\alpha \Rightarrow \beta$ and α ***Consequent:*** β
  - **Examples:** Establish the following:

    {Q} |-(P ⇒ Q) i.e., P ⇒ Q is a deductive consequence of {Q}.

{Hypothesis}	Q	(1)
{Axiom A1}	Q ⇒(P ⇒ Q)	(2)
{MP, (1,2)}	**P ⇒ Q**	**proved**

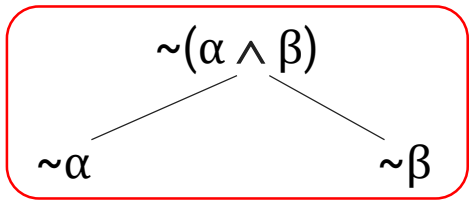
Semantic tableau system in proportional logic

- Earlier approaches require
  - construction of proof of a formula from given set of formulae and are called direct methods.
  - In **semantic tableaux**,
    - the set of rules are applied systematically on a formula or set of formulae to establish its consistency or inconsistency.
  - *Semantic tableau*
    - binary tree constructed by using semantic rules with a formula as a root
  - Assume α and β be any two formulae.
- ***Semantic Tableaux Rules***

Rule 1: A tableau for the a formula $(\alpha \wedge \beta)$ is constructed by adding both α and β to the same path (branch). This can be represented by

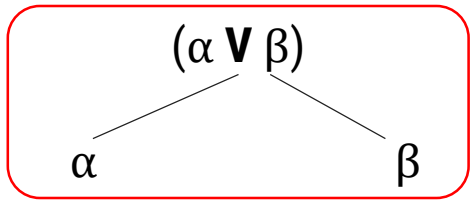
$(\alpha \wedge \beta)$
α
β

Rule 2: A tableau for the a formula $\sim(\alpha \wedge \beta)$ is constructed by adding two alternative paths one containing $\sim\alpha$ and other containing $\sim\beta$. This can be represented by

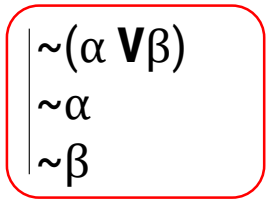


Semantic tableau system in proportional logic

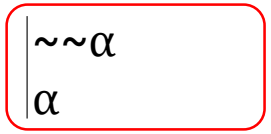
Rule 3: A tableau for the a formula (α **V** β) is constructed by adding two alternative paths one containing $\sim\alpha$ and other containing $\sim\beta$. This can be represented by



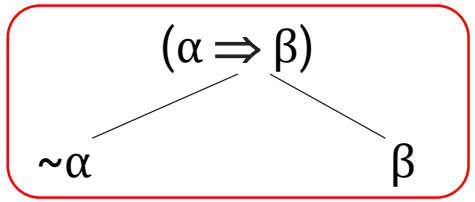
Rule 4: A tableau for the a formula (α **V** β) is constructed by adding both α and β to the same path (branch). This can be represented by



Rule 5:



Rule 6:

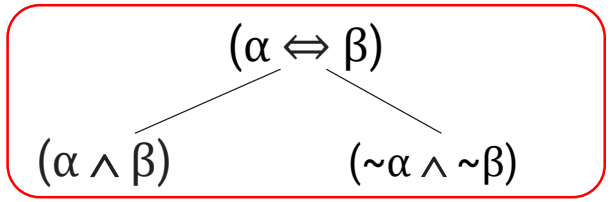


Semantic tableau system in proportional logic

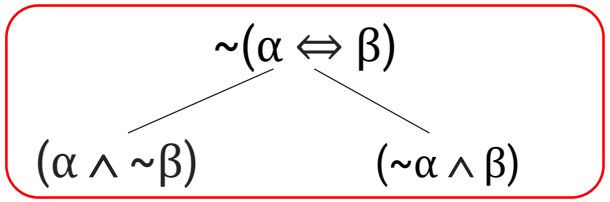
Rule 7:

$\sim(\alpha \Rightarrow \beta)$
α
$\sim \beta$

Rule 8: $(\alpha \Leftrightarrow \beta) \equiv (\alpha \wedge \beta) \vee (\sim\alpha \wedge \sim\beta)$



Rule 9: $\sim(\alpha \Leftrightarrow \beta) \equiv (\alpha \wedge \sim\beta) \vee (\sim\alpha \wedge \beta)$

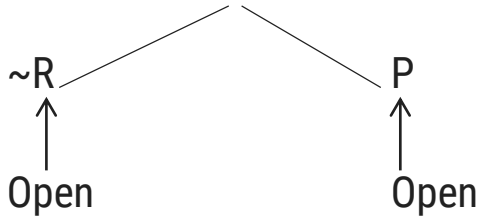


Semantic tableau system in proportional logic

- ***Consistency and Inconsistency***
  - If an atom P and ~ P appear on a same path of a semantic tableau,
    - then inconsistency is indicated and such path is said to be c**ontradictory** or **closed** (finished) path.
    - Even if one path remains **non contradictory** or **unclosed** (open), then the formula α at the root of a tableau is **consistent**.
  - **Contradictory tableau** (or **finished tableau):**
    - It defined to be a tableau in which all the paths are contradictory or closed (finished).
  - If a tableau for a formula α at the root is a contradictory tableau**,**
    - then a formula α is said to be inconsistent.
  - Example:

    Show that α: $(Q \wedge \sim R) \wedge (R \Rightarrow P)$ is consistency and finds the its model

{Tableau Root}	$(Q \wedge \sim R) \wedge (R \Rightarrow P)$	(1)
{Apply rule 1 to 1}	$(Q \wedge \sim R)$	(2)
	$(R \Rightarrow P)$	(3)
{Apply rule 1 to 2}	Q	
{Apply rule 6 to 3}	~R	



{Q=T, R=F} and {P=T, Q=T, R=F} are models for α

Resolution Refutation

- Resolution is a procedure, which gains its efficiency from the fact that it operates on statements that have been converted to a very convenient standard form.
- Resolution produces proofs by refutation.
- In other words, to prove a statement (i.e., to show that it is valid), resolution attempts to show that the negation of the statement produces a contradiction with the known statements (that it is un-satisfiable).
- The resolution procedure is a simple iterative process: at each step, two clauses, called the parent clauses, are compared (resolved), resulting into a new clause that has been inferred from them.
- The new clause represents ways that the two parent clauses interact with each other.

Predicate Logic

- Predicate logic builds heavily upon the ideas of proposition logic to provide a more powerful system for expression and reasoning. As we have already mentioned, a **predicate** is just a function with a range of two values, say false and true.
- **Predicate logic** deals with the combination of predicates using the propositional operators we have already studied. It also adds one more interesting element, the "quantifiers".
- The meaning of predicate logic expressions is suggested by the following:

Expression + Interpretation + Assignment = Truth Value

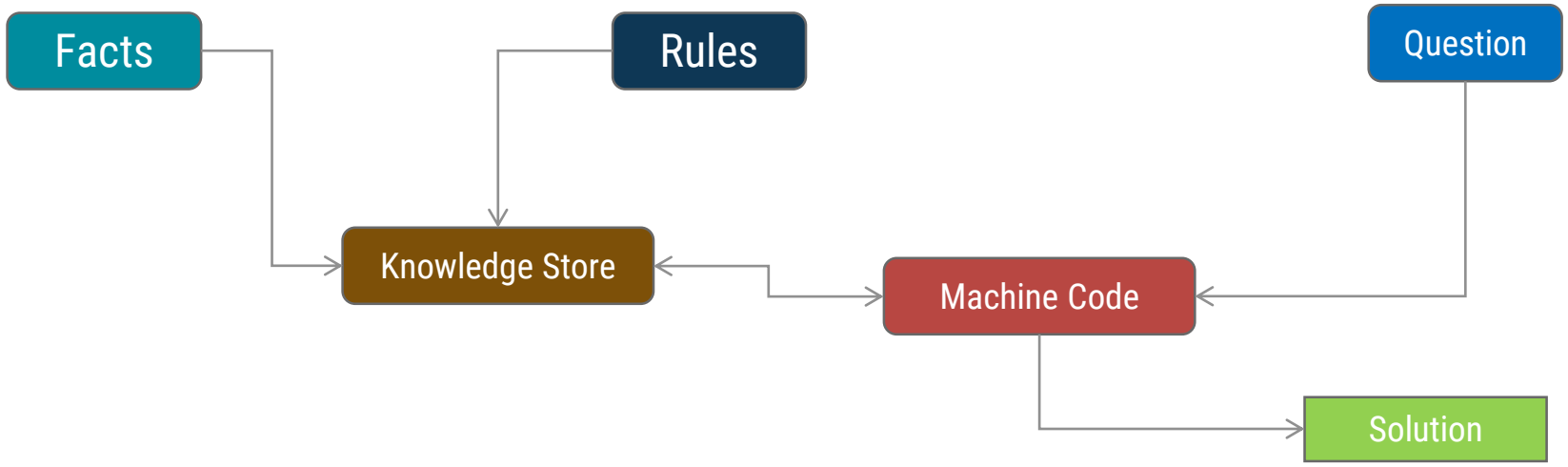
- An **interpretation** for a predicate logic expression consists of:
  - a domain for each variable in the expression
  - a predicate for each predicate symbol in the expression
  - a function for each function symbol in the expression
- An **assignment** for a predicate logic expression consists of:
  - a value for each variable in the expression
- **Example**

  Consider the expression: x < y || (y < z && z < x)

  Here || and && are propositional operators and < is a predicate symbol

Logic Programming

- The logic programming used to address information in rationale writing computer programs is a clausal structure which is a subset of first-request predicate logic.
- It is utilized because the first-request logic is surely known and ready to address every single computational issue.
- Information is controlled utilizing the goal surmising framework, which is needed for demonstrating hypotheses in clausal-structure logic.
- Ex: X is an integer.
  - Here ‘X’ is –Variable
  - ‘Is an integer’ - Fact



Introduction to Knowledge Representation

Knowledge *Representation*

- *Knowledge representation (KR)* is an important issue in both cognitive science and artificial intelligence.
  - In cognitive science, it is concerned with the way people store and process information and
  - In artificial intelligence (AI), main focus is to store knowledge so that programs can process it and achieve human intelligence.
- There are different ways of representing knowledge e.g.
  - predicate logic,
  - semantic networks,
  - extended semantic net,
  - frames,
  - conceptual dependency etc.
- In predicate logic, knowledge is represented in the form of rules and facts as is done in Prolog.

