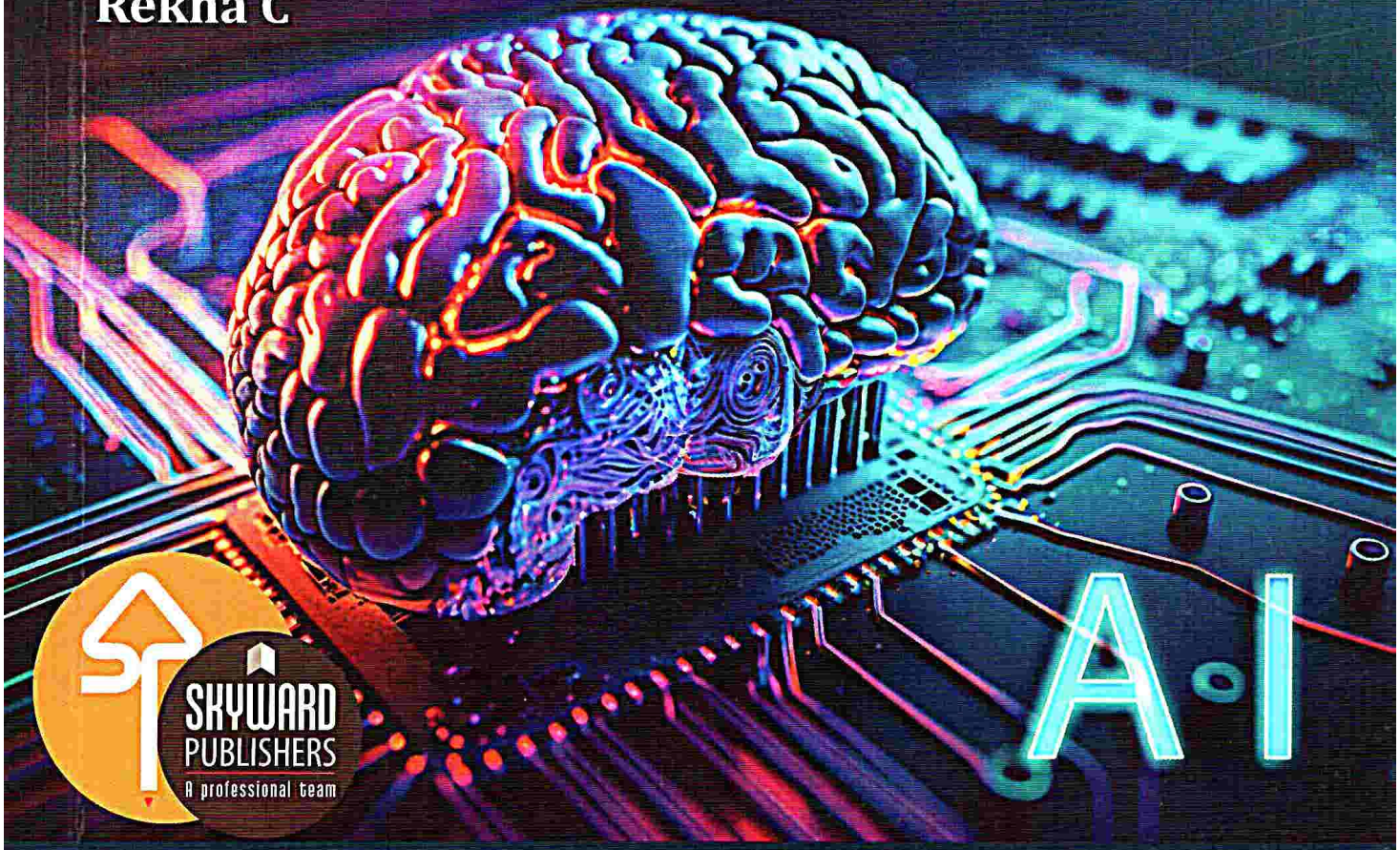
As per the New NEP Syllabus

ARTIFICIAL INTELLIGENCE

As Per the New NEP Syllabus for BCA 5th Semester Course of
Bengaluru City University & Bangalore University

Rekha C



SKYWARD PUBLISHERS
A professional team

ARTIFICIAL INTELLIGENCE

As per the New NEP Syllabus for BCA 5th Semester Course of
Bengaluru City University and Bangalore University

Authored By

Prof. Rekha. C

M.C.A.,M.Phil.,B.Ed.,ADSE(Ph.D), K.SET

HOD in Department of Computer Science
Soundarya Institute of Management and Science,
Bangalore.

Skyward Publishers
#157, 7th Cross, 3rd Main Road, Chamarajpet
Bangalore-18. Phone : 080-26603535 / 43706620,
Mob: 9611185999
E-mail: skyward.publishers@gmail.com
Website: www.skywardpublishers.co.in

Syllabus

Total: 60 Hrs

Unit - I [15 Hours]

Introduction to AI: What is AI? **Intelligent Agents:** Agents and environment, the concept of Rationality, the nature of the environment, the structure of agents; **Problem-solving:** Problem Solving agents; Uninformed search strategies: DFS, BFS; Informed Search: Best First Search, A* search, AO* search, Means End Analysis. Adversarial Search & Games: Two-player zero-sum games, Minimax Search, Alpha-Beta pruning.

Unit - II [15 Hours]

Knowledge-based Agents, The Wumpus world as an example world, Logic, Propositional logic, First-order predicate logic, Propositional versus first-order inference, Unification and lifting, Forward chaining, Backward chaining, Resolution, Truth maintenance systems.

Knowledge in Learning, What is learning? Types of Learning,: Rote Learning, Learning by Taking Advice, Learning in Problem Solving, Learning from Examples, Winston's Learning Program, Decision Trees.

Unit - III [15 Hours]

Introduction to Planning: Blocks World problem, Strips; Handling Uncertainties: Nonmonotonic reasoning, Probabilistic reasoning, Fuzzy logic; **Robotics:** Fundamentals of Robotics, Robot Kinematics; Computer Vision: Introduction to image processing and classification, object detection.

Unit - IV [15 Hours]

Natural Language Processing: Introduction, Syntactic Processing, Semantic Analysis, Discourse and Pragmatic Processing; **Expert Systems:** Architecture and role of expert systems, two case studies of Expert Systems; **Introduction to Machine Learning:** Supervised learning, unsupervised learning, reinforcement learning; **Neural Networks:** Introduction, basics of ANN, Deep Learning with basics of CNN, RNN, LSTM and their applications.

CONTENTS

UNIT - I

CHAPTER 1

INTRODUCTION TO AI

- What Is Intelligence ?
- What is Artificial Intelligence ?
- History of AI
- Artificial Intelligence Disciplines
- Types of Artificial Intelligence
- Advantages of Artificial Intelligence
- Disadvantages of Artificial Intelligence
- Top Technologies used in Artificial Intelligence
- Review Questions

1.1 What Is Intelligence ?

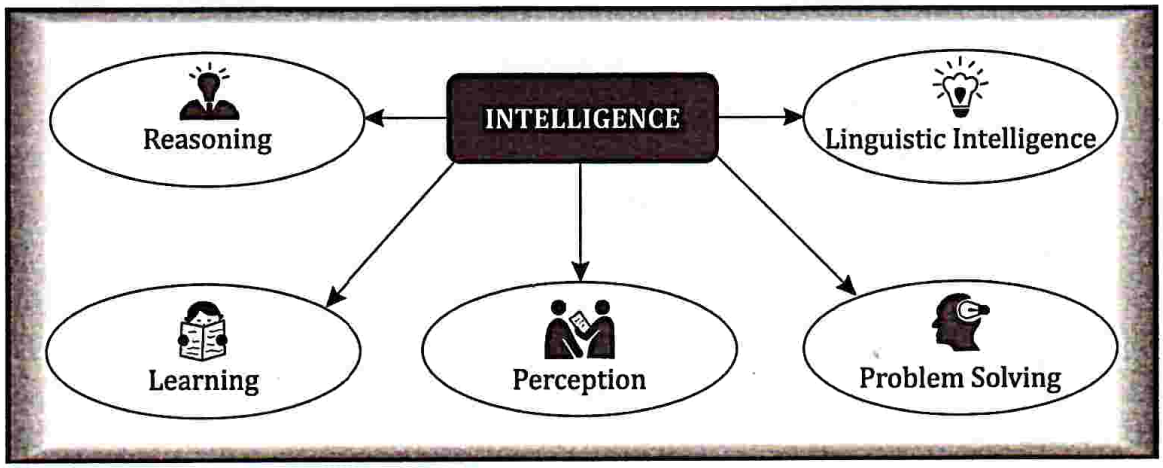
"It is not my aim to surprise or shock you – but the simplest way I can summarize is to say that there are now in the world machines that can think, that learn, and that create. Moreover, their ability to do these things is going to increase rapidly until - in a visible future - the range of problems they can handle will be coextensive with the range to which human mind has been applied." - by Herbert A Simon (1957)

The ability of a system to calculate, reason, perceive relationships and analogies, learn from experience, store and retrieve information from memory, solve problems, comprehend complex ideas, use natural language fluently, classify, generalize, and adapt new situations.

What is Intelligence composed of ?

As described by Howard Gardner, an American developmental psychologist, the Intelligence is intangible. It is composed of –

1. Reasoning
2. Learning
3. Problem Solving
4. Perception
5. Linguistic Intelligence



1. **Reasoning** – It is the set of processes that enables us to provide basis for judgement, making decisions, and prediction. There are broadly two types –

Inductive Reasoning	Deductive Reasoning
It conducts specific observations to makes broad general statements.	It starts with a general statement and examines the possibilities to reach a specific, logical conclusion.
Even if all of the premises are true in a statement, inductive reasoning allows for the conclusion to be false.	If something is true of a class of things in general, it is also true for all members of that class.
Example – "Nita is a teacher. Nita is studious. Therefore, All teachers are studious."	Example – "All women of age above 60 years are grandmothers. Shalini is 65 years. Therefore, Shalini is a grandmother.

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

The ability of learning is possessed by humans, some animals, and AI-enabled systems. Learning is categorized as –

a. **Auditory Learning:** It is learning by listening and hearing. For example, students listening to recorded audio lectures.
b. **Episodic Learning:** To learn by remembering sequences of events that one has witnessed or experienced. This is linear and orderly.
c. **Motor Learning:** It is learning by precise movement of muscles. For example, picking objects, Writing, etc.
d. **Observational Learning:** To learn by watching and imitating others. For example, child tries to learn by mimicking her parent.
e. **Perceptual Learning:** It is learning to recognize stimuli that one has seen before. For example, identifying and classifying objects and situations.
f. **Relational Learning:** It involves learning to differentiate among various stimuli on the basis of relational properties, rather than absolute properties. For Example, Adding 'little less' salt at the time of cooking potatoes that came up salty last time, when cooked with adding say a tablespoon of salt.
g. **Spatial Learning:** It is learning through visual stimuli such as images, colors, maps, etc. For Example, A person can create roadmap in mind before actually following the road.
h. **Stimulus-Response Learning:** It is learning to perform a particular behavior when a certain stimulus is present. For example, a dog raises its ear on hearing doorbell.

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

   Problem solving also includes decision making, which is the process of selecting the best suitable alternative out of multiple alternatives to reach the desired goal are available.

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

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

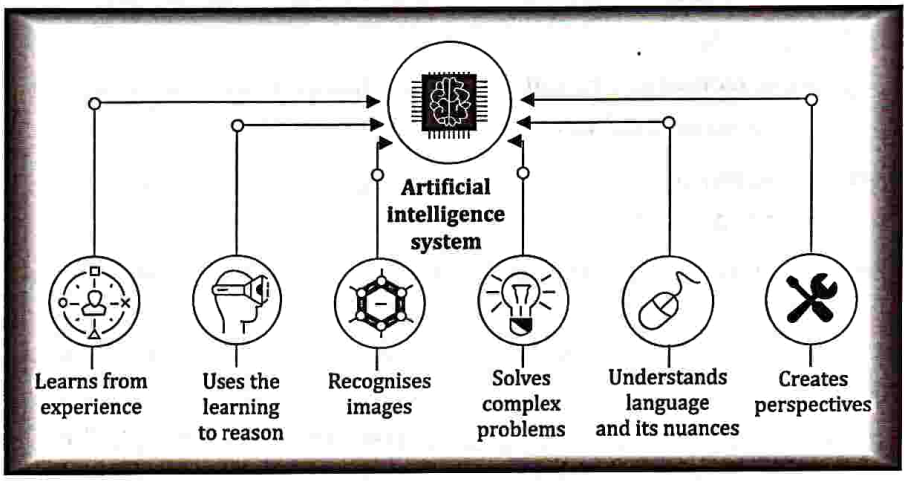
5. **Linguistic Intelligence** – It is one's ability to use, comprehend, speak, and write the verbal and written language. It is important in interpersonal communication.

