1. Course Objectives

1. The student should be made to study the concepts of Artificial Intelligence.
2. The student should be made to learn the methods of solving problems using Artificial Intelligence.
3. The student should be made to introduce the concepts of Expert Systems.
4. To understand the applications of AI, namely game playing, theorem proving, and machine learning.
5. To learn different knowledge representation techniques

2. Pre-requisite

1. Knowledge in Computer Programming.
2. A course on "Mathematical Foundations of Computer Science".
3. Background in linear algebra, data structures and algorithms, and probability.

3. Syllabus

L	T	P	C
3	0	0	3

(23A31301T) ARTIFICIAL INTELLIGENCE

UNIT - I

Introduction: AI problems, foundation of AI and history of AI intelligent agents: Agents and Environments, the concept of rationality, the nature of environments, structure of agents, problem solving agents, problem formulation.

UNIT - II

Searching- Searching for solutions, uniformed search strategies – Breadth first search, depth first Search. Search with partial information (Heuristic search) Hill climbing, A*, AO* Algorithms, Problem reduction, Game Playing-Adversial search, Games, mini-max algorithm, optimal decisions in multiplayer games, Problem in Game playing, Alpha-Beta pruning, Evaluation functions.

UNIT - III

Representation of Knowledge: Knowledge representation issues, predicate logic- logic programming, semantic nets- frames and inheritance, constraint propagation, representing knowledge using rules, rules-based deduction systems. Reasoning under uncertainty, review of probability, Bayes' probabilistic interferences and dempster Shafer theory.

UNIT - IV

Logic concepts: First order logic. Inference in first order logic, propositional vs. first order inference, unification & lifts forward chaining, Backward chaining, Resolution, learning from observation Inductive learning, Decision trees, Explanation based learning, Statistical Learning methods, Reinforcement Learning.

UNIT - V

Expert Systems: Architecture of expert systems, Roles of expert systems – Knowledge Acquisition Meta knowledge Heuristics. Typical expert systems – MYCIN, DART, XCON: Expert systems shells.

Textbooks:

1. S. Russel and P. Norvig, "Artificial Intelligence – A Modern Approach", Second Edition, Pearson Education.
2. Kevin Night and Elaine Rich, Nair B., "Artificial Intelligence (SIE)", Mc Graw Hill

Reference Books:

1. David Poole, Alan Mackworth, Randy Goebel," Computational Intelligence: a logical approach", Oxford University Press.
2. G. Luger, "Artificial Intelligence: Structures and Strategies for complex problem solving", Fourth Edition, Pearson Education.
3. J. Nilsson, "Artificial Intelligence: A new Synthesis", Elsevier Publishers.
4. Artificial Intelligence, Saroj Kaushik, CENGAGE Learning.

Online Learning Resources:

1. https://ai.google/
2. https://swayam.gov.in/nd1_noc19_me71/preview

4. Course Outcomes

Course Code	Course Outcome Statement	Cognitive /affective Level of the course outcome	Course Outcome
23A31301T	Explain the foundations of AI and various Intelligent Agents	Apply L3	CO1
23A31301T	Apply search strategies in problem solving and game playing	Apply L3	CO2
23A31301T	Apply problem solving strategies with knowledge representation techniques for applications of AI	Apply L4	CO3
23A31301T	Explain logical agents and first-order logic	Apply L3	CO4
23A31301T	Describe the basics of Learning and Expert Systems	Apply L4	CO5

5. LECTURE PLAN
UNIT I

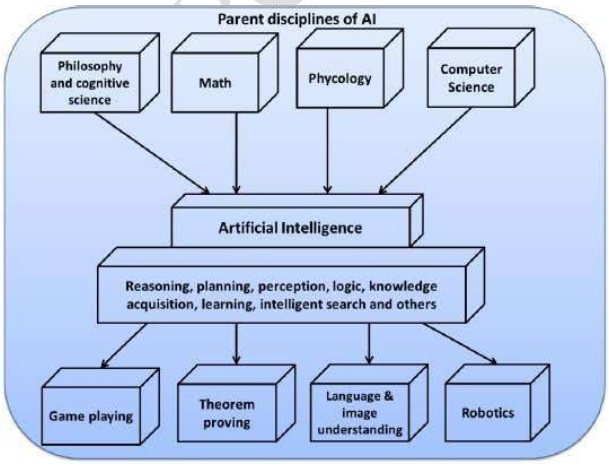
S.No.	Topic	No. of Periods	Proposed lecture	Actual lecture	Pertaining CO(s)	Taxonomy Level	Mode of Delivery
			Period	Period			
1	Introduction- AI Problems	1			CO1	L2	MD1, MD5
2	Foundation of AI, History of AI	1			CO1	L1	MD1, MD5
3	Intelligent Agents and Environments	1			CO1	L2	MD1, MD5
4	The Concept of Rationality	1			CO1	L2	MD1, MD5
5	Nature of environments, Structure of Agents	1			CO1	L2	MD1, MD5
6	Problem solving agents, Problem formulation	1			CO1	L2	MD1, MD5
7	Problem solving agents-Example Problems	1			CO1	L3	MD1, MD5

UNIT - I

Introduction: AI problems, foundation of AI and history of AI intelligent agents: Agents and Environments, the concept of rationality, the nature of environments, structure of agents, problem solving agents, problem formulation.

1. Introduction:

Artificial Intelligence (AI) refers to the ***"development of computer systems for performing tasks that require human intelligence".*** The *goal of AI* is *to create machines* to carry out diverse *tasks* for *identifying patterns* and *making decisions* based on the collected information. This can be achieved through *AI techniques* (machine learning and neural networks to natural language processing and robotics) which enable *machines to learn, reason, and perform tasks previously reserved for human intelligence.* The field of AI is unique, sharing border with *disciplines* like Mathematics, Computer Science, Philosophy, Psychology, Biology, Cognitive Science and many others.



- Although there is no clear definition of AI or even Intelligence, it can be described as an attempt *"to build machines that like humans can think and act, able to learn and use knowledge to solve problems"* on their own.
- Artificial Intelligence (AI), a term coined by emeritus Stanford Professor John McCarthy in 1956, was defined by him as ***"the science and engineering of making intelligent machines"***

Definition: Artificial intelligence, commonly known as AI, *is a branch of computer science that aims to develop systems capable of performing tasks that normally require human intelligence.*

1.1 Organization of AI:

To design intelligent systems, it is important to categorize them into four categories (Luger and Stubberfield 1993), (Russell and Norvig, 2003) by **definitions**:

Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

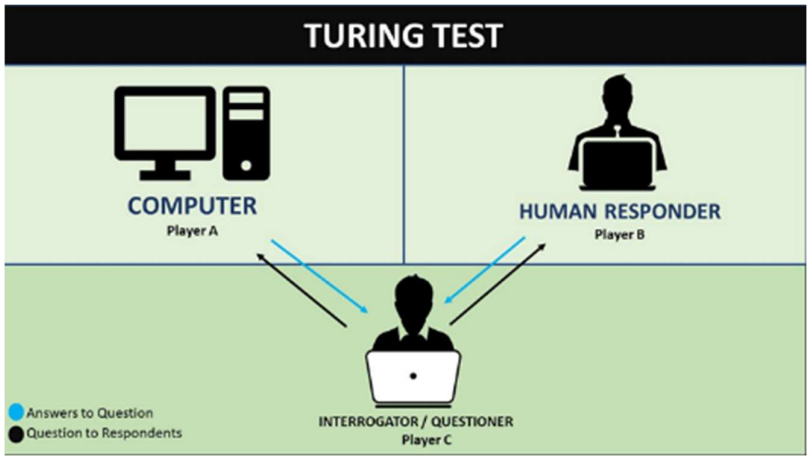
Figure 1.1 Some definitions of artificial intelligence, organized into four categories.

The four approaches in more detail are as follows:

1. **Systems that think like humans (The cognitive modelling approach):** The idea behind this approach is to *determine whether the computer thinks like a human.* It is an area of **psychology/*cognitive science*.** One must know functioning of brain and its mechanism for processing information. **Neural network** is a computing model for processing information similar to brain.
2. **Systems that act like humans (The Turing Test approach)**: This approach was designed by ***Alan Turing in 1950.***

- The ideology behind this approach is that *"a computer passes the test if a human interrogator, after asking some written questions, cannot identify whether the written responses come from a human or from a computer".*

- This test is used to evaluate a computer acting like humans.



The features required for a machine to pass the Turing test:

- *Natural language processing* to enable it to communicate successfully in English (or some other human language) with the interrogator.
- *Knowledge representation* to store information provided before or during the interrogation.
- *Automated reasoning* to use the stored information to answer questions and to draw new conclusions.
- *Machine learning* to adapt to new circumstances and to detect and infer patterns.

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

- *Computer Vision* to perceive objects in the environment i.e. to recognize the interrogator's actions and other objects during a test.
- *Robotics* to fulfill mechanical tasks.

3. **Systems that think rationally (The laws of thought approach):** 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: *"John is a human, and all humans are mortal"*
   then one can conclude logically that, *"John is mortal"*

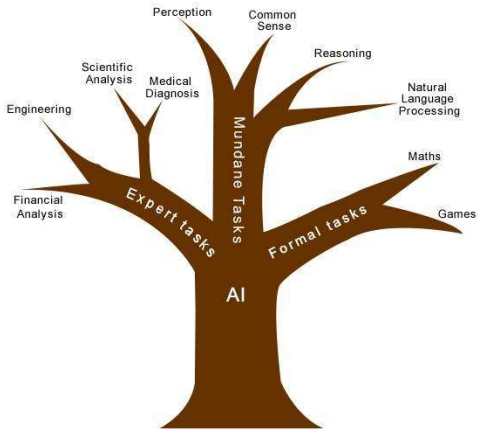
4. **Systems that act rationally (The rational agent approach):** It tries to define AI, with the concept of so-called rational agents. An ***agent*** is just something that *acts* and a rational agent is one that acts to achieve the best outcome when there is uncertainty. ***Rational behavior means doing right thing.***

	Like humans	Well
Think	GPS	Rational agents
Act	Eliza	Heuristic systems

1.2 AI Problems:

Problems in Artificial Intelligence (AI) come in different forms, each with its own set of challenges and potential for innovation. From image recognition to natural language processing, AI problems can be classified into three types of tasks:

1. **Mundane Tasks:** These tasks are done routinely by people and some other animals. (For e.g. Perception, robotics, natural language, Vision, Speech)
   - *Recognizing* people, objects.
   - Communicating (through *natural language*).
   - *Navigating* around obstacles on the streets.



2. **Formal Tasks:** Tasks which require logic and constraints to function. (For e.g. Mathematics-logic, calculus, algebra Theorem proving, Games chess, checkers, verification,).

3. **Expert tasks:** These tasks cannot be done by all people and can only be performed by skilled specialists. (For e.g. financial analysis, medical diagnostics, engineering, scientific analysis, consulting)

- Much of the early work in the field of AI was focused on *formal tasks*, such as *Game Playing and Theorem proving*. Initially computers could perform well at those tasks, but no computer is fast enough to overcome the combinational explosion generated by most problems.
- Another early foray into AI focused on *common-sense reasoning*. Again, no attempt was made to create a program with a large amount of knowledge.
- This is the reason why AI work is more prospering in the *Expert Tasks* domain now, as the expert task domain needs *expert knowledge* without common sense, which can be easier to represent and handle.

1.2.1 Characteristics of AI problems:

The problems in artificial intelligence exhibit certain traits that contribute to their complexity:

Traits	Issues
Uncertainty	AI problems often involve incomplete or uncertain information, making it difficult to determine the best course of action.
Complexity	AI problems can be highly complex, with many variables and interdependencies that need to be considered.
Ambiguity	Many AI problems have multiple possible interpretations, leading to ambiguity in the decision-making process.
Nonlinearity	The relationships between variables in AI problems are often nonlinear, requiring sophisticated algorithms to model and solve them.
Incompleteness	The available information for AI problems is usually incomplete, which can contribute to the difficulty in finding optimal solutions.

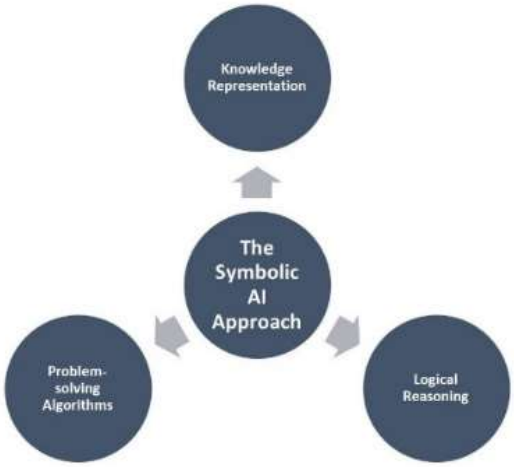
Noise	Noise in the data can introduce errors and make the problem-solving process more challenging.

1.2.2 Approaches & Techniques to AI problems:

Artificial Intelligence problems span a very broad spectrum. The various approaches and techniques provide **solutions** to AI problems.

❖ **Approaches**

1. **Symbolic AI** interprets human intelligence through *symbol manipulation, rules* and *logical reasoning*. Basic concepts are broken down into symbols and the AI processes them into something understandable to solve the problem.



2. **Sub-symbolic approach** is how an AI *recognizes things or patterns* without any specific knowledge. sub symbolic AI focuses on the use of *numerical representations and machine learning algorithms* to extract patterns from data. This approach, also known as "connectionist" or "neural network".

3. **Statistical approach** is how AI *solves specific problems using mathematics tools* and *statistical models* such as information theory, decision theory, etc., to develop AI algorithms. This approach has resulted in greater accuracy and reproducibility in ***data mining***.

❖ **AI Technique:**

AI techniques provide solutions to a variety of these problems. Artificial Intelligence techniques refer to a ***"set of methods and algorithms used to develop intelligent systems that can perform tasks requiring human-like intelligence."*** These techniques encompass the various approaches like:

- *Machine Learning*: This approach involves the building of algorithms to learn patterns in data and make predictions based on it.
- *Deep Learning:* Deep learning uses ***neural networks*** to learn features directly from data, making it suitable for domains with complex and unstructured data. The main applications of deep learning can be divided into computer vision, natural language processing (NLP).
- *Natural Language Processing*: Natural Language Processing involves programming computers to process human languages to facilitate interactions between humans and computers.
- *Computer Vision*: Computer Vision equips machines with the ability to interpret visual information from the world.
- *Data Mining:* Data mining is the process of extracting knowledge or insights from large amounts of data using various **statistical** and computational techniques.
- *Automation & Robotics:* Robotic process automation is designed to carry out high-volume, repetitive jobs while being capable of adapting to changing conditions.

1.3 The Goals of AI:

"Artificial Intelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit characteristics we associate with intelligence in human behaviour – understanding language, learning, reasoning, solving problems, and so on." (**Barr & Feigenbaum, 1981**)

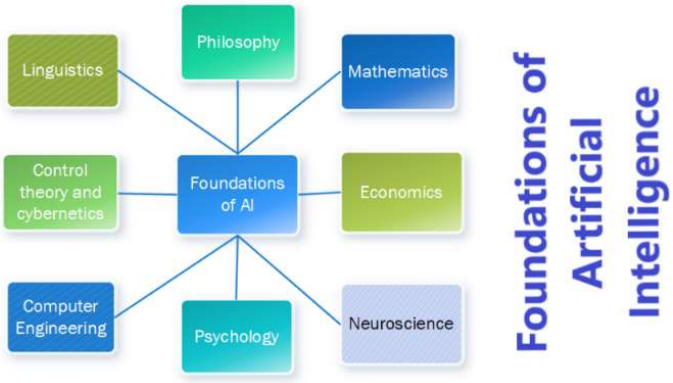
Following are the two main goals of Artificial Intelligence:

- **Scientific Goals:** "*To implement human intelligence in machines*"- Creating systems that understand, think, learn on own, and behave like humans.
- **Engineering Goals (***To create Expert Systems***):** To *solve* real world *problems* using AI techniques such as knowledge representation, learning, rule systems, search, and so on.

Traditionally, computer scientists and engineers have been more interested in the engineering goal, while psychologists, philosophers and cognitive scientists have been more interested in the scientific goal. **"General intelligence is among the field's long-term goals."**

2. The Foundation of AI:

To accomplish the traditional goals for a machine or software, Artificial Intelligence requires the following disciplines:



1. **Philosophy** (the study of the fundamental nature of knowledge):
   - *Where does knowledge come from?*
   - *How does knowledge lead to action?*
   - Aristotle (384–322 B.C.), was the first to formulate a precise set of laws governing the rational part of the mind. He developed an informal system of syllogisms for proper reasoning to generate valid conclusions.
     E.g. *all dogs are animals; all animals have four legs.*
     therefore, all dogs have four legs

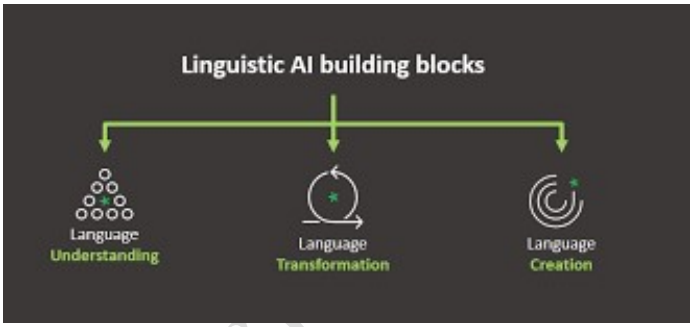
2. **Mathematics:** Formal science requires a level of mathematical formalization in three fundamental areas: *logic, computation, and probability.*
   - Mathematical development really began with the work of George Boole (1815-1864), who worked out the details of propositional or *Boolean logic.*
   - The Italian Gerolamo Cardanao (1501-1576) first framed the idea of *probability,*
   - Thomas Bayes (1702-1761) proposed a rule for updating probabilities to deal with uncertain information.
   - *Logic and Computation:* The first nontrivial algorithm is thought to be Euclid's algorithm for computing greatest common divisors (GCD).
3. **Economics:** Economics is the study of "*how people make choices that lead to preferred outcomes or utility.*"

- The science of economics got its start in 1776, when Scottish philosopher Adam Smith treats it as a science.
- *Decision theory*, which combines probability theory with utility theory, provides a formal and complete framework for decisions (economic or otherwise) made under

4. **Neuroscience:** How does the brain work?
   - Early studies (1824) relied on injured and abnormal people to understand what parts of the brain work. More recent studies use accurate sensors to correlate brain activity to human thought. *Moore's law states* that computers will have as many gates as humans have neurons in 2020.
5. **Psychology and cognitive science:** *How do humans and animals think and act?*
   - Behaviorism movement, led by John Watson (1878-1958) discovered a lot about rats and pigeons but had less success at *understanding humans.*
   - *Cognitive psychology* views the brain as an information processing device. A common view among psychologists is that cognitive theory should be like a *computer program.* (Anderson 1980)
6. **Computer engineering:** How can we build an efficient computer? For artificial intelligence to succeed, we need two things: *intelligence and an artifact.* The *computer* has been the *artifact*(object) of choice.

   The first operational computer was the electromechanical Heath Robinson, built in 1940 by Alan Turing's team for a single purpose: deciphering German messages.

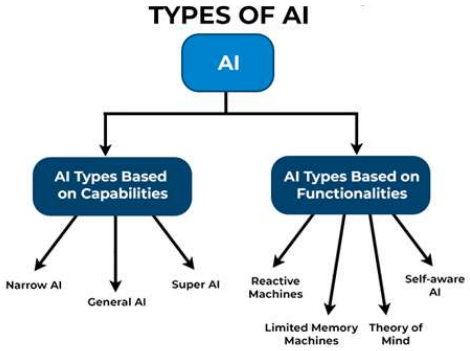
7. **Control theory and cybernetics:** How can artifacts operate under their own control?
   - Machines can *modify* their *behavior* in response to the environment (sense/action loop). Water-flow regulator, steam engine governor, thermostat, The theory of stable feedback systems (**1894**)
   - Weiner's book *cybernetics* (**1948**) awoke public to the possibility of artificially intelligent machines
8. **Linguistics:** How does language relate to thought?
   - Modern linguistics and AI were —born at about the same time, and grew up together, intersecting in a hybrid field *called computational linguistics or natural language processing.*



   - The problem of *understanding language* soon turned out to be considerably more complex than it seemed in 1957.
   - *Knowledge representation* (the study of how to put knowledge into a form that a computer can reason with)- tied to language and informed by research in linguistics.

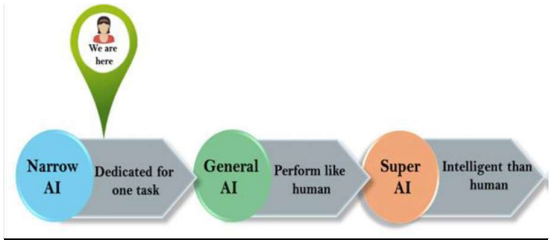
3.1 Types of Artificial Intelligence: The Strong and Weak AI

After taking a brief look at various disciplines that contribute towards AI, now let us look at the concept of **strong and weak AI** which also gives foundation for developing automated systems.



- **Type-1(Based on Capabilities):**

1. **Narrow AI (Weak AI):** This form of AI uses *Machine learning* which *is* designed and trained for specific tasks or domains, such as speech recognition, image classification, or recommendation systems. Narrow AI excels within defined parameters but *lacks general human-like intelligence.* **E.g.**: IBM's Watson, Siri and Alexa are weak AI.
2. **General AI (Strong AI):** General AI aims to exhibit human-like intelligence *(machine intelligence)* and *cognitive abilities* across a wide range of tasks. This form of AI is hypothetical and remains a *long-term goal of AI* research. It can find the solution to a problem and works beyond a preprogrammed algorithm. **E.g.** Visual perception, speech recognition, decision making, and translations between languages.
3. **Super AI (Artificially Super AI): Super AI** is AI that to go beyond in excellence than human intelligence and ability (*machine consciousness*). E.g. It's the best at everything — maths, science, medicine, etc...,



- **Type-2 (Based on functionalities):**

4. **Reactive machines:** Reactive machines are AI systems that have no memory and are task specific, meaning that an input always delivers the same output.
   - **Machine learning** models tend to be reactive machines because they take customer data, such as purchase or search history, and use it to

deliver recommendations to the same customers. **Ex**: Beat at chess by IBM's supercomputer Deep Blue, Netflix recommendations.