Procedural and Declarative Knowledge

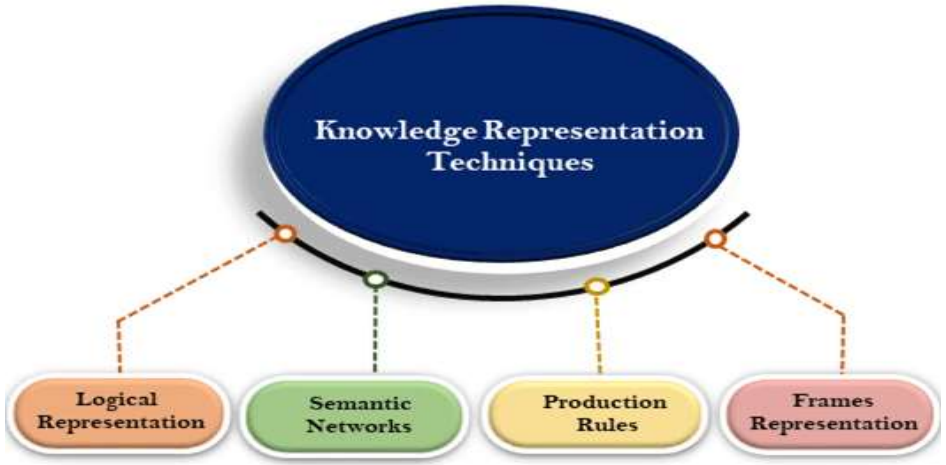
- Procedural Knowledge :
  - Procedural Knowledge is the type of knowledge which clarifies how a particular thing can be accomplished.
  - It emphasize how to do something to solve a given problem.
  - A representation in which the control information that is necessary to use the knowledge is embedded in the knowledge itself for e.g. computer programs, directions, and recipes; these indicate specific use or implementation;
- Declarative Knowledge :
  - Declarative Knowledge is the type of knowledge which tells the basic knowledge about something and it is more popular than Procedural Knowledge.
  - It emphasize what to do something to solve a given problem.
  - A statement in which knowledge is specified, but the use to which that knowledge is to be put is not given. For example, laws, these are the facts which can stand alone, not dependent on other knowledge.

Differences Between Declarative knowledge and Procedural Knowledge

Procedural knowledge	Declarative knowledge
• It is also known as Interpretive knowledge.	• It is also known as Descriptive knowledge.
• Procedural Knowledge means how a particular thing can be accomplished.	• While Declarative Knowledge means basic knowledge about something.
• High efficiency and Low modifiability	• Good modifiability and good readability
• Low perceptive adequacy (better for knowledge engineers)	• Suitable for independent facts
• Produces creative, reflective thought and promoters critical thinking and independent decision making	• Good cognitive matching (better for domain experts and end-users) and low computational efficiency.

Knowledge *Representation*

- *Knowledge representation (KR)* is an important issue in both cognitive science and artificial intelligence.
  - In cognitive science, it is concerned with the way people store and process information and
  - In artificial intelligence (AI), main focus is to store knowledge so that programs can process it and achieve human intelligence.
- There are mainly four ways of knowledge representation which are given as follows:
  - Logical Representation
  - Semantic Network Representation
  - Frame Representation
  - Production Rules



- In predicate logic, knowledge is represented in the form of rules and facts as is done in Prolog.

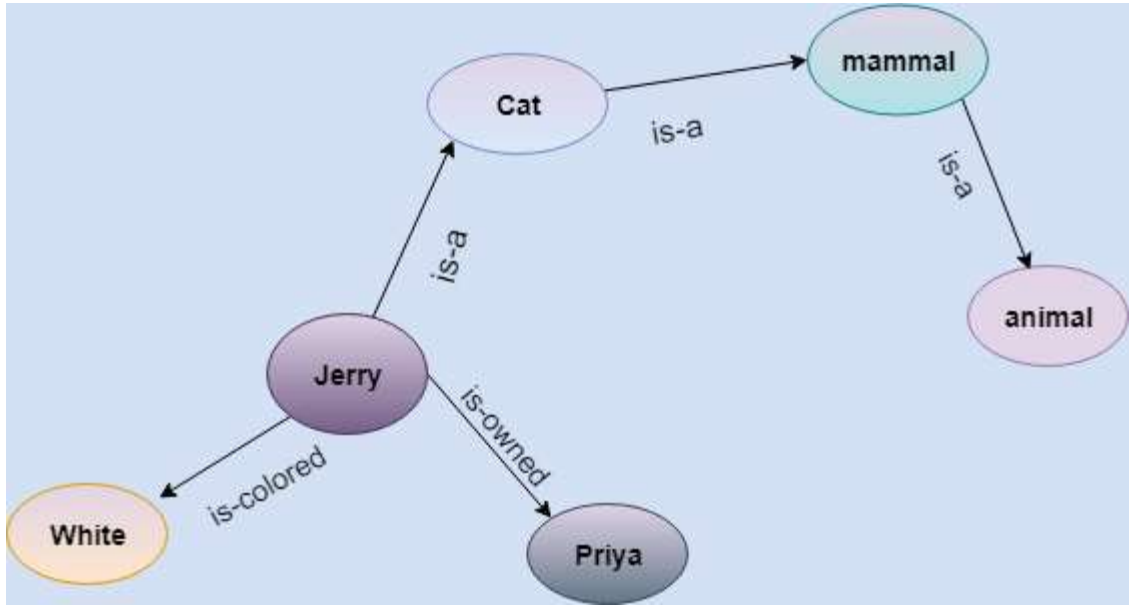
Logical Representation

- Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation.
- Logical representation means drawing a conclusion based on various conditions.
- This representation lays down some important communication rules.
- It consists of precisely defined syntax and semantics which supports the sound inference.
- Each sentence can be translated into logics using syntax and semantics.
- **Syntax:**
  - Syntaxes are the rules which decide how we can construct legal sentences in the logic.
  - It determines which symbol we can use in knowledge representation.
  - How to write those symbols.
- **Semantics:**
  - Semantics are the rules by which we can interpret the sentence in the logic.
  - Semantic also involves assigning a meaning to each sentence.
- Logical representation can be categorized into mainly two logics:
  - Propositional Logics
  - Predicate logics

Semantic networks

- Semantic networks are alternative of predicate logic for knowledge representation.
- In Semantic networks, we can represent our knowledge in the form of graphical networks.
- This network consists of nodes representing objects and arcs which describe the relationship between those objects.
- Semantic networks can categorize the object in different forms and can also link those objects.
- Semantic networks are easy to understand and can be easily extend
- This representation consist of mainly two types of relations:
  - IS-A relation (Inheritance)
  - Kind-of-relation

- **Statements:**
  - Jerry is a cat.
  - Jerry is a mammal
  - Jerry is owned by Priya.
  - Jerry is brown colored.
  - All Mammals are animal.



Frame Representation

- A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world.
- Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations.
- It consists of a collection of slots and slot values. These slots may be of any type and sizes.
- Slots have names and values which are called facets.

Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

Production Rules

- Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:
  - The set of production rules
  - Working Memory
  - The recognize-act-cycle
- In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out.
- The condition part of the rule determines which rule may be applied to a problem. And the action part carries out the associated problem-solving steps.
- This complete process is called a recognize-act cycle.
- Example:

  IF (at bus stop AND bus arrives) THEN action (get into the bus)

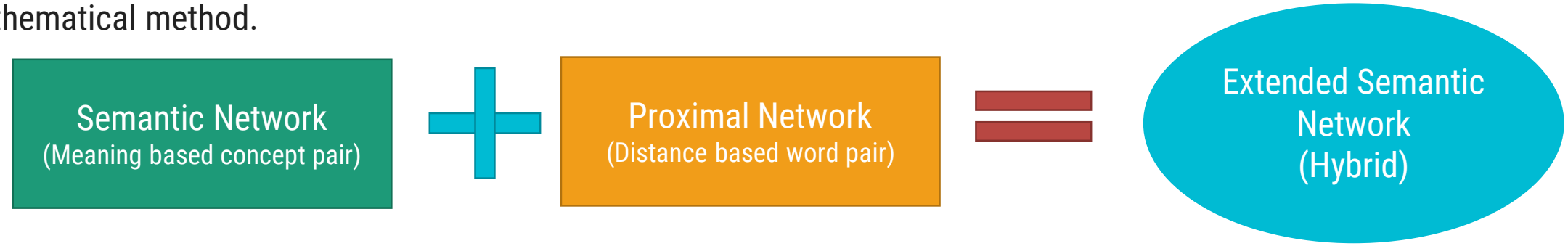
  IF (on the bus AND paid AND empty seat) THEN action (sit down).

  IF (on bus AND unpaid) THEN action (pay charges).

  IF (bus arrives at destination) THEN action (get down from the bus).