1.2 What is Artificial Intelligence ?

1. Artificial intelligence (AI), sometimes called **machine intelligence**, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals, such as "learning" and "problem solving.
2. The term artificial intelligence was first coined by John McCarthy in 1956 when he held the first academic conference on the subject. But the journey to understand if machines can truly think

began much before that. In Vannevar Bush's seminal work **As We May Think**, he proposed a system which amplifies people's own knowledge and understanding. Five years later Alan Turing wrote a paper on the notion of machines being able to simulate human beings and the ability to do intelligent things, such as playing a Chess.

3. In computer science AI system is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Thus Artificial Intelligence system can be viewed as :



4. AI is about teaching the machines to learn, to act, and think as humans would do. We can organize AI definition into 4 categories:
   - The definitions on top are concerned with thought processes and reasoning, whereas the ones on the bottom address behaviour.
   - The definitions on the left measure success in terms of conformity to human performance whereas the ones on the right measure against an ideal performance measure called rationality.
   - A system is rational if it does the "right thing," given what it knows.
   - Historically, all four approaches to AI have been followed, each by different people with different methods.
   - A human-centered approach must be in part an empirical science, involving observations and hypotheses about human behaviour.
   - A rationalist's approach involves a combination of mathematics and engineering. The various groups have both disparaged and helped each other. Let us look at the four approaches in more detail

Thinking Humanly	Thinking Rationally
"The exciting new effort to make computers think machines with minds, in the full and literal sense." (Haugeland, 1985)	"The study of mental faculties through the use of computational models."(Charniak and McDermott, 1985)
"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning." (Bellman, 1978)	"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.‖ (Pooleet al., 1998) —AI is concerned with intelligent behaviour in artifacts." (Nilsson, 1998)

Thinking humanly: The cognitive modelling approach

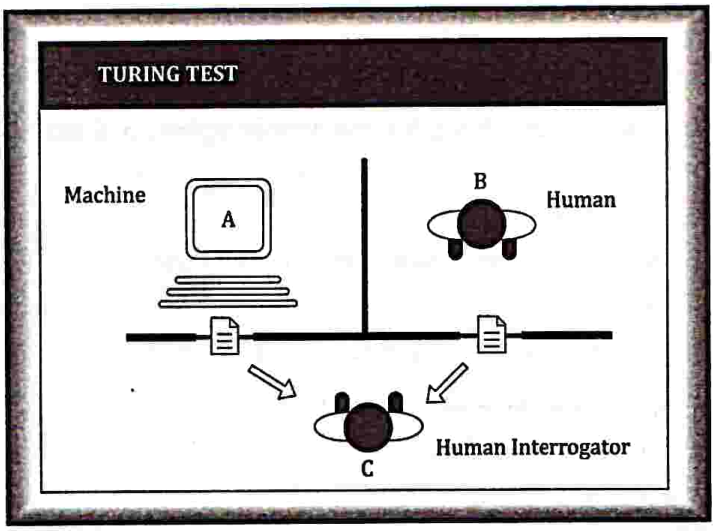
If we are going to say that a given program thinks like a human, we must have some way of determining how humans think. We need to get inside the actual workings of human minds.

There are three ways to do this:

1. through introspection-trying to catch our own thoughts as they go by
2. through psychological experiments-observing a person in action; and
3. through brain imaging-observing the brain in action.

Acting humanly: The Turing Test approach

1. The Turing Test, proposed by Alan Turing (1950), was designed to provide a satisfactoryoperational definition of intelligence.
2. A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer
3. This test is used to evaluate a computer acting like humanly.



4. For current scenarios the computer would need to possess the following capabilities:
   - **natural language processing** to enable it to communicate successfully in English
   - **knowledge representation** to store what it knows or hears;
   - **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 the patterns.

5. **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.
6. To pass the total Turing Test, the computer will need
   - **computer vision** to perceive objects, and
   - **robotics** to manipulate objects and move about.

These six disciplines compose most of AI

Thinking rationally: The "laws of thought" approach

Aristotle was one of the first to attempt to codify - right thinking,|| that is, irrefutable reasoning processes. His **syllogisms** provided patterns for argument structures that always yielded correct conclusions when given correct premises.

Eg. Socrates is a man;

all men are mortal;

therefore, Socrates is mortal.|| - logic

There are two main obstacles to this approach.

1. it is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain.
2. Second, there is a big difference between solvinga problem —in principle|| and solving it in practice.

Acting rationally: The rational agent approach

- An **agent** is just something that acts.
- All computer programs do something, but computer agents are expected to do more: operate autonomously, perceive their environment, persist over a prolonged time period, and adapt to change, and create and pursue goals.
- A **rational agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.
- In the - laws of thought|| approach to AI, the emphasis was on correct inferences.
- On the other hand, correct inference is not all of rationality; in some situations, there is no provably correct thing to do, but something must still be done.
- For example, recoiling from a hot stove is a reflex action that is usually more successful than a slower action taken after careful deliberation.

Other approaches are :

i. **Machine Learning approach**: This approach involves training machines to learn from data and improve performance on specific tasks over time. It is widely used in areas such as image and speech recognition, natural language processing, and recommender systems.

ii. **Evolutionary approach**: This approach is inspired by the process of natural selection in biology. It involves generating and testing a large number of variations of a solution to a

problem, and then selecting and combining the most successful variations to create a new generation of solutions.

iii. **Neural Networks approach**: This approach involves building artificial neural networks that are modelled after the structure and function of the human brain. Neural networks can be used for tasks such as pattern recognition, prediction, and decision-making.

iv. **Fuzzy logic approach**: This approach involves reasoning with uncertain and imprecise information, which is common in real-world situations. Fuzzy logic can be used to model and control complex systems in areas such as robotics, automotive control, and industrial automation.

v. **Hybrid approach**: This approach combines multiple AI techniques to solve complex problems. For example, a hybrid approach might use machine learning to analyze data and identify patterns, and then use logical reasoning to make decisions based on those patterns.

1.3 History of AI

Artificial Intelligence is not a new word and not a new technology for researchers. Following are some milestones in the history of AI which defines the journey from the AI generation to till date development.

1950	• Alan Turing published "Computing Machinery and Intelligence," introducing the Turing test and opening the doors to what would be known as AI.
1951	• Marvin Minsky and Dean Edmonds developed the first artificial neural network (ANN) called SNARC using 3,000 vacuum tubes to simulate a network of 40 neurons.
1952	• Arthur Samuel developed Samuel Checkers-Playing Program, the world's first program to play games that was self-learning
1956	• John McCarthy, Marvin Minsky, Nathaniel Rochester and Claude Shannon coined the term artificial intelligence in a proposal for a workshop widely recognized as a founding event in the AI field.
1958	• Frank Rosenblatt developed the perceptron, an early ANN that could learn from data and became the foundation for modern neural networks. • John McCarthy developed the programming language Lisp, which was quickly adopted by the AI industry and gained enormous popularity among developers.
1959	• Arthur Samuel coined the term machine learning in a seminal paper explaining that the computer could be programmed to outplay its programmer. • Oliver Selfridge published "Pandemonium: A Paradigm for Learning," a landmark contribution to machine learning that described a model that could adaptively improve itself to find patterns in events.
1964	• Daniel Bobrow developed STUDENT, an early natural language processing (NLP) program designed to solve algebra word problems, while he was a doctoral candidate at MIT.
1965	• Edward Feigenbaum, Bruce G. Buchanan, Joshua Lederberg and Carl Djerassi developed the first expert system, Dendral, which assisted organic chemists in identifying unknown organic molecules.

1966	• Joseph Weizenbaum created Eliza, one of the more celebrated computer programs of all time, capable of engaging in conversations with humans and making them believe the software had humanlike emotions. • Stanford Research Institute developed Shakey, the world's first mobile intelligent robot that combined AI, computer vision, navigation and NLP. It's the grandfather of self-driving cars and drones.
1968	• Terry Winograd created SHRDLU, the first multimodal AI that could manipulate and reason out a world of blocks according to instructions from a user.
1969	• Arthur Bryson and Yu-Chi Ho described a backpropagation learning algorithm to enable multilayer ANNs, an advancement over the perceptron and a foundation for deep learning. • Marvin Minsky and Seymour Papert published the book Perceptrons, which described the limitations of simple neural networks and caused neural network research to decline and symbolic AI research to thrive.
1973	• James Lighthill released the report "Artificial Intelligence: A General Survey," which caused the British government to significantly reduce support for AI research.
1980	• Symbolics Lisp machines were commercialized, signaling an AI renaissance. Years later, the Lisp machine market collapsed.
1981	• Danny Hillis designed parallel computers for AI and other computational tasks, an architecture similar to modern GPUs.
1984	• Marvin Minsky and Roger Schank coined the term AI winter at a meeting of the Association for the Advancement of Artificial Intelligence, warning the business community that AI hype would lead to disappointment and the collapse of the industry, which happened three years later.
1985	• Judea Pearl introduced Bayesian networks causal analysis, which provides statistical techniques for representing uncertainty in computers.
1988	• Peter Brown et al. published "A Statistical Approach to Language Translation," paving the way for one of the more widely studied machine translation methods.
1989	• Yann LeCun, Yoshua Bengio and Patrick Haffner demonstrated how convolutional neural networks (CNNs) can be used to recognize handwritten characters, showing that neural networks could be applied to real-world problems.
1997	• Sepp Hochreiter and Jürgen Schmidhuber proposed the Long Short-Term Memory recurrent neural network, which could process entire sequences of data such as speech or video. • IBM's Deep Blue defeated Garry Kasparov in a historic chess rematch, the first defeat of a reigning world chess champion by a computer under tournament conditions.
2000	• University of Montreal researchers published "A Neural Probabilistic Language Model," which suggested a method to model language using feedforward neural networks.
2006	• Fei-Fei Li started working on the ImageNet visual database, introduced in 2009, which became a catalyst for the AI boom and the basis of an annual competition for image recognition algorithms. • IBM Watson originated with the initial goal of beating a human on the iconic quiz show Jeopardy! In 2011, the question-answering computer system defeated the show's all-time (human) champion, Ken Jennings.

2009	• Rajat Raina, Anand Madhavan and Andrew Ng published "Large-Scale Deep Unsupervised Learning Using Graphics Processors," presenting the idea of using GPUs to train large neural networks.
2011	• Jürgen Schmidhuber, Dan Claudiu Cireşan, Ueli Meier and Jonathan Masci developed the first CNN to achieve "superhuman" performance by winning the German Traffic Sign Recognition competition. • Apple released Siri, a voice-powered personal assistant that can generate responses and take actions in response to voice requests.
2012	• Geoffrey Hinton, Ilya Sutskever and Alex Krizhevsky introduced a deep CNN architecture that won the ImageNet challenge and triggered the explosion of deep learning research and implementation.
2013	• China's Tianhe-2 doubled the world's top supercomputing speed at 33.86 petaflops, retaining the title of the world's fastest system for the third consecutive time. • DeepMind introduced deep reinforcement learning, a CNN that learned based on rewards and learned to play games through repetition, surpassing human expert levels. • Google researcher Tomas Mikolov and colleagues introduced Word2vec to automatically identify semantic relationships between words.
2014	• Ian Goodfellow and colleagues invented generative adversarial networks, a class of machine learning frameworks used to generate photos, transform images and create deepfakes. • Diederik Kingma and Max Welling introduced variational autoencoders to generate images, videos and text. • Facebook developed the deep learning facial recognition system DeepFace, which identifies human faces in digital images with near-human accuracy.
2016	• DeepMind's AlphaGo defeated top Go player Lee Sedol in Seoul, South Korea, drawing comparisons to the Kasparov chess match with Deep Blue nearly 20 years earlier. • Uber started a self-driving car pilot program in Pittsburgh for a select group of users.
2017	• Stanford researchers published work on diffusion models in the paper "Deep Unsupervised Learning Using Nonequilibrium Thermodynamics." The technique provides a way to reverse-engineer the process of adding noise to a final image. • Google researchers developed the concept of transformers in the seminal paper «Attention Is All You Need,» inspiring subsequent research into tools that could automatically parse unlabeled text into large language models (LLMs). • British physicist Stephen Hawking warned, "Unless we learn how to prepare for, and avoid, the potential risks, AI could be the worst event in the history of our civilization."
2018	• Developed by IBM, Airbus and the German Aerospace Center DLR, Cimon was the first robot sent into space to assist astronauts. • OpenAI released GPT (Generative Pre-trained Transformer), paving the way for subsequent LLMs. • Groove X unveiled a home mini-robot called Lovot that could sense and affect mood changes in humans.