5. **Limited memory machines:** The next type of AI in its evolution is limited memory. This algorithm imitates the way our brains neurons work together, meaning that it gets smarter as it receives more data to train on**.**
   - **Deep learning** algorithms improve natural language processing (NLP), image recognition, and other types of reinforcement learning. **Ex:** Self-driving cars**.**
6. **Theory of mind:** The previous two types of AI, reactive machines and limited memory, are types that currently exist. If it is developed, theory of mind AI could have the potential to understand the world and how other entities have thoughts and emotions.
7. **Self-awareness:** The grand finale for the evolution of AI would be to design systems that have a sense of self, a conscious understanding of their existence. This type of AI does not exist yet.

3. History of AI:

The idea of artificial intelligence was introduced in the **1950s by Alan Mathison Turing**, a mathematician and computer scientist. Turing's paper named "Computing Machinery and Intelligence" included the question "**Can machines think?"**.

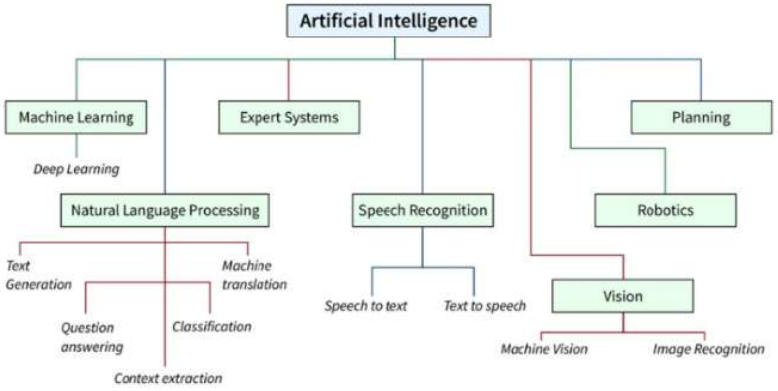
Later, in **1956, John McCarthy coined the term artificial intelligence** during the first AI conference held at Dartmouth College, New Hampshire. John McCarthy is called "The Father of Artificial Intelligence". Here is the history of AI during 20th century –

Year	Milestone / Innovation
1923	Karel Čapek play named "Rossum's Universal Robots" (RUR) opens in London, first use of the word "robot" in English.
1943	Foundations for neural networks laid.
1945	Isaac Asimov, a Columbia University alumnus, coined the term *Robotics*.

1950	Alan Turing introduced the Turing Test for evaluation of intelligence and published *Computing Machinery and Intelligence*. Claude Shannon published *Detailed Analysis of Chess Playing* as a search.
1956	John McCarthy coined the term *Artificial Intelligence*. Demonstration of the first running AI program at Carnegie Mellon University.
1958	John McCarthy invents LISP programming language for AI.
1964	Danny Bobrow's dissertation at MIT showed that computers can understand natural language well enough to solve algebra word problems correctly.
1965	Joseph Weizenbaum at MIT built *ELIZA*, an interactive problem that carries on a dialogue in English.
1969	Scientists at Stanford Research Institute Developed *Shakey*, a robot, equipped with locomotion, perception, and problem solving.
1973	The Assembly Robotics group at Edinburgh University built *Freddy*, the Famous Scottish Robot, capable of using vision to locate and assemble models.
1979	The first computer-controlled autonomous vehicle, Stanford Cart, was built.
1985	Harold Cohen created and demonstrated the drawing program, *Aaron*.
1990	Major advances in all areas of AI − • Significant demonstrations in machine learning • Case-based reasoning • Multi-agent planning • Scheduling • Data mining, Web Crawler • natural language understanding and translation • Vision, Virtual Reality • Games
1997	The Deep Blue Chess Program beats the then world chess champion, Garry Kasparov.
2000	Interactive robot pets have become commercially available. MIT displays *Kismet*, a robot with a face that expresses emotions. The robot *Nomad* explores remote regions of Antarctica and locates meteorites.

4. Sub Areas of AI:

Artificial intelligence (AI) applications are software programs that use **AI techniques** to perform specific **tasks**.



- **AI in Natural language processing (NLP)**: NLP allows computers to understand and generate human language. This technology is used in a variety of applications, such as *machine translation, spam filtering, and sentiment analysis.* **E.g.:** AltaVista's translation of web pages. Translation of Caterpillar Truck manuals into 20 languages.
- **AI in Vision Systems** – These systems understand, interpret, and comprehend visual input on the computer. For example,
    - A spying Aero plane takes photographs, which are used to figure out spatial information or map of the areas.
    - Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.
- **AI in Machine learning (ML)**: ML allows computers to learn from data and improve their performance over time. This technology is used in a variety of applications, such as *predictive analytics, fraud detection, and recommendation systems.*
- **AI in Robotics**: Robotics is the branch of AI that deals with the design, construction, and operation of robots. Robots are used in a variety of applications, such as *manufacturing, healthcare, and space exploration.*
    - **Self-Moving Robots:** AI makes robots smart at moving around on their own. It's like giving them a built-in GPS and a clever brain.

- **Object Recognition and Manipulation:** This is super useful, especially in places like warehouses, where they can do things like sorting and packing items accurately.

- **AI in Expert Systems:** Application-specific systems that rely on obtaining the knowledge of human experts in an area and programming that knowledge into a system.
- Suggestion for the spelling error while typing in the Google search box.
- MYCIN system for diagnosing bacterial infections of the blood and suggesting treatments.
- *DEC's* XCON system for custom hardware configuration. Radiotherapy treatment planning.

  NASA developed a system for classifying very faint areas in astronomical images into either stars or galaxies.
- **AI in Speech Recognition (speech-to-text recognition):** It is the capacity of a machine or program to recognize spoken words and transform them into text. PEGASUS spoken language interface to American Airlines' EAASY SABRE reservation system, which allows users to obtain flight information and make reservations over the telephone.
- **AI in Mathematical Theorem Proving** Use inference methods to prove new theorems.
- **AI in Scheduling and Planning**: Dynamic analysis and replanning tool (*DART*)used in Desert Storm and Desert Shield operations performs automated planning and scheduling for transportation. European space agency planning and scheduling of spacecraft assembly, integration and verification.

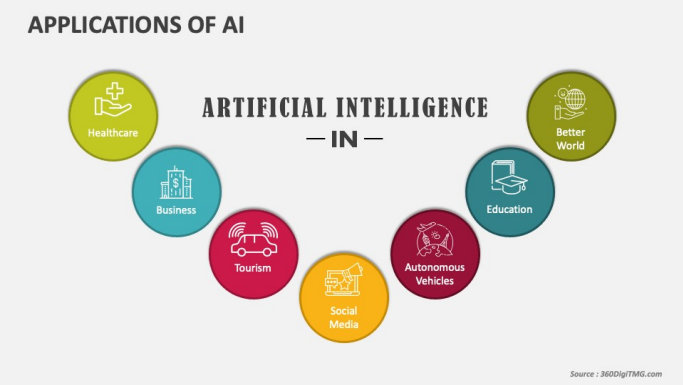
5. Applications of AI

Applications of AI in a wide variety of industries including healthcare, finance, manufacturing etc. are as follows:

AI in Astronomy: Machines can assist with data processing in the astronomical sector, such as capturing new stars, extraterrestrial planets, and even dark matter.

AI in Gaming:AI plays crucial role in strategic games such as *chess (***Deep Blue Chess** program beat world champion Gary Kasparov), *poker*, *tic-tac-toe*, *etc.*, where machine can think of large number of possible positions based on **heuristic** knowledge.

AI in Healthcare: AI is used for medical diagnosis, drug discovery, and predictive analysis of diseases.



- **AI in Finance:** AI helps in credit scoring, fraud detection, and financial forecasting.
- **AI in Retail:** AI is used for product recommendations, price optimization, and supply chain management.
- **AI in Manufacturing:** AI helps in quality control, predictive maintenance, and production optimization.
- **AI in Transportation:** AI is used for autonomous vehicles, traffic prediction, and route optimization.
- **AI in Automotive Industry:** AI is being used to develop *self-driving cars. Advanced Driver Assistance Systems* ***(ADAS)*** autonomously adjust your vehicle's speed while on the highway, swiftly engage the brakes when detecting potential hazards.
- **AI in Social Media:** *Virtual Assistants and AI Chatbots* on social media are used for customer support, answering frequently asked questions, and handling simple requests. AI-powered *Sentiment Analysis tools* can figure out how people feel (happy, sad, or neutral) on social media based on their comments or posts in message.
- **AI in Entertainment:** Recommendation of Content**:** AI looks at what customers have liked before, such as movies or music, and suggests new things that they might enjoy.
- **AI in Agriculture:** (*Crop Observation and Control)* ***AI***, with the help of various sensors, acts as a guardian for crops on the farm. AI controls several machines like tractors and drones. These machines can plant seeds, remove weeds, and spray stuff on crops all by themselves.
- **AI in E-commerce:** Personalized Product recommendations, dynamic pricing, virtual shopping assistants etc.,

Review Questions:

Part A – Two Marks

1. Distinguish Intelligence and Artificial Intelligence. L1, CO1
2. What is A.I. technique? Also tell the areas where it can be used. L2, CO1
3. What are the goals of AI? L1,CO1 *Refer 1.2*
4. List the advantages and disadvantages of AI?L1,CO1 Refer 10
5. What is meant by Turing test? Refer 1.2

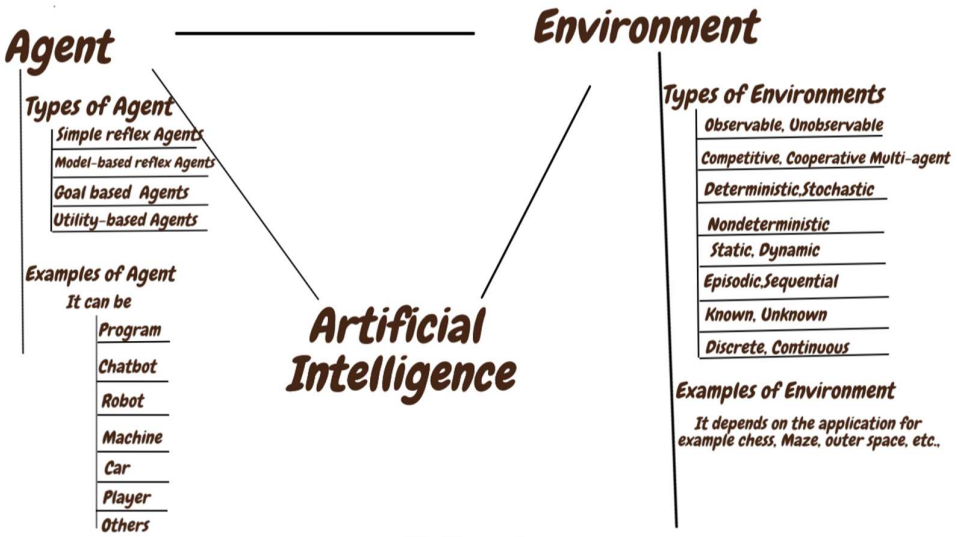
Part B -Ten Marks

1. Define AI? Explain the organization of AI. L2, CO1 *Refer 1,1.1*
2. Describe the Turing test for machine intelligence. Highlights its important features. L2, CO1 *Refer 1.1*
3. Explain about the foundations of AI. L2, CO1 *Refer 2*
4. Classify AI problems and Discuss the characteristics of AI Problems.L2, CO1 *Refer 1.2,1.2.2*
5. Write a short note on history of AI along with milestones. L2, CO1 *Refer 3*

Chapter 2-Intelligent Agents

1. Agents and Environment:

An AI system is composed of an agent and its environment. The agents act in their environment. An environment in artificial intelligence is the surrounding of the agent.



So, we define-

Agent as *"an independent program or an autonomous entity that is designed to perceive its environment through* ***sensors*** *and acts upon that environment through* ***actuators".***

Agents run through a cycle of *perception, thought, and action.*

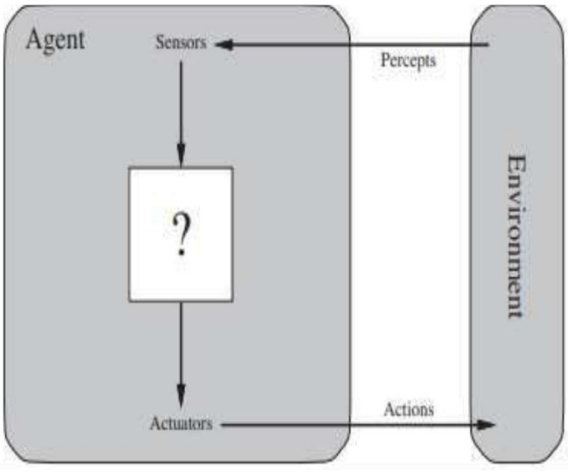
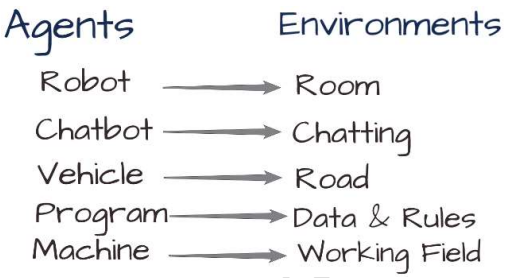


Figure: Agents interact with environments through sensors and actuators.

Examples of an agents include:

- A **human agent** has sensory organs such as eyes, ears, nose, tongue and skin parallel to the sensors, and other organs such as hands, legs, mouth, for actuators and physical world as environment.
- A **robotic agent** can have cameras, infrared range finder, NLP for *sensors* and various motors for *actuators*, physical world as environment.
- A **software agent** can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen. Operating system as environment.



1.1 Agent Terminology:

- **Performance Measure of Agent** – It is the criteria which determines how successful an agent is.
- **Behavior of Agent** – It is the action that agent performs after any given sequence of percepts.
- **Percept** – It is agent's perceptual inputs at a given instance.
- **Percept Sequence** – It is the history of all that an agent has perceived till date.
- **Agent Function** – It is a map from the precept sequence to an action.

$$\mathbf{f: P^* \rightarrow A}$$

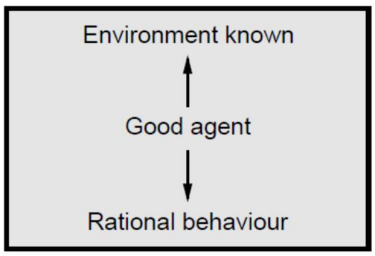
- **Agent Program:** A software/algorithm where agent function is typically implemented.
- **Intelligent Agent:** An intelligent agent is **an autonomous entity** which act upon an environment using sensors and actuators for achieving goals.

2. The Concept of Rationality:

In AI, rational agents are closely related to **intelligent agents** or **autonomous programs** that mimic human intelligence. It operates under the premise of *rationality*.

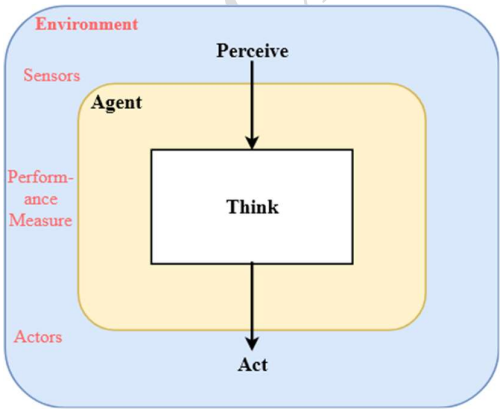
Rationality can be judged based on following points:

- The existence of a *performance measure* that defines the success criterion.
- The agent having *prior knowledge* about its environment.
- The best possible *actions* for an agent to perform. (*actuator dependency*)
- The agent's *percept sequence*



2.1 Rational Agent

Definition- "*A rational agent is an agent which has clear preference, models uncertainty, and acts in a way to* ***maximize its performance measure*** *with all possible actions.*"



A rational agent is said to perform the right things. A rational agent is anything that

- *Perceive* its environment through sensors
- *Think* and make decisions based on logical reasoning and
- *Act* through actuators

AI is about creating rational agents to use for ***game theory and decision theory*** for various real-world scenarios.

2.2 P.E.A.S Representation:

A task environment is a **"problem"** to which **rational agent** is a solution. In designing a rational agent, we need to specify the task environment by **PAGE** (Percept, Action, Goal, Environment) or **P.E.A.S Representation**.

- **P**erformance **M**easure It defines the success of an agent. It evaluates the criteria that determines whether the system performs well.
- **E**nvironment refers to the physical or virtual surroundings and conditions in which the AI system operates.
- **A**ctuators These are the mechanisms through which an AI agent interacts with its environment, such as motors, wheels, or speakers.
- **S**ensors These devices enable an AI agent to perceive its surroundings, gathering necessary data for decision-making, like cameras, microphones, or thermometer.

Example of Agents with their PEAS representation

Agent	Performance measure	Environment	Actuators	Sensors
Medical Diagnose	Healthy patient Minimized cost	Patient Hospital Staff	Tests Treatments	Keyboard (Entry of symptoms)
Vacuum Cleaner	Cleanness Efficiency Battery life Security	Room Table Wood floor Carpet Various obstacles	Wheels Brushes Vacuum Extractor	Camera Dirt detection sensor Cliff sensor Bump Sensor Infrared Wall Sensor
Part picking Robot	Percentage of parts in correct bins.	Conveyor belt with parts, Bins	Jointed Arms Hand	Camera Joint angle sensors.

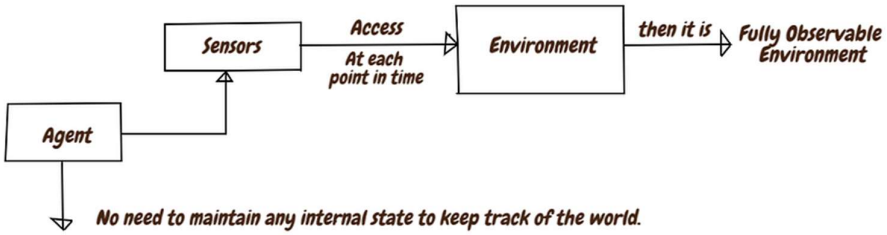
3. The Nature of Environments:

An **environment** in artificial intelligence is the **surrounding of the agent.** The agent takes input from the environment through sensors and delivers the output to the environment through actuators. **The environment is mostly said to be non-feministic.**

3.1 Properties of Environment:

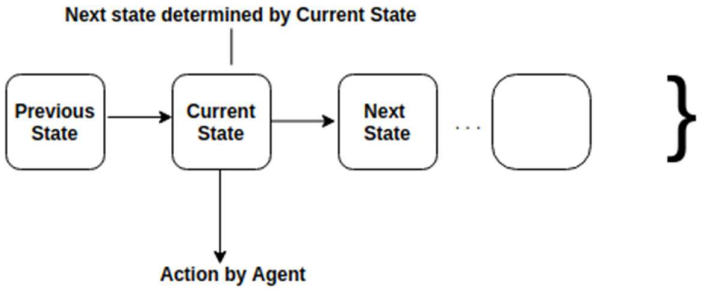
The environment in Artificial Intelligence refers to the outside circumstances in which an agent functions to accomplish a particular task. There are several types of environments:

1. **Fully observable (vs. partially observable):** A fully observable environment is one in which **the agent has complete information about the current state of the environment.** The agent has direct access to all environmental features that are necessary for making decisions.
   Example: board games like **chess or checkers-** the board is fully observable, and so are the opponent's moves.



- A partially observable environment is one in which the **agent does not have complete information** about the current state of the environment. The agent can only observe a subset of the environment, and some aspects of the environment may be hidden or uncertain.
- **Example: driving a car in traffic**. The environment is partially observable because what's around the corner is not known.

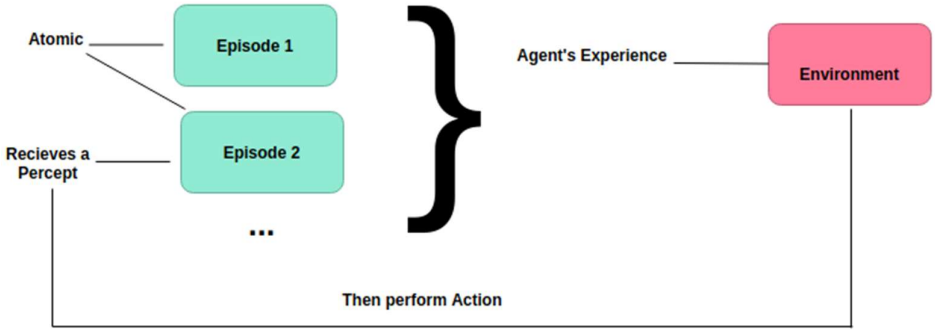
2. **Deterministic (vs. stochastic):** We say the environment is deterministic if the **next state of the environment is completely determined by the current state** and the action executed by the agent.
   Example: simple mathematical equations, where the outcome of each operation is precisely defined.



- A stochastic environment is one in which the **outcome of an action is uncertain** and involves probability.
- **Examples**: games of chance like **poker or roulette**, where the outcome of each action is influenced by random factors like the shuffle of cards or the spin of a wheel.

3. **Episodic (vs. sequential):** In an Episodic task environment, each of the agent's actions is divided into atomic incidents or episodes. **There is no dependency between current and previous incidents**. In each incident, an agent receives input from the environment and then performs the corresponding action.

   Example: Consider an example of **Pick and Place robot**, which is used to detect defective parts from the conveyor belts. Here, every time robot(agent) will make the decision on the current part i.e. there is no dependency between current and previous decisions.



- In a **Sequential environment**, the **current or previous decisions can affect all future decisions.** The next action of the agent depends on what action he has taken previously and what action he is supposed to take in the

future. **Example: Checkers-** Where the previous move can affect all the following moves.