Extended Semantic networks

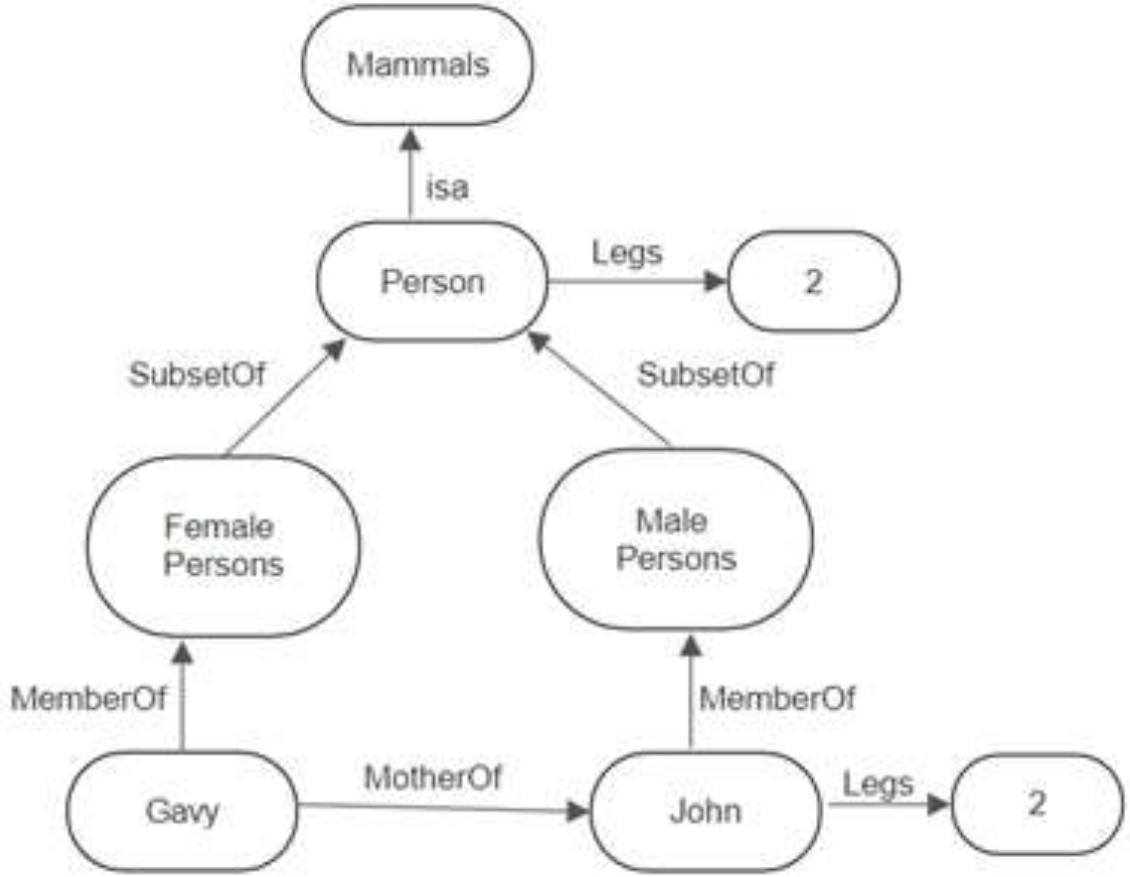
- Extended Semantic Network is data representing network resulting from the collaboration involving two networks, one automatically constructed proximal network and the second manually constructed semantic network.
- Here, the primary idea is to develop a modern approach combining the features of man and machine theory of concept , which can be of enormous use in the latest knowledge representation, classification, pattern matching and ontology development fields.
- We propose to visualize a novel method for knowledge representation partly based on mind modeling and partly on the mathematical method.



- All the links used in connecting the node is based on the JML links, consisting of four different types of associative lines as shown below.

Extended Semantic networks

- Example of Extended Semantic Network Knowledge Representation



Important Questions

Short Questions:

1. Define Knowledge.
2. What is Knowledge Representation?.
3. List out the approaches for Knowledge Representation.
4. What Propositional Logic?
5. Define Predicate logic.
6. What is semantic network?
7. Define Frame Representation for knowledge.
8. What is Proximal network?.
9. What is Axiomatic system?
10. What are the rules for Semantic tableau?
11. What is Natural deduction system?
12. What is Logic Programming?
13. Differentiate the inductive logic and deductive logic.
14. Define the Resolution Refutation Method.
15. What is Propositional Calculus?

Important Questions

Long Questions:

1. Define knowledge representation. Explain List the different approaches for Knowledge Representation?
2. Explain the different rules in Sematic tableaux system in propositional logic.
3. Explain the different rules in natural deduction system in propositional logic.
4. Explain the different rules in Axiomatic system in propositional logic.
5. Define Logic . Explain the different types of logics?
6. Explain the knowledge representation using semantic network.
7. Explain the knowledge representation using frames network.
8. Explain the knowledge representation using Extended semantic network.
9. Explain the knowledge representation using predicate logic.
10. Explain the Resolution refutation in proportional logic.

Thank You!

Unit-3
Expert System, Uncertainty Measure - Probability Theory

Mr D Srinivas
Computer Science and Engineering
www.srinivas-materials.blogspot.com
srinivascsedept@gmail.com
9347556447

Outline

- **Introduction to Expert System**
  - Introduction
  - Phases in Building Expert System
  - Expert System Architecture
  - Expert System vs Traditional System
  - Truth Maintenance System
  - Application of Expert System
  - List of Shells and Tools
- **Introduction to Uncertainty Measure – Probability Theory**
  - Introduction
  - Probability Theory,
  - Bayesian Belief Networks,
  - Certainty Factory Theory,
  - Dempster - Shafer Theory

Introduction to Expert System

Introduction to Expert System

Expert System

- An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert.
- **Expert System** is an interactive and reliable computer-based decision-making system which uses both facts and heuristics to solve complex decision-making problems.
- The concept of expert systems was first developed in the 1970s by Edward Feigenbaum, professor and founder of the Knowledge Systems Laboratory at Stanford University.
- Feigenbaum explained that the world was moving from data processing to knowledge processing, a transition which was being enabled by new processor technology and computer architectures.
- An expert system solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base.
- The knowledge is extracted from its knowledge base using the reasoning and inference rules according to the user queries.
- It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of the performance of an expert system is based on the expert's knowledge stored in its knowledge base.
- The more knowledge stored in the KB, the more that system improves its performance.
- One of the most common example is, making of a medical diagnosis expert system in which a medical diagnosis expert system lets the user diagnose his disease without going to a real doctor.
- Typically, an expert system incorporates a knowledge base containing accumulated experience and an inference or rules engine -- a set of rules for applying the knowledge base to each particular situation that is described to the program.
- The system's capabilities can be enhanced with additions to the knowledge base or to the set of rules.
- Current systems may include Machine Learning capabilities that allow them to improve their performance based on experience, just as humans do that particular domain. These systems are designed for a specific domain, such as medicine, science, etc.

Introduction to Expert System

Expert Systems in Artificial Intelligence

- The Expert System in AI can resolve many issues which generally would require a human expert. It is based on knowledge acquired from an expert. Artificial Intelligence and Expert Systems are capable of expressing and reasoning about some domain of knowledge.

Examples of Expert Systems

- **MYCIN:** It was based on backward chaining and could identify various bacteria that could cause acute infections. It could also recommend drugs based on the patient's weight. It is one of the best Expert System Example.
- **DENDRAL:** Expert system used for chemical analysis to predict molecular structure.
- **PXDES:** An Example of Expert System used to predict the degree and type of lung cancer
- **CaDet:** One of the best Expert System Example that can identify cancer at early stages

Characteristics of Expert System

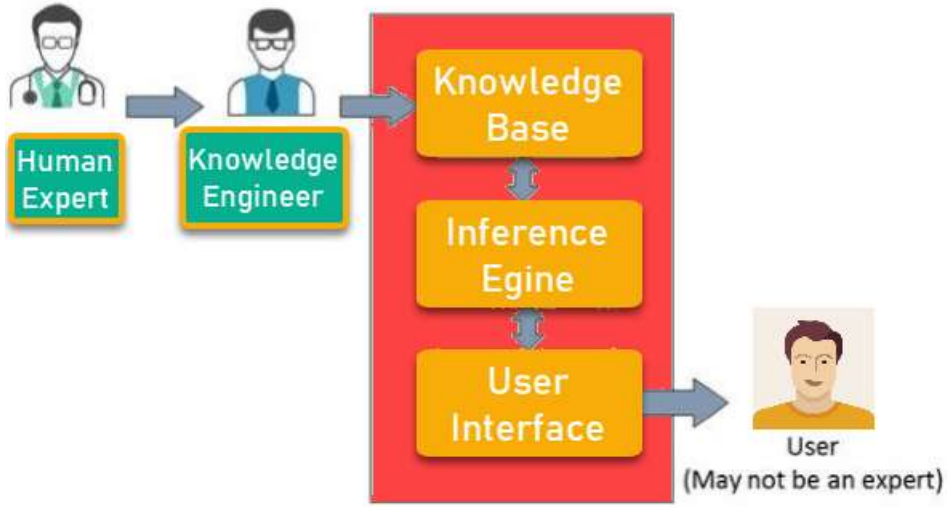
- Following are the important Characteristics of Expert System in AI:
  - **The Highest Level of Expertise:**
    - The Expert system in AI offers the highest level of expertise. It provides efficiency, accuracy and imaginative problem-solving.
  - **Right on Time Reaction:**
    - An Expert System in Artificial Intelligence interacts in a very reasonable period of time with the user. The total time must be less than the time taken by an expert to get the most accurate solution for the same problem.
  - **Good Reliability:**
    - The Expert system in AI needs to be reliable, and it must not make any a mistake.
  - **Flexible:**
    - It is vital that it remains flexible as it the is possessed by an Expert system.
  - **Effective Mechanism:**
    - Expert System in Artificial Intelligence must have an efficient mechanism to administer the compilation of the existing knowledge in it.
  - **Capable of handling challenging decision & problems:**
    - An expert system is capable of handling challenging decision problems and delivering solutions.

Components of Expert System

The Expert System in AI consists of the following given components:

1. User Interface

- The user interface is the most crucial part of the Expert System Software. This component takes the user's query in a readable form and passes it to the inference engine. After that, it displays the results to the user. In other words, it's an interface that helps the user communicate with the expert system.



2. Inference Engine

- The inference engine is the brain of the expert system. Inference engine contains rules to solve a specific problem. It refers the knowledge from the Knowledge Base. It selects facts and rules to apply when trying to answer the user's query. It provides reasoning about the information in the knowledge base. It also helps in deducting the problem to find the solution. This component is also helpful for formulating conclusions.

3. Knowledge Base

- The knowledge base is a repository of facts. It stores all the knowledge about the problem domain. It is like a large container of knowledge which is obtained from different experts of a specific field.

Other Key terms used in Expert Systems

- **Facts and Rules**
  - A fact is a small portion of important information. Facts on their own are of very limited use. The rules are essential to select and apply facts to a user problem.
- **Knowledge Acquisition**
  - The term knowledge acquisition means how to get required domain knowledge by the expert system. The entire process starts by extracting knowledge from a human expert, converting the acquired knowledge into rules and injecting the developed rules into the knowledge base.