2019	• Microsoft launched the Turing Natural Language Generation generative language model with 17 billion parameters. • Google AI and Langone Medical Center's deep learning algorithm outperformed radiologists in detecting potential lung cancers.
2020	• The University of Oxford developed an AI test called Curial to rapidly identify COVID-19 in emergency room patients. • Open AI released the GPT-3 LLM consisting of 175 billion parameters to generate humanlike text models. • Nvidia announced the beta version of its Omniverse platform to create 3D models in the physical world. • DeepMind's AlphaFold system won the Critical Assessment of Protein Structure Prediction protein-folding contest.
2021	• Open AI introduced the Dall-E multimodal AI system that can generate images from text prompts. • The University of California, San Diego, created a four-legged soft robot that functioned on pressurized air instead of electronics.
2022	• Google software engineer Blake Lemoine was fired for revealing secrets of Lamda and claiming it was sentient. • DeepMind unveiled AlphaTensor "for discovering novel, efficient and provably correct algorithms." • Intel claimed its FakeCatcher real-time deepfake detector was 96% accurate. • OpenAI released ChatGPT in November to provide a chat-based interface to its GPT-3.5 LLM.
2023	• OpenAI announced the GPT-4 multimodal LLM that receives both text and image prompts. • Elon Musk, Steve Wozniak and thousands more signatories urged a six-month pause on training «AI systems more powerful than GPT-4.»
Beyond 2023	• In business, 55% of organizations that have deployed AI always consider AI for every new use case they're evaluating, according to a 2023 Gartner survey. By 2026, Gartner reported, organizations that "operationalize AI transparency, trust and security will see their AI models achieve a 50% improvement in terms of adoption, business goals and user acceptance."

1.4 Artificial Intelligence Disciplines

The aim of AI is to develop program for simulating learning of human being and its a complex process which requires knowledge of various disciplines like Computer Science, Biology, Psychology, Linguistics etc... Learning all these things are really complex task and that's why in project team we have to hire resources from many fields.

Following are major disciplines used with Artificial Intelligence:

1. Philosophy
2. Mathematics
3. Economics
4. Neuroscience
5. Psychology
6. Computer engineering

7. Control theory and cybernetics
8. Linguistics
9. Big Data

1. **Philosophy:**

   Philosophy is very important as it attempts to answer important questions like "Can a machine act intelligently?", "Can it solve like human being?", "Are computer intelligence is like Human one?" etc...

2. **Mathematics:**

   Mathematics is used to write the logic and algorithm for machine learning. Philosophy thinks and defines particular intelligence and way it should work. But here comes the intelligence of Mathematicians to come out with calculations and algorithms for learning. Good knowledge of mathematics is a must skills to be able to develop model of AI.

3. **Economics:**

   - How should we make decisions so as to maximize payoff?
   - How should we do this when others may not go along?
   - How should we do this when the payoff may be far in the future?

   Lots of Economics reasoning is used in developing programs for Artificial Intelligence.

4. **Neuroscience:**

   Study of Neuroscience providing information about how human brain works and how neurons responds to a particular event. This enables AI scientists to develop programming model to work like human brain.

5. **Psychology:**

   The Psychology is used to study and find the process of thinking of humans and animals. This discipline enables data science to understand the Brain, Behaviour and Person which is essential to make things like human brain.

6. **Computer engineering:**

   Computer engineer write the codes for making the neural network for artificial intelligence. It then updates the values/properties of the neural network based on the data provided to the system. This way Artificial Intelligence is achieved. Computer programmer should have very high programming skills along with the knowledge of Mathematics and other disciplines used with AI.

7. **Control theory and cybernetics:**

   This theory describes how things operate under their own control. It is scientific study of working of humans, animals and machines control and communicate with each other. Knowledge of this technology is also very important.

8. **Linguistics:**

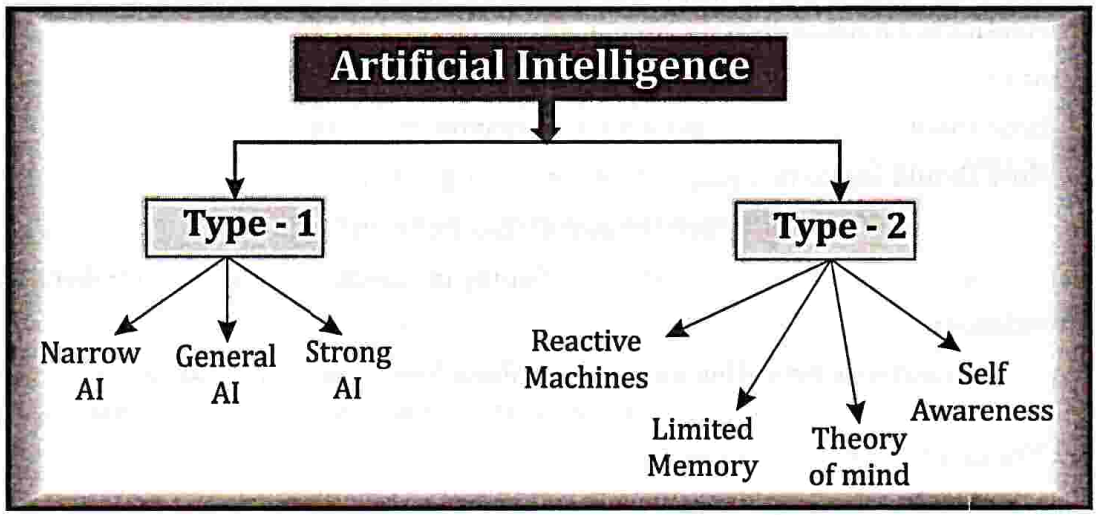
   The modern Linguistics is called computational linguistics or natural language processing. The natural language processing allows the intelligent system to communicate through language such as English. To natural language processing experience is also a must for developing Artificial Intelligence system for machines.

9. **Big Data:**

   Big Data is fueling the growth of Artificial Intelligence as it provides a platform for saving and querying huge data sets. AI requires to process a lot of data and its not possible to save data in one computer and Big Data plays big role here. Big Data also provides distributed computing environment which can be used for training the model on distributed system.

1.5 Types of Artificial Intelligence

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

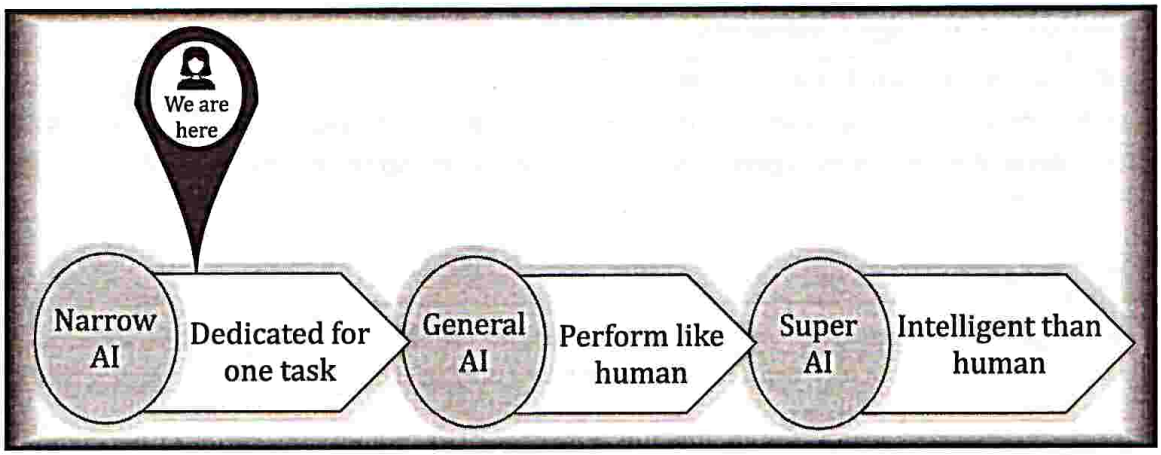


AI TYPE -1 Based on functionality

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

- The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
- Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
- The worldwide researchers are now focused on developing machines with General AI.
- As systems with general AI are still under research, and it will take lots of efforts and time to develop such systems.

3. **Super AI / Strong AI:**
   - Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI.
   - Some key characteristics of strong AI include capability include the ability to think, to reason,solve the puzzle, make judgments, plan, learn, and communicate by its own.
   - Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task



AI TYPE -2 Based on functionality

1. **Reactive Machines:**
   - Purely reactive machines are the most basic types of Artificial Intelligence.
   - Such AI systems do not store memories or past experiences for future actions.
   - These machines only focus on current scenarios and react on it as per possible best action.
   - IBM's Deep Blue system is an example of reactive machines.
   - Google's AlphaGo is also an example of reactive machines.
2. **Limited Memory:**
   - Limited memory machines can store past experiences or some data for a short period of time.
   - These machines can use stored data for a limited time period only.

- Self-driving cars are one of the best examples of Limited Memory systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road.

3. **Theory of Mind:**
   - Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.
   - This type of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.
4. **Self-Awareness:**
   - Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.
   - These machines will be smarter than human mind.
   - Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

1.6 Advantages of Artificial Intelligence

Advantages of Artificial Intelligence

1. **Reduction in Human Error:** One of the biggest benefits of Artificial Intelligence is that it can significantly reduce errors and increase accuracy and precision. The decisions taken by AI in every step is decided by information previously gathered and a certain set of algorithms. When programmed properly, these errors can be reduced to null.

   Example

   An example of the reduction in human error through AI is the use of robotic surgery systems, which can perform complex procedures with precision and accuracy, reducing the risk of human error and improving patient safety in healthcare.

2. **Zero Risks:** Another big benefit of AI is that humans can overcome many risks by letting AI robots do them for us. Whether it be defusing a bomb, going to space, exploring the deepest parts of oceans, machines with metal bodies are resistant in nature and can survive unfriendly atmospheres. Moreover, they can provide accurate work with greater responsibility and not wear out easily.

   Example

   One example of zero risks is a fully automated production line in a manufacturing facility. Robots perform all tasks, eliminating the risk of human error and injury in hazardous environments.

3. **24 × 7 Availability:** There are many studies that show humans are productive only about 3 to 4 hours in a day. Humans also need breaks and time offs to balance their work life and personal life. But AI can work endlessly without breaks. They think much faster than humans and perform multiple tasks at a time with accurate results. They can even handle tedious repetitive jobs easily with the help of AI algorithms.

Example

An example of this is online customer support chatbots, which can provide instant assistance to customers anytime, anywhere. Using AI and natural language processing, chatbots can answer common questions, resolve issues, and escalate complex problems to human agents, ensuring seamless customer service around the clock.

4. **Digital Assistance:** Some of the most technologically advanced companies engage with users using digital assistants, which eliminates the need for human personnel. Many websites utilize digital assistants to deliver user-requested content. We can discuss our search with them in conversation. Some chatbots are built in a way that makes it difficult to tell whether we are conversing with a human or a chatbot.

Example

We all know that businesses have a customer service crew that must address the doubts and concerns of the patrons. Businesses can create a chatbot or voice bot that can answer all of their clients' questions using AI.

5. **New Inventions:** In practically every field, AI is the driving force behind numerous innovations that will aid humans in resolving the majority of challenging issues.

   For instance, recent advances in AI-based technologies have allowed doctors to detect breast cancer in a woman at an earlier stage.

Example

Another example of new inventions is self-driving cars, which use a combination of cameras, sensors, and AI algorithms to navigate roads and traffic without human intervention. Self-driving cars have the potential to improve road safety, reduce traffic congestion, and increase accessibility for people with disabilities or limited mobility. They are being developed by various companies, including Tesla, Google, and Uber, and are expected to revolutionize transportation.

6. **Unbiased Decisions:** Human beings are driven by emotions, whether we like it or not. AI on the other hand, is devoid of emotions and highly practical and rational in its approach. A huge advantage of Artificial Intelligence is that it doesn't have any biased views, which ensures more accurate decision-making.

Example

An example of this is AI-powered recruitment systems that screen job applicants based on skills and qualifications rather than demographics. This helps eliminate bias in the hiring process, leading to an inclusive and more diverse workforce.

7. **Perform Repetitive Jobs:** We will be doing a lot of repetitive tasks as part of our daily work, such as checking documents for flaws and mailing thank-you notes, among other things. We may use artificial intelligence to efficiently automate these menial chores and even eliminate "boring" tasks for people, allowing them to focus on being more creative.