4. **Static (vs. dynamic):** is one in which the **environment does not change over time.** The state of the environment remains constant, and the agent's actions do not affect the environment. **Ex:** C**ross word puzzles**

   A **dynamic** environment is one in which **the environment changes over time** and the agent's actions can affect the future state of the environment. **Examples** include **video games or robotics** applications.

5. **Discrete (vs. continuous):** If an environment consists of a **finite number of actions** that can be deliberated in the environment to obtain the output, it is said to be a discrete environment. **Example:** The game of **chess** is discrete as it has only a finite number of moves though the number of moves might vary.

   In contrast, a continuous environment is one in which the **state and action** spaces are **continuous and infinite**. **Ex: robotics or control systems**.

6. **Single agent (vs. multiagent):** An environment consisting of **only one agent** is said to be a single-agent environment. **Ex:** An agent solving a crossword puzzle.

   An environment involving **more than one agent** is a multi-agent environment. **Examples**: multiplayer games, traffic simulations.

4. Structure of an AI Agent:

The task of AI is to design an **agent program** which is an implementation of the agent function. Structure of an Intelligent Agent is a combination of agent, architecture and agent program and can be viewed as:

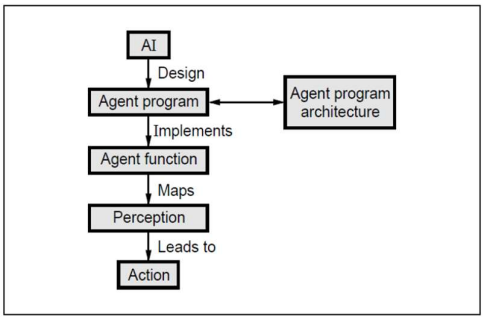
Agent = Architecture + Agent Program

Architecture is a device with sensors and actuators that the agent executes on. for example, a robotic car, a camera, and a PC.

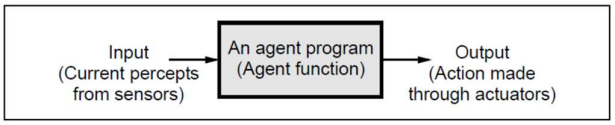
Agent function is a map from the percept sequence to an action which takes the entire percept history.

Agent program is an implementation of the agent function which executes on a computing device called architecture and takes the current percept as input from sensors and return action to the actuators.

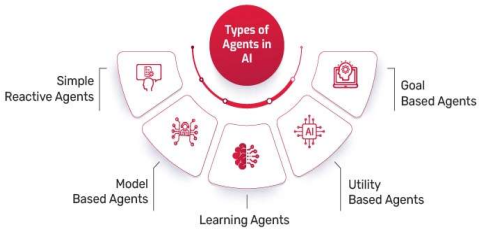
A) Schematic of Agent's Structure



B) Role of agent architecture in agent program



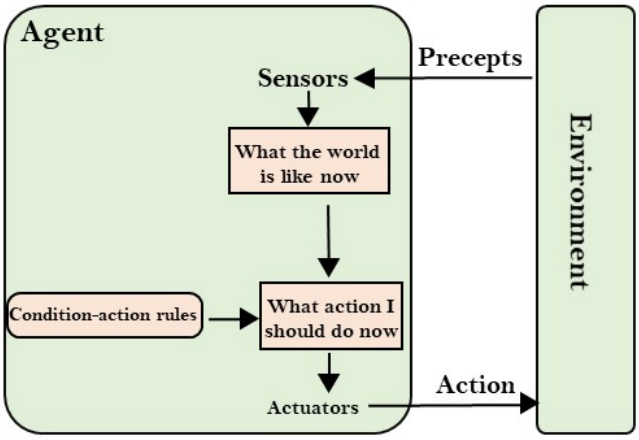
4.1 Types of Intelligent Agents in AI Environment:



Here are the five main types of AI agents:

1. **Simple reflex agents** are programmed to only succeed in the fully observable environment and take decisions based on the current precepts and ignore the rest of the percept history.

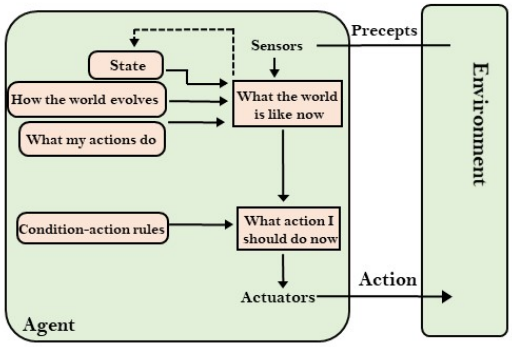
- The Simple reflex agent works on Condition-action rule, which means it maps the current state to action.
- Such as **a Room Cleaner agent**, it works only if there is dirt in the room. which means it maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.



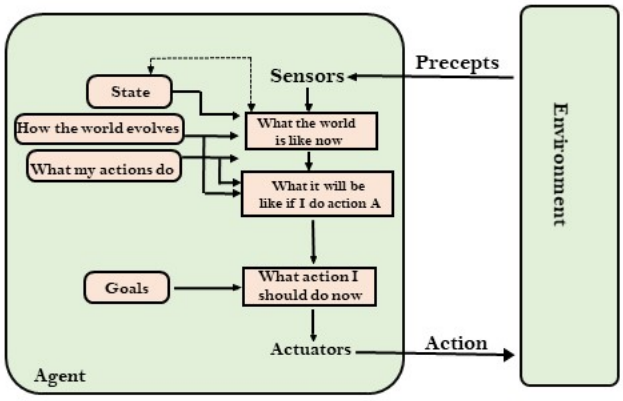
Problems for the simple reflex agent design approach:

- They have very limited intelligence
- They do not have knowledge of non-perceptual parts of the current state
- Mostly too big to generate and to store.
- Not adaptive to changes in the environment.

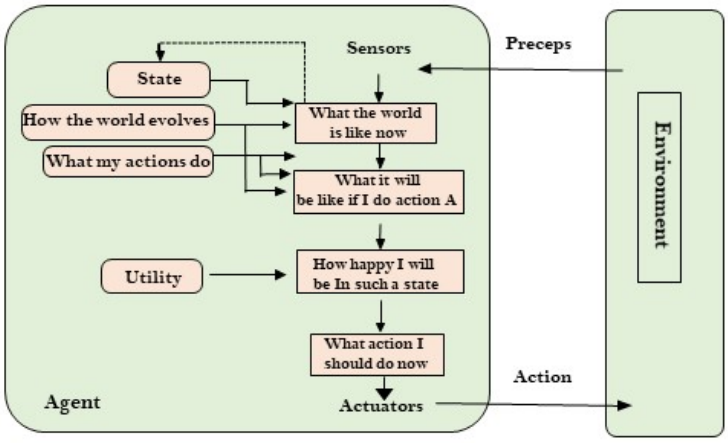
2. **Model-based reflex agents** can work in a partially observable environment and track the situation. A model-based agent has two important factors:
   Model: It is knowledge about "*how things happen in the world*," so it is called a Model-based agent.
   Internal State: It is a representation of the current state based on percept history. Updating the agent state requires information about:
   - How the world evolves independently from the agent?
   - How do the agent's actions affect the world?



3. **Goal-based agents** execute a program to achieve specific goals and take actions based on evaluating the current state of the environment. These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. They usually require search and planning. The goal-based agent's behavior can easily be changed.



4. **Utility-based agents** consider the potential outcomes of their actions and choose the best way to achieve the goal. The utility function maps each state to a real number to check how efficiently each action achieves the goals. They choose actions based on a **preference (utility)** for each state.

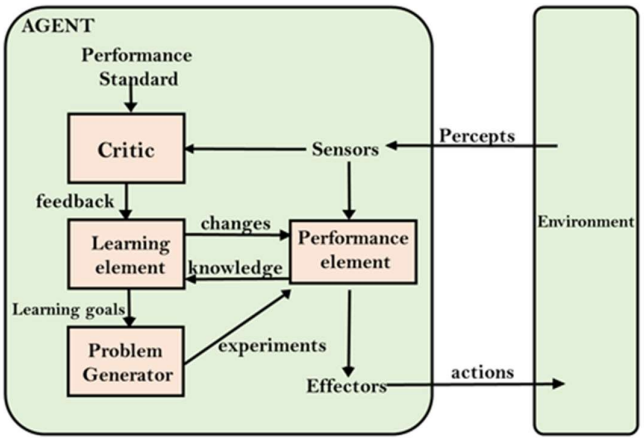


5. **Learning agents** execute machine learning techniques to improve their decision-making over time.

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

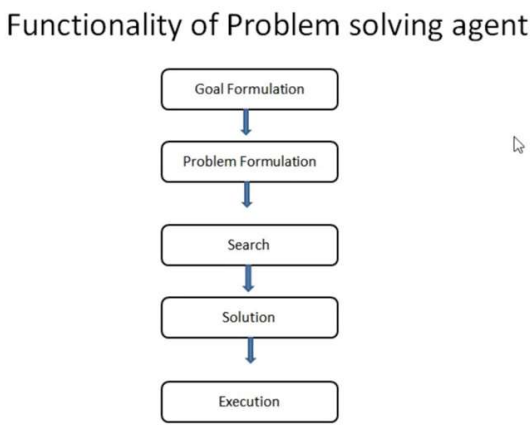
A learning agent has mainly four conceptual components, which are:

a. **Learning element:** It is responsible for making improvements by learning from the environment.
b. **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.
c. **Performance element:** It is responsible for selecting external action.
d. **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.



4.2 Problem Solving Agents in AI

A problem-solving agent is a goal-based agent and focuses on achieving goals. It decides what to do by finding sequence of actions that lead to desirable states.



1) **Goal formulation** define the problems that the agents need to solve.
2) **Problem formulation** is the process of deciding what actions and states to consider, given a goal.

The five components involved in problem formulation:

i. **Initial State:** This state requires an initial state for the problem which starts the AI agent towards a specified goal. In this state new methods also initialize problem domain solving by a specific class.

ii. **Action:** This stage of problem formulation works with function with a specific class taken from the initial state and all possible actions done in this stage.

iii. **Transition:** This stage of problem formulation integrates the actual action done by the previous action stage and collects the final stage to forward it to their next stage.

iv. **Goal test:** This stage determines that the specified goal achieved by the integrated transition model or not, whenever the goal achieves stop the action and forward into the next stage to determines the cost to achieve the goal.

v. **Path costing:** This component of problem-solving numerical assigned what will be the cost to achieve the goal. It requires all hardware, software and human working cost.

3) **Search:** The process of looking for a sequence of actions that reaches the goal is called search**.** A search algorithm takes a problem as input and returns a solution in the form of an action sequence.

4) **Solution:** It finds the best algorithm out of various algorithms, which may be proven as the best optimal solution. Finding a solution of a problem involves following phases:
   a. **Problem definition:** Detailed specification of inputs and acceptable system solutions.
   b. **Problem analysis:** Analyse the problem thoroughly.
   c. **Knowledge Representation:** collect detailed information about the problem and define all possible techniques.
   d. **Problem-solving:** Selection of best techniques.
5) **Execution phase:** Once a solution is found, the carrying actions it recommends is called the execution phase.

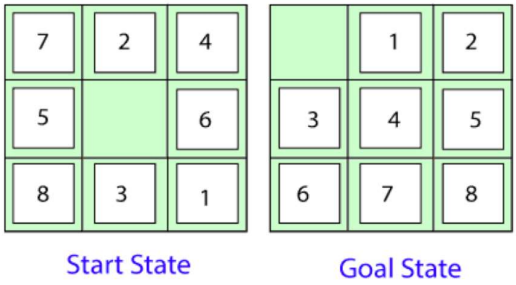
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

A. Toy Problems

1. 8-Puzzle Problem

States: A state of the 8-puzzle is a 3x3 grid, with eight numbered tiles and a blank space. The **objective** is to reach the goal state as shown in Figure. We can slide four adjacent tiles (**left, right, above, and below**) into the empty space.

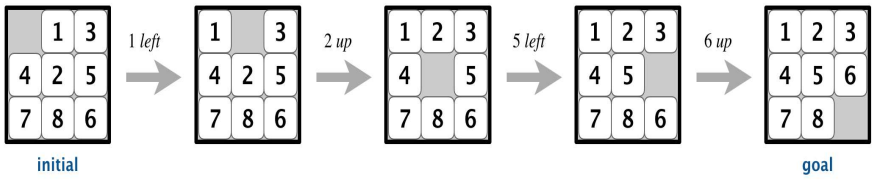


The problem formulation is as follows:

- **Initial State:** We can start from any state as the initial state.
- **Actions:** Here, actions or the movement of the blank space is defined, i.e., either left, right, up or down. The empty space cannot move diagonally and can take only one step at a time.
- **Transition Model:** Given the state and action it returns the resulting state.
- **Goal test:** It identifies whether we have reached the correct goal-state.
- **Path cost:** The path cost is the number of steps in the path where the cost of each step is 1.

Note: The 8-puzzle belongs to the family of **sliding-block puzzles.**

State Space for 8-Puzzle problem



2. Vacuum World

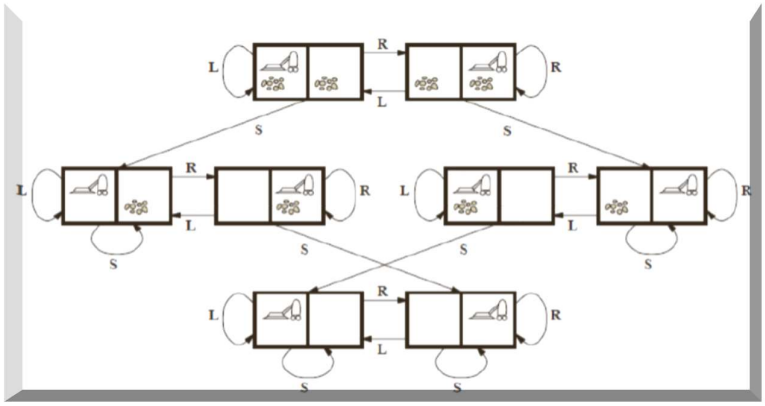
States: The state is determined by both the agent location and the dirt locations. The agent is in one of two locations, each of which might or might not contain dirt. Thus, there are $2 \times 2^2 = 8$ possible world states.

The problem formulation is as follows:

- **Initial state:** Any state can be designated as the initial state.
- **Actions**: Each state has just three actions: Left, Right, and Suck.
- **Transition model:** The actions have their expected effects, except that moving Left in the leftmost square, moving Right in the rightmost square, and Sucking in a clean square have no effect. The transition model defines a state space.
- **Goal test:** This checks whether all the squares are clean.
- **Path cost:** Each step costs 1, so the path cost is the number of steps in the path.

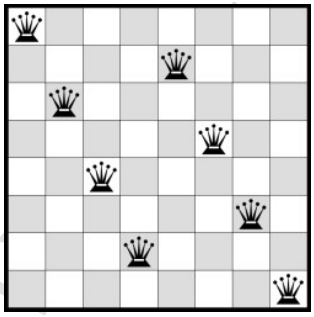
State space for the vacuum world

- Links denote actions: L = Left, R =Right, S = Suck.



3. 8- Queens problem

The goal of the 8-queens problem is to place 8 queens on the chessboard such that no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal).



For this problem, there are two main kinds of formulation:

1. Incremental formulation: It starts from an empty state where the operator augments a queen at each step.

Following steps are involved in this formulation:

States: Arrangement of any 0 to 8 queens on the chessboard.

Initial State: An empty chessboard

Actions: Add a queen to any empty box.

Transition model: Returns the chessboard with the queen added in a box.

Goal test: Checks whether 8-queens are placed on the chessboard without any attack.

Path cost: There is no need for path cost because only final states are counted.

In this formulation, there is approximately **1.8 x 1014** possible sequence to investigate.

2. Complete-state formulation: It starts with all the 8-queens on the chessboard and moves them around, saving from the attacks.

Following steps are involved in this formulation

States: Arrangement of all the 8 queens one per column with no queen attacking the other queen.

Actions: Move the queen at the location where it is safe from the attacks.

This formulation is better than the incremental formulation as it reduces the state space from **1.8 x 1014** to **2057**, and it is easy to find the solutions.

B. Real-World Problems

1. **Route-finding problems** are defined in terms of specified locations and transitions along links between them.
   - **E.g.** Traveling salesperson problem (**TSP**): It is a touring problem where the salesman can visit each city only once. The objective is to find the shortest tour and sell-out the stuff in each city.
   - Route-finding algorithms are used in a variety of applications such as Web sites and in-car systems that provide driving directions.
2. **VLSI layout problem** requires positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitances, and maximize manufacturing yield.
3. **Air Line Travel Problem**

   The airline travel problem is specifying as follows:

   1. **States:** Each is represented by a location (e.g., an airport) and the current time.
   2. **Initial state:** This is specified by the problem.
   3. **Transition state:** This returns to the states resulting from taking any scheduled flight (further specified by seat class and location), leaving later than the current time plus the within-airport transit time, from the current airport to another.
   4. **Goal Test:** Are we at the destination by some prespecified time?
   5. **Path cost:** This depends upon the monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of Date, type of airplane, frequent-flyer mileage awards, and so on.

Review Questions:

Unit-I

Part A - Two Marks

1. Define Intelligent Agent.L1, CO1

2. Distinguish an agent function and an agent program. L2, CO1

3. State the concept of Rationality? L1, CO1 *Refer 2.*

4. Define Rational agent? L1, CO1 *Refer 2.1*

5. What is a task environment? How is it specified? L1, CO1 *Refer 3.1*

6. Give the structure of an agent in an environment? L1, CO1 *Refer 4*

7. What are the four different kinds of agent programs? L2, CO1 *Refer 4.1*

Part B – Ten Marks

1. Elaborate in detail about reflex and goal-based agents. Illustrate the use of learning agents. L2, CO1 *Refer 4.1*

2. List the properties of task environment? Explain in detail about each property with examples. L2, CO1 *Refer 4*

3. What is a simple problem-solving agent? Explain it briefly. L2, CO1 *Refer 4.2*

4. What is problem formulation? Formulate 8 puzzle problem as AI problem. L3, CO1 *Refer 4.2,4.2(A)*

5. Explain the applications of AI? L2, CO1 *Refer 5*

.

UNIT - II

Searching- Searching for solutions, uniformed search strategies – Breadth first search, depth first Search. Search with partial information (Heuristic search) Hill climbing, A*, AO* Algorithms, Problem reduction, Game Playing-Adversial search, Games, mini-max algorithm, optimal decisions in multiplayer games, Problem in Game playing, Alpha-Beta pruning,

1. Searching:

Searching is a step-by-step procedure to solve a search-problem in search space.

a. Searching for solutions:

"Solution to a search problem is a sequence of actions, that transforms the start state to the goal state".

This plan is achieved through search strategies or algorithms.

b. Search Algorithm:

The search algorithm takes **"***a problem as input and returns a solution in the form of action sequence***".** Each of these algorithms will have:

A problem **graph,** containing the start node S and the goal node G.

A **strategy,** describing the way the **graph** will be traversed to get to G.

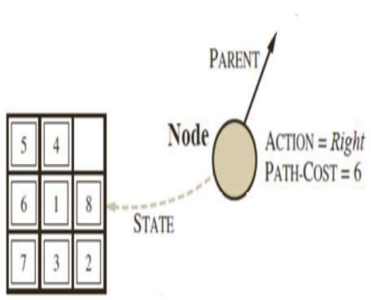
A **frontier(fringe**) is a **data structure** Queue (FIFO), or Stack (LIFO) used to store all the possible states (nodes) that you can go from the current states.

c. Search Tree:

A tree representation of search problem is called Search tree. The root of the search tree is the root node which corresponds to the initial state.

For each **Node** of the **tree**, we have a structure that contains four components:

- **State:** The state in the state space to which the node corresponds.
- **Parent node:** The node in the search tree that generated this node.
- **Action:** The action that was applied to the parent to generate the node.
- **Path-cost:** The cost, traditionally denoted by **g(n),** of the path from the initial state to the node, as indicated by the parent pointers.
- **Depth:** The number of steps along the path from initial state.



d. Properties of Search Algorithms:

The Criteria to measure the performance of different search strategies:

Completeness: A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

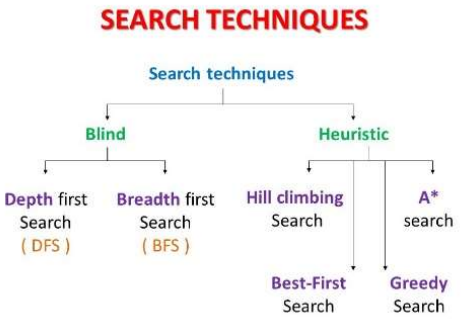
Optimality: If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

Time Complexity: Time complexity is a measure of time for an algorithm to complete its task.

Space Complexity: It is the maximum storage space required at any point during the search, as the complexity of the problem.

2. Types of search algorithms

Based on the search problems we can classify the search algorithms into uninformed (Blind search) search and informed search (Heuristic search) algorithms.



2.1 Uninformed (Blind) Search Strategies

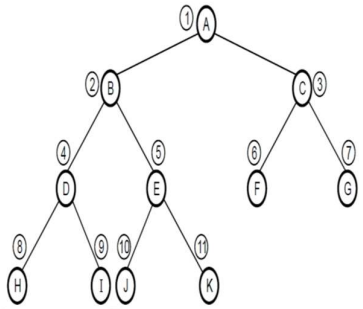
A way in which *search tree is searched without any information* about the search space like initial state operators and test for the goal, so it is also called **blind search.**

It can be divided into six main types:

1. Breadth-first Search
2. Depth-first Search
3. Depth-limited Search
4. Iterative deepening depth-first search
5. Uniform cost search
6. Bidirectional Search

a. Breadth-First Search Algorithm

- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- It always expands the Shallow nodes first.
- BFS is implemented by calling TREE-SEARCH (problem, FIFO-QUEUE ())
- Queue data structure puts all newly generated successors at the end of the queue.



Path: A-B-C-D-E-F-G-H-I-J-K

Advantages:

- BFS guarantees the **shortest path** to the goal in terms of the number of actions.
- It's also complete, meaning it will find a solution if one exists.