The process of Building An Expert Systems

- 1. Identification:
- 2. Conceptualisation:
- 3. Formalisation (Designing):
- 4. Implementation:
- 5. Testing (Validation, Verification and Maintenance):

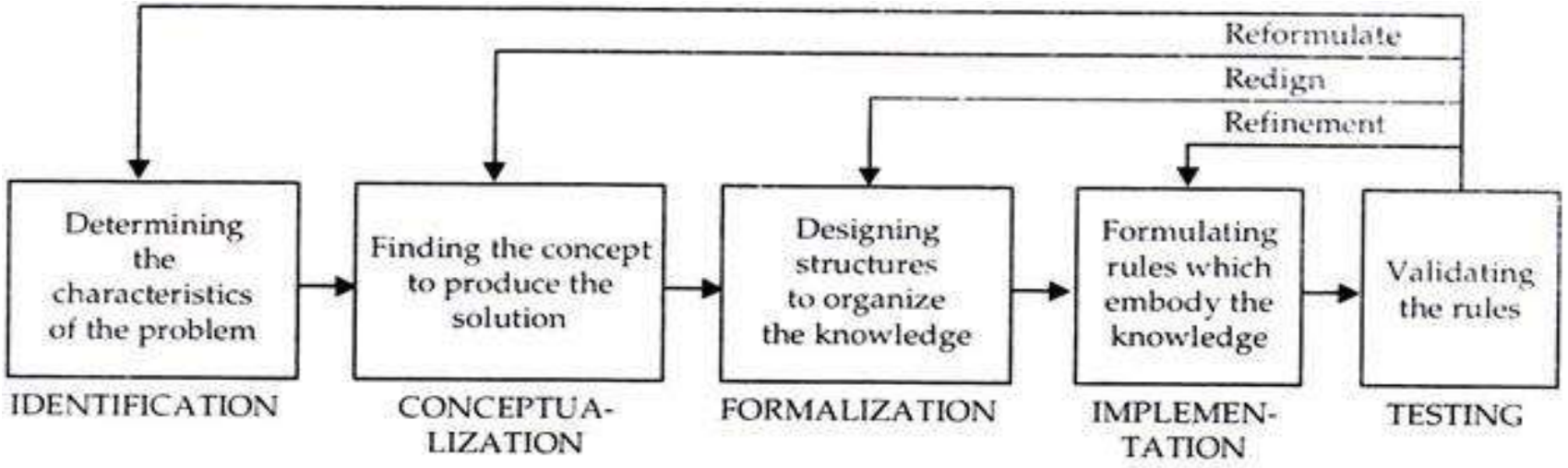


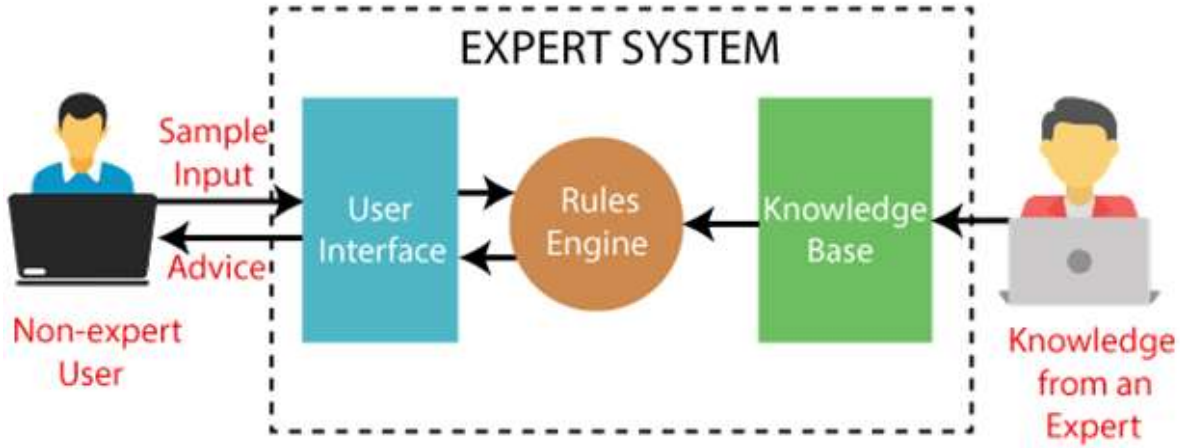
Fig. 12.8. *The five stages of expert system development.*

The process of Building An Expert Systems

- **1. Identification:**
  - Knowledge engineer finds out important features of the problem with the help of domain expert (human).
  - He tries to determine the type and scope of the problem, the kind of resources required, goal and objective of the ES.
- **2. *Conceptualization Phase*:**
  - In this phase, knowledge engineer and domain expert decide the concepts, relations and control mechanism needed to describe a problem solving.
- ***3. Formalization Phase:***
  - It involves expressing the key concepts and relations in some framework supported by ES building tools.
  - Formalized knowledge consists of data structures, inference rules, control strategies and languages for implementation.
- **4. *Implementation Phase:***
  - During this phase, formalized knowledge is converted to working computer program initially called prototype of the whole system.
- **5. *Testing Phase:***
  - It involves evaluating the performance and utility of prototype systems and revising it if need be.
  - Domain expert evaluates the prototype system and his feedback help knowledge engineer to revise it.

Expert System Architecture

- An expert system is a set of programs that manipulate encoded knowledge to solve problems in a specialized domain that normally requires human expertise.
- An expert system's knowledge is obtained from expert sources and coded in a form suitable for the system to use in its inference or reasoning processes.
- The expert knowledge must be obtained from specialists or other sources of expertise, such as texts, journal, articles and databases.
- This type of knowledge usually requires much training and experience in some specialized field such as medicine, geology, system configuration, or engineering design.
- Once a sufficient amount of expert knowledge has been acquired, it must be encoded in some form, loaded into a knowledge base, then tested, and refined continually throughout the life of the system.



Expert System Architecture

Knowledge

- The data is collection of facts. The information is organized as data and facts about the task domain. **Data, information,** and **past experience** combined together are termed as 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 judgement, one's ability of evaluation, and guessing.
- **Knowledge representation**
  - It is the method used to organize and formalize the knowledge in the knowledge base. It is in the form of IF-THEN-ELSE rules.

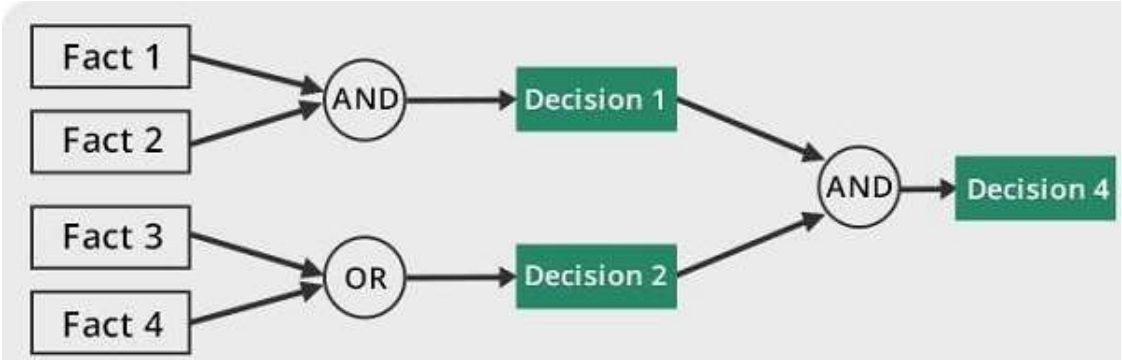
Inference Engine

- Use of efficient procedures and rules by the Inference Engine is essential in deducting a correct, flawless solution.
- 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 the Inference Engine uses the following strategies –
  - Forward Chaining
  - Backward Chaining

Expert System Architecture

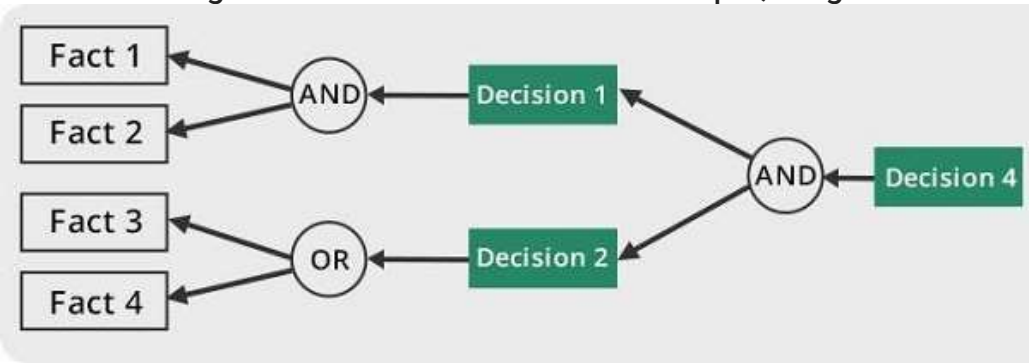
Forward Chaining

- It is a strategy of an expert system to answer the question, **"What can happen next?"**
- Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.
- This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.



Backward Chaining

- With this strategy, an expert system finds out the answer to the question, **"Why this happened?"**
- On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans.



Expert System Architecture

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-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence.
- It explains how the ES has arrived at a particular recommendation. The explanation may appear in the following forms –
  - Natural language displayed on screen.
  - Verbal narrations in natural language.
  - Listing of rule numbers displayed on the screen.

Conventional System vs. Expert System

Conventional System	Expert System
Knowledge and processing are combined in one unit.	Knowledge database and the processing mechanism are two separate components.
The programme does not make errors (Unless error in programming).	The Expert System may make a mistake.
The system is operational only when fully developed.	The expert system is optimized on an ongoing basis and can be launched with a small number of rules.
Step by step execution according to fixed algorithms is required.	Execution is done logically & heuristically.
It needs full information.	It can be functional with sufficient or insufficient information.

Advantages and Disadvantages

Advantages

- Below are the main advantages/benefits of Expert Systems in Artificial Intelligence (AI):
  - It improves the decision quality
  - Cuts the expense of consulting experts for problem-solving
  - It provides fast and efficient solutions to problems in a narrow area of specialization.
  - It can gather scarce expertise and used it efficiently.
  - Offers consistent answer for the repetitive problem
  - Maintains a significant level of information
  - Helps you to get fast and accurate answers
  - A proper explanation of decision making
  - Ability to solve complex and challenging issues
  - Artificial Intelligence Expert Systems can steadily work without getting emotional, tensed or fatigued.