Example

An example of this is using robots in manufacturing assembly lines, which can handle repetitive tasks such as welding, painting, and packaging with high accuracy and speed, reducing costs and improving efficiency.

8. **Daily Applications:** Today, our everyday lives are entirely dependent on mobile devices and the internet. We utilize a variety of apps, including Google Maps, Alexa, Siri, Cortana on Windows, OK Google, taking selfies, making calls, responding to emails, etc. With the use of various AI-based techniques, we can also anticipate today's weather and the days ahead.

Example

About 20 years ago, you must have asked someone who had already been there for instructions when you were planning a trip. All you need to do now is ask Google where Bangalore is. The best route between you and Bangalore will be displayed, along with Bangalore's location, on a Google map.

9. **AI in Risky Situations:** One of the main benefits of artificial intelligence is this. By creating an AI robot that can perform perilous tasks on our behalf, we can get beyond many of the dangerous restrictions that humans face. It can be utilized effectively in any type of natural or man-made calamity, whether it be going to Mars, defusing a bomb, exploring the deepest regions of the oceans, or mining for coal and oil.

Example

For instance, the explosion at the Chernobyl nuclear power facility in Ukraine. As any person who came close to the core would have perished in a matter of minutes, at the time, there were no AI-powered robots that could assist us in reducing the effects of radiation by controlling the fire in its early phases.

10. **Faster Decision-making:** Faster decision-making is another benefit of AI. By automating certain tasks and providing real-time insights, AI can help organizations make faster and more informed decisions. This can be particularly valuable in high-stakes environments, where decisions must be made quickly and accurately to prevent costly errors or save lives.

Example

An example of faster decision-making is using AI-powered predictive analytics in financial trading, where algorithms can analyze vast amounts of data in real time and make informed investment decisions faster than human traders, resulting in improved returns and reduced risks.

11. **Pattern Identification:** Pattern identification is another area where AI excels. With its ability to analyze vast amounts of data and identify patterns and trends, AI can help businesses and organizations better understand customer behavior, market trends, and other important factors. This information can be used to make better decisions and improve business outcomes.

Example

An example of pattern identification is the use of AI in fraud detection, where machine learning algorithms can identify patterns and anomalies in transaction data to detect and prevent fraudulent activity, improving security and reducing financial losses for individuals and organizations.

12. **Medical Applications:** AI has also made significant contributions to the field of medicine, with applications ranging from diagnosis and treatment to drug discovery and clinical trials. AI-powered tools can help doctors and researchers analyze patient data, identify potential health risks, and develop personalized treatment plans. This can lead to better health outcomes for patients and help accelerate the development of new medical treatments and technologies.

1.7 Disadvantages of Artificial Intelligence

Disadvantages of Artificial Intelligence

1. **High Costs:** The ability to create a machine that can simulate human intelligence is no small feat. It requires plenty of time and resources and can cost a huge deal of money. AI also needs to operate on the latest hardware and software to stay updated and meet the latest requirements, thus making it quite costly.
2. **No Creativity:** A big disadvantage of AI is that it cannot learn to think outside the box. AI is capable of learning over time with pre-fed data and past experiences, but cannot be creative in its approach. A classic example is the bot Quill who can write Forbes earning reports. These reports only contain data and facts already provided to the bot. Although it is impressive that a bot can write an article on its own, it lacks the human touch present in other Forbes articles.
3. **Unemployment:** One application of artificial intelligence is a robot, which is displacing occupations and increasing unemployment (in a few cases). Therefore, some claim that there is always a chance of unemployment as a result of chatbots and robots replacing humans.

   For instance, robots are frequently utilized to replace human resources in manufacturing businesses in some more technologically advanced nations like Japan. This is not always the case, though, as it creates additional opportunities for humans to work while also replacing humans in order to increase efficiency.
4. **Make Humans Lazy:** AI applications automate the majority of tedious and repetitive tasks. Since we do not have to memorize things or solve puzzles to get the job done, we tend to use our brains less and less. This addiction to AI can cause problems to future generations.
5. **No Ethics:** Ethics and morality are important human features that can be difficult to incorporate into an AI. The rapid progress of AI has raised a number of concerns that one day, AI will grow uncontrollably, and eventually wipe out humanity. This moment is referred to as the AI singularity.
6. **Emotionless:** Since early childhood, we have been taught that neither computers nor other machines have feelings. Humans function as a team, and team management is essential for achieving goals. However, there is no denying that robots are superior to humans when functioning effectively, but it is also true that human connections, which form the basis of teams, cannot be replaced by computers.
7. **No Improvement:** Humans cannot develop artificial intelligence because it is a technology based on pre-loaded facts and experience. AI is proficient at repeatedly carrying out the same task, but if we want any adjustments or improvements, we must manually alter the codes. AI cannot be accessed and utilized akin to human intelligence, but it can store infinite data.

   Machines can only complete tasks they have been developed or programmed for; if they are asked to complete anything else, they frequently fail or provide useless results, which can have significant negative effects. Thus, we are unable to make anything conventional.

1.8 Top Technologies used in Artificial Intelligence

Some of the few top technologies rocking the world are listed below:

1. **Natural Language Generation:** AI converts the data into a readable form allow the system to interact ideas with perfect accuracy. It is widely used in customer services to generate reports and pull market data.

2. **Speech Recognition:** Siri is the best example of speech recognition which understands and interacts with the voice response of human language by mobile apps.
3. **Virtual Agents:** The Chatbot is a suitable example that is programmed to interact with a human.
4. **Machine Learning Platform:** The main aim is to develop techniques that enable the computer to learn. They are currently developed for prediction and acts as an audience management tool. It is most profitable for digital marketing.
5. **AI Optimized Hardware:** The new graphics and processing unit are designed and developed to perform Artificial Intelligent oriented tasks.
6. **Decision Management:** Intelligent machines are designed to frame new rules and logic to AI systems for setting up, prolonged maintenance and optimum tuning and make you run a profitable organization.
7. **Deep Learning Platform:** It is mainly used for classification and pattern recognition for large scale data.
8. **Biometrics:** This technology is used to identify and analyze the human attributes and physical features of a body's shape and form.
9. **Robotic Process Automation:** It uses scripts and mimics the human process and fed to a robot to complete it effectively.
10. **Digital Twin:** A digital twin is software that joins the space between physical systems and the digital world.
11. **Cyber Defense:** It acts as a firewall that detects, prevent and provides timely support to fight against any threat which is yet to affect information and infrastructure.
12. **Compliance:** It is an agreement between the employee and organization to follow the standard policies and rules of the organization.
13. **Knowledge Worker Aid:** AI technology also can widely help employees at work, especially those in knowledge work.
14. **Peer to Peer Networks:** When multiple systems are connected and share resources without the data going through the server computer. It is also used in cryptocurrencies.
15. **Content Writing:** Artificial intelligence helps in content writing such as articles, blogs, reports by suggesting possible words that suit well for sentences and also provide spell correction and grammatical mistakes to their online world.
16. **Emotion Recognition:** The technology permits the software to "scan" the sentiments on a human face using high-level image processing or audio processing, now at the point where you can catch "micro-expressions," or complex body language ideas, and vocal accent that reveals a person's feelings.
17. **Image Recognition:** Image recognition is the method of recognizing and distinguishing an object or trait in a digital image or video, and AI is frequently being piled on top of this technology to great effect.
18. **Marketing Automation:** Marketing automation enables companies to increase engagement and improve performance to grow income faster. It uses software to automate client segmentation, data integration, and campaign management, and streamlines repeated tasks, providing vital minds to get back to doing what they do best.

Summary

- ***Intelligence*** is an ability of a system to calculate, reason, perceive relationships and analogies, learn from experience, store and retrieve information from memory, solve problems, comprehend complex ideas, use natural language fluently, classify, generalize, and adapt new situations.
- Intelligence composed of ***Reasoning, Learning. Problem Solving, Perception, Linguistic Intelligence***
- Reasoning broadly divided into ***Inductive Reasoning & Deductive Reasoning***
- Learning is categorized into ***Auditory Learning, Episodic Learning, Motor Learning, Observational Learning, Perceptual Learning, Relational Learning, Spatial Learning, Stimulus-Response Learning***
- Artificial intelligence (AI), sometimes called *machine intelligence*, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals, such as "learning" and "problem solving
- AI definition into 4 categories : ***Thinking Humanly(***The cognitive modelling approach), ***Thinking Rationally(***The "laws of thought" approach), ***Acting Humanly(***The Turing Test approach), ***Acting Rationally(***The rational agent approach).
- Other approaches are : Machine Learning approach, Evolutionary approach, Neural Networks approach, Fuzzy logic approach, Hybrid approach
- Major disciplines used with Artificial Intelligence are : ***Philosophy, Mathematics, Economics, Neuroscience, Psychology, Computer engineering, Control theory and cybernetics, Linguistics and Big Data***
- Artificial Intelligence broadly classified into two types:

  Type-1 (Based on Capabilities) – Weak AI or Narrow AI, General AI, Super AI or Strong AI

  Type-2 (Based on functionality) – Reactive Machines, Limited Memory, Theory of mind, Self-Awareness.
- Advantages of AI:

  Reduction in Human error
  24 × 7 Availability
  New Inventions
  Perform repetitive jobs
  AI in risk situations
  Pattern identification
  Medical applications
  Zero Risks
  Digital Assistance
  Unbiased decisions
  Daily applications
  Faster decision making
- Disadvantages of AI:

  High cost
  Unemployment
  No ethics
  No Improvement
  No Creativity
  Make humans lazy
  Emotionless

- Top technologies used in AI:
  - Natural Language Generation
  - Virtual Agents
  - AI Optimized hardware
  - Deep learning platform
  - Robotic Process Automation
  - Cyber Defence
  - Knowledge worker aid
  - Content writing
  - Emotion recognition
  - Speech Recognition
  - Machine Learning platform
  - Decision Management
  - Biometrics
  - Digital twin
  - Compliance
  - Peer to peer network
  - Image recognition

1.9 Review Questions

Short Answer Questions

1. Define Intelligence.
2. What is intelligence composed of?
3. What do you mean by Linguistic intelligence?
4. Define AI.
5. List 4 categories of AI.
6. List any 4 major disciplines of AI.
7. Compare Weak AI with Strong AI
8. What are Reactive machines in AI.
9. List any 4 advantages of AI.
10. List any 4 disadvantages of AI.
11. List any 4 top technologies used in AI

Long Answer Questions

1. Explain Intelligence System?
2. Explain the four categories of AI
3. Explain the Turning Test approach of AI with a neat diagram.
4. List any 8 mile stones of AI.
5. Explain the major disciplines of AI
6. Explain in detail the types of AI.
7. Explain any 4 advantages of AI in detail
8. Explain any 4 disadvantages of AI in detail.
9. List and explain any 4 technologies used in AI

❖❖❖❖

UNIT - I

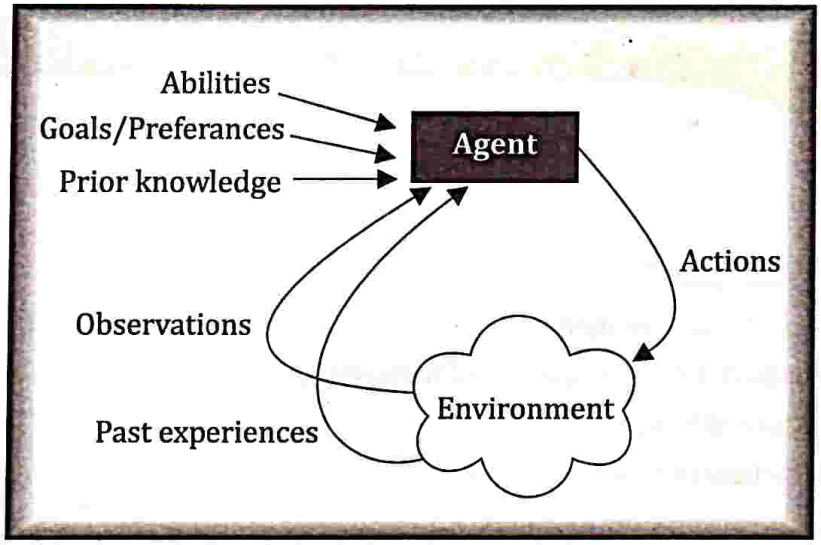
CHAPTER 2

INTELLIGENT AGENTS

- What is an Agent ?
- Agents & its Environment
- Good Behaviour : The Concept of Rationality
- The Nature of Environment
- The Structure of Intelligent Agents
- PEAS Representation
- Review Questions

2.1 What is an Agent ?

- In artificial intelligence, an agent is a computer program or system that is designed to perceive its environment, make decisions and take actions to achieve a specific goal or set of goals. The agent operates autonomously, meaning it is not directly controlled by a human operator but it takes inputs from environment based on abilities, preferences, prior knowledge, observations and past experiences and then finally formulate the actions.