Limitations:

- It can be **memory-intensive**, especially in large state spaces.
- It may not be efficient for searching deep or infinite spaces.

Properties:

Complete?? Yes (if b is finite)

Time?? $1 + b + b^2 + b^3 + \ldots + b^d + b(b^d - 1) = O(b^{d+1})$, i.e., exp. in d

Space?? $O(b^{d+1})$ (keeps every node in memory)

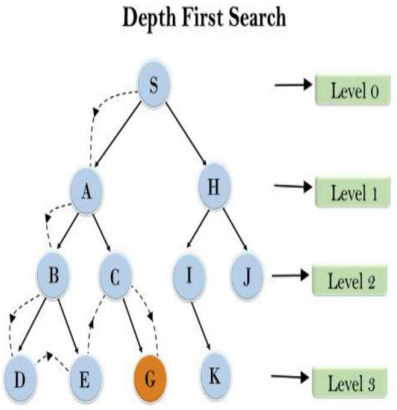
Optimal?? No, unless step costs are constant

Space is the big problem; can easily generate nodes at 100MB/sec
so 24hrs = 8640GB.

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    frontier ← a FIFO queue with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node ← POP(frontier)  /* chooses the shallowest node in frontier */
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
                frontier ← INSERT(child, frontier)
```

b. Depth First Search:

- Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
- It always expands to the ***deepest node*** in the frontier of the search tree.
- DFS is implemented in TREE-SEARCH using a last-in first-out (**LIFO**) queue data structure.
- Generated states are added at the front of the Queue.
    - Root node--->Left node ----> right node



Advantages:

It is memory-efficient and well-suited for deep or infinite state spaces.

It can find a solution quickly if it's not too far from the initial state.

Limitations: It does not guarantee the shortest path, and it may get stuck in infinite loops or deep branches

Properties:

Complete?	No: fails in infinite-depth spaces, spaces with loops Modify to avoid repeated states along path ➔ complete finite spaces
Time?	$O(b^m)$ where m is the maximum depth of search tree terrible if m is much larger than d (depth of shallowest solution) but if solutions are dense, may be much faster than breadth-first
Space?	O(bm) linear space
Optimal?	No

```
Algorithm DFS (v)
//Given an undirected (OR directed) graph G = (V, E) with
//n vertices and an array visited initially set
// to zero, this algorithm visits all vertices
//reachable from v. G and visited [ ] are global.
{
      visited [v] : = 1;
   for each vertex w adjacent from v do
   {
      if (visited [w] = 0 then DFS (w);
   }
}
```

- Time complexity - $O(b^d)$.
- Space complexity - $O(b^d + 1)$.

2.2 Search with partial information:

If environment is not fully observable and knowledge of the states and actions is incomplete, then agent has different types of task environment. Such an environment leads to three distinct types of problems -

- Sensor less or conformant problem - Agent may have no sensors i.e. no idea where it is; solution is a sequence.
- Contingency problem – Percepts provide new information about current state; solution is a tree or policy; often interleave search and execution. A contingency

problem is called as adversarial problem if the uncertainty is caused by the actions of another agent.

- Exploration problem – When states and actions of the environment are unknown.

2.3 Heuristic search:

In an informed search, *problem information is available* which can guide the search. Informed search is also called a *Heuristic search*.

Informed search strategies can find a solution more efficiently than an uninformed search strategy.

They are:

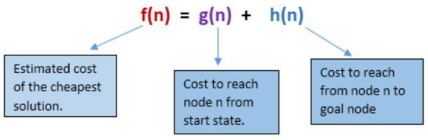
1. Hill Climbing
2. A* Search
3. AO* search

An example of informed search algorithms is a traveling salesman problem.

Heuristic function: It is represented by h(n), and it calculates the cost of an optimal path from the current state to goal state. The value of the heuristic function is always positive.

2.3.1. A* Search Algorithm

The A* (A-star) algorithm is a popular **pathfinding** and **graph traversal** algorithm used to find the shortest path between two nodes in a graph. It employs a "heuristic estimate" as well as the cost to reach the node. Hence, we can combine both costs as follows, and this sum is called a **fitness number f(n).**



Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node, then return success and stop, otherwise

Step 4: Expand node n and generate all its successors and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to Step 2.

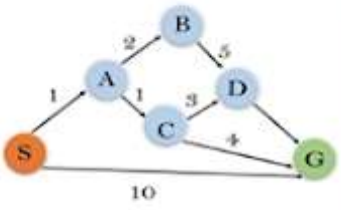
Advantages: A* guarantees an optimal solution (shortest path) if an admissible heuristic is used. It's also efficient in terms of the number of nodes expanded.

Limitations: The quality of the heuristic matters. If the heuristic is not admissible, the optimality guarantee is lost.

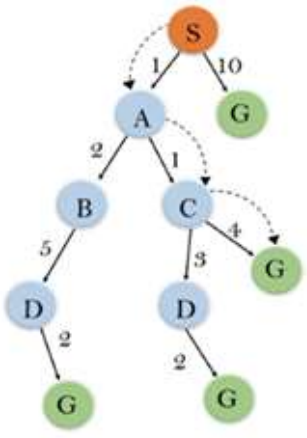
Applications: Tower Défense is a type of strategy video game where the goal is to defend a player's territories or possessions by obstructing enemy attackers, usually achieved by placing defensive structures on or along their path of attack.

Example:

In this example, we will traverse the given graph using the A* algorithm. The heuristic value h (n) of all states is given in the below table. Here we will use OPEN and CLOSED list.



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



Initialization: {(S, 5)}

Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

Iteration 4 will give the result, as S--->A--->C--->G

it provides the optimal path cost=6

2.3.2 Hill-Climbing:

The Hill-Climbing search algorithm is simply a loop that continually moves in the direction of increasing value – that is, **uphill**.

It terminates when it reaches a "**peak**" where no neighbor has a higher value.

The idea behind hill climbing is as follows:

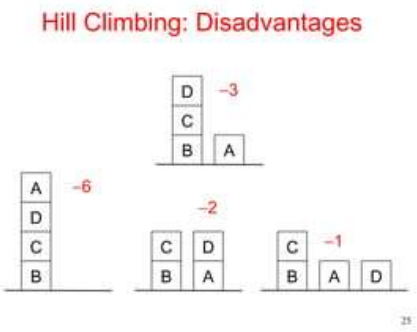
1. Pick a random point in the search space.
2. Consider all the neighbors of the current state.
3. Choose the neighbor with the best quality and move to that state.
4. Repeat 2 thru 4 until all the neighbouring states are of lower quality.
5. Return the current state as the solution state.

Algorithm:

Function HILL-CLIMBING(*Problem*) returns a solution state Inputs: *Problem*, problem
Local variables: *Current*, a node
Next, a node
Current = MAKE-NODE(INITIAL-STATE[*Problem*])
Loop do
Next = a highest-valued successor of *Current*
If VALUE[Next] < VALUE[Current] then return *Current*
Current = *Next*
End

Problems with Hill Climbing:

- The main problem with hill climbing (which is also sometimes called ***gradient descent***) is that we are not guaranteed to find the best solution.
- Hill Climbing can get stuck in local optima, which are suboptimal solutions.
- The choice of the initial solution can significantly impact the algorithm's performance.



Applications: Hill Climbing finds applications in network routing, hyperparameter tuning, game strategy optimization, chip design, and more.

2.4 Problem Reduction

- Problem reduction search is a basic problem-solving technique of AI.
- A problem can be divided into a set of sub problems, where each sub problem can be solved separately and a combination of these will be a solution.
- The AND-OR (AO*) graphs are used for representing the solution.

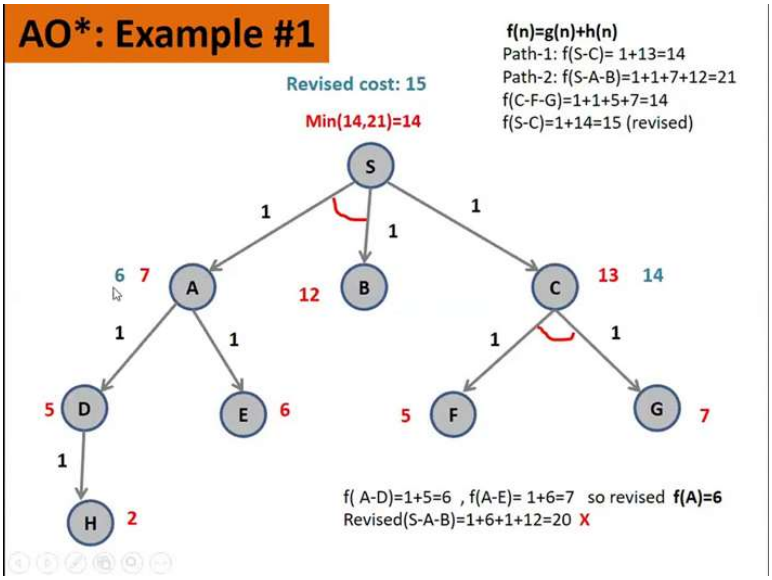
2.4.1 AO* algorithm

The AO* method **divides** any given difficult **problem into a smaller group** of problems that are then resolved **using the AND-OR** graph concept.

The AND side of the graph represents a set of tasks that must be completed to achieve the main goal, while the OR side of the graph represents different methods for accomplishing the same main goal.

The evaluation function in AO*: **f(n)** = Actual cost + Estimated cost

$$f(n) = g(n) + h(n)$$



2.4.1 Game Playing -Adversarial search

Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called **adversarial searches**, often known as **Games**.

Games are modelled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

Formalization of the problem:

A game can be formally defined as a search problem with the following components:

- The initial state, which includes the board position and identifies the player to move.
- A successor function, which returns a list of (*move*, *state*) pairs.
- A terminal state-States where the game has ended.
- A utility function (also called an objective function or payoff function), which give a numeric value for the terminal states.

2- Person Games

Players: We call them **Max** and **Min**.

Initial State: Includes board position and whose turn it is.

Operators: These correspond to legal moves.

Terminal Test: A test applied to a board position which determines whether the game is over. In chess, for example, this would be a checkmate or stalemate situation.

Utility Function: A function which assigns a numeric value to a terminal state. For example, in chess the outcome is win (+1), lose (-1) or draw (0).

Note that by convention, we always measure utility relative to **Max.**

2.4.2 Mini-max Algorithm

1. Generate the whole game tree.

2. Apply the utility function to leaf nodes to get their values.

3. Use the utility of nodes at level n to derive the utility of nodes at level n-1.

4. Continue backing up values towards the root (one layer at a time).

5. Eventually the backed-up values reach the top of the tree, at which point Max chooses the move that yields the highest value. This is called the minimax decision because it maximizes the utility for Max on the assumption that Min will play perfectly to minimize it.

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v
```

```
function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v
```

```
function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

2.4.3 Alpha-beta pruning

Alpha-beta pruning is an optimization technique for the minimax algorithm. By performing pruning, we can eliminate large part of the tree from consideration.

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

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

The main condition which required for alpha-beta pruning is: **α>=β**

Key points about alpha-beta pruning:

- 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.

```
function ALPHA-BETA-SEARCH(state) returns an action
   inputs: state, current state in game
   v ← MAX-VALUE(state, −∞, +∞)
   return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
   inputs: state, current state in game
           α, the value of the best alternative for MAX along the path to state
           β, the value of the best alternative for MIN along the path to state
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for a, s in SUCCESSORS(state) do
      v ← MAX(v, MIN-VALUE(s, α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Review Questions

Unit-II

Part-A

Distinguish uninformed and informed search strategies?
Define heuristic functions
List the criteria to measure the performance of search strategies.
What are the advantages of Breadth-First Search.
Define Search tree and write its properties.
What is AO* Search?
What is game tree?
Define Evaluation function.

Part-B

Describe searching for solutions. Give the structure of search tree.
Elucidate search algorithms with uninformed search strategies.
Define Heuristic search? Explain any two Heuristic Search Techniques
Write in detail the Breadth-First Search algorithm. Illustrate with example

Explain about A* search in detail.
Illustrate the heuristic Hill climbing algorithm with an example.
What is problem reduction. Illustrate AO* search with example.
Describe Adversial Search in game-playing with an example.
Explain Minmax search in gaming.
Elaborate in detail about alpha-beta pruning with example.

LECTURE PLAN

UNIT III

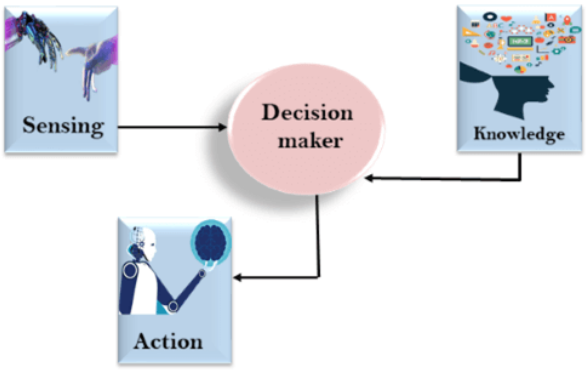
S.No.	Topic	No. of Periods	Proposed lecture	Actual lecture	Pertaining CO(s)	Taxonomy Level	Mode of Delivery
			Period	Period			
1	Knowledge Representation issues	1			CO3	L2	MD1, MD5
2	Predicate logic	1			CO3	L1	MD1, MD5
3	Logic Programming	1			CO3	L2	MD1, MD5
4	Semantic Nets	1			CO3	L1	MD1, MD5
5	Frames and Inheritance	1			CO3	L1	MD1, MD5
6	Constraint Propagation	1			CO3	L2	MD1, MD5
7	Representing knowledge using Rules	1			CO3	L2	MD1, MD5
8	Rules-based deduction systems	1			CO3	L2	MD1, MD5
9	Reasoning under uncertainty	1			CO3	L3	MD1, MD5
10	Review of probability	1			CO3	L1	MD1, MD5
11	Bayes' probabilistic interferences	1			CO3	L1	MD1, MD5
12	Dempster Shafer theory	1			CO3	L1	MD1, MD5

UNIT - III

Representation of Knowledge: Knowledge representation issues, predicate logic- logic programming, semantic nets- frames and inheritance, constraint propagation, representing knowledge using rules, rules-based deduction propagation, representing knowledge using rules, rules-based deduction systems. Reasoning under uncertainty, review of probability, Bayes' probabilistic interferences and dempster Shafer theory.

1. Introduction

Knowledge Representation in AI *describes the representation of knowledge in a structured form*. The fundamental *goal* of knowledge representation is to make *inferences* and *draw conclusions*. To achieve this, various *knowledge representation techniques* can be used, such as *logical representation, semantic network representation, frame representation, and production rules.*



As we can see in above diagram, there is one decision maker which *act* by *sensing* the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.

Any knowledge representation system should possess the *properties* such as *learning, efficiency in acquisition, representational adequacy and inferential adequacy*.

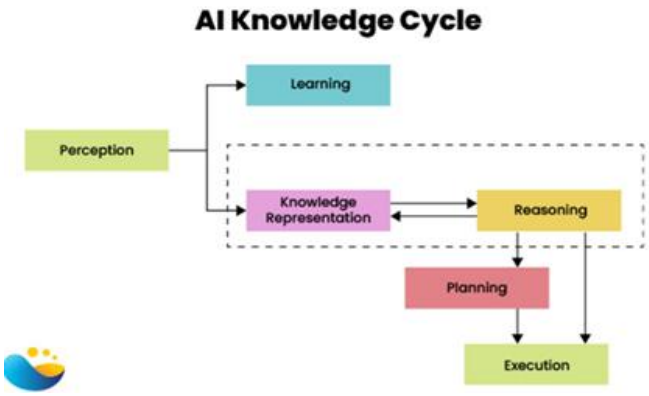
Knowledge representation in AI is not just about storing data in a database, it allows a *machine to learn from that knowledge* and behave intelligently like a human being.

For *example*, a knowledge representation system might be used to *build a chatbot* that can answer questions about a particular topic *or* a *recommendation engine* that can suggest products based on a user's preferences.

1.1 Cycle of Knowledge Representation in AI

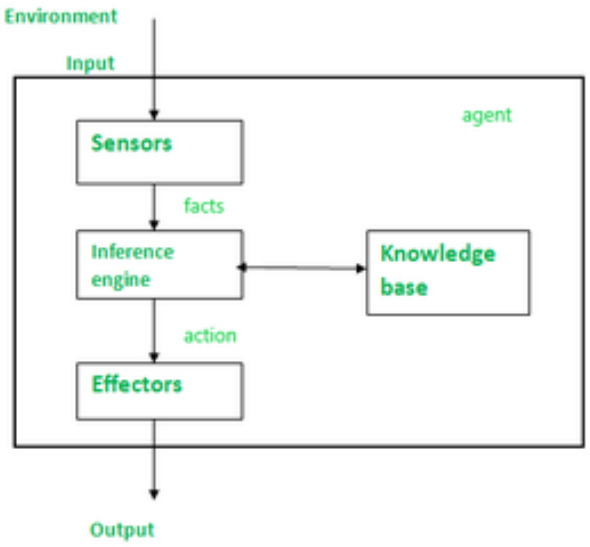
Artificial Intelligent Systems usually consist of various components to display their intelligent behavior. Some of these components include:

- Perception
- Learning
- Knowledge Representation & Reasoning
- Planning
- Execution



1.2 What to Represent:

The knowledge that needs to be represented in AI are *objects, events, performance, facts, meta-knowledge, and Knowledgebase.*



1. Objects: Objects refer to things in the world that have physical properties and can be observed, touched, or manipulated. *Object-oriented programming* is an example of a technique that uses objects to represent knowledge in AI

2. Events: Events refer to actions or occurrences that take place in the world. Event-based systems use events to represent knowledge in AI. Examples of events include *driving a car, cooking food.*

3. Performance: Performance refers to the behavior of agents or systems that perform a task. Performance-based systems use performance to represent knowledge in AI.

4. Facts refer to propositions that are either true or false. They are statements that can be verified using evidence or logical deduction. Example: "*the sky is blue*,"

5. Meta-Knowledge refers to knowledge about knowledge. It helps machines reason about the quality and validity of the knowledge they are using.

6. Knowledge Base: A knowledge base is a collection of facts, rules, procedures, and another knowledge relevant to a particular domain.

1.3 Properties of Knowledge Representation

A good knowledge representation system must possess the following properties:

* Representational Accuracy: KR system should have the ability to represent all kind of required knowledge.

* Inferential Adequacy: KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.

* Inferential Efficiency: The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.

* Acquisitional efficiency: The ability to acquire the new knowledge easily using automatic methods.

1.4 Knowledge Representation Issues

1. Incompleteness and Uncertainty: Incompleteness and uncertainty can lead to incorrect diagnoses or treatment recommendations.

2. Scalability Problems: Scalability issues can result in slow response times and hamper the performance of AI-driven recommendation systems.

3. Representation Bias: Representation bias occurs when knowledge representations favour certain perspectives or groups, leading to unfair or inaccurate decision-making.

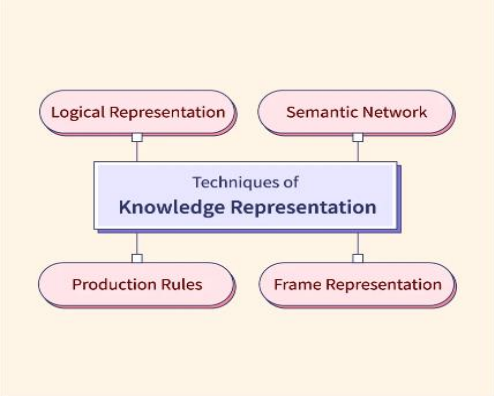
4. Handling Context and Context-Dependent Knowledge**:** AI systems must be able to handle context-dependent knowledge to provide accurate guidance.

5. The Symbol-Grounding Problem: The symbol-grounding problem is concerned with connecting abstract symbols in AI systems to real-world entities.

6. Ontology Development and Maintenance Challenges: Ontologies are fundamental for knowledge representation, but developing and maintaining them can be complex.

1.5 Knowledge Representation Techniques

The Knowledge Representation models/mechanisms are often based on:



1) **Logical Representation:** This method employs formal logic to represent information systematically.
2) **Semantic Networks:** These graphical representations connect related concepts, making it easier to visualize relationships.
3) **Production Rules:** Utilized to establish a set of rules for decision-making and problem-solving.
4) **Frames Representation:** This technique structures information into frames or templates, enhancing data organization.

1.5.1 Logical Representation

Logical representation means *drawing a conclusion based on various conditions. 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 and 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.

➢ **Advantages:**

* Logical representation helps to perform logical reasoning.
* This representation is the basis for the programming languages.

➢ **Disadvantages:**

* Logical representations have some restrictions and are challenging to work with.
* This technique may not be very natural, and inference may not be very efficient.

1.5.2 Approaches of Logical Representation

Logical representation can be categorised into mainly two logics:

1. Predicate Logic
2. Propositional Logic

A. Predicate Logic in AI

Predicate logic in artificial intelligence, also known as first-order logic (FOL), *is a formal system used in logic and mathematics to represent and reason about complex expressions in easier forms using predicates, variables, and quantifiers.* It can represent negation, conjunction, disjunction, and many more types of statements by using symbols.

∧	*and* [conjunction]
∨	*or* [disjunction]
⇒	*implies* [implication]
¬	*not* [negation]
∀	*For all*
∃	*There exists*

Figure: Logic symbols used in Predicate Logic

➢ **Basic components of predicate logic are:**

* Predicates are *statements* or propositions that can be either *true* or *false* depending on the values of their arguments.
* Variables are symbols that can take on different values. In predicate logic, variables are used to *represent objects* or entities in the domain of discourse.
* Constants are *specific* values that do *not change*. For instance, in a knowledge base about people, "Alice" and "Bob" might be constants representing *specific* individuals.