Disadvantages

- Below are the disadvantages/limitations of Expert System in AI:
  - Unable to make a creative response in an extraordinary situation
  - Errors in the knowledge base can lead to wrong decision
  - The maintenance cost of an expert system is too expensive
  - Each problem is different therefore the solution from a human expert can also be different and more creative

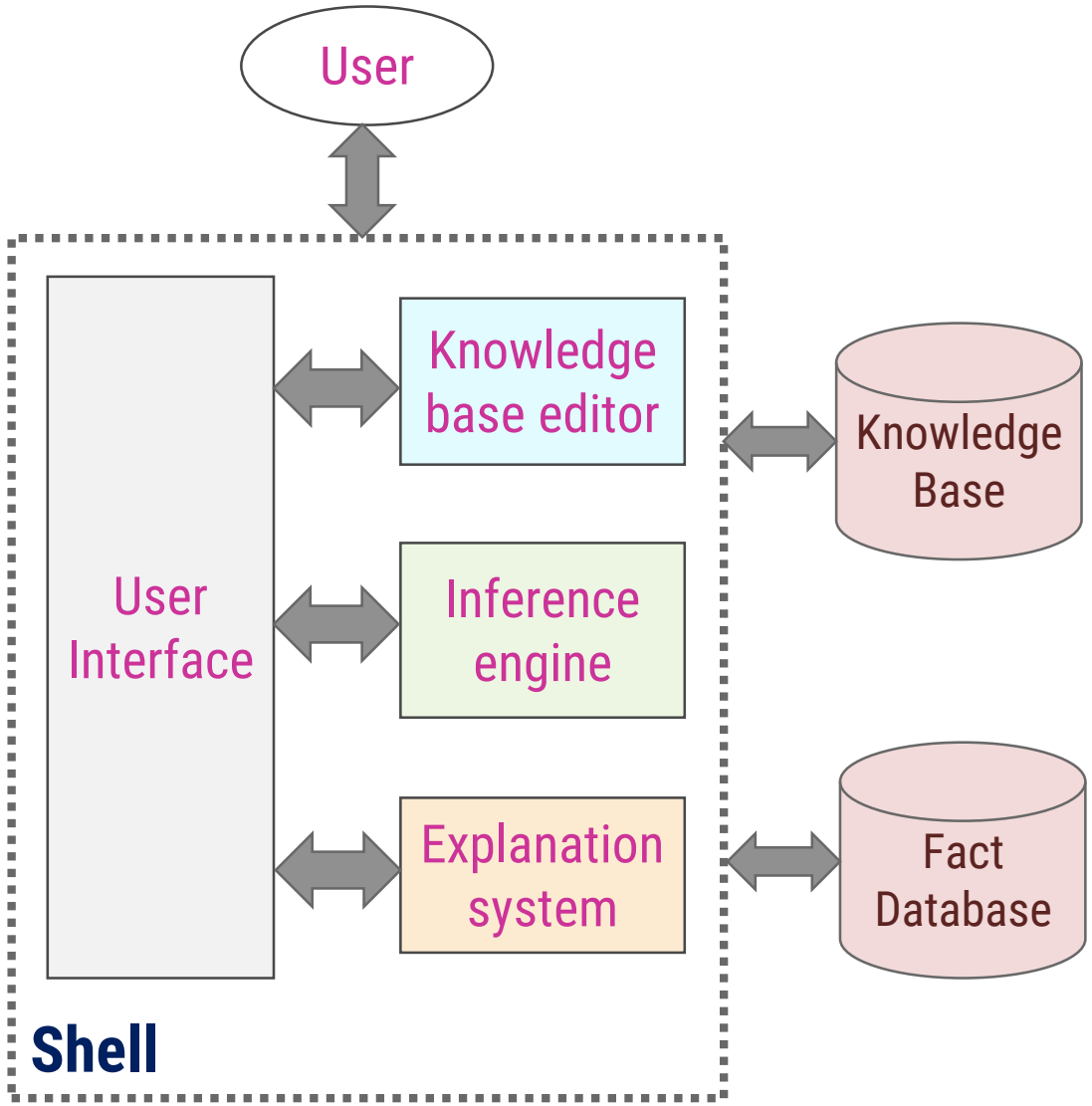
Applications of Expert Systems

- The Expert systems have found their way into most areas of knowledge work. The applications of expert systems technology have widely proliferated to industrial and commercial problems.
  - **Diagnosis and Troubleshooting of Devices and Systems**
    - Medical diagnosis was one of the first knowledge areas to which Expert system technology was applied in 1976. However, the diagnosis of engineering systems quickly surpassed medical diagnosis.
  - **Planning and Scheduling**
    - The Expert system's commercial potential in planning and scheduling has been recognized as very large. Examples are airlines scheduling their flights, personnel, and gates; the manufacturing process planning and job scheduling;
  - **Configuration of Manufactured Objects from sub-assemblies**
    - Configuration problems are synthesized from a given set of elements related by a set of constraints. The Expert systems have been very useful to find solutions. For example, modular home building and manufacturing involving complex engineering design.
  - **Design and Manufacturing**
    - Here the Expert systems assist in the design of physical devices and processes, ranging from high-level conceptual design of abstract entities all the way to factory floor configuration of manufacturing processes.

Applications of Expert Systems

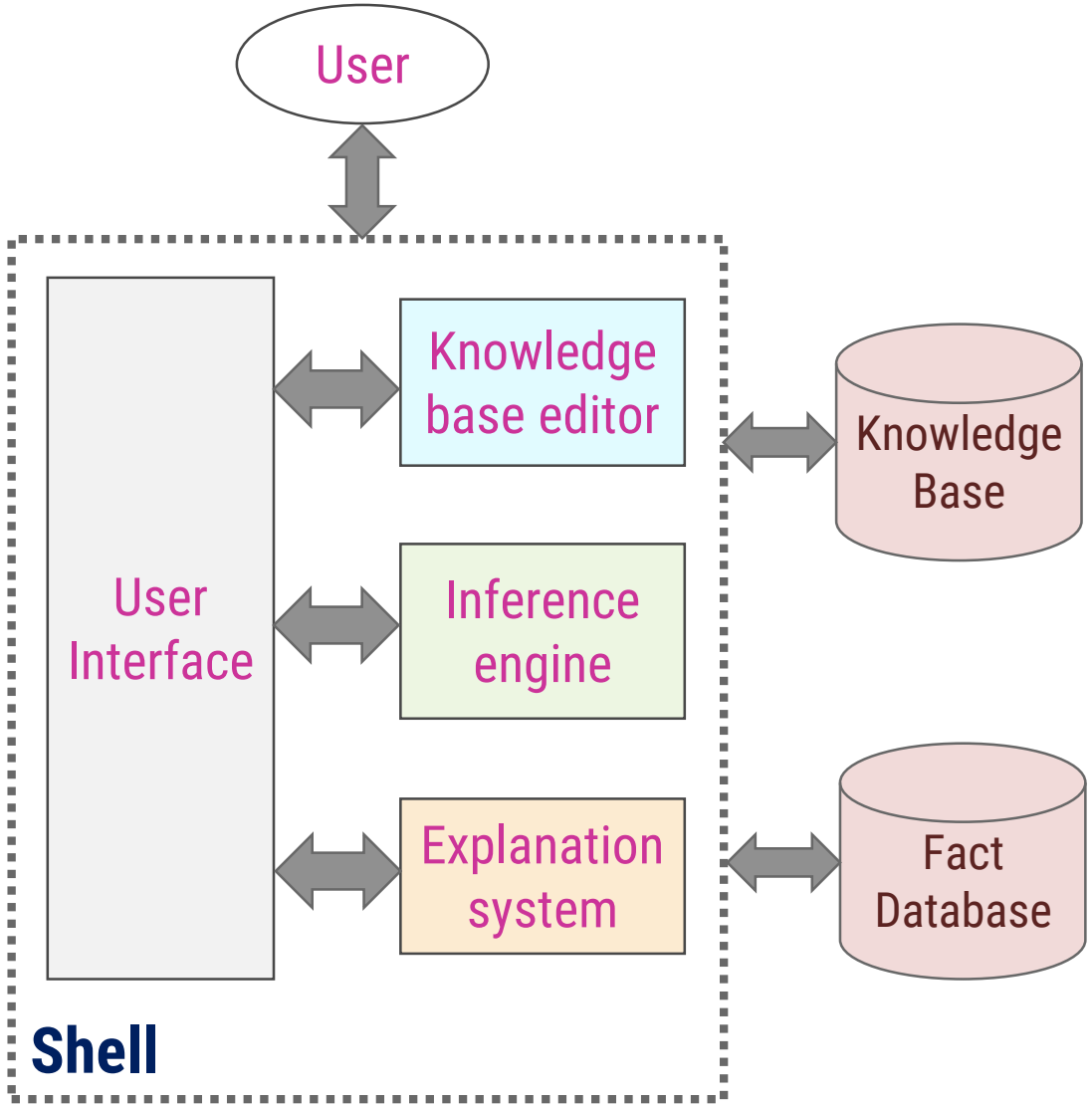
- **Knowledge Publishing**
  - This is relatively new, but also potentially explosive area. Here the primary function of the Expert system is to deliver knowledge that is relevant to the user's problem.
  - The two most widely known Expert systems are :
    1. An advisor on appropriate grammatical usage in a text,
    2. It is a tax advisor on tax strategy, tactics, and individual tax policy.
- **Process Monitoring and Control**
  - Here Expert system does analysis of real-time data from physical devices, looking for anomalies, predicting trends, controlling optimality and failure correction.
  - Examples of real-time systems that actively monitor processes are found in the steel making and oil refining industries.
- **Financial Decision Making**
  - The financial services are the vigorous user of expert system techniques.
  - Advisory programs have been created to assist bankers in determining whether to make loans to businesses and individuals.
  - Insurance companies to assess the risk presented by the customer and to determine a price for the insurance.
  - ES are used in typical applications in the financial markets / foreign exchange trading.

Expert System Shells



- An **Expert system shell** is a software development environment which contains the basic components of expert systems.
- A shell is associated with a prescribed method for building applications by configuring and instantiating these components.
- The expert system that does not contain domain-specific or case-specific information are contained within the expert system shell.
- This shell is a general toolkit that can be used to build a number of different expert systems, depending on which knowledge base is added to the shell.
- The user of the expert system interfaces with it through a user interface, which provides access to the inference engine, the explanation system, and the knowledge-base editor.
- The inference engine is the part of the system that uses the rules and facts to derive conclusions.