- An agent can be: (few listed)
  - **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
  - **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
  - **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.
  - **Gaming agents:** These are agents that are designed to play games, either against human opponents or other agents. Examples of gaming agents include chess-playing agents and poker-playing agents.
  - **Fraud detection agents:** These are agents that are designed to detect fraudulent behavior in financial transactions. They can analyse patterns of behavior to identify suspicious activity and alert authorities. Examples of fraud detection agents include those used by banks and credit card companies
  - **Traffic management agents:** These are agents that are designed to manage traffic flow in cities. They can monitor traffic patterns, adjust traffic lights, and reroute vehicles to minimize congestion. Examples of traffic management agents include those used in smart cities around the world.

Hence the world around us is full of agents such as thermostat, cell phone, camera, and even we are also agents.

- Agents can be classified into different types based on their characteristics, such as whether they are **reactive** or **proactive**, whether they have a **fixed** or **dynamic** environment, and whether they are **single** or **multi-agent** systems.
    - **Reactive agents** are those that respond to immediate stimuli from their environment and take actions based on those stimuli.
    - **Proactive agents,** on the other hand, take initiative and plan ahead to achieve their goals.
    - The environment in which an agent operates can also be fixed or dynamic. **Fixed environments** have a static set of rules that do not change, while **dynamic environments** are constantly changing and require agents to adapt to new situations.
    - **Multi-agent** systems involve multiple agents working together to achieve a common goal. These agents may have to coordinate their actions and communicate with each other to achieve their objectives.

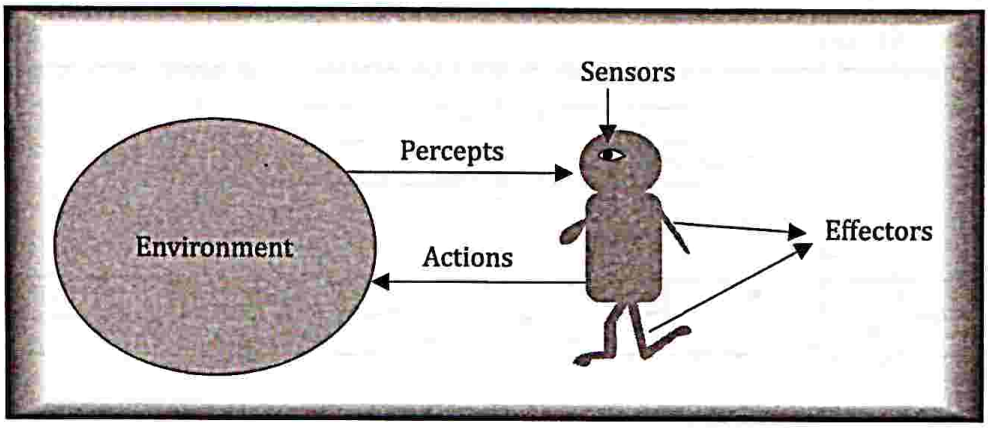
2.2 Agents & its Environment

An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents and can be viewed as:

- Perceiving its environment through **sensors**

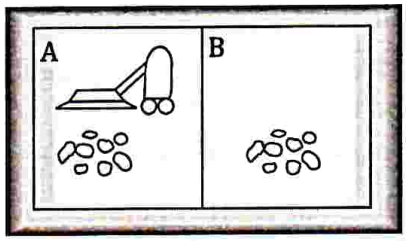
  [Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.]
- Acting upon that environment through **actuators**

  [Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.]
- **Effectors:** Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.

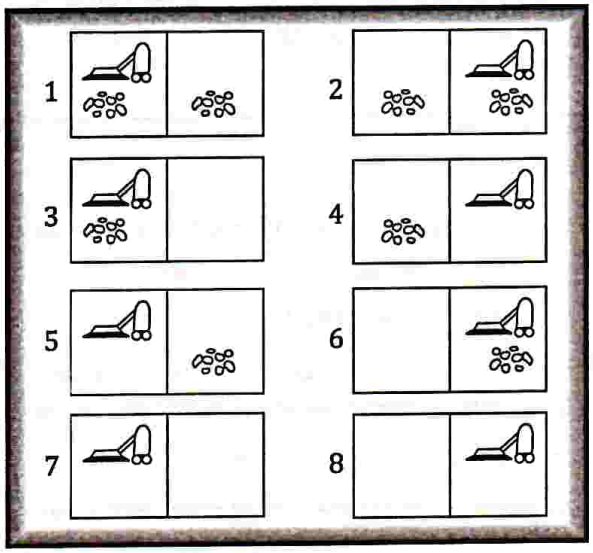


The term **percept** refers to the agent's perceptual inputs at any given instant. An agent's percept sequence is the complete history of everything the agent has ever perceived. Mathematically speaking, we say that an agent's behaviour is described by the **agent function** that maps any given percept sequence to an action **f[P-->A].** Internally, the agent function for an artificial agent will be implemented by an **agent program.**

To illustrate these ideas, we use a very simple example—the vacuum-cleaner world shown in Figure.



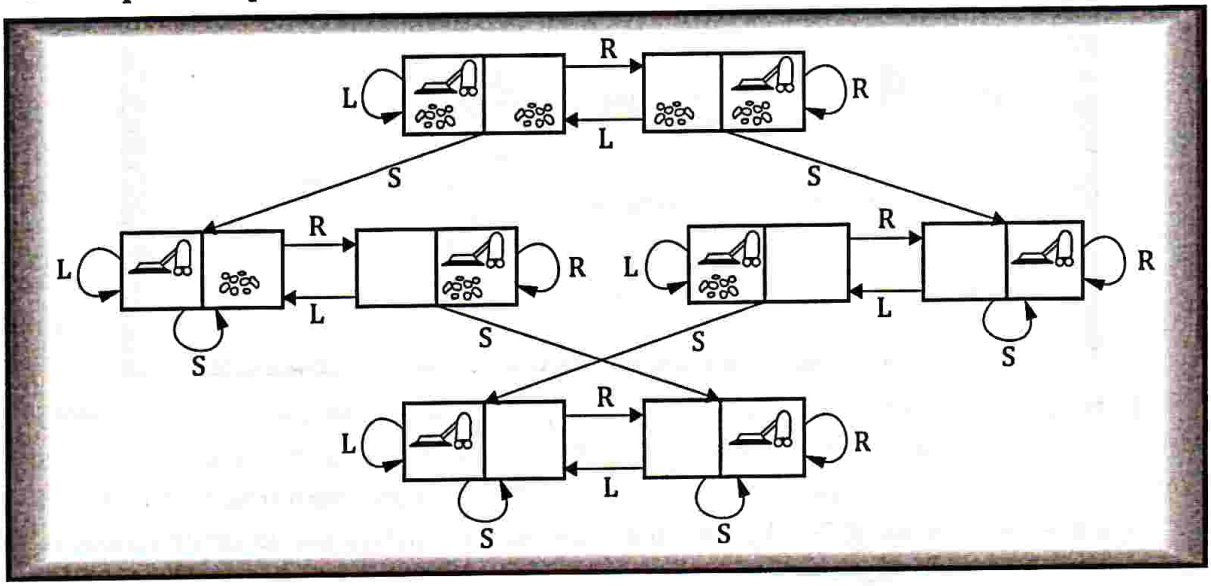
Let's suppose that the world has just two rooms. The robot can be in either room and there can be dirt in zero, one, or two rooms.



Goal formulation: intuitively, we want all the dirt cleaned up. Formally, the goal is { state 7, state 8 }.

Problem formulation(Actions):Left, Right, Suck, NoOp

State Space Graph:



Agent function

```
function REFLEX-VACCUM-AGENT (location, status)
{
If status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
}
```

The above example is a very simple agent function with the following: if the current square is dirty, then suck; otherwise, move to the other square. A partial tabulation of this agent function is shown below.

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
.	:
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
:	.

2.3 Good Behaviour : The Concept of Rationality

- Rationality is nothing but status of being reasonable, sensible, and having good sense of judgment. Rationality is concerned with expected actions and results depending upon what the agent has perceived. Performing actions with the aim of obtaining useful information is an important part of rationality.
- A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept sequence. The problem the agent solves is characterized by Performance Measure, Environment, Actuators, and Sensors (PEAS).
- Rationality of an agent depends on the following –
    - The performance measures, which determine the degree of success.
    - Agent's Percept Sequence till now.
    - The agent's prior knowledge about the environment.
    - The actions that the agent can carry out.

This leads to a definition of a **rational agent:**

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

For better understanding Let us consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not; this is the agent function. **Is this a rational agent?** That depends! First, we need to say what the performance measure is, what is known about the environment, and what sensors and actuators the agent has Let us assume the following:

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known a priori but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The Left and Hight actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are Left, Right, and Suck.
- The agent correctly perceives its location and whether that location contains dirt.

We claim that under these circumstances the agent is indeed **rational**; its expected performance is at least as high as any other agent'.

Eventually the same agent would be **irrational** under different circumstances. For example, once all the dirt is cleaned up, the agent will oscillate needlessly back and forth; if the performance measure includes a penalty of one point for each movement left or right, the agent will fare poorly. A better agent for this case would do nothing once it is sure that all the squares are clean. If clean squares can become dirty again, the agent should occasionally check and re-clean them if needed. If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares A and B.

2.4 The Nature of Environment

Some programs operate in the entirely artificial environment confined to keyboard input, database, computer file systems and character output on a screen. In contrast, some software agents (software robots or softbots) exist in rich, unlimited softbots domains. The simulator has a very detailed, complex environment. The software agent needs to choose from a long array of actions in real time. A softbot designed to scan the online preferences of the customer and show interesting items to the customer works in the real as well as an artificial environment.

The most famous artificial environment is the **Turing Test environment**, in which one real and other artificial agents are tested on equal ground. This is a very challenging environment as it is highly difficult for a software agent to perform as well as a human.

Turing Test

Two persons and a machine to be evaluated participate in the test. Out of the two persons, one plays the role of the tester. Each of them sits in different rooms. The tester is unaware of who is machine and who is a human. He interrogates the questions by typing and sending them to both intelligences, to which he receives typed responses.

This test aims at fooling the tester. If the tester fails to determine machine's response from the human response, then the machine is said to be intelligent.

Properties of Environment

The environment has multifold properties –

- **Discrete / Continuous –** If there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete (For example, chess); otherwise it is continuous (For example, driving).
- **Observable / Partially Observable –** If it is possible to determine the complete state of the environment at each time point from the percepts it is observable; otherwise it is only partially observable.
- **Static / Dynamic –** If the environment does not change while an agent is acting, then it is static; otherwise it is dynamic.
- **Single agent / Multiple agents –** The environment may contain other agents which may be of the same or different kind as that of the agent.
- **Accessible / Inaccessible –** If the agent's sensory apparatus can have access to the complete state of the environment, then the environment is accessible to that agent.
- **Deterministic / Non-deterministic** – If the next state of the environment is completely determined by the current state and the actions of the agent, then the environment is deterministic; otherwise it is non-deterministic.
- **Episodic / Non-episodic –** In an episodic environment, each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself. Subsequent episodes do not depend on the actions in the previous episodes. Episodic environments are much simpler because the agent does not need to think ahead.

The below table lists the properties of a number of familiar environments.