* Quantifiers are used to specify the *scope of variables* in logical expressions. There are two types of quantifiers in predicate logic –

a) *Universal Quantifier* **(∀):** it indicates that the statement within the quantifier is *true for all objects* in the domain

b) *Existential Quantifier* **(∃):** it indicates that there exists *at least one object* for which the statement within the quantifier is true

➢ **Examples of Predicate Logic in AI:**

1. **Predicate Example**: "IsHungry(x)"

Predicate Symbol: IsHungry

Argument: x (variable representing an object)

Meaning: These predicate asserts that a specific object represented by "x" is hungry.

2. **Universal Quantification**: ∀x IsHuman(x) → IsMortal(x)

This statement uses the universal quantifier to claim that for all objects "x" in the domain, if "x" is human, then "x" is mortal.

3. **Existential Quantification**: ∃x IsHungry(x)

This statement uses the existential quantifier to claim that there is at least one object "x" in the domain that is hungry.

➢ **Characteristics of Predicate Logic**

* The Logical inference is allowed.
* More accurate knowledge representation of facts of the real world.
* Program designing is its application area.
* Better theoretical foundation.

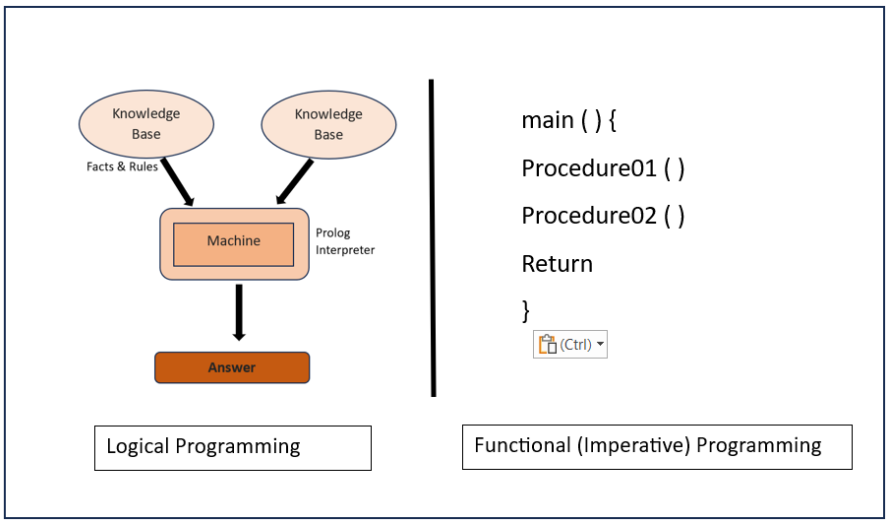
➢ **Applications of Predicate Logic:**

* In predicate logic knowledge is represented in the form of *facts* & *rules* as in done in *prolog*.
* Predicates help AI planners search for a sequence of actions to achieve a goal.
* Predicates allow machine learning algorithms to *discover patterns* and *make predictions*
* Predicates are used in natural language processing for understanding the *semantics* of sentences.
* Predicates allow expert systems to make *informed decisions and solve problems* by using facts & rules.

1.5.3 Introduction to Programming Languages

A programming language includes syntax, semantics of programs and *the* computation.

1) Procedural Language: The program specifies a computation by saying *"how" it* is to be performed. *FORTRAN, C, and Object-oriented languages* fall under this general approach.



2) Declarative Language: The program specifies a computation by giving the *logical properties* of a correct answer. LISP, *P*rolog and Logic Data Language (LDL) are called logic programming languages.

1.5.4 Logic Programming

- Logic program is a collection of logic statements which express facts and rules about a problem domain.
- Logic programming specify a computation in terms of logical relations between entities.
- Computation determines whether, a particular conclusion follows from those logical statements.
- In logic programming, using the knowledge base (collection of facts and rules), the machine can find answers to the given questions.

- Major programming language families include ***Prolog, LDL, LISP***

➢ **Characteristics of Logic program**

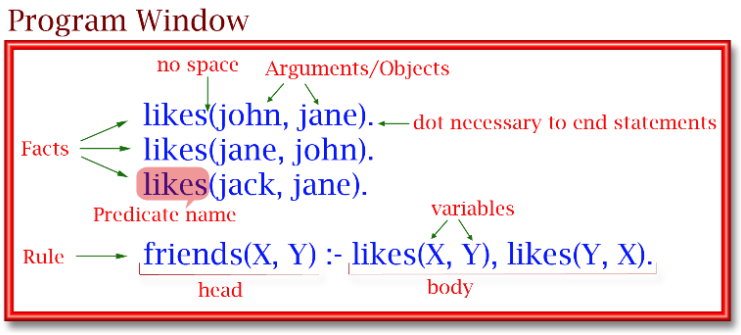
1. Logic program is characterized by set of relations and inferences. – program consists of a set of axioms and a goal statement.
2. Rules of inference determine whether the axioms are sufficient to ensure the truth of the goal statement.
3. Execution of a logic program corresponds to the construction of a proof of the goal statement from the axioms.
4. In logic programming, logic is used to represent knowledge and inference is used to manipulate it.

1.5.5 PROLOG

Prolog as the name itself suggests, is the short form of **PRO**gramming in **LOG**ics

Definition: "It is a logical and declarative programming language designed for developing logic-based AI applications".

Developers can "set rules & facts" around a problem, and then Prolog's interpreter will use that information to automatically execute those plans to find solutions.



Prolog programs consist of set of three different *Horn clauses* (": -") -

1. **Facts** – The fact is a predicate that is true

   Syntax: Relation (object1, object2).

   Example: Father (James, Robert)

2. **Rules** – Rules consists of a head and body separated by conditional clauses ": - "

   Syntax: rule name (object1, object2): - fact (object1, object2).

   Example: Parent (person1, person2): - Father (James, Robert).

3. **Queries** – To run a prolog program, we need some questions, and those questions can be answered by the given facts and rules. Knowledge Base can be considered like database, against which we can query.

Query syntax: ?: - odd number (7).

Output: No.

Explanation: As our knowledge base does not contain the above fact, output is *No*

- Prolog uses uppercase letters for **variables** and lowercase for **constants**.
- Commas separate literals in the body, and Period marks the end of a sentence

➢ **Advantages:**

1. Easy to build database. Doesn't need a lot of programming effort.
2. Pattern matching is easy. Search is recursion based.
3. It has built in list handling. Makes it easier to play with any algorithm involving lists.

➢ **Disadvantages:**

1. LISP logic programming language dominates over prolog with respect to I/O features.
2. Sometimes input and output is not easy.

➢ **Applications of Prolog:**

* It plays a vital role in automation system.
* Intelligent Database Retrieval
* Natural Language Understanding
* Specification Language
* Machine Learning
* Robot Planning
* Problem Solving

1.6 Semantic Network 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 extended.
* This representation consists of mainly two types of relations:

a. IS-A Relation (Inheritance)

b. Kind-of-Relation

Example: Statements are represented in the form of nodes and arcs.

Statements: a) Man is a human.

b) All humans has name.

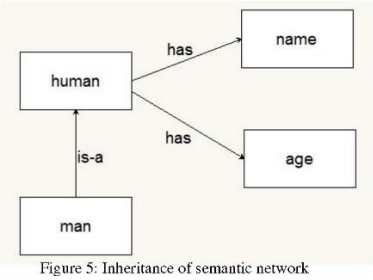


Figure 5: Inheritance of semantic network

In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object relates to another object by some relation.

➢ **Advantages of Semantic network:**

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

➢ **Drawbacks in Semantic representation:**

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions.
2. It is not possible to build a vast semantic network to model human-like memory (Which has 1015 neurons and links) to store the information.
3. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.
4. Difficulties in modelling sophisticated knowledge structures.
5. These networks are not intelligent and depend on the creator of the system.

1.7 Frame Representation

Frames in artificial intelligence are a structured knowledge representation technique. It consists of a collection of slots and facets (**fillers**). A frame is also known as slot-filler knowledge representation in artificial intelligence.

➢ **Key Characteristics of Frames:**

- Structure: Frames are organized hierarchically, with frames containing slots (attributes) and fillers (values) representing specific information about objects or concepts.

- **Attributes and Values:** Frames focus on modelling knowledge using attribute-value pairs. Each frame has attributes (slots) associated with specific values (fillers).
- Inheritance: Frames often support inheritance, allowing subframes to inherit attributes and values from their parent frames.
- Use Cases: Frames are well-suited for structured representation in knowledge-based systems, and expert systems. Including *Natural language processing and machine visions*.

➢ **Frame Representation example for a "person"**

Slots(attribute)	**Filler**(value)
Name	John Smith
Age	35
Gender	Male
Address	123 Main Street
Phone Number	(555) 123-4567

➢ **Advantages of frame representation:**

1. It makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many AI applications
3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

➢ **Disadvantages of frame representation:**

1. Inference mechanism cannot be smoothly proceeded by frame representation.
2. Frame representation has a much-generalized approach.

1.8 Frame Inheritance-Hierarchical Structures

Inheritance is an essential feature of frame-based system. In this hierarchical arrangement, *frames inherit properties and attributes from their parent frames*. This means that frames can automatically acquire and extend their knowledge, based on their position in the frame hierarchy.

➢ **Illustrating Frame Inheritance with an example:**

In frame-based systems, frames are organized hierarchically.

- Consider a frame representing a "**Car**."

- This frame might contain **slots** such as "Make," "Model," "Year," and "Color."
- Each of these slots has corresponding **fillers**, providing specific values.
- Additionally, the "Car" frame may be organized hierarchically within a larger category called "Vehicle," inheriting certain properties and attributes.

➢ **Advantages:**

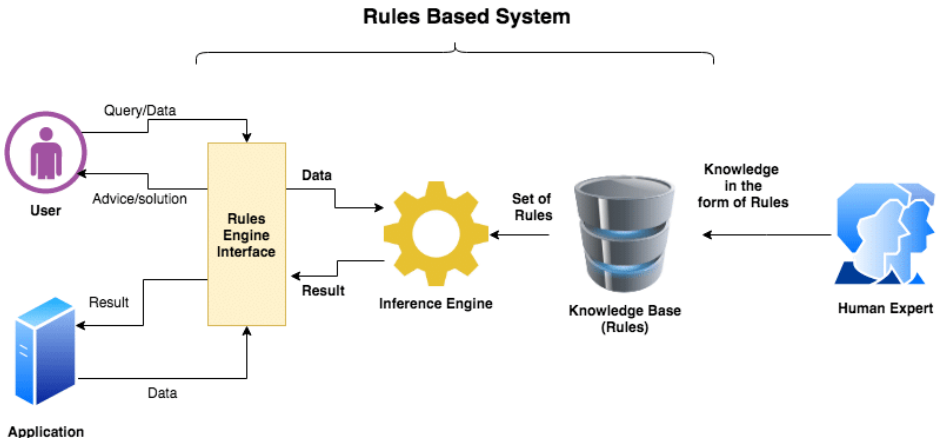
Inheritance **simplifies** knowledge representation in several ways:

- Reduces Redundancy: Inheritance eliminates the need to redundantly specify common attributes and values for each specific frame.
- Maintains Consistency: Inheritance ensures that related frames share consistent information.
- Facilitates Extensibility: New frames can be added to the hierarchy without the need to specify all their attributes from scratch.
- Enables Classification: Inheritance allows frames to be classified based on their position in the hierarchy.
- Supports Abstraction: Hierarchical frames can represent abstract and specific concepts.

1.9 Representing Knowledge using Rules:

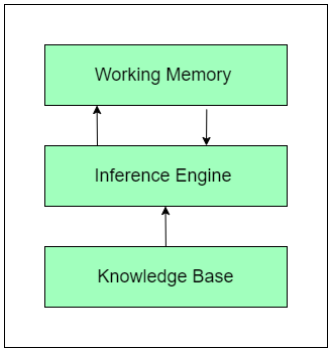
Rule based systems in AI express knowledge using a *set of rules* or *production rules* to generate judgements or suggestions. Rules are expressed in the following form:

"If<condition> then <conclusion>"



1.9.1 Components of Rule-Based System

A typical rule-based system comprises several key components:



1. Rules are conditional statements that define the system's behavior. "IF condition THEN action."
2. Knowledge Base: This is the repository where all the rules and facts are stored.
3. Inference Engine applies the rules to the knowledge base to derive conclusions or make decisions.
4. Working Memory: This is a dynamic component that holds the current facts. It is updated as the inference engine applies rules.
5. User Interface allows users to interact with the system, input data, and receive outputs or recommendations.

1.9.2 Rule-based Deduction System

A rule-based deduction system is a computer system that uses *rules* & *reasoning* to *solve problems* in a specific domain. The rules are often written as **if-then** statements, such as **"If X happens, then do Y"**.

Here's how a rule-based system works:

1. Create rules and facts: A developer creates a list of rules and facts for the system.
2. Analyse data: The system processes the input data against the rules.
3. Perform actions: The system follows the rules and performs any programmed functions.

- **Advantages**
  - Accuracy: Rule-based systems operate by cause and effect, and only within their rule set.
  - Ease of use: Rule-based systems require only small amounts of simple data to perform tasks and repetitive processes.
  - Speed: With the proper training, rule-based systems can make informed decisions quickly and efficiently.
- **Disadvantages:**

- Lack of Learning Capability: Rule-based systems are exact and do not have learning capabilities. Including too many rules can slow down a system and introduce complexity.
- Scalability: Altering existing rules or incorporating new rules can introduce time-consuming and expensive complications.
- High Maintenance Costs: To keep the rules accurate and up to date, rule-based systems need continual maintenance.

➢ **Applications of Rule-based System**

A classic example of a rule-based system is the domain-specific expert system that uses rules to make deductions or choices.

- Help a doctor choose the correct diagnosis based on a cluster of symptoms.
- Select tactical moves to play a game.

2. Constraint Propagation in AI

Artificial Intelligence (AI) encompasses a variety of methods and techniques to solve complex problems efficiently. One such technique is constraint propagation, which plays a crucial role in areas like scheduling, planning, and resource allocation. A method of inference that assigns values to variables characterizing a problem in such a way that some conditions (called constraints) are satisfied. This process can result in more domain reductions.

Definition: Constraint propagation is a process in artificial intelligence (AI) that uses "*constraints to reduce the possible values of variables and find new constraints*".

Key Concepts

1. Variables: Elements that need to be assigned values.
2. Domains: Possible values that can be assigned to the variables.
3. Constraints: Rules that define permissible combinations of values for the variables.

How Constraint Propagation Works

- Constraint propagation works by iteratively narrowing down the domains of variables based on the constraints.
- This process continues until no more values can be eliminated from any domain. The primary goal is to reduce the search space and make it easier to find a solution.

Steps in Constraint Propagation

1. **Initialization**: Start with the initial domains of all variables.
2. **Propagation**: Apply constraints to reduce the domains of variables.
3. **Iteration**: Repeat the propagation step until a stable state is reached, where no further reduction is possible.

Example: Consider a simple CSP with two variables, X and Y, each with domains {1, 2, 3}, and a constraint X ≠ Y.

Constraint propagation will iteratively reduce the domains as follows:

- If X is assigned 1, then Y cannot be 1, so Y's domain becomes {2, 3}.
- If Y is then assigned 2, X cannot be 2, so X's domain is reduced to {1, 3}.
- This process continues until a stable state is reached.

3. Reasoning in AI

Reasoning in Artificial Intelligence refers to the *"process by which AI systems analyse information, make inferences, and draw conclusions to solve problems or make decisions"*.

Types of Reasoning in AI

1) Probabilistic Reasoning in AI: Probabilistic reasoning involves dealing with uncertainty and making decisions based on probabilities. AI systems use statistical models to assess the likelihood of different outcomes and make informed choices.
2) Default Reasoning in AI: It is a type of non-monotonic reasoning where conclusions are drawn based on default assumptions unless explicitly contradicted.
3) Statistical Reasoning in AI: statistical reasoning involves the use of statistical methods to analyse data, identify patterns, and make predictions.

3.1 Probabilistic Reasoning with uncertainty in Artificial intelligence

A) Uncertainty:

- Till now, knowledge representation using first-order logic and propositional logic gives **certainty**, which means we were sure about the predicates.
- With this knowledge representation, we might write A→B, which means if A is true then B is true.
- But consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

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

B) Causes of uncertainty:

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

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

C) Probabilistic Reasoning:

Probabilistic Reasoning is a way of knowledge representation where we apply the concept of probability with logic to indicate the uncertainty in knowledge.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

- Bayes' rule
- Bayesian Statistics

3.2 Review of Probability

Let's understand some common terms:

1. **Probability:** Probability can be defined "*as a chance that an uncertain event will occur.*" It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.
   * $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
   * $P(A) = 0$, indicates total uncertainty in an event A.
   * $P(A) = 1$, indicates total certainty in an event A.

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

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

2. **Event:** Each possible outcome of a variable is called an event.
3. **Sample space:** The collection of all possible events is called sample space.
4. **Random variables:** Random variables are used to represent the events and objects in the real world.

5. **Prior probability:** The prior probability of an event is probability computed before observing new information.
6. **Posterior Probability:** The probability that is calculated after all evidence or information has considered. It is a combination of prior probability and new information.
7. **Conditional probability** is a "probability of occurring an event when another event has already happened".

   Let's suppose, we want to calculate the event A when event B has already occurred,

   * The probability of A under the conditions of B, it can be written as:

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

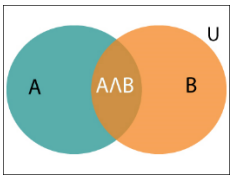
   Where, $P(A \wedge B)$ = Joint probability of A and B

   $P(B)$ = Marginal probability of B.

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

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

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



3.3 Bayes Theorem

In AI, Bayes' Theorem is used in probabilistic reasoning, machine learning, and data science to model and handle uncertainty. Bayes theorem can be derived using product rule and conditional probability of event A with known event B:

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

OR

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

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

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

The above equation (a) is called as **Bayes' rule** or **Bayes' theorem**. This equation is basic of most modern AI systems for **probabilistic inference**.

* P(A|B) is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.
* P(B|A) is called the **likelihood**, in which we consider that hypothesis is true, then we calculate the probability of evidence.
* P(A) is called the **prior probability**, probability of hypothesis before considering the evidence.
* P(B) is called **marginal probability**, pure probability of an evidence.

➢ **Application of Bayes' theorem in Artificial intelligence:**

* It is used to calculate the next step of the robot when the already executed step is given.
* Bayes' theorem is helpful in weather forecasting.
* It can solve the Monty Hall problem.
* Applications include spam email classification, medical diagnosis, natural language processing, and Bayesian networks.

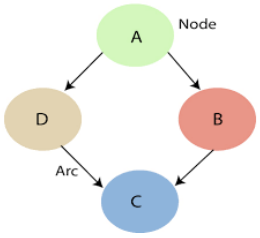
3.4 Bayesian Networks (aka Belief Networks)

Bayesian Networks, also known as Bayes Nets, Belief Nets, Causal Nets, and Probability Nets are "*space-efficient data structure for computing any value in the full joint probability distribution of the set of random variables*".

Bayesian Network can be used for building models from data and expert's opinions, and it consists of two parts:

- Directed Acyclic Graph
- Table of conditional probabilities.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.

- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.

Properties:

1. Can be used to reason:
   - o Forward (top-down) from causes to effects -- **predictive reasoning** (**causal reasoning**)
   - o Backward (bottom-up) from effects to causes -- **diagnostic reasoning**
2. Captures both qualitative and quantitative relationships between variables

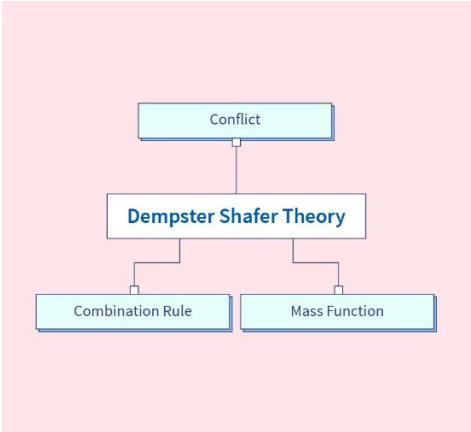
3.5 Dempster Shafer Theory

Dempster-Shafer Theory is a *"mathematical framework for reasoning about uncertainty"* which was proposed by Arthur P. Dempster in 1967 and Glenn Shafer in the 1960s. This theory was released because of the following reason: -

* Bayesian theory is only concerned about single evidence.
* Bayesian probability cannot describe ignorance.

Definition: Dempster-Shafer theory (DST), also known as belief function theory or evidence theory, is a framework for representing, quantifying, and managing uncertainty in artificial intelligence.

Dempster Shafer Theory stands as a valuable tool in the field of artificial intelligence to *solve problems, handle uncertainty* effectively and make more *accurate decisions*.



3.5.1 The uncertainty in this model is given by: -

1. **Conflict of evidence:** In DST, uncertainty arises from conflicting evidence or incomplete information. It Considers all possible outcomes.

2. **Combination Rule:** Dempster's rule of combination, combine belief functions from different sources.

3. **Mass functions:** In particular, a mass function must assign a mass of zero to the empty set (corresponding to no information), and a mass of one to the entire set (corresponding to complete certainty).

4. **Belief and plausibility**: Given a mass function, we can calculate two measures of uncertainty. They are:

a) **Belief:** The belief of a set A is the sum of the masses of all the focal elements which include A. Belief represents the **lower bound** of uncertainty (how much we believe A). A **belief function** assigns a degree of belief (or plausibility) to each possible outcome.

b) **Plausibility** is the sum of the masses of all the focal elements which intersect A. Plausibility represents the **upper bound** of uncertainty (how much we can accept A). Plausibility (denoted by Pl) is thus related to Bel by:

Pl(p) = 1 − Bel(~p). It also ranges from 0 to 1

Dempster–Shafer Theory in AI, can analyse the evidence, assign masses to subsets of possible conclusions, and calculate beliefs and plausibility where, **belief<= plausibility**

3.5.2 Characteristics of Dempster Shafer Theory

Dempster Shafer Theory in artificial intelligence (AI) exhibits several notable characteristics:

1. **Uncertainty Representation:** it provides a way to represent and reason incomplete evidence.
2. **Conflict of Evidence:** The DST allows for the combination of multiple sources of evidence of evidence.
3. **Handling Ignorance:** Dempster Shafer Theory, ignorance is gradually diminished through the accumulation of additional evidence.
4. **Decision-Making Ability:** By deriving measures such as belief, probability and plausibility from the combined belief function it helps in decision making.