Expert System Shells

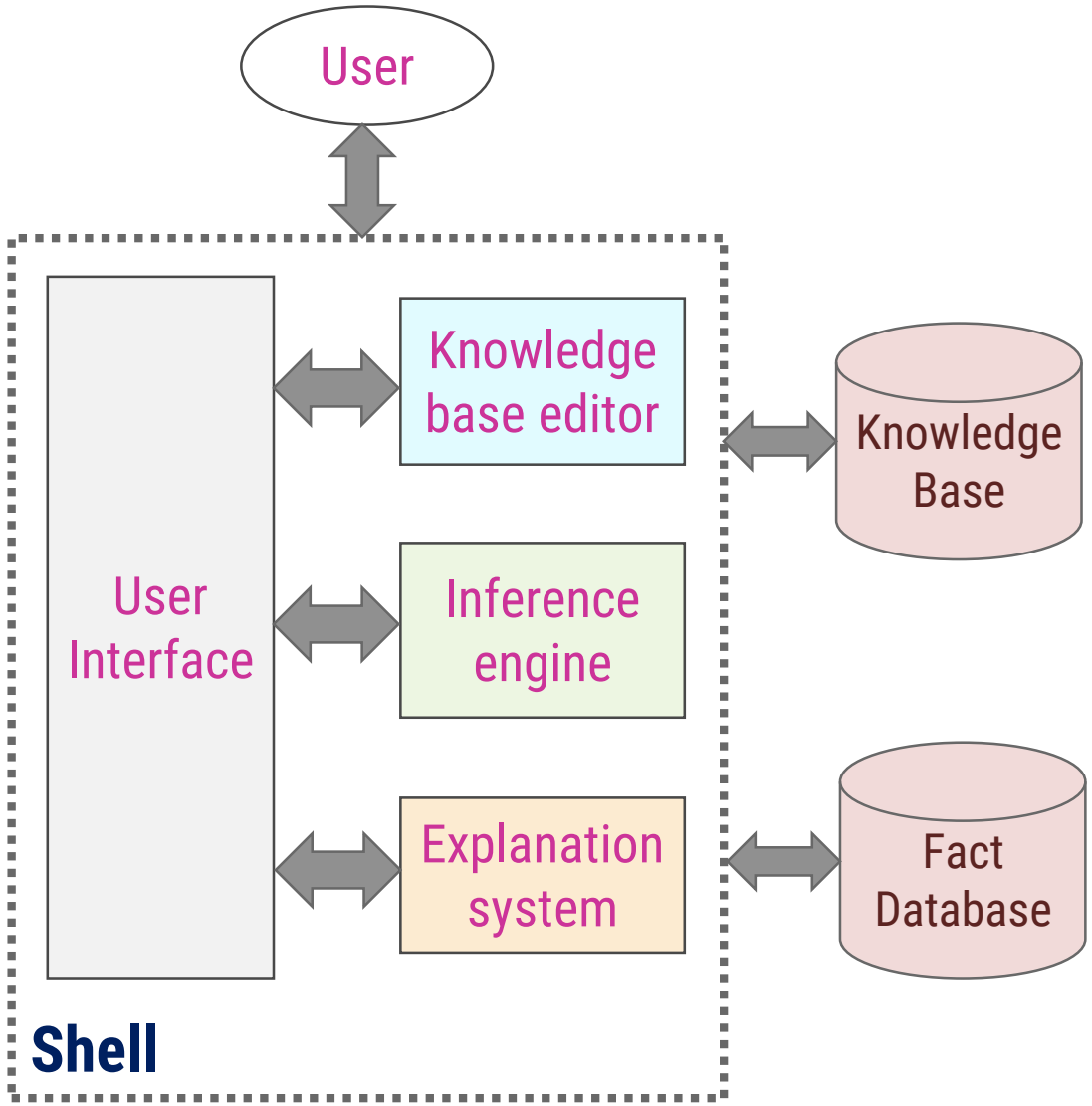


- The inference engine will use forward chaining, backward chaining, or a combination of the two to make inferences from the data that are available to it..
- The knowledge-base editor allows the user to edit the information that is contained in the knowledge base.

- The inference engine will use forward chaining, backward chaining, or a combination of the two to make inferences from the data that are available to it..
- The knowledge-base editor allows the user to edit the information that is contained in the knowledge base.

- The knowledge-base editor is not usually made available to the end user of the system but is used by the knowledge engineer or the expert to provide and update the knowledge that is contained within the system.
- The **knowledge base** contains the specific domain knowledge that is used by an expert to derive conclusions from facts.

Expert System Shells



- In the case of a rule-based expert system, this domain knowledge is expressed in the form of a series of rules.
- The explanation system provides information to the user about how the inference engine arrived at its conclusions.
- This can often be essential, particularly if the advice being given is of a critical nature, such as with a medical diagnosis system.
- If the system has used faulty reasoning to arrive at its conclusions, then the user may be able to see this by examining the data given by the explanation system.
- The **fact database** contains the case-specific data that are to be used in a particular case to derive a conclusion.
- In the case of a medical expert system, this would contain information that had been obtained about the patient's condition.

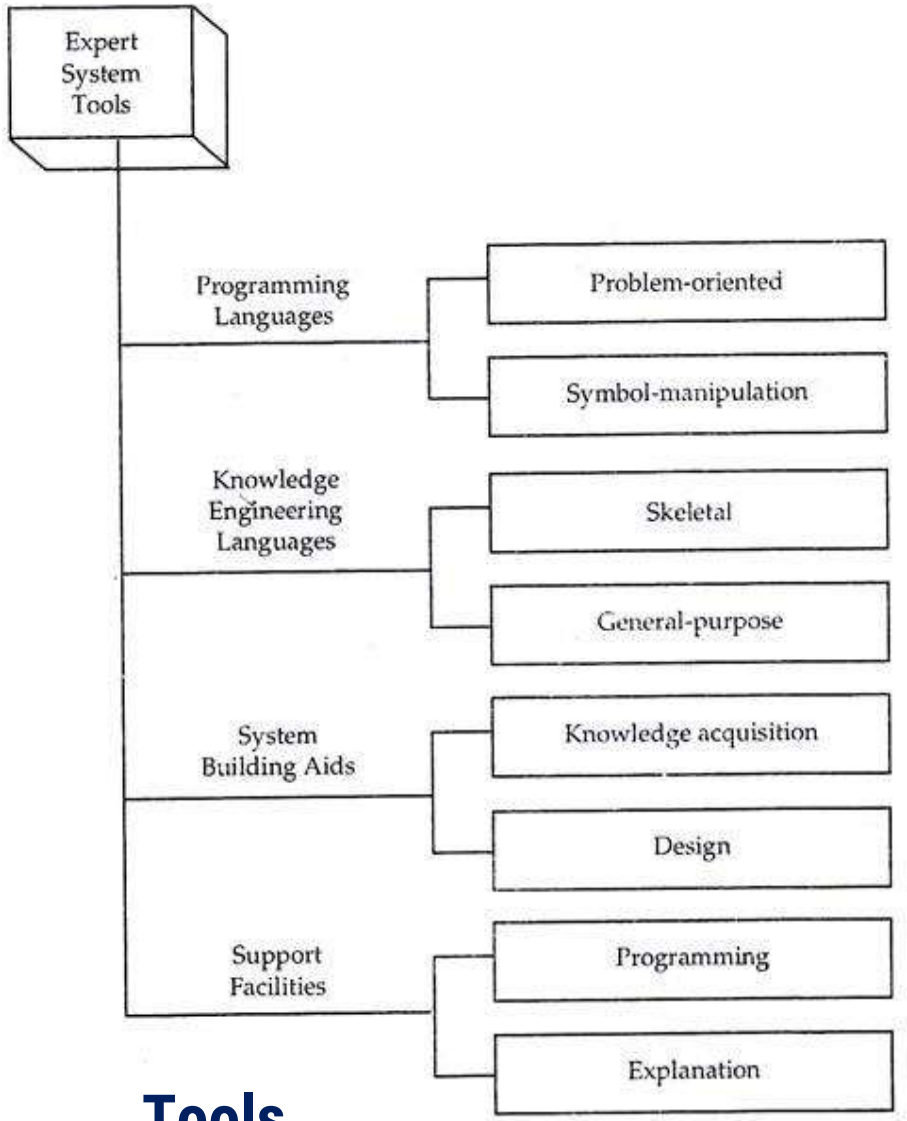
Expert System Tools

1. Programming Languages:

- Most important programming languages used for expert system applications are generally either problem-oriented languages, such as FORTRAN and PASCAL, or symbol-manipulation languages, such as LISP and PROLOG.
- Problem oriented languages are designed for particular classes of problems; e.g., FORTRAN has convenient features for performing algebraic calculations and is most applicable to scientific, mathematical and statistical problem area.

2. Knowledge Engineering Languages:

- A knowledge engineering language is a sophisticated tool for developing expert systems, consisting of an expert system building language integrated into an extensive support environment.
- A programming language is an artificial language development to acquire knowledge, accept/reject knowledge to control and direct the operation of a computer.