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	**Partially**	**Multi**	**Stochastic**	**Sequential**	**Dynamic**	**Continuous**
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive. English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Several of the answers in the table depend on how the task environment is defined. We have listed the medical-diagnosis task as single-agent because the disease process in a patient is not profitably modeled as an agent; but a medical-diagnosis system might also have to deal with recalcitrant

patients and skeptical staff, so the environment could have a multiagent aspect. Furthermore, medical diagnosis is episodic if one conceives of the task as selecting a diagnosis given a list of symptoms; the problem is sequential if the task can include proposing a series of tests, evaluating progress over the course of treatment, and so on. Also, many environments are episodic at higher levels than the agent's individual actions. For example, a chess tournament consists of a sequence of games; each game is an episode because (by and large) the contribution of the moves in one game to the agent's overall performance is not affected by the moves in its previous game. On the other hand, decision making within a single game is certainly sequential

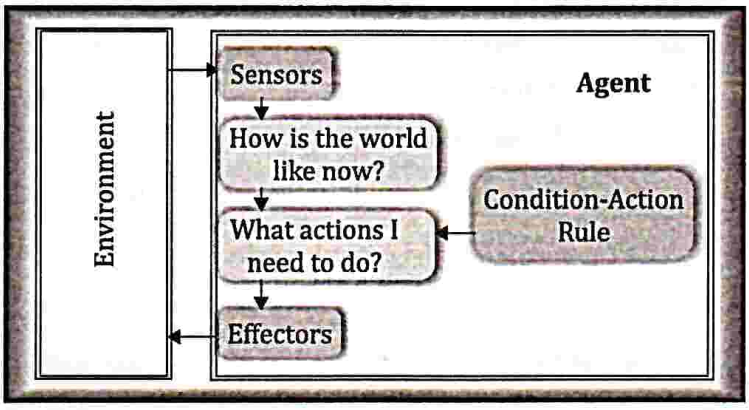
2.5 The Structure of Intelligent Agents

To understand the structure of Intelligent Agents, we should be familiar with Architecture and Agent programs. **Architecture** is the machinery that the agent executes on. It is a device with sensors and actuators, for example, a robotic car, a camera, and a PC. **An agent program** is an implementation of an agent function. **An agent function** is a map from the percept sequence(history of all that an agent has perceived to date) to an action.

Agent = Architecture + Agent Program

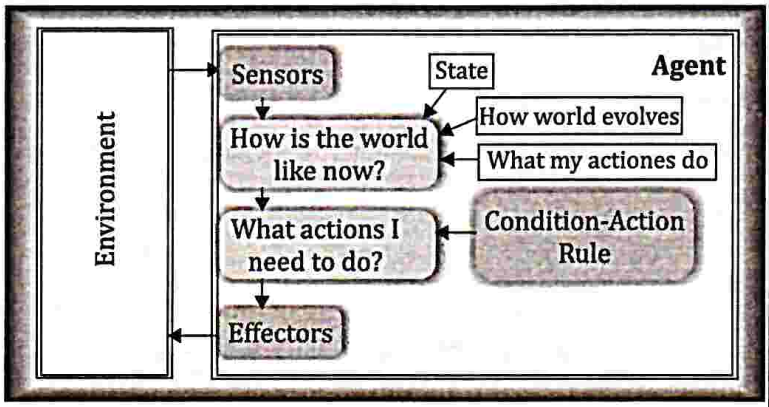
we outline five basic kinds of agent programs that embody the principles underlying almost all intelligent systems:

(a) Simple Reflex Agents



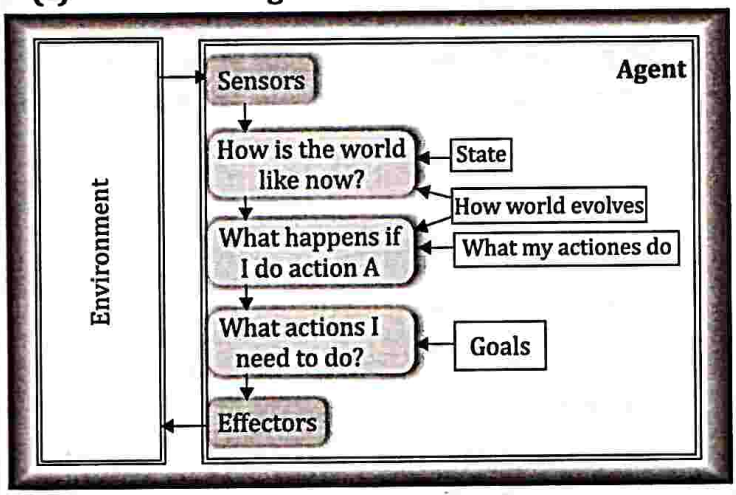
These agents perform actions using the current percept, rather than the percept history. The condition-action rule is used as the basis for the agent function. In this category, a fully observable environment is ideal for the success of the agent function.

(b) Model based Reflex Agents



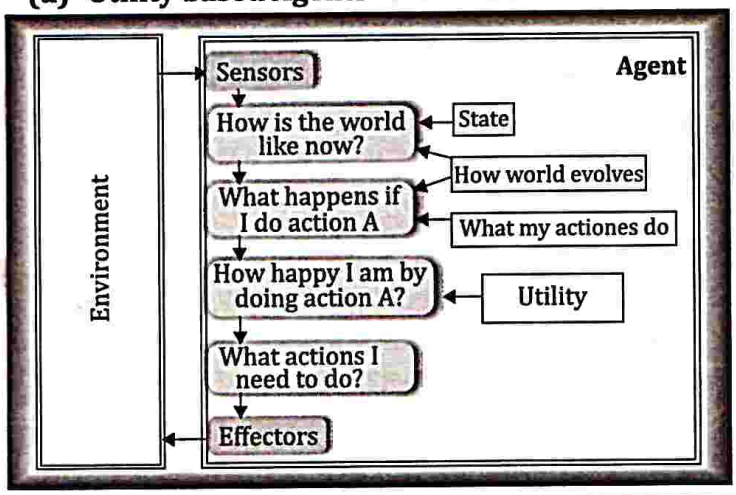
Unlike simple reflex agents, model-based reflex agents consider the percept history in their actions. The agent function can still work well even in an environment that is not fully observable. These agents use an internal model that determines the percept history and effect of actions. They reflect on certain aspects of the present state that have been unobserved..

(c) Goal based Agents



These agents have higher capabilities than model-based reflex agents. Goal-based agents use goal information to describe desirable capabilities. This allows them to choose among various possibilities. These agents select the best action that enhances the attainment of the goal.

(d) Utility based Agents

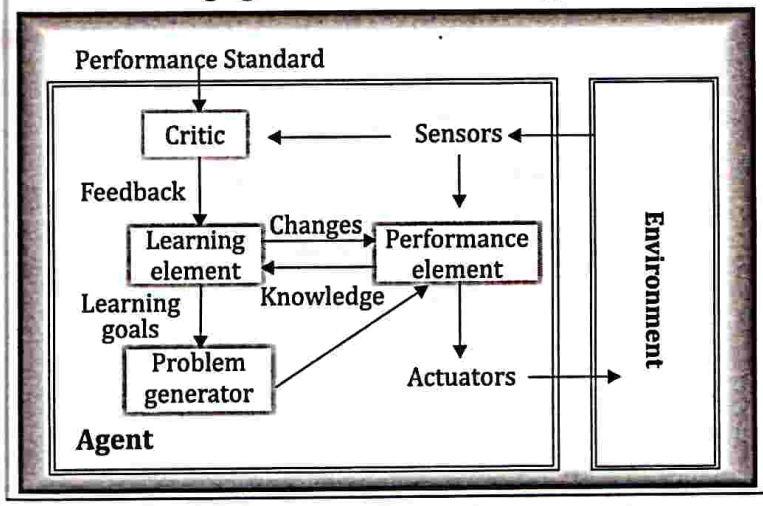


These agents make choices based on utility. They are more advanced than goal-based agents because of an extra component of utility measurement. Using a utility function, a state is mapped against a certain measure of utility. A rational agent selects the action that optimizes the expected utility of the outcome.

(e) Learning Agents

These are agents that have the capability of learning from their previous experience.

Learning agents have the following elements.



- The learning element: This element enables learning agents to learn from previous experiences.
- The critic: It provides feedback on how the agent is doing.
- The performance element: This element decides on the external action that needs to be taken.
- The problem generator: This acts as a feedback agent that performs certain tasks such as making suggestions (new) and keeping history.

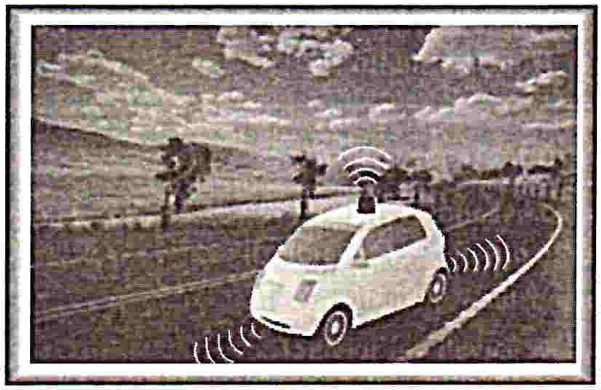
2.6 PEAS Representation

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model. It is made up of four words:

- P: Performance measure
- E: Environment
- A: Actuators
- S: Sensors

Here performance measure is the objective for the success of an agent's behavior.

PEAS for self-driving cars:



Let's suppose a self-driving car then PEAS representation will be:

- **Performance:** Safety, time, legal drive, comfort
- **Environment:** Roads, other vehicles, road signs, pedestrian
- **Actuators:** Steering, accelerator, brake, signal, horn
- **Sensors:** Camera, GPS, speedometer, odometer, accelerometer, sonar.

Other examples :

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.

Summary

- In artificial intelligence, **an agent** is a computer program or system that is designed to perceive its environment, make decisions and take actions to achieve a specific goal or set of goals.
- An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.
- Agents can be classified into different types based on their characteristics, such as whether they are ***reactive*** or ***proactive***, whether they have a ***fixed*** or ***dynamic*** environment, and whether they are ***single*** or ***multi-agent*** systems.
- An ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure, on the basis of (a) Its percept sequence (b) Its built-in knowledge base
- The most famous artificial environment is the **Turing Test environment**, in which one real and other artificial agents are tested on equal ground.
- **Turing test :** Two persons and a machine to be evaluated participate in the test. Out of the two persons, one plays the role of the tester. Each of them sits in different rooms. The tester is unaware of who is machine and who is a human. He interrogates the questions by typing and sending them to both intelligences, to which he receives typed responses.
- The environment has multifold properties –(a) Discrete / Continuous, (b) Observable / Partially Observable, (c) Static / Dynamic , (d) Single agent / Multiple agents, (e) Accessible / Inaccessible, (f) Deterministic / Non-deterministic and (g) Episodic / Non-*episodic*
- Agent's structure can be viewed as –Agent = Architecture + Agent Program

  Architecture = the machinery that an agent executes on.

  Agent Program = an implementation of an agent function.
- Five basic kinds of agent programs – (a) Simple Reflex agents (b) Model based Reflex agents (c) Goal based Agents (d) Utility based agents (e) Learning agents
- PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model. It is made up of four words: (a) P: Performance measure (b) E: Environment (c) A: Actuators and (d)S: Sensors

2.7 Review Questions

Short Answer Questions

1. What is an Intelligent agent ?
2. Define Agents & its environment.
3. What is Ideal Rational Agent?
4. Define Turning Test.
5. List any 4 properties of Environment.
6. Define agent function with an example.
7. What is Simple Reflex agent?
8. What is Model based reflex agent ?
9. What is Goal based agent ?
10. What is Utility based agent ?
11. What is learning agent ?
12. What do you mean by PEAS representation?

Long Answer Questions

1. With a neat diagram explain agents and its environment.
2. With a neat diagram explain the vacuum-cleaner world example.
3. Write a note on Concept of Rationality.
4. Discuss the various properties of environment with an example.
5. Explain the structure of Intelligent Agents.
6. Write a note on PEAS representation with an example.

UNIT - I

CHAPTER 3

PROBLEM SOLVING AGENTS

- Problem Solving Techniques in AI
  - Problem-solving agent
  - Problem Definition
  - Steps performed by Problem-solving agent
  - Example Problems
    - Toy Problems
    - Real world Problems
- Search Algorithms in Artificial Intelligence
  - Search Algorithm Terminologies:
  - Properties of Search Algorithms:
  - Types of search algorithms
- AND-OR graphs
- Types of Algorithms in Adversarial Search
  - Minmax Algorithm
  - Alpha-Beta Pruning
- Review Questions

3.1 Problem Solving Techniques in AI

The **reflex agents** are known as the simplest agents because they directly map states into actions. Unfortunately, these agents fail to operate in an environment where the mapping is too large to store and learn. **Goal-based agent**, on the other hand, considers future actions and the desired outcomes.

Here, we will discuss one type of goal-based agent known as a **problem-solving agent**, which uses atomic representation with no internal states visible to the problem-solving algorithms.

3.1.1. Problem-solving agent

The problem-solving agent performs precisely by defining problems and its several solutions.

- According to psychology, "a problem-solving refers to a state where we wish to reach to a definite goal from a present state or condition."
- According to computer science, a problem-solving is a part of artificial intelligence which encompasses a number of techniques such as algorithms, heuristics to solve a problem.

Therefore, a problem-solving agent is a goal-driven agent and focuses on satisfying the goal.