3.5.3 Advantages & Disadvantages

Advantages

* As we add more information, the uncertainty interval reduces.
* DST has a much lower level of ignorance.
* Enhances the robustness of decision-making processes in AI systems.
* Handling of incomplete or conflicting information encountered in artificial intelligence.

Disadvantages

* In this, computation complexity is high, as we must deal with 2n sets
* The process of combining evidence necessitates careful modelling and calibration.
* The interpretation of belief and plausibility values in DST may possess subjectivity, biases.

****

Review Questions Unit-III

PART-A (Two Marks)

1. Define knowledge representation and list its properties. *Refer 1.1*
2. Write the merits and demerits of Logical Representation. *Refer 1.5.1*
3. What is semantic nets representation. Give the diagrammatic representation with an example. *Refer 1.6*
4. Write the Frame representation for *"Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England"* *Refer 1.7*
5. Define predicate logic. Name the components of predicate logic. *Refer 1.5.1(A)*
6. Compare declarative and procedural representation of knowledge using rules. *Refer 1.9*
7. Define Logic programming in AI. *Refer 1.5.4*
8. What is reasoning in AI? Write the types of reasoning in AI. *Refer 3*
9. Define probability. What is the formula for probability of occurrence. *Refer 3.2*
10. State Dempster Shafer Theory. *Refer 3.5*

PART-B (Ten Marks)

11. Explain knowledge representation in AI? *Refer1.1,1.2*
12. Illustrate knowledge representation using Predicate Logic. *Refer 1.5.1(A)*
13. Explain Semantic Network Representation. *Refer 1.6*

14. What are Frames in AI? Demonstrate with example. *Refer 1.7*
15. Discuss in detail about Rule Based Systems. *Refer 1.9*
16. How constraint propagation works in AI. Illustrate with example. *Refer 2*
17. Illustrate Frame Inheritance with example. *Refer 1.8*
18. Explain Probabilistic Reasoning under uncertainty. *Refer 3.1*
19. Describe Bayes' Theorem and its application in AI. *Refer 3.3*
20. Elaborate Dempster Shafer Theory. *Refer 3.5*

****

LECTURE PLAN

UNIT IV

S.No.	Topic	No. of Periods	Proposed lecture Period	Actual lecture Period	Pertaining CO(s)	Taxonomy Level	Mode of Delivery
1	First order logic	1			CO4	L2	MD1, MD5
2	Inference in first order logic	1			CO4	L2	MD1, MD5
3	Propostional vs First order inference	1			CO4	L2	MD1, MD5
4	Unification & Lifts	1			CO4	L1	MD1, MD5
5	Forward chaining	1			CO4	L1	MD1, MD5

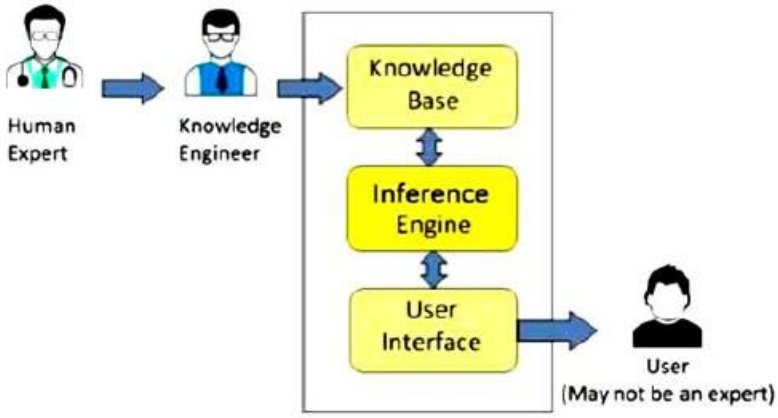
6	Backward chaining	1			CO4	L1	MD1, MD5
7	Resolution	1			CO4	L1	MD1, MD5
8	Learning from observation	1			CO4	L2	MD1, MD5
9	Inductive learning	1			CO4	L2	MD1, MD5
10	Decision Tree	1			CO4	L2	MD1, MD5
11	Explanation based learning	1			CO4	L2	MD1, MD5
12	Statistical learning methods	1			CO4	L2	MD1, MD5
13	Reinforcement Learning	1			CO4	L2	MD1, MD5

UNIT - IV

Logic concepts: First order logic. Inference in first order logic, propositional vs. first order inference, unification & lifts forward chaining, Backward chaining, Resolution, learning from observation Inductive learning, Decision trees, Explanation based learning, Statistical Learning methods, Reinforcement Learning.

1. Introduction

Logical AI involves representing knowledge of an agent's world, its goals and the current situation by **sentences in logic**. Logical AI deal with using **knowledge base** and **rule sets** to make intelligent decisions.



An *Inference Engine* is a component of the expert system that applies logical rules to the knowledge base to deduce new information. It interprets and evaluates the facts in the knowledge base to provide an answer. *A knowledgebase is a structured collection of facts about the system's domain.*

1.1 First-Order logic

First Order Logic (FOL) can be defined as "*a collection of objects, their attributes, and relations among them to represent knowledge*". It's also known as Predicate Logic.

- **Objects:** A, B, people, numbers, colors, squares, pits, wars, theories, wumpus, ...
- **Relations:** It can be unary relation such as: red, round, is adjacent, or n-any relation such as: the sister of, brother of, has colour, comes between
- **Function:** Father of, best friend, third inning of, end of, ...
- First-order logic also has two main parts as a natural language:
  - Syntax
  - Semantics

Syntax of First-Order logic: In first-order logic, the syntax of FOL determines which **set of symbols** represents a logical expression.

The basic elements of FOL syntax are as follows:

Constant	1, 2, A, John, Mumbai, cat,
Variables	x, y, z, a, b,
Predicates	Brother, Father, >,
Functions	sqrt,
Connectives	∧, v, ¬, ⇒, ⇔

Equality	==
Quantifier	∀, ∃

a) Atomic sentences: Predicate can be used to represent **atomic sentences** (term1, term2, term3....., term n).

Ravi and Ajay are brothers: => Brothers (Ravi, Ajay).

b) Complex Sentences: Connectives are used to join atomic sentences to form complex sentences.

c) Subject: The major component of the sentence is the subject.

d) Predicate: A predicate is a relationship that ties two atoms together in a sentence.

Consider the following statement: **"x is an integer."**

e) Quantifiers: These are the symbols that allow you to determine or identify the variable's range and scope in a logical expression. There are two different kinds of quantifiers:

Universal Quantifier (for all, everyone, everything): Universal Quantifier used to express that a statement is true for all objects in the domain.

Statement: All man drink coffee

Logic representation: ∀xman(x)→drink(coffee).

It will be read as: There are all x where x is a man who drink coffee.

Existential Quantifier, (for some, at least one)**:** Existential quantifiers are a sort of quantifier that expresses that a statement is true for at least one instance of something within its scope.

Statement: Some boys are intelligent.

Logic representation: ∃x: boys(x)∧intelligent(x)

It will be read as: There are some x where x is a boy who is intelligent.

1.2 Inference in First Order Logic

In artificial intelligence, we need intelligent computers which can create new logic from old logic or by evidence, so generating the conclusions from evidence and facts is termed as Inference. *"Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences".*

A. Substitution:

Substitution is a fundamental operation performed on *terms* and *formulas*. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL. If we write F[a/x], so it refers to *substitute a constant* "a" in place of variable "x".

B. FOL Inference rules for quantifiers:

There are some Inference rules that can be applied to sentences with quantifiers to obtain sentences without quantifiers. *These rules will lead us to make the conversion.*

1. Universal Generalization: This rule can be used if we want to show that every element has a similar property. It can be represented as:

$$\frac{P(c)}{\forall x\, P(x)}$$

Ex: P(c): "A byte contains 8 bits",
∀ x P(x): "All bytes contain 8 bits.", it will also be **true**.

2. Universal Instantiation: The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from ∀ x P(x) for any object in the universe of discourse. It can be represented as:

$$\frac{\forall x\, P(x)}{P(c)}$$

Ex: IF "Every person like ice-cream"=> ∀x P(x) so, we can infer that,
"John likes ice-cream" => P(c)

3. Existential Instantiation is also known as Existential Elimination. This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c. It's written like this:

$$\frac{\exists x\, P(x)}{P(c)}$$

Ex: From the given sentence: ∃x Crown(x) ∧ OnHead(x, John),So we can infer: Crown(K) ∧ OnHead(K, John),

The above used K is a constant symbol, which is called Skolem constant. The Existential instantiation is a special case of **Skolemization process.**

4. Existential introduction is also known as an Existential Generalization. This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P. It's written like this: $\frac{P(c)}{\exists x P(x)}$

Ex: Let's say that "Priyanka got good marks in English."
Therefore, "someone got good marks in English"

C. Generalized Modus Ponens Rule:

For the inference process in FOL, we have a *single inference rule* which is called Generalized Modus Ponens. It is lifted version of Modus ponens. Generalized Modus Ponens can be summarized as, " *P implies Q and P is asserted to be true, therefore Q must be True."*

According to Modus Ponens, for atomic sentences *pi, pi', q.* Where there is a substitution θ such that SUBST (θ, pi',) = SUBST (θ, pi), it can be represented as:

$$\frac{p1', p2', \ldots, pn', (p1 \wedge p2 \wedge \ldots \wedge pn \Rightarrow q)}{SUBST(\theta, q)}$$

1.3 Propositional Logic Vs. First Order Logic

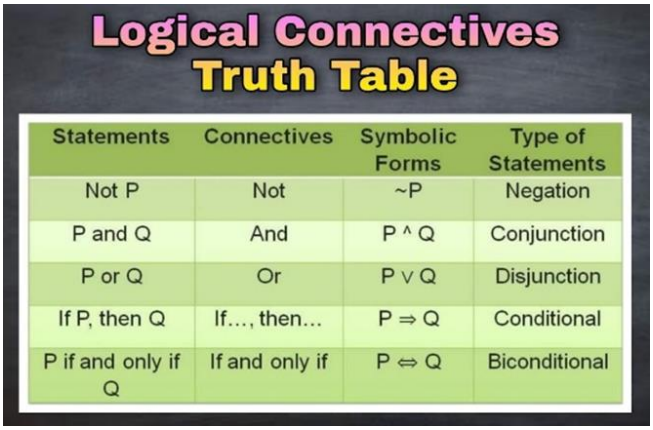
In knowledge representation, two fundamental forms of logic are used:

a.Propositional Logic b. First Order Logic

A. Propositional logic: *Propositional logic or sentential logic, deals with propositions that can be either true or false and uses logical connectives to form complex expressions.* However, propositional logic is *limited* because it cannot represent relationships between objects or quantify over them.

The basic components of propositional logic include:

- Propositions: Basic statements that are either true or false.
- Logical Connectives: Operators such as AND (∧), OR (∨), NOT (¬), IMPLIES (→) and BICONDITIONAL (↔) used to combine propositions.
- Truth Values: Each proposition has a truth value of either true (T) or false (F).



Example:

- P: "It is raining."
- Q: "The ground is wet."

Using logical connectives, we can form complex expressions like P→Q (*If it is raining, then the ground is wet*).

➢ **Applications:**

- *Digital Circuit Design*: Representing and analysing the behaviour of logic gates and circuits where each component can be represented as a true/false variable, and the overall circuit can be analysed using *logical connectives*.
- *Expert Systems:* Encoding simple rules and facts for decision-making systems.
- *Truth Tables:* Evaluating the truth values of logical expressions based on various combinations of input values.

B. First-order logic: *First-order logic, or predicate logic, extends propositional logic by introducing predicates and quantifiers, allowing us to express more complex statements.* First-order logic *cannot represent higher-order concepts*, such as statements about other statements or sets of objects.

The basic components of first-order logic include:

* *Constants*: Specific objects in the domain (e.g., Alice, Bob).
* *Variables*: Symbols that can represent any object in the domain (e.g., x, y).
* *Predicates*: Functions that map objects to truth values (e.g., Likes (Alice, Ice Cream)).
* *Quantifiers*: Symbols that indicate the scope of a statement (e.g., ∀ (for all), ∃ (exists)).
* *Logical Connectives*: Same as in propositional logic.

Basic Elements of First Order Logic

- Constants KingJohn, 2, NUS,...
- Predicates Brother, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives ¬, ⇒, ∧, ∨, ⇔
- Equality =
- Quantifiers ∀, ∃

➢ **Applications:**

* Ontology Modelling: Representing knowledge about categories, properties, and relationships between concepts in a domain.

* Semantic Web**:** Encoding information about web resources and their relationships to enable intelligent searching and data integration.

* Automated Reasoning: Developing systems that can reason about knowledge, make inferences, and answer queries based on a set of axioms and rules.

* Natural Language Processing: Understanding and generating human language by modelling the relationships and properties of words and sentences.

C. Key Differences Summarized:

Feature	**Propositional Logic**	**First-Order Logic**
Basic Unit	Propositions	Predicates, constants, variables
Expressiveness	Limited to true/false statements. Not able to represent natural language statements	Expressive, like natural language it can represent objects relations and functions.
Quantifiers	None	Universal (∀) and Existential (∃)
Syntax	Combines propositions using logical connectives	Uses predicates and quantifiers
Semantics	Truth tables	Interpretation over a domain
Inference Algorithms	DPLL, GSAT Fast in practice	Unification, forward chaining, backward chaining, Prolog, theorem proving.
Use Cases	Simple problems (e.g., circuit design, rule-based systems)	Complex problems (e.g., AI reasoning, ontology modeling)
Example	P→Q	$\forall x \exists y (Likes(x, y))$

1.4 Unification

Unification is a process of making two different logical atomic expressions identical by finding a substitution. *"It takes two literals as input and makes them identical using substitution"*. **Unification** is a key component of all first-order Inference algorithms**.**

Let Ψ_1 and Ψ_2 be two atomic sentences and σ be a unifier such that, $\Psi_1\sigma = \Psi_2\sigma$, then it can be expressed as UNIFY (Ψ_1, Ψ_2).

$$UNIFY\ (p, q) = \theta \text{ where } SUBST\ (\theta, p) = SUBST\ (\theta, q)$$

θ *is our Most General Unifier (MGU) value (if one exists).*

P (x, y) ..... (i)

P (a, f(z)) ..... (ii)

Substitute x with a, and y with f(z) in the first expression, and it will be represented as **a/x** and **f(z)/y.**

With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: **[a/x, f(z)/y]**.

Following are some basic conditions for unification:

- Predicate symbol must be same.
- Number of Arguments in both expressions must be identical.
- Unification will fail if there are two similar variables present in the same expression.

1.5 Lifting

Lifting is a concept that extends the utility of unification, enabling more flexible and abstract handling of knowledge representations in AI. Lifted Inference rule require finding substitutions that make different logical expressions look identical (same). That is unification.

It is often associated with answer set programming (ASP), a logic programming paradigm that serves as a foundation for AI knowledge representation and reasoning.

1.6 Resolution:

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. *Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form.* Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence.

* **Clause**: Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.

* **Conjunctive Normal Form**: A sentence represented as a conjunction of clauses is said to be *conjunctive normal form (CNF).*

This rule is also called the *binary resolution rule* because it only resolves exactly two literals.

➢ **Steps for Resolution:**

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

➢ **The resolution inference rule:**

The resolution rule for first-order logic is simply a lifted version of the propositional rule. Resolution can resolve two clauses if they contain complementary literals, which are assumed to be standardized apart so that they share no variables.

$$\frac{l_1 \vee \ldots\ldots \vee l_{k,} \qquad m_1 \vee \ldots\ldots \vee m_n}{\text{SUBST}(\theta, l_1 \vee \ldots\ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots\ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots\ldots \vee m_n)}$$

Where, l_i and m_j are complementary literals.

1.7 Forward Chaining and Backward Chaining in AI

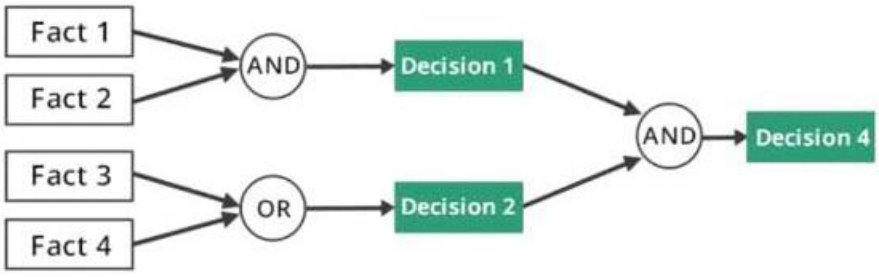
In First-Order Logic, inference is used to derive new facts or sentences from existing ones. To recommend a solution, the Inference Engine (part of expert system) uses the following strategies:

a. Forward Chaining b. Backward Chaining

A. 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. Forward chaining *is a form of reasoning that starts with simple facts in the knowledge base and applies inference rules in the forward direction to extract more data until a goal is reached.*

For example, *prediction of share market* status as an effect of changes in interest rates.

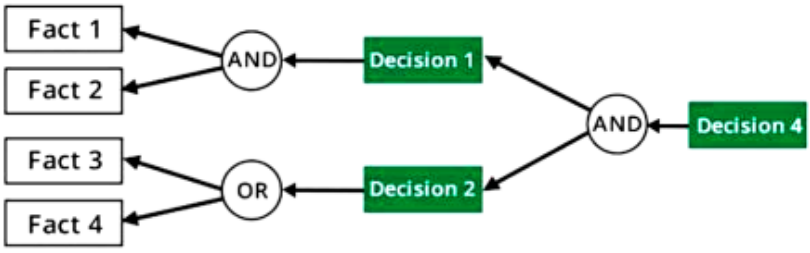


➢ Properties of Forward-Chaining:

- It is a process of making a *conclusion* based on known *facts or data*, by starting from the initial state and reaches the goal state.
- It is a *down-up* approach, as it moves from bottom to top.
- Also called as *data driven* as we reach to the goal using available data.
- It applies the *Breadth-First Strategy*.
- Its goal is to *get the conclusion*.
- It applies inference rules (*Modus Ponens*) in the *forward direction* to extract more data until a goal is reached.
- *Slow* as it must use all the rules.
- Commonly used in the Expert Systems, such as *CLIPS, business, and production rule systems*.

B. Backward Chaining:

- With this strategy, an expert system finds out the answer to the question, *"Why this happened?"*. Based on 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. *This algorithm works backward from the goal, chaining through rules to find known facts that support the proof.* For example, *diagnosis of blood cancer* in humans.



➢ Properties of backward chaining:

- In backward chaining, the goal is broken into *sub-goal* or sub-goals to prove the facts true.
- It is known as a *top-down approach*.
- It is called a *goal-driven approach*, as a list of goals decides which rules are selected and used.

- It operates in *backward direction* i.e. it works from goal to reach initial state.
- The backward-chaining method mostly used a *depth-first search strategy* for proof.
- Its goal is to get the *possible facts,* or the required data based on *Modus Ponens inference rule*.
- *Fast* as it must use only a few rules.
- Backward-chaining algorithm is used in *game theory, automated theorem proving tools, inference engines, proof assistants*, and various AI applications.

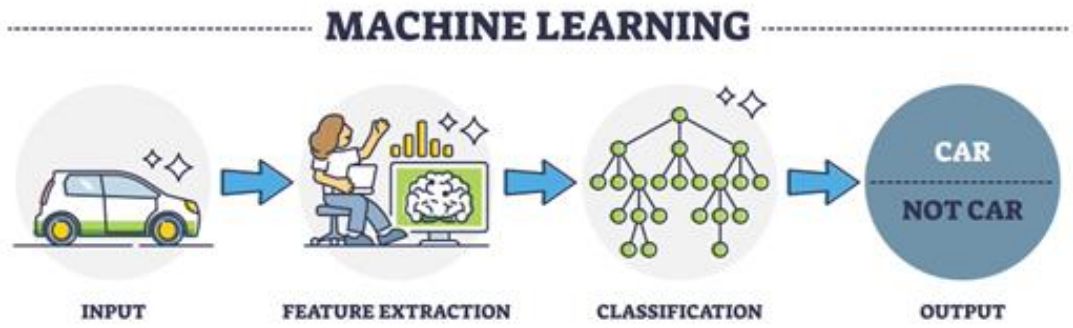
2. Learning

Learning is essential for unknown environments. It modifies the agent's decision mechanisms to improve performance.

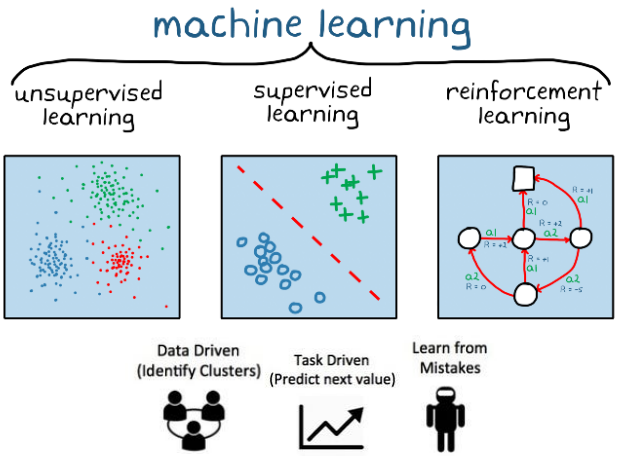
* *"Learning is constructing or modifying representations of what is being experienced"*. (*Michalski, 1986*)
* *"A computer program learns if it improves its performance at some tasks through experience"* (*Mitchell, 1997*).