Tools

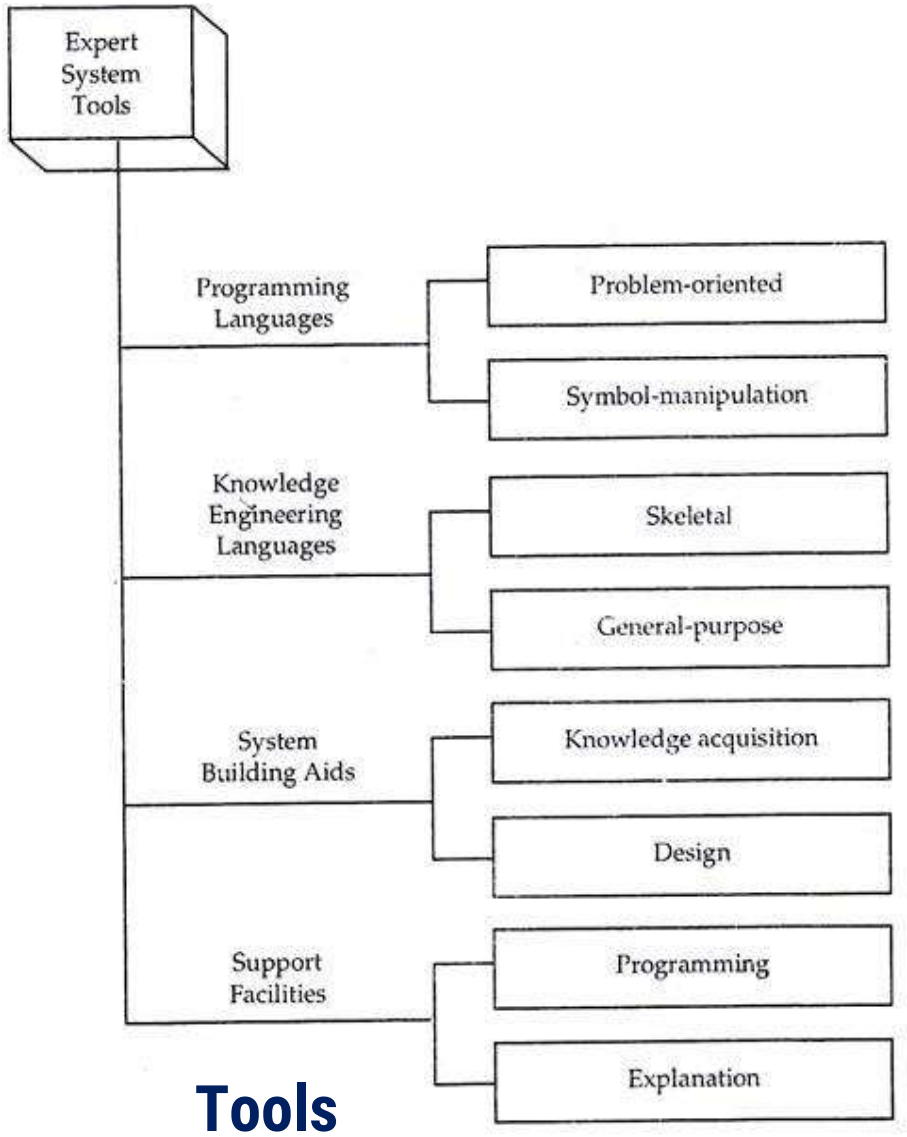
Expert System Tools

3. System-Building Aids:

- The system-building aids consist of programs which help acquire and represent the domain expert's knowledge and programs which help design the expert system under construction.
- These programs address very difficult tasks; many are research tools just beginning to evolve into practical and useful aids, although a few are offered as full-blown commercial systems

4. Tool Support Environment Facilities:

- These are simply software packages which come with each tool to make it more user friendly and more efficient.
- **Environments which can be further categorised into:**
  - (i) Those required during the development t of the expert system programs, such as debugging aids and knowledge base editors.
  - (ii) Those required to enhance the capabilities of the developed programs, such as input/output facilities and explanation mechanism. Although few expert system tools support all these tools, they all support some of them. These facilities are usually available as part of K.E. language and are designed to work specially with that language.



Tools

Introduction to Uncertainty Measure – Probability Theory

Introduction to Uncertainty

- **Uncertainty**
  - Most intelligent systems have some degree of uncertainty associated with them.
  - Uncertainty may occur in KBS because of the problems with the data.
    - Data might be missing or unavailable.
    - Data might be present but unreliable or ambiguous due to measurement errors, multiple conflicting measurements etc.
    - The representation of the data may be imprecise or inconsistent.
    - Data may just be expert's best guess.
    - Data may be based on defaults and the defaults may have exceptions.
    - Given numerous sources of errors, the most KBS requires the incorporation of some form of uncertainty management.
    - For any form of uncertainty scheme, we must be concerned with three issues.
    - How to represent uncertain data?
    - How to combine two or more pieces of uncertain data?
    - How to draw inference using uncertain data?
    - Probability is the oldest theory with strong mathematical basis.
    - Other methods for handling uncertainty are Bayesian belief network, Certainty factor theory etc.

Introduction to Probability Theory

- Probability is a way of turning opinion or expectation into numbers.
- It lies between 0 to 1 that reflects the likelihood of an event.
- The chance that a particular event will occur = the number of ways the event can occur divided by the total number of all possible events.

- **Example:** The probability of throwing two successive heads with a fair coin is 0.25

  Total of four possible outcomes are :

  HH, HT, TH & TT

  Since there is only one way of getting HH,

  probability = ¼ = 0.25

- **Axioms of Probability**
  - Let S be a sample space, A and B are events.
  - P(A) >= 0
  - P(S) = 1
  - P(A') = 1 - P(A)
  - P(AUB) = P(A) + P(B) – P(A ^ B)
  - If events A and B are mutually exclusive, then

    P(A U B) = P(A) + P(B),
  - In general, for mutually exclusive events A1,…,An in S
    - P(A1 U A2 U… U An) = P(A1) + P(A2) + …+ P(An)

Introduction to Probability Theory

- Probability quantifies the uncertainty of the outcomes of a random variable / event.
- Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network.
- Marginal probability is the probability of an event, irrespective of other random variables.
  - Marginal Probability: The probability of an event irrespective of the outcomes of other random variables, e.g. P(A).
- The joint probability is the probability of two (or more) simultaneous events, often described in terms of events A and B from two dependent random variables, e.g. X and Y. The joint probability is often summarized as just the outcomes, e.g. A and B.
  - Joint Probability: Probability of two (or more) simultaneous events, e.g. P(A and B) or P(A, B).
- The conditional probability is the probability of one event given the occurrence of another event, often described in terms of events A and B from two dependent random variables e.g. X and Y.
  - Conditional Probability: Probability of one (or more) event given the occurrence of another event,
    e.g. P(A given B) or P(A | B).

Introduction to Probability Theory

- The joint probability can be calculated using the conditional probability :

$$P(A, B) = P(A \mid B) * P(B)$$

- The joint probability is symmetrical : $P(A, B) = P(B, A)$

- The conditional probability can be calculated using the joint probability:

$$P(A \mid B) = P(A, B) / P(B)$$

- The conditional probability is not symmetrical : $P(A \mid B) != P(B \mid A)$

Bayes' Theorem

- In statistics and probability theory, the Bayes' theorem (also known as the Bayes' rule) is a mathematical formula used to determine the conditional probability of events.
- Essentially, the Bayes' theorem describes the probability of an event based on prior knowledge of the conditions that might be relevant to the event.
- The Bayes' theorem is expressed in the following formula:

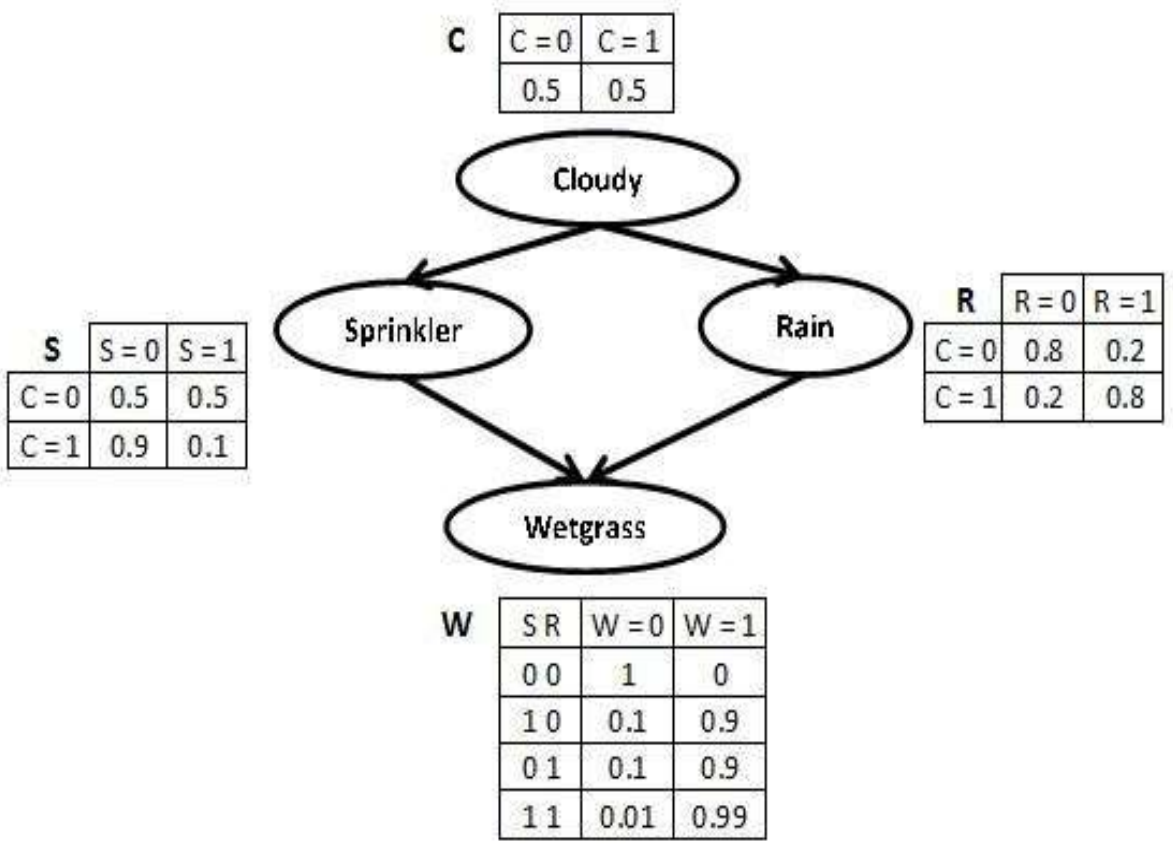
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

  - Where:
  - $P(A|B)$ – the probability of event A occurring, given event B has occurred
  - $P(B|A)$ – the probability of event B occurring, given event A has occurred
  - $P(A)$ – the probability of event A
  - $P(B)$ – the probability of event B
  - Note that events A and B are independent events