3.1.2 Problem Definition

To build a system to solve a particular problem, we need to do four things:

1. **Define the problem precisely:** This definition must include specification of the initial situations and also final situations which constitute (i.e) acceptable solution to the problem.
2. **Analyse the problem** (i.e) important features have an immense (i.e) huge impact on the appropriateness of various techniques for solving the problems.
3. **Isolate and represent the knowledge to solve the problem.**
4. **Choose the best problem** – solving techniques and apply it to the particular problem.

3.1.3 Steps performed by Problem-solving agent

- **Goal Formulation:** It is the first and simplest step in problem-solving. It organizes the steps/ sequence required to formulate one goal out of multiple goals as well as actions to achieve that goal. Goal formulation is based on the current situation and the agent's performance measure
- **Problem Formulation:** It is the most important step of problem-solving which decides what actions should be taken to achieve the formulated goal. There are following five components involved in problem formulation:
    - Initial State: It is the starting state or initial step of the agent towards its goal.
    - Actions: It is the description of the possible actions available to the agent.
    - Transition Model: It describes what each action does.
    - Goal Test: It determines if the given state is a goal state.
    - Path cost: It assigns a numeric cost to each path that follows the goal. The problem solving agent selects a cost function, which reflects its performance measure.

Remember, an optimal solution has the lowest path cost among all the solutions.

Note:

Initial state, actions, and **transition** model together define the **state-space** of the problem implicitly. State-space of a problem is a set of all states which can be reached from the initial state followed by any sequence of actions. The state-space forms a directed map or graph where nodes are the states, links between the nodes are actions, and the path is a sequence of states connected by the sequence of actions.

- **Search:** It identifies all the best possible sequence of actions to reach the goal state from the current state. It takes a problem as an input and returns solution as its output.
- **Solution:** It finds the best algorithm out of various algorithms, which may be proven as the best optimal solution.
- **Execution:** It executes the best optimal solution from the searching algorithms to reach the goal state from the current state.

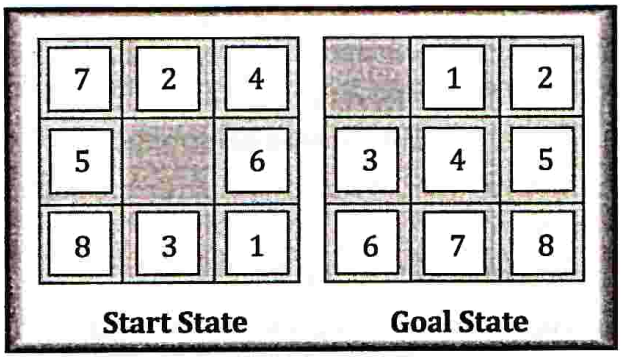
3.1.4 Example Problems

Basically, there are two types of problem approaches

- **Toy Problem:** It is a concise and exact description of the problem which is used by the researchers to compare the performance of algorithms.
- **Real-world Problem:** It is real-world based problems which require solutions. Unlike a toy problem, it does not depend on descriptions, but we can have a general formulation of the problem

3.1.4.1 Toy Problems

- **8 Puzzle Problem:** Here, we have a 3×3 matrix with movable tiles numbered from 1 to 8 with a blank space. The tile adjacent to the blank space can slide into that space. The objective is to reach a specified goal state similar to the goal state, as shown in the below figure. In the figure, our task is to convert the current state into goal state by sliding digits into the blank space.



In the above figure, our task is to convert the current(Start) state into goal state by sliding digits into the blank space.

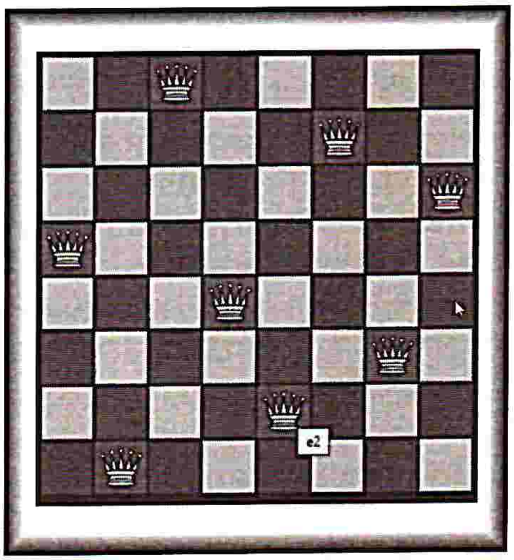
The problem formulation is as follows:

- States: It describes the location of each numbered tiles and the blank tile.
- Initial State: We can start from any state as the initial state
- Actions: Here, actions of the blank space is defined, i.e., either left, right, up or down
- Transition Model: It returns the resulting state as per the given state and actions.
- 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 problem is a type of sliding-block problem which is used for testing new search algorithms in artificial intelligence.

- **8-queens problem:** The aim of this problem is to place eight queens on a chessboard in an order where no queen may attack another. A queen can attack other queens either diagonally or in same row and column. From the following figure, we can understand the problem as well as its correct solution.



It is noticed from the above figure that each queen is set into the chessboard in a position where no other queen is placed diagonally, in same row or column. Therefore, it is one right approach to the 8-queens problem.

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 × 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

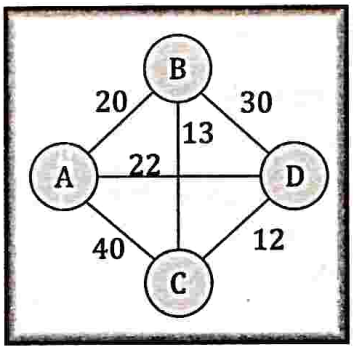
   - States: Arrangement of all the 8 queens one per column with no queen attacking the other queen.
   - Actions: Move the queen at the location where it is safe from the attacks.

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

3.1.4.2 Real World Problems

- **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.

  Example: Salesman has list of cities, each of which must be visited exactly once.
  - In list, there are direct roads between each pair of cities.
  - To find Route, salesman should follow shortest possible round trip.
  - State is represented as pair of any two cities and distance between them.



$(A \xrightarrow{20} B \xrightarrow{13} C \xrightarrow{12} D \xrightarrow{22} A) = 67$ (i)

$(A \xrightarrow{20} B \xrightarrow{30} D \xrightarrow{12} C \xrightarrow{40} A) = 102$

$(A \xrightarrow{22} D \xrightarrow{30} B \xrightarrow{13} C \xrightarrow{40} A) = 105$

$(A \xrightarrow{22} D \xrightarrow{12} C \xrightarrow{13} B \xrightarrow{20} A) = 67$ (ii)

$(A \xrightarrow{40} C \xrightarrow{12} D \xrightarrow{30} B \xrightarrow{20} A) = 102$

$(A \xrightarrow{40} C \xrightarrow{13} B \xrightarrow{30} D \xrightarrow{22} A) = 105$

The best shortest path is either (i) or (ii) & the cost of the path is 67.

- **VLSI Layout problem:** In this problem, millions of components and connections are positioned on a chip in order to minimize the area, circuit-delays, stray-capacitances, and maximizing the manufacturing yield. The layout problem is split into two parts:
  - Cell layout: Here, the primitive components of the circuit are grouped into cells, each performing its specific function. Each cell has a fixed shape and size. The task is to place the cells on the chip without overlapping each other.
  - Channel routing: It finds a specific route for each wire through the gaps between the cells

- **Protein Design:** The objective is to find a sequence of amino acids which will fold into 3D protein having a property to cure some disease
- **Robot Navigation** is a generalization of the route-finding problem described earlier. Rather than following a discrete set of routes, a robot can move in a continuous space with (in principle) an infinite set of possible actions and states. For a circular robot moving on a flat surface, the space is essentially two-dimensional. When the robot has arms and legs or wheels that must also be controlled, the search space becomes many-dimensional. Advanced techniques are required just to make the search space finite. We examine some of these methods in Chapter 25. In addition to the complexity of the problem, real robots must also deal with errors in their sensor readings and motor controls.

Example **Water Jug Problem**

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

The state space for this problem can be described as the set of ordered pairs of integers (x, y), such that x = 0,1,2,3 or 4 and y = 0, 1, 2, or 3;

x represents the number of gallons of water in the 4-gallon jug, and y represents the quantity of water in the 3-gallon jug.

- The **start state** is (0, 0).
- The **goal state** is (2, n), for any value of n. since the problem does not specify how many gallons need to be in the 3-gallon jug.

The operators are represented as rules:

- Whose left sides are matched against the current state and
- Whose right sides describe the new state that results from applying the rule.

Production Rules for the Water Jug Problem

1.	(x, y) if x < 4	→	(4; y)	Fill the 4-gallon jug
2.	(x, y) if y < 3 →		(x, 3)	Fill the 3-gallon jug
3.	(x, y) if x > 0	→	(x –d, y)	Pour some water out of the 4-gallon jug
4.	(x, y) if y > 0	→	(x, y – d)	Pour some water out of the 3-gallon jug
5.	(x, y) if x > 0	→	(0, y)	Empty the 4-gallon jug on the ground
6.	(x, y) if y > 0	→	(x, 0)	Empty the 3-gallon jug on the ground

7.	(x, y) if x + y > 4 and x > 0	→	(4,y-(4-x))	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8.	(x, y) if x + y > 3 and x > 0	→	(x-(3-y), 3)	Pour water from the 4 gallon jug into the 3-gallon jug until the 3-gallon jug is full
9.	(x, y) if x + y < 4 and y > 0	→	(x + y, 0)	Pour all the water from the 3-gallon jug into the 4-gallon jug
10.	(x, y) → if x + y < and x > 0		(0, x + y)	Pour all the water from the 4-gallon jug into the 3-gallon jug
11.	(0, 2) →		(2, 0)	Pour the 2 gallons from the 3-gallon jug into the 4-gallon jug
12.	(2, y)	→	(0, y)	Empty the 2 gallons in the 4-gallon jug on the ground

Assumptions:

- We can fill a jug from the pump,
- We can pour water out of a jug onto the ground,
- We can pour water from one jug to another, and
- No other measuring devices are available.

A control structure that loops through a simple cycle in which

- A rule whose left side matches the current state is chosen,
- The appropriate change to the state is made as described in the corresponding right side, and
- The resulting state is checked to see if it corresponds to a goal state.
- As long as it does not, the cycle continues.

Clearly the speed with which the problem gets solved depends on the mechanism that is used to select the next operation to be performed. There are several sequences of operators that solves this problem. One such sequence is shown in Table.

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

Table: One Solution to the Water Jug Problem

Summarizing what we have just said, in order to provide a formal description of a problem, we must do the following:

1. Define a state space that contains all the possible configurations of the relevant objects and perhaps some impossible ones.
2. Specify one or more states within that space that describe possible situations from which the problem-solving process may start. These states are called the initial states.
3. Specify one or more states that would be acceptable as solutions to the problem. These states are called goal states.
4. Specify a set of rules that describe the actions or operators available keeping in mind the following issues:
   - What unstated assumptions are present in the informal problem description
   - How general should the rules be?
   - How much of the work required to solve the problem should be precomputed and represented in the rules?

The problem can then be solved by using the rules, in combination with an appropriate control strategy, to move through the problem space until a path from an initial state to a goal state is found. Thus the process of search is fundamental to the problem-solving process.

3.2 Search Algorithms in Artificial Intelligence

Several of the fundamental ways that AI solves every challenge is through searching. These searching algorithms are used by rational agents or problem-solving agents for select the most appropriate answers. Intelligent entities use molecular representations and seem to be frequently main objective when finding solutions. Depending upon that calibre of the solutions they produce, most searching algorithms also have attributes of completeness, optimality, time complexity, and high computational.

3.2.1 Search Algorithm Terminologies:

* **Search:** Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
  a. Search Space: Search space represents a set of possible solutions, which a system may have.
  b. Start State: It is a state from where agent begins the search.
  c. Goal test: It is a function which observe the current state and returns whether the goal state is achieved or not.
* **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
* **Actions:** It gives the description of all the available actions to the agent.
* **Transition model:** A description of what each action do, can be represented as a transition model.
* **Path Cost:** It is a function which assigns a numeric cost to each path.

* **Solution:** It is an action sequence which leads from the start node to the goal node.
* **Optimal Solution:** If a solution has the lowest cost among all solutions.

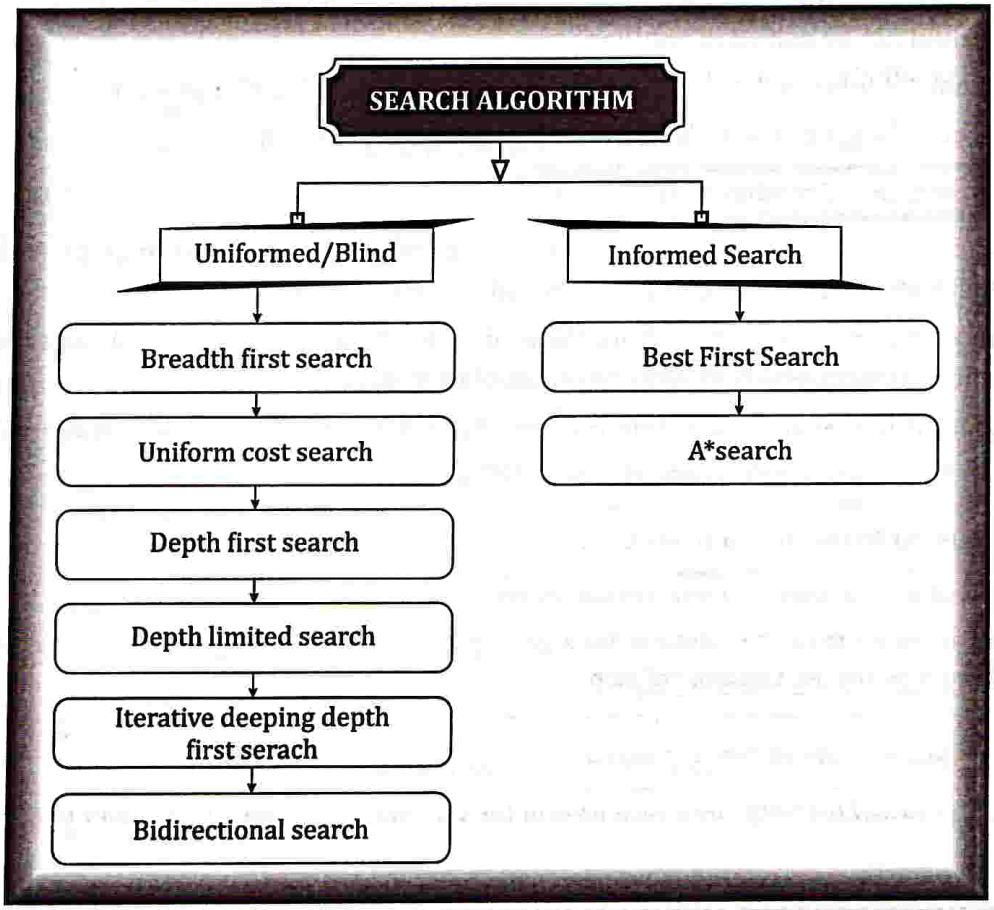
3.2.2 Properties of Search Algorithms:

Following are the four essential properties of search algorithms to compare the efficiency of these algorithms:

- **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

3.2.3 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.



1. **Uninformed/Blind Search:**

   The uninformed search does not contain any domain knowledge such as closeness, the location of the goal. It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes. Uninformed search applies 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 examines each node of the tree until it achieves the goal node.

   It can be divided into five main types:

   - Breadth-first search
   - Depth-first search
   - Bidirectional Search
   - Uniform cost search
   - Iterative deepening depth-first search

2. **Informed Search**

   Informed search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy. Informed search is also called a Heuristic search.

   A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time. Informed search can solve much complex problem which could not be solved in another way.

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

   - Greedy Best First Search
   - A* Search

Uninformed Search : Breadth-first Search

Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.

- 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.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

Advantages of Breadth-first Search

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

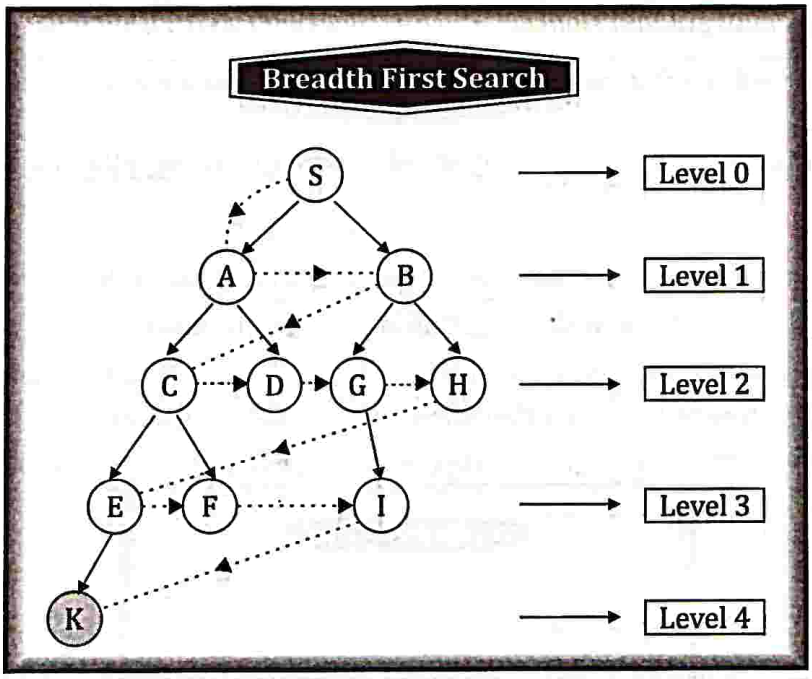
Disadvantages of Breadth-first Search

1. It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
2. BFS needs lots of time if the solution is far away from the root node.

Examples

In the below tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

$$S \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow G \rightarrow H \rightarrow E \rightarrow F \rightarrow I \rightarrow K$$



Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$$T(b) = 1+b^2+b^3+.....+ b^d = O(b^d)$$

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if path cost is a non-decreasing function of the depth of the node.

Uninformed Search : Depth-first Search

- Depth-first search isa recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.

Advantages of Depth-first Search

1. DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
2. It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

Disadvantages of Depth-first Search

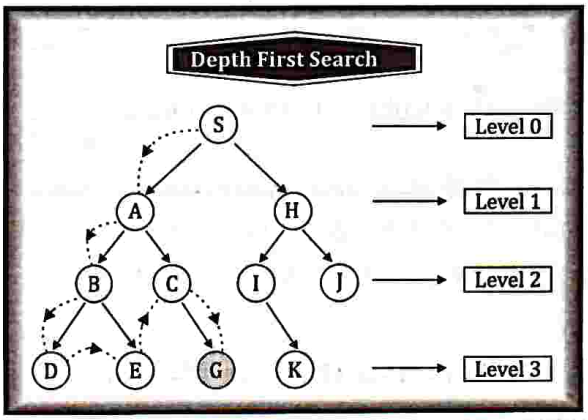
1. There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
2. DFS algorithm goes for deep down searching and sometime it may go to the infinite loop

Examples

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

Root node--->Left node ----> Right node.

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



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

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

$$T(n)= 1+ n^2+ n^3 +.....+ n^m=O(n^m)$$

Where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)

Space Complexity: DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is O(bm).

Informed Search : Best First Search

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both

algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e. **f(n)= g(n)**

Algorithm

Step 1 : Place the starting node into the OPEN list.

Step 2 : If the OPEN list is empty, Stop and return failure.

Step 3 : Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.

Step 4 : Expand the node n, and generate the successors of node n.

Step 5 : Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.

Step 6 : For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.

Step 7 : Return to Step 2.

Advantages of Best First Search

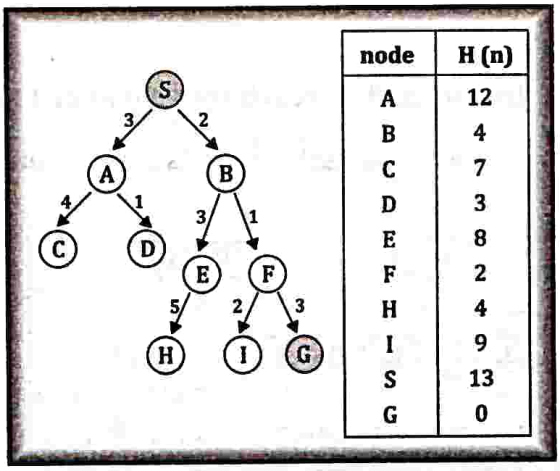
1. Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
2. This algorithm is more efficient than BFS and DFS algorithms.

Disadvantages of Best First Search

1. It can behave as an unguided depth-first search in the worst case scenario.
2. It can get stuck in a loop as DFS.
3. This algorithm is not optimal.

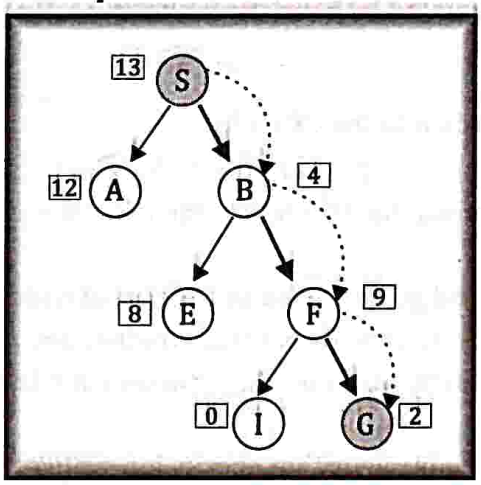
Examples

Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, we are using two lists which are OPEN and CLOSED Lists. Following are the iteration for traversing the above example.



Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

Iteration 1: Open [A], Closed [S, B]

Iteration 2: Open [E, F, A], Closed [S, B]

: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]

: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: S----> B----->F----> G

Time Complexity: The worst case time complexity of Greedy best first search is $O(b^m)$.

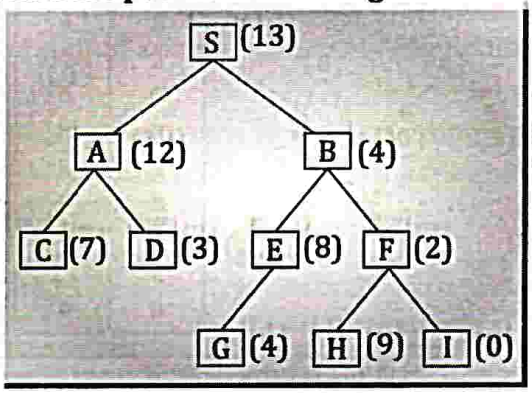
Space Complexity: The worst case space complexity of Greedy best first search is O(bm). Where, m is the maximum depth of the search space.

Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal : : Greedy best-first search is not optimal

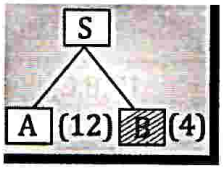
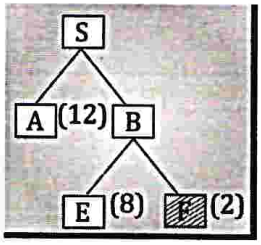
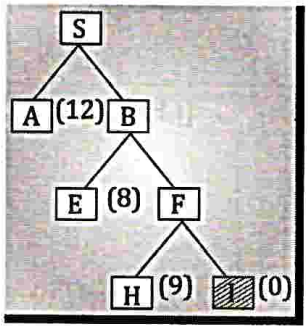
Problem : 1

Consider the following tree, find the path to reach the goal node I from S using BFS Algorithm.



Solution :

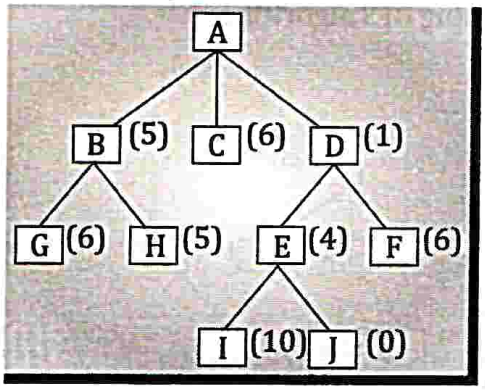
Note : S is the initial node and I is the goal node [since its heuristic value is 0].

Steps		Open Node	Close Node
1.		[S]	-
2.		[B,A]	[S]
3.		[F,E,A]	[S,B]
4.		[I,E,H,A]	[S,B,F]
			[S,B,F,I]

Therefore the path to reach the goal node is S → B → F → I.

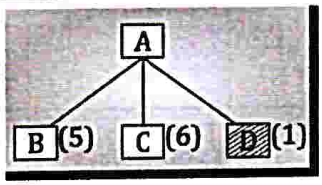
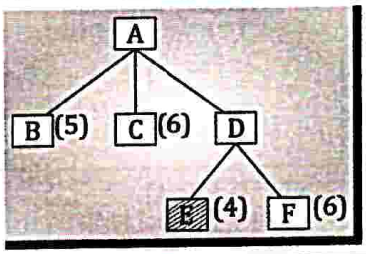