➢ **Machine Learning:**

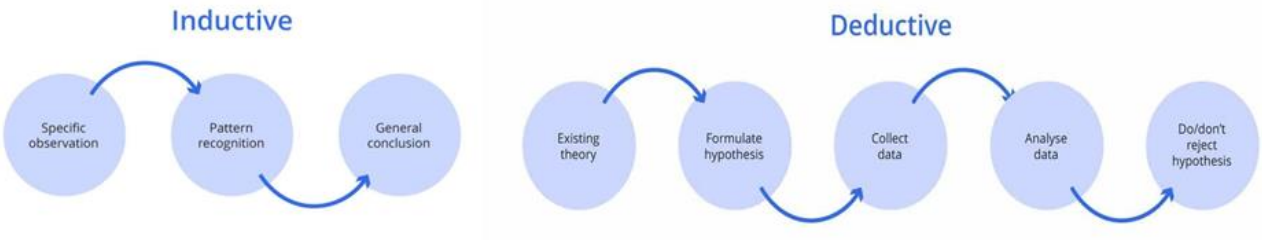
In the field of artificial intelligence known as machine learning, algorithms are developed that can *learn patterns and make judgments or predictions without being explicitly programmed.*



➢ **Categorization of Machine Learning Algorithm:**



- The two main methods used in machine learning are:



Aspect	Inductive Learning	Deductive Learning
Start with	Observations	A theory
Conclude with	A theory	Observations
Learning approach	Bottom-up	Top-down
Data need	Many specific examples	Established general rules
Generalization	Good at finding patterns	Excels at precise predictions
Handling Noise	Can be affected by noise	More robust with solid rules
Interpretation	Less clear due to data patterns	Easier to interpret, explicit rules
Thinking Skills	Creative, critical, inductive	Analytical, deductive
Applications	Real-life problem-solving	Abstract concept application
Reasoning	Goes from specifics to general	Goes from general to specific

2.1 Observational learning

Observational learning is an important concept in artificial intelligence (AI). AI systems can learn by observing the behaviour of humans or other AI systems. *It is a process where humans or animals learn from each other's actions, gestures, and behavior.*

There are four key elements that are involved in observational learning:

1. Attention – the individual needs to pay attention to the behavior that is being demonstrated.
2. Retention – the individual needs to remember or retain the information or behavior they have observed.
3. Reproduction – the individual needs to be able to reproduce the behavior that they have observed.
4. Motivation – the individual needs to be motivated to reproduce the behavior that they have observed.

For example, an AI system can learn to play a game by observing the behavior of a human player. The AI system can observe the human's movements, decision-making process, and strategies, and then use that information to improve its own performance in the game.

2.2 Inductive Learning Algorithm (ILA)

Inductive Learning Algorithm (ILA) is an *iterative and inductive machine learning algorithm that is used for generating a set of classification rules, which produces rules of the form* "IF-THEN", for a set of examples, producing rules at each iteration and appending to the set of rules.

- **Advantages:**
  - * Inductive learning models are flexible and adaptive, they are well suited for handling difficult, complex, and dynamic information.
  - * Finding hidden patterns and relationships in data.
  - * They can efficiently handle huge volumes of data.
  - * Inductive learning models may learn from examples without explicit programming where the rules are ambiguous
- **Disadvantages:**
  - * May overfit to data
  - * computationally costly
  - * Limited interpretability
  - * If the data is inaccurate or inadequate, the model may not perform effectively.
- **Applications:** Inductive learning algorithms are used in a variety of applications, including *credit risk assessment, disease diagnosis, face recognition, and autonomous driving.*

➢ **Illustration with an Example:**

Example No.	Place type	weather	location	decision
1.	hilly	winter	kullu	Yes
2.	mountain	windy	Mumbai	No
3.	mountain	windy	Shimla	Yes
4.	beach	windy	Mumbai	No
5.	beach	warm	goa	Yes
6.	beach	windy	goa	No
7.	beach	warm	Shimla	Yes

✈ **Subset – 1**

Example No.	Place type	weather	location	decision
1.	hilly	winter	kullu	Yes
2.	mountain	windy	Shimla	Yes
3.	beach	warm	goa	Yes
4.	beach	warm	Shimla	Yes

✈ **Subset – 2**

Example No.	Place type	weather	location	decision
5.	mountain	windy	Mumbai	No
6.	beach	windy	Mumbai	No
7.	beach	windy	goa	No

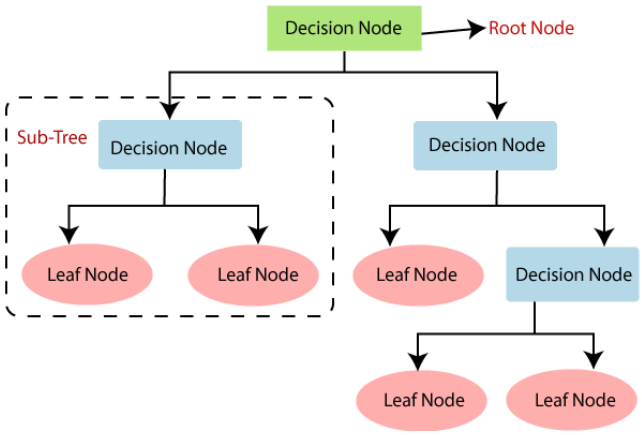
✈ **Finally, we get the Rule Set:**

* Rule 1: IF the weather is warm THEN the decision is yes.
* Rule 2: IF the place type is hilly THEN the decision is yes.
* Rule 3: IF the location is Shimla, THEN the decision is yes.
* Rule 4: IF the location is Mumbai, THEN the decision is no.
* Rule 5: IF the place type is beach AND the weather is windy THEN the decision is No

2.3 Decision Tree Learning:

A Decision tree is a *tree-like structure that represents a set of decisions and their possible consequences*. Each node in the tree represents a decision, and each branch represents an

outcome of that decision. The leaves of the tree represent the final decisions or predictions. Decision trees are used in various fields such as *machine learning, data mining, and statistics*.



➢ **Structure of a Decision Trees:**

* Root Node: The decision tree's starting node, which represents the complete dataset.
* Leaf Node: Leaf nodes are the final output node, and the tree cannot be divided further after getting a leaf node.
* Branches/Sub Tree: A tree formed by splitting the tree.
* Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
* Splitting Criteria**:** Metrics like information gain, entropy, or the Gini Index are used to calculate the optimal split.
* Decision Rules: Rules that govern the splitting of data at each branch node.
* Attribute Selection: The process of choosing the most informative attribute for each split.
* Pruning**:** Pruning is the process of removing the unwanted branches from the tree.

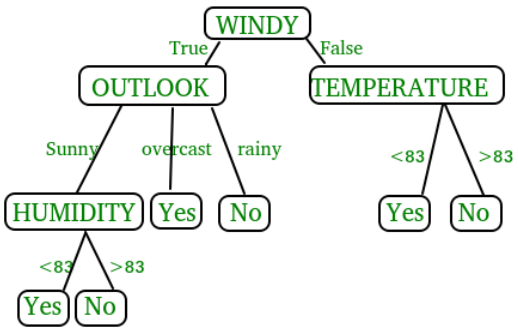
➢ **How does the Decision Tree algorithm Work?**

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

➢ **Illustration with an example:**

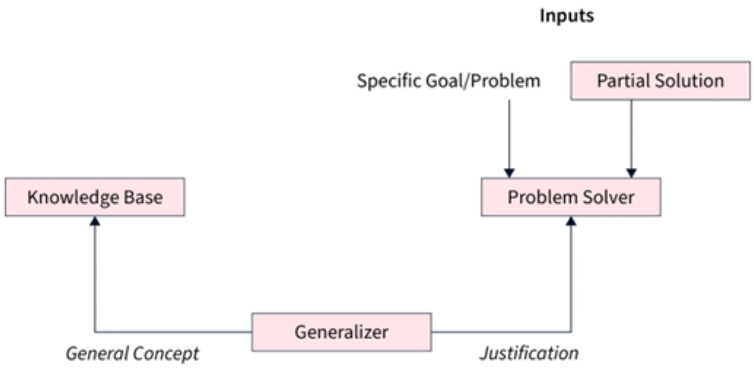
This decision tree classifies Saturday mornings according to whether they are suitable for playing tennis.

For example, the instance (Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong) would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance i.e., the tree predicts that Play Tennis = **no**.



2.4 Explanation-Based Learning

Explanation-based learning in artificial intelligence is *a problem-solving method that involves agent learning by analysing specific situations and connecting them to previously acquired information*. EBL algorithms incorporate *logical reasoning* and *domain knowledge* to make predictions and identify patterns.



➢ **Key Characteristics of EBL:**

i. Use of Domain Knowledge: This knowledge helps the system to generalize the learned concept to new, similar situations.

ii. Focused Learning: EBL focuses on understanding the essential features of an example that are necessary to achieve a goal or solve a problem.

iii. Efficiency: Since EBL can learn from a single example, it is computationally efficient compared to other learning methods that require large datasets for training.

➢ **How Explanation-Based Learning Works?**

Explanation-Based Learning follows a systematic process that involves the following steps:

1. Input Example: The learning process begins with a single example that the system needs to learn from. This example is typically a positive instance of a concept that the system needs to understand.
2. Domain Knowledge: The system uses domain knowledge, which includes rules, concepts, and relationships relevant to the problem domain. This knowledge is crucial for explaining why the example is valid.
3. Explanation Generation: The system generates an explanation for why the example satisfies the concept. This involves identifying the relevant features and their relationships that make the example a valid instance.
4. Generalization: Once the explanation is generated, the system generalizes it to form a broader concept that can apply to other similar examples. This generalization is typically in the form of a rule or a set of rules that describe the concept.
5. Learning Outcome: The outcome of EBL is a generalized rule or concept that can be applied to new situations. The system can now use this rule to identify or solve similar problems in the future.

➢ Advantages of Explanation-Based Learning

- Efficiency in Learning: EBL can learn effectively from a single example, making it efficient in situations where data is scarce or expensive to obtain.
- Understanding and Generalization: EBL focuses on understanding the rationale behind examples, leading to more robust generalizations that can be applied to a wide range of situations.
- Interpretable Models: The rules or concepts learned through EBL are often more interpretable than those learned through other methods, making it easier to understand and trust the system's decisions.

➢ Challenges and Limitations

- Dependency on Domain Knowledge: EBL relies heavily on accurate and comprehensive domain knowledge. If the domain knowledge is incomplete or incorrect, the system may generate flawed explanations and generalizations.
- Limited to Well-Defined Problems: EBL is most effective in well-defined problem domains where the rules and relationships are clear. It may struggle in more complex or ambiguous domains.

• Complexity of Explanation Generation: Generating explanations can be computationally intensive, especially in domains with complex relationships and many features.

➢ **Applications**

Explanation-based learning in the artificial intelligence approach has a wide range of applications, from medical diagnosis to fraud detection.

2.5 Statistical Machine Learning

Statistical Machine Learning *involves using statistical techniques to develop models that can learn from data and make predictions or decisions.*

Popular statistical machine learning techniques are:

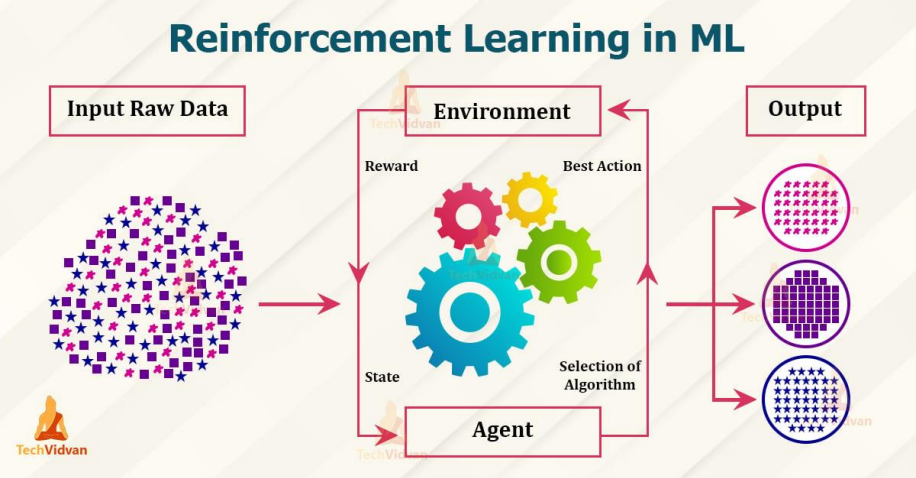
1. **Linear Regression:** It is also seen as a *supervised learning algorithm that captures the connection between a dependent variable and independent variables.*
   Statistics assist in estimating coefficients, conducting hypothesis tests, and evaluating the significance of the relationships, providing valuable insights and a deeper understanding of the data.
2. **Logistic Regression:** Logistic regression is a *statistical classification algorithm that estimates the probability of categorical outcomes based on independent variables*. By applying a logistic function, it predicts the occurrence of a particular class.
3. **Decision-Tree:** Decision trees are versatile algorithms that *use statistics to split data based on features, creating a tree-like structure for classification or regression*. They are intuitive, interpretable, and handle categorical and numerical data. Statistics-based measurements, such as *Gini impurity or information gain*, are often incorporated to guide the splits throughout the tree construction process.
4. **Random forest:** Random Forest is an *ensemble learning method that improves prediction accuracy by combining multiple decision trees*. It employs *sampling* to randomly select subsets of features and data for building the trees. The predictions of these individual trees are then aggregated to make the final prediction.
5. **SVM:** SVM is a powerful algorithm that can be used for *classification and regression* tasks. It uses statistical principles *to create a boundary between different groups of data points*, making it easier to tell them apart. By optimizing this boundary, SVM reduces the chances of making mistakes and improves overall accuracy.

6. **K-Nearest neighbour KNN:** KNN is a simple, yet effective algorithm used for *classifying data points based on the majority vote of their nearest neighbors.* It is suitable for both classification and regression problems and does not require training.

   In KNN, statistical measures are utilized to determine the proximity between data points, helping to *identify the nearest neighbors*. The majority vote of the nearest neighbors is then used to classify or predict the target variable.

2.6 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning focused on making decisions to maximize cumulative rewards in each situation. RL involves learning through experience. In RL, an agent learns to achieve a goal in an uncertain, potentially complex environment by performing actions and receiving feedback through rewards or penalties.

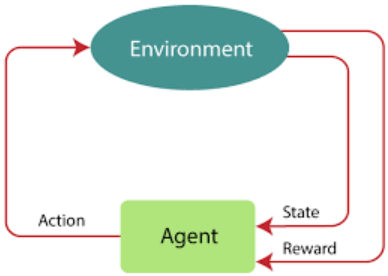


➢ Key Concepts of Reinforcement Learning:

* Agent: The learner or decision-maker.
* Environment: Everything the agent interacts with.
* State: A specific situation in which the agent finds itself.
* Action: All possible moves the agent can make.
* Reward: Feedback from the environment based on the action taken.

➢ How Reinforcement Learning Works

RL operates on the *principle of learning optimal behavior through trial and error*. The agent takes actions within the environment, receives rewards or penalties, and adjusts its behavior to maximize the cumulative reward.



This learning process is characterized by the following elements:

* Policy: A strategy used by the agent to determine the next action based on the current state.
* Reward Function: A function that provides a scalar feedback signal based on the state and action.
* Value Function: A function that estimates the expected cumulative reward from a given state.
* Model of the Environment: A representation of the environment that helps in planning by predicting future states and rewards.

➢ **Main points in Reinforcement learning –**

* Input: The input should be an initial state from which the model will start

* Output: There are many possible outputs as there are a variety of solutions to a particular problem.

* Training: The training is based upon the input; The model will return a state, and the user will decide to reward or punish the model based on its output.

* The model keeps continues to learn.

* The best solution is decided based on the maximum reward.

➢ **Advantages:**

1. Reinforcement learning can be used to solve very complex problems that cannot be solved by conventional techniques.
2. The model can correct the errors that occurred during the training process.
3. In RL, training data is obtained via the direct interaction of the agent with the environment
4. This is useful in real-world applications where the environment may change over time or is uncertain.
5. Reinforcement learning can be used to solve a wide range of problems, including those that involve decision making, control, and optimization.
6. Reinforcement learning is a flexible approach that can be combined with other machine learning techniques, such as deep learning, to improve performance.

➢ **Disadvantages:**

1. Reinforcement learning is not preferable to use for solving simple problems.
2. Reinforcement learning needs a lot of data and a lot of computation.
3. In RL, If the reward function is poorly designed, the agent may not learn the desired behavior.
4. Reinforcement learning can be difficult to debug and interpret.

2.6.1 Types of Reinforcement:

1) Positive: Positive Reinforcement is defined as when an event, occurs due to a particular *behavior, increases the strength and the frequency* of the behavior. In other words, it has a positive effect on behavior.

➢ **Advantages:**

* Maximizes Performance
* Sustain Change for a long period of time.
* Too much Reinforcement can lead to an overload of states which can diminish the results.

2) Negative: Negative Reinforcement is defined as *strengthening of behavior because a negative condition is stopped or avoided.*

➢ **Advantages:**

* Increases Behavior.
* Provide defiance to a minimum standard of performance.
* It Only provides enough to meet up the minimum behavior.

2.6.2 Applications of Reinforcement Learning:

i) Robotics: Automating tasks in structured environments like manufacturing.

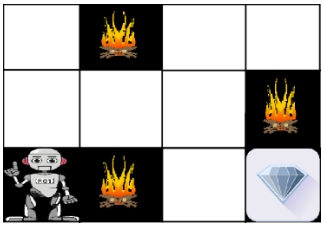
ii) Game Playing: Developing strategies in complex games like chess.

iii) Industrial Control: Real-time adjustments in operations like refinery controls.

iv) Personalized Training Systems: Customizing instruction based on individual needs.

2.6.3 Illustration with Example: Navigating a Maze

The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.



The above image shows the robot, diamond, and fire. *The goal of the robot is to get the reward that is the diamond and avoid the hurdles that are fired.* The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least hurdles. *Each right step will give the robot a reward and each wrong step will subtract the reward of the robot.* The total reward will be calculated when it reaches the final reward that is the diamond.

2.7 Comparison Table

Criteria	Supervised ML	Unsupervised ML	Reinforcement ML
Definition	Learns by using labelled data	Trained using unlabeled data without any guidance.	Works on interacting with the environment
Type of data	Labelled data	Unlabeled data	No – predefined data
Type of problems	Regression and classification	Association and Clustering	Exploitation or Exploration
Supervision	Extra supervision	No supervision	No supervision
Algorithms	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori	Q – Learning, SARSA
Aim	Calculate outcomes	Discover underlying patterns	Learn a series of action
Application	Risk Evaluation, Forecast Sales	Recommendation System, Anomaly Detection	Self-Driving Cars, Gaming, Healthcare

Review Questions:

PART-A (TWO MARKS)

1. Mention the two fundamental logics in AI with any two key differences. *Refer 1.3(C)*
2. Write the basic elements of FOL syntax. *Refer 1.1*
3. State Generalized Modus Ponen's rule with representation. *Refer 1.2(C)*
4. Define Unification and Lifting.*Refer1.4,1.5*
5. Write the Notation and Steps for Resolution. *Refer 1.6*
6. List the various Learning Methods in AI. *Refer 2*
7. Draw the schematic diagram for Decision tree and Explanation based learning. *Refer 2.3,2.4*
8. Define Observational learning and list the key elements. *Refer 2.1*
9. Give the diagrammatic representation for the Inductive Learning Algorithm with an example. *Refer 2.2*
10. Write the key concepts of Reinforcement Learning. *Refer 2.6*

PART-B (TEN MARKS)

1. Define Logic in AI. Classify FOL and Propositional logic. *Refer 1.3*
2. Explain in detail about First order logic. *Refer 1.2*
3. What is inference in FOL? Discuss FOL Inference rules for quantifiers. *Refer 1.2*
4. Distinguish Forward and Backward chaining. *Refer 1.7*
5. Demonstrate briefly about various Learning Algorithms. *Refer 2*
6. Define Inductive Learning? Explain with an example. *Refer 2.2*
7. Elucidate the structure and working of Decision Trees with an example. *Refer 2.3*
8. Write the characteristics and working of Explanation-Based Learning. *Refer 2.4*
9. Discuss about various methods in Statistical Learning. *Refer 2.5*

10. Illustrate the working of Reinforcement Learning. Formulate Navigating a Maze problem. *Refer 2.6,2.6.3*

LECTURE PLAN

UNIT V

S.No.	Topic	No. of Periods	Proposed lecture	Actual lecture	Pertaining CO(s)	Taxonomy Level	Mode of Delivery
			Period	Period			
1	Architecture of expert systems	1			CO4	L2	MD1, MD5
2	Roles of expert systems	1			CO4	L2	MD1, MD5
3	Knowledge Acquisition	1			CO4	L2	MD1, MD5
4	Meta knowledge Heuristics	1			CO4	L1	MD1, MD5
5	Typical expert systems–MYCIN	1			CO4	L1	MD1, MD5
6	DART	1			CO4	L1	MD1, MD5
7	XCON: Expert systems shells.	1			CO4	L1	MD1, MD5

UNIT - V

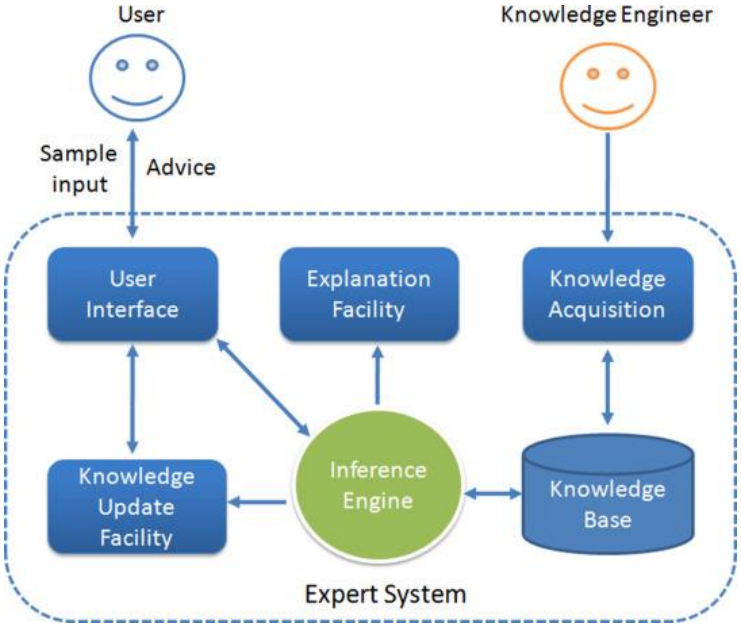
Expert Systems: Architecture of expert systems, Roles of expert systems – Knowledge Acquisition Meta knowledge Heuristics. Typical expert systems – MYCIN, DART, XCON: Expert systems shells.