Bayesian network

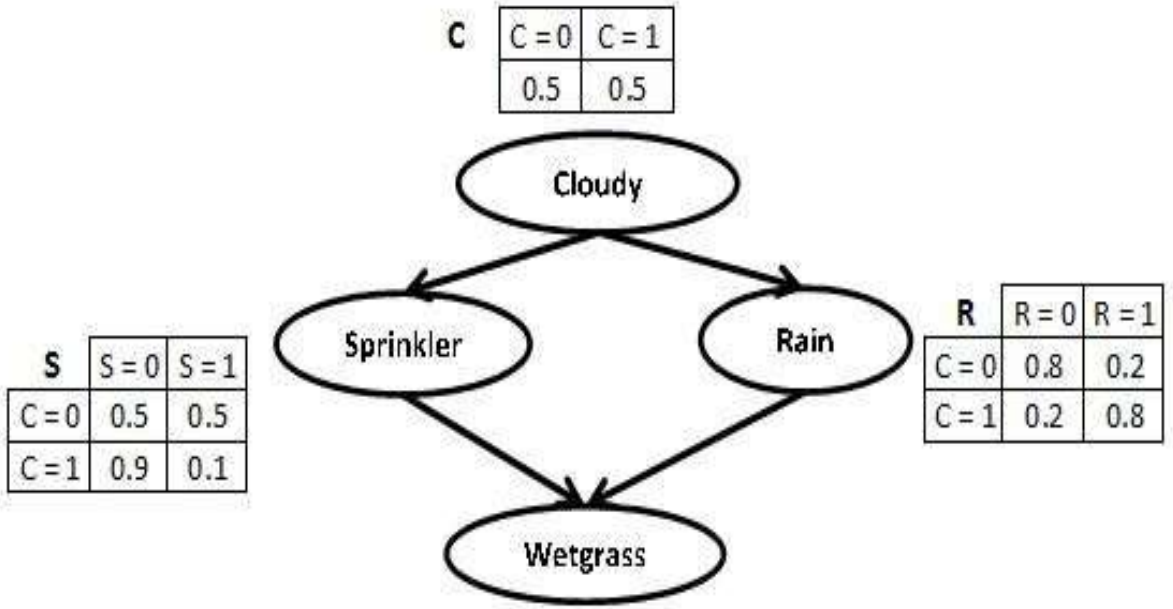
- A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph.
- In a directed acyclic graph, each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable.
- It is also called a Bayes network, belief network, decision network, or Bayesian model.
- Bayesian Network represents the dependency among events and assigning probabilities to them.
- Thus ascertaining how probable or what is the change of occurrence of one event given the other.
- It can be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty.

Bayesian network - Example



- Whether the grass is wet, W, depends on whether the sprinkler has been used, S, or whether it has rained, R.
- Whether the sprinkler is used depends on whether it is cloudy, similarly for whether it has rained.
- The probability of the grass being wet is conditionally independent of it being cloudy, given information about the sprinklers and whether it has rained.
- This joint probability may be expressed as
  P(C, S, R, W) = P(C)P(S|C)P(R|C,S)P(W|C,S, R)

Bayesian network - Example



C	C = 0	C = 1
	0.5	0.5

S	S = 0	S = 1
C = 0	0.5	0.5
C = 1	0.9	0.1

R	R = 0	R = 1
C = 0	0.8	0.2
C = 1	0.2	0.8

W	S R	W = 0	W = 1
	0 0	1	0
	1 0	0.1	0.9
	0 1	0.1	0.9
	1 1	0.01	0.99

- It is cloudy, what's the probability that the grass is wet?
- So, we want to compute P(W = T|C = T).
- By the chain rule of probability, the joint probability of all the nodes in the graph above is,

$$P(C,S,R,W) = P(C)*P(S|C)*P(R|C,S) * P(W|C,S,R)$$
$$P(C,S,R,W) = 0.99 \times 0.1 \times 0.8 + 0.90 \times 0.1 \times 0.2 + 0.90 \times 0.9 \times 0.8 + 0.00 \times 0.9 \times 0.2$$
$$P(C,S,R,W) = 0.7452$$

Certainty Factor

- A Certainty Factor (CF) is a numerical estimates of the belief or disbelief on a conclusion in the presence of set of evidence. Different methods for adopting Certainty Factor have been adopted.
  1. Use a scale from 0 to 1, where 0 indicates certainly false (total disbelief), 1 indicates definitely true (total belief). Other values between 0 to 1 represents varying degrees of beliefs and disbeliefs.
  2. use a scale from −1 to +1 where -1 indicates certainly false, +1 indicates definitely true, and intermediate values represent varying degrees of certainty, with 0 meaning unknown.
- The weights express the perceived certainty of a fact being true.
- The use of certainty factors is similar to probabilistic reasoning but is less formally related to probability theory.
- There are many schemes for treating uncertainty in rule based systems. The most common are
  - Adding certainty factors.
  - Adoptions of Dempster-Shafer belief functions.
  - Inclusion of fuzzy logic.

Certainty Factor in a Rule based System

- In a rule based system, a rule is an expression of the form "if A then B" where A is an assertion and B can be either an action or another assertion.
- A problem with rule-based systems is that often the connections reflected by the rules are not absolutely certain or deterministic, and the gathered information is often subject to uncertainty.
- In such cases, a certainty measure is added to the premises as well as the conclusions in the rules of the system.
- A rule then provides a function that describes : how much a change in the certainty of the premise will change the certainty of the conclusion.
- In its simplest form, this looks like :

If A (with certainty x) then B (with certainty f(x))

Certainty Factor in a Rule based System

- Each rule has a certainty attached to it. Once the identities of the virus/bacteria are found, it then attempts to select a therapy by which the disease can be treated.
- A certainty factor (CF [h, e]) is defined in terms of two components:

1. MB[h, e] - a measure (between 0 and 1) of belief in hypothesis "h" given the evidence "e".
   - MB measures the extent to which the evidence supports the hypothesis.
   - It is zero if the evidence fails to support the hypothesis.
2. MD[h, e] - a measure (between 0 and 1) of disbelief in hypothesis "h" given the evidence "e".
   - MD measures the extent to which the evidence supports the negation of the hypothesis. It is zero if the evidence support the hypothesis.

$$CF[h, e] = MB[h, e] - MD[h, e]$$

Dempster – Shafer Theory

- In Dempster-Shafer Theory we consider sets of propositions and assign an interval to each of them in which the degree of belief must lie.

[Belief, Plausibility]

- Belief (denoted as Bel) measures the strength of the evidence in favor of a set of propositions.
- It ranges from 0 (no evidence) to 1 (definite certainty)
- Plausibility (Pl) is Pl(s) = 1- Bel(¬s)
- It also ranges from 0 to 1 and measures the extent to which evidence in favor of ¬s leaves room for belief in s.
- In short, if we have certain evidence in favor of (¬s), then Bel(not(s)) will be 1 and Pl(s) will be 0.
- This tells us that the only possible value for Bel(s) is also 0.
- The interval, also tells about the amount of information that we have.
- If we have no evidence we say that the hypothesis is in the range of [0, 1].

Dempster – Shafer Theory

- Let's take an example where we have some mutually exclusive hypothesis.
- Let the set {Allergy, Flu, Cold, Pneumonia} be denoted by θ and we want to attach some measure of belief to elements of θ.
- The key function we use here is a Probability Density Function, denoted by m.
- The function m, is not only defined for elements of θ but also all subsets of it.
- We must assign m so that the sum of all the m values assigned to subsets of θ is 1.
- At the beginning we have m as under $\theta = (1.0)$
- If we get an evidence of 0.6 magnitude that the correct diagnosis is in the set {Flu, Cold, Pneu} then,

{Flu, Cold, Pneu} = (0.6)
θ = (0.4)

Dempster – Shafer Theory

- Now to move further, let's consider we have two belief function m1 and m2.
- Let X be the set of subsets of θ to which m1 assigns a nonzero value and let Y be the corresponding set for m2.
- We define m3, as a combination of the m1 and m2 to be,

$$\mathbf{m}_3(\mathbf{Z}) = \frac{\sum_{\mathbf{X}\cap\mathbf{Y}=\mathbf{Z}} \mathbf{m}_1(\mathbf{X}) \cdot \mathbf{m}_2(\mathbf{Y})}{1 - \sum_{\mathbf{X}\cap\mathbf{Y}=\emptyset} \mathbf{m}_1(\mathbf{X}) \cdot \mathbf{m}_2(\mathbf{Y})}$$

Dempster – Shafer Theory

- For example, suppose m1 corresponds to our belief after observing fever:

$$m1 = \{ F, C, P\} = 0.6 \text{ and } \theta = (0.4)$$

- suppose m2 corresponds to our belief after observing runny nose:

$$m2 = \{ A, F, C\} = 0.8 \text{ and } \theta = (0.2)$$

- Then we can compute their combination m3 using the following table.

		{A,F,C }	(0.8)	Θ	(0.2)
{F,C,P }	(0.6)	{F,C }	(0.48)	{F,C,P}	(0.12)
Θ	(0.4)	{A,F,C }	(0.32)	Θ	(0.08)

- So we produce a new, combined m3 as,

{Flu, Cold}	(0.48)	{All, Flu, Cold}	(0.32)	{Flu, Cold, Pneu}	(0.12)
				Θ	(0.08)

Important Questions

Short Questions:

1. Define the various phases in building the expert system .
2. Describe the characteristics of expert system.
3. List out the advantages of expert system.
4. What is the forward chaining in expert system?
5. What are the components in expert system?
6. What is the backward chaining in expert system?
7. List out the disadvantages of expert system.
8. What is role of inference engine in expert system?
9. What is a knowledge base in expert system?
10. Define a user interface in expert system?
11. What is Dempster - Shafer Theory?
12. Define a Certainty Factor.
13. What is a Probability Theory?
14. Describe Bayesian Belief Networks.
15. What is knowledge aquistion?

Important Questions

Long Questions:

1. Explain the architecture of expert system .
2. Explain the Phases in Building Expert System.
3. Differentiate between traditional system and expert system.
4. List and explain the applications of Expert System with examples.
5. Discuss the characteristics of expert system. Explain the truth Maintenance system.
6. List out the List of Shells and Tools?
7. Explain the Probability Theory with examples.
8. Explain the Certainty Factory Theory with examples.
9. Give the brief note on Dempster - Shafer Theory .
10. Discuss in detail about Bayesian Belief Networks.

Thank You!