1. Introduction

An expert system is an *AI software that is designed to solve complex problems by using knowledge from its knowledge base to provide decision-making ability like a human-expert.* The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

1.1 Architecture of an Expert System

The process of building an expert system is often called **knowledge engineering.**



- **Key Components of Expert System:**

The components of an expert system typically include the knowledge base, inference engine with explanation module, user interface, and learning module.

a) **Knowledge Base:**

The knowledge base stores/represents facts and rules. It consists of knowledge in a particular domain as well as rules to solve a problem, procedures and intrinsic data relevant to the domain.

1. Factual Knowledge – facts about a specific subject domain and it is accepted by Knowledge Engineers (for example, Ramesh is an analyst).
2. Heuristic Knowledge (*rules of thumb*) – This knowledge is based on experience, the capacity to guess, and evaluation. (for example, *IF Ramesh is an analyst, THEN he needs a workstation*).

b) Inference Engine (Rules of Engine):

It applies inference rules to the knowledge base to derive conclusions or recommendations. With the help of an inference engine, the system extracts the knowledge from the knowledge base.

- **Inference engines are divided into two types:**
  - Deterministic Inference Engine: In this form of an inference engine, the conclusions are presumed to be correct. It is founded on *facts* and regulations.
  - Probabilistic Inference Engine: This sort of inference engine is based on probability and contains *uncertainty* in findings.
- **The following modes are used by the inference engine to generate solutions:**
  i. Forward Chaining: This is a data-driven reasoning approach where the system starts with available data and applies rules to deduce new information until a conclusion is reached or no further inferences can be made.
  ii. Backward Chaining: This is a goal-driven reasoning approach where the system starts with a goal or conclusion and works backward to find evidence or data supporting that goal. It is useful for diagnostic systems that need to identify the causes of observed symptoms.

c) User Interface

It is an interface that helps a non-expert user to interact with the expert system to find a solution. It takes queries as an input in a readable format and passes it to the inference engine. Example**:** Text-Based interface, Graphical User Interface (GUI)

d) Knowledge Acquisition:

The function of this component is to allow the expert system to acquire more and more knowledge from various sources and store it in the knowledge base.

e) Learning Module (Optional):

Some expert systems incorporate a learning mechanism to improve performance. Examples include supervised, unsupervised, reinforcement learning.

f) **Explanation Module:**

This module helps the expert system to give the user an explanation about how the expert system reached a particular conclusion.

1.2 How Expert Systems Work?

Expert systems operate by following a structured approach:

1. Input Data: Users provide data or *queries* related to a specific problem or scenario.
2. Processing: The *inference engine* processes the input data using the rules in the knowledge base to generate conclusions or recommendations.
3. Output: The system presents the results or solutions to the user through the *user interface.*
4. Explanation: If applicable, the system explains how the *conclusions* were reached, providing insights into the *reasoning* process.

1.3 Advantages of Expert Systems

1. **Consistency in Decision-Making**: Expert systems apply the same set of rules to every problem.
2. **24/7 Availability**: Expert systems can operate continuously, offering advice or making decisions anytime.
3. **Efficient Problem Solving**: They can process large amounts of data quickly and reach decisions faster than a human expert.
4. **Cost-Effective**: Once developed, an expert system can be a cost-effective solution for organizations, as it reduces the need for constant human intervention.
5. **Knowledge Preservation**: Expert systems capture and store human expert knowledge, which can be used over time.

1.4 Disadvantages of Expert Systems

1. **Lack of Common Sense**: Expert systems lack the ability to apply common sense or judgment to unusual or novel situations.
2. **Inability to Learn and Adapt**: Traditional expert systems do not learn from new experiences or evolve over time. Their knowledge base must be manually updated, which can be labour-intensive and slow.
3. **High Initial Development Costs**: Building and implementing an expert system can require significant upfront investment.
4. **Dependence on Expertise**: engineers are needed for the design and development of the

Expert system,

5. **Limited Scope**: They are not suitable for handling tasks that require human intuition, creativity, or emotional intelligence.

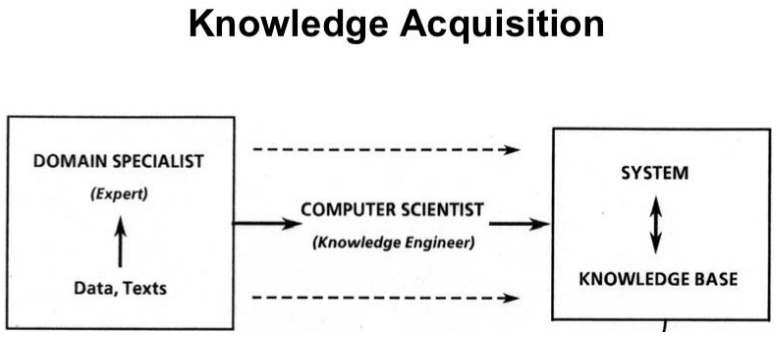
1.5 Applications of Expert Systems

1. **Medical Diagnosis**: Expert systems assist doctors by analyzing symptoms and medical history to suggest possible diagnoses or treatment options. For example, MYCIN, an early expert system, helped identify bacterial infections and recommend antibiotics.
2. **Financial Services**: In finance, expert systems are used for credit scoring, fraud detection, and investment advice. They analyze financial data and patterns to make informed decisions.
3. **Technical Support**: Expert systems can troubleshoot and provide solutions for technical issues. They guide users through problem-solving steps based on pre-defined rules and knowledge.
4. **Manufacturing**: In manufacturing, expert systems help optimize production processes, perform quality control, and manage inventory by analyzing data and making recommendations.

1.6 Roles of expert systems:

1. Knowledge Acquisition

Knowledge acquisition refers to the process of gathering, organizing, and structuring knowledge from human experts or other sources to be used in an expert system. This knowledge is often codified into rules or facts that the system uses to make decisions.



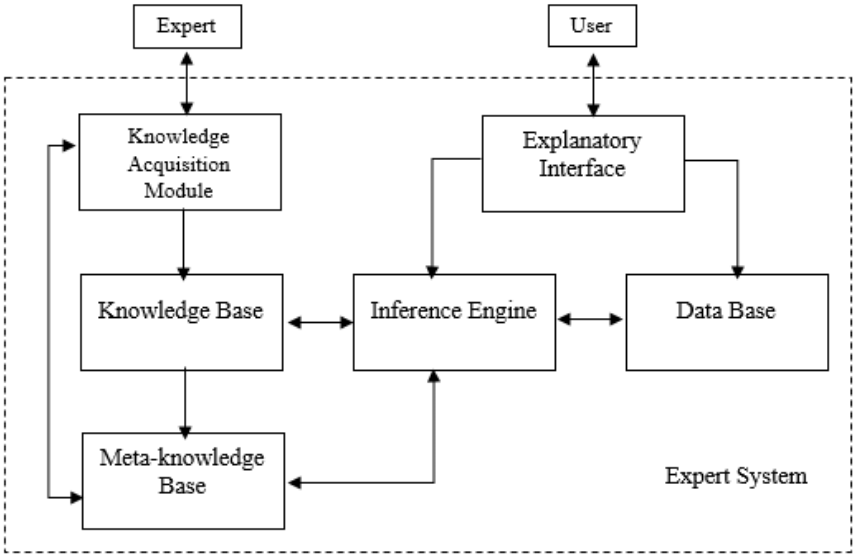
There are two main types of knowledge acquisition:

- **Direct Knowledge Acquisition:** Involves directly interviewing experts, reading literature, or analysing databases to extract relevant information.
- **Indirect Knowledge Acquisition:** Uses automated tools, machine learning, or other AI techniques to derive knowledge patterns from data or experiences.

The quality of the knowledge acquisition process directly affects the system's performance and accuracy.

2. Meta-Knowledge

Meta-knowledge is "knowledge about knowledge." In expert systems, meta-knowledge refers to the understanding of how the knowledge base is structured and how to use it efficiently.



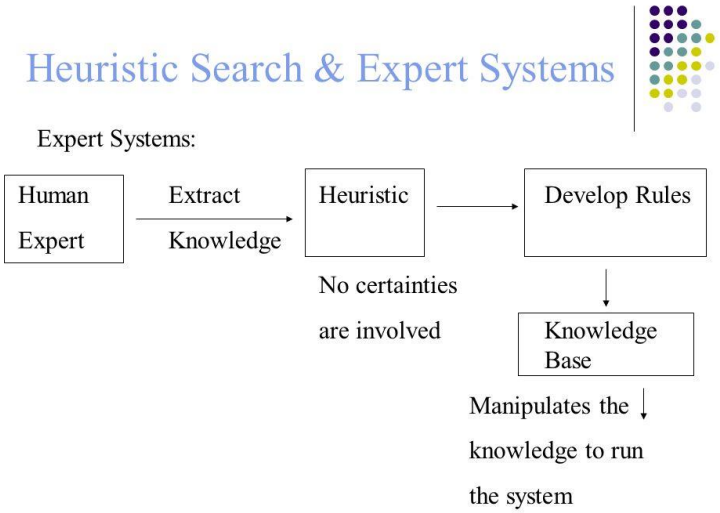
It involves knowing:

- How and when to apply specific rules.
- The strategies or methods to search and infer the best solution.
- The level of confidence in certain rules or pieces of knowledge.

Meta-knowledge helps an expert system manage uncertainty, optimize performance, and prioritize actions. It provides context and reasoning guidelines for the system's knowledge.

3. Heuristics

Heuristics are rules of thumb or practical shortcuts that help in making decisions more efficiently. They are not guaranteed to produce the optimal solution but often lead to good enough solutions faster.



In expert systems, heuristics allow for:

- **Efficient Problem Solving:** By reducing the search space or focusing on the most likely solutions.
- **Decision-Making:** Where complex reasoning may not be feasible or where time constraints exist.
- **Human-Like Reasoning:** Mimicking how human experts use experience or intuition to solve problems.

Expert systems use heuristics to simplify problem-solving processes, especially in situations where complete information or resources may not be available.

1.6.1 Relationship Between Meta-Knowledge and Heuristics in Expert Systems

In expert systems, meta-knowledge and heuristics often work hand in hand. Meta-knowledge can be used to decide when and how to apply certain heuristics to solve a problem. For instance:

- **Heuristics:** Practical rules of thumb that simplify and speed up decision-making.
- **Meta-Knowledge:** Acts as a supervisory layer, guiding the inference engine in using the knowledge base and heuristics effectively.

In this representation:

- The **inference engine** is at the core, interacting with both **knowledge** and **heuristics**.
- **Meta-knowledge** monitors and optimizes the whole process, ensuring correct usage of both rule-based knowledge and heuristic shortcuts.

❖ **Examples of Meta-Knowledge and Heuristics in Expert Systems**

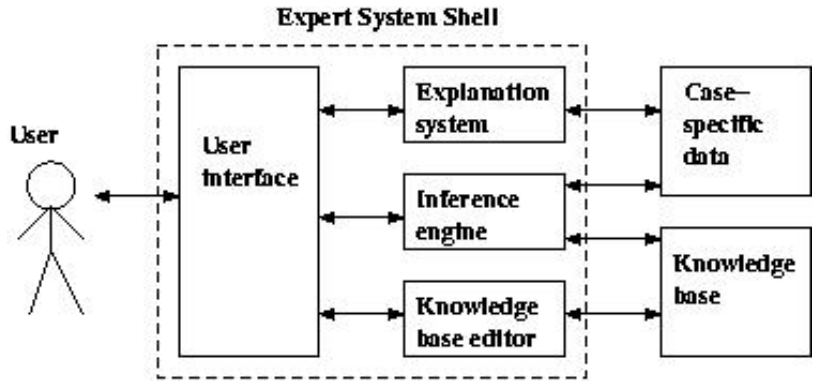
- MYCIN (Medical Diagnosis): An early expert system used in medicine, MYCIN employed heuristic rules for diagnosing bacterial infections and prescribing

treatments. Its meta-knowledge allowed it to assess the certainty of its conclusions, providing users with confidence scores based on the available evidence.

- XCON (Computer Configuration): XCON was used by Digital Equipment Corporation to configure computer systems. It utilized heuristics for selecting compatible hardware components based on user requirements and meta-knowledge to manage conflicts between different rules.

1.7 Expert Systems/Shells:

The E.S shell simplifies the process of creating some knowledgebase. It is the shell that processes the information entered by a user relates it to the concepts contained in the knowledge base and provides an assessment or solution for a particular problem.



❖ Key Components of Expert System Shells:

1. **Knowledge Base:**
   - Stores facts and rules related to the domain of expertise.
   - The user inputs domain-specific knowledge in the form of **if-then rules**, frames, or objects.
2. **Inference Engine:**
   - The core component that applies logical rules to the knowledge base to deduce new information or make decisions.
   - Two main reasoning methods:
     - **Forward chaining:** Starts with known facts and applies rules to infer new facts until a goal is reached.
     - **Backward chaining:** Starts with a goal and works backward to see if there is evidence to support that goal.
3. **User Interface:**
   - Provides interaction between the user and the system, typically through a command-line or graphical interface.

- Allows users to input queries and receive explanations or advice.

4. **Explanation Facility:**
   - Explains how the system reached a particular conclusion or decision, helping users understand the reasoning process.
5. **Knowledge Acquisition Tool:**
   - Assists in adding or updating the knowledge base, typically through a guided interface for non-programmers.

❖ **Advantages of Using Expert System Shells:**

- **Pre-built Components:** Saves time by providing ready-made modules for inference and knowledge management.
- **No Need for Low-Level Programming:** Users can focus on entering domain knowledge without dealing with complex coding.
- **Flexibility:** Can be adapted to various domains, including medical diagnosis, financial decision-making, and engineering.

❖ **Limitations:**

- **Knowledge Engineering Required:** Still requires expert knowledge to define rules or facts.
- **Performance Issues:** Large knowledge bases can slow down processing in complex systems.
- **Lack of Creativity:** Shells are limited to the rules and knowledge provided, lacking the ability to "think" beyond them.

❖ **Examples of Expert System Shells:**

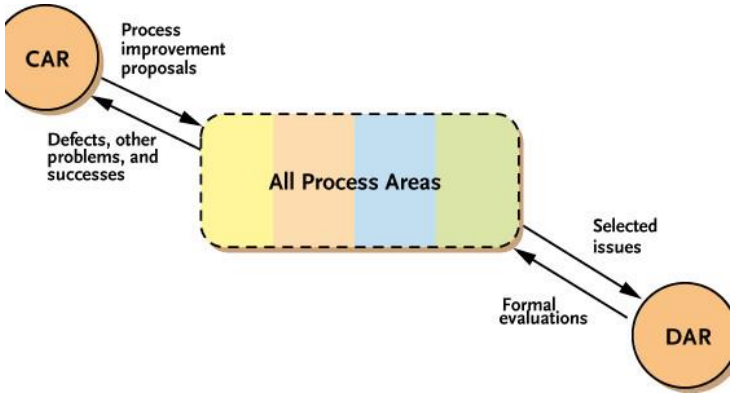
1. **CLIPS (C Language Integrated Production System):**
   - An open-source shell designed for developing rule-based expert systems.
   - Supports forward chaining and backward chaining.
2. **EMYCIN (Empty MYCIN):**
   - A shell derived from the MYCIN expert system, used to build medical and other rule-based systems.
3. **Drools:**
   - A business rule management system (BRMS) that serves as an expert system shell for creating decision automation solutions.
4. **JESS (Java Expert System Shell):**
   - A Java-based shell used for building rule-based systems, widely used in academia and industry.

1.8 Typical Expert Systems Shells:

Expert system shells like **DART**, **MYCIN**, and **XCON** represent early advancements in the development of artificial intelligence, particularly in the field of expert systems.

1.8.1 DART (Decision Analysis and Resolution Tool)

- **Purpose:** DART is designed for decision support within expert systems, focusing on multi-criteria decision analysis and resolution. It is more of a methodology or tool than a standalone expert system, often integrated into broader decision support systems or expert system shells.



- **Functionality:**
  - Provides structured decision-making by analyzing different alternatives based on weighted criteria.
  - Supports decision-making processes in domains where multiple options need to be evaluated against performance, risk, and other factors.
- **Domain Applications:**

  DART can be applied in various fields like business management, engineering, finance, and healthcare, where decisions must be made by comparing alternatives and trade-offs.

❖ **Key Features:**

- Multi-criteria decision-making (e.g., cost vs. benefit analysis).
- Integration with inference engines and rule-based systems to automate decision-making.
- Uncertainty handling and what-if analysis.

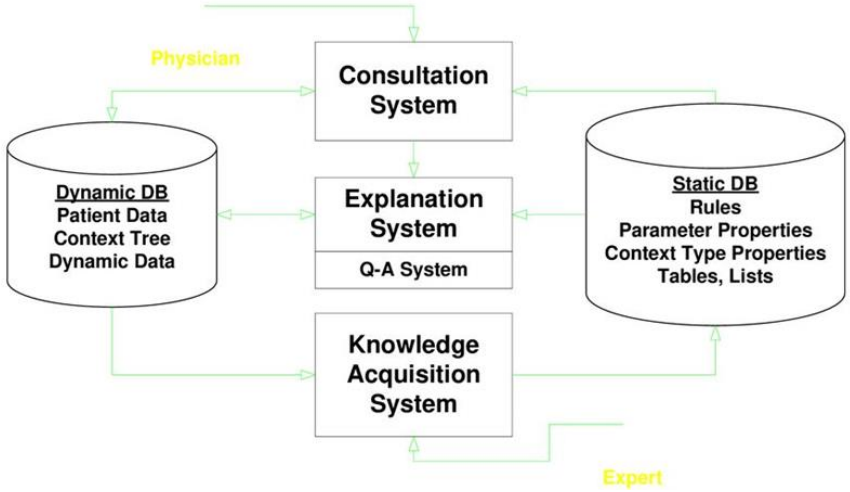
1.8.2 MYCIN

- **Development:** MYCIN was developed in the early 1970s at Stanford University by Edward Shortliffe as a rule-based expert system focused on **medical diagnosis** and **treatment recommendations** for infectious diseases, particularly bacterial infections.

- **Purpose:** It was designed to assist physicians by providing recommendations on which antibiotics to prescribe for patients, based on a set of symptoms and laboratory results.

❖ **Key Components of MYCIN Architecture:**

- **Knowledge Base**: A set of rules encoding domain expertise.
- **Inference Engine**: Performs backward chaining to apply rules to the current problem.
- **User Interface**: Interacts with the user through questions and answers.
- **Explanation System**: Provides the reasoning behind the system's conclusions.
- **Certainty Handling**: Manages uncertain data using certainty factors.



❖ **Key Features:**

- Rule-based inference engine (over 600 rules).
- Certainty factor-based reasoning to handle incomplete or uncertain data.
- Natural language processing for user interaction (though limited by the technology at the time).
- Foundational for many later medical expert systems.

1.8.3 XCON (eXpert CONfigurer)

- **Development:** XCON (originally called R1) was developed by **John McDermott** at **Digital Equipment Corporation (DEC)** in the late 1970s to solve a specific business problem—configuring VAX computer systems.
- **Purpose:** XCON was designed to help DEC's sales and manufacturing teams configure the complex hardware components of their VAX computers, ensuring that they were properly assembled and compatible.

- **Functionality:**
  - XCON is a rule-based system with hundreds of if-then rules, which automates the process of configuring complex hardware systems by ensuring that all the necessary components are included and correctly interconnected.
  - It applies **forward chaining**—starting from the facts provided by the user (e.g., specific hardware parts required) and then working forward through rules to arrive at a configuration solution.
- **Domain Applications:** Primarily used in the **manufacturing** and **sales** domains for configuring complex systems, especially for computer hardware and network design.

❖ **Key Features:**

- Rule-based system designed for complex configuration tasks.
- Forward chaining inference engine to apply knowledge rules to specific facts.
- Used a large set of rules (eventually over 10,000) to cover a wide range of hardware configurations.
- Ensured that systems were error-free and complete, saving DEC time and reducing human errors in configuration.

1.8.4 Comparison of DART, MYCIN, and XCON:

Feature	DART	MYCIN	XCON (R1)
Type	Decision Analysis and Resolution Tool	Rule-Based Medical Expert System	Rule-Based Configuration Expert System
Primary Domain	General Decision Support	Medicine (Infectious Diseases)	Manufacturing and Sales (Computer Systems)
Inference Method	Multi-Criteria Decision Making	Backward Chaining (Rule-Based)	Forward Chaining (Rule-Based)
Handling Uncertainty	Probabilistic and multi-criteria Models	Certainty Factors	Limited (Focuses on Configuration)
Primary Use Case	Decision Optimization	Diagnosis and Treatment of Infections	VAX Computer Configuration
Impact	Enhanced decision-making in various domains	Pioneering medical AI	Successful commercial AI system

Review Questions:

Part-A (Two Marks)

Define Expert system.
Write the advantages and disadvantages of expert systems.
List the components of expert systems.
Define inference engine.
What is knowledge base?
What is meant by knowledge acquisition?
Compare two different expert system shells (e.g., CLIPS and Jess)
Write the functionality of XCON.
Give the architecture of MYCIN in medical diagnosis.
Draw the schematic diagram for expert system system shell.

Part-B (Ten Marks)

With a neat sketch explain the architecture of expert system.
Explain the role of the knowledge base and inference engine in the architecture of an expert system.
Discuss the role of knowledge acquisition in Expert System.
What is meta-knowledge in expert systems? Provide the functionality and examples.
How do heuristics improve the performance of expert systems? Provide examples from real-world applications.
How do meta-knowledge and heuristics work together in real-world scenarios.
Examine the architecture and functionality of the MYCIN expert system in medical diagnosis.
Discuss the role of the DART in expert system.
Compare and contrast MYCIN, DART, and XCON expert systems
Explain the key components of an expert system shell and discuss the benefits and limitations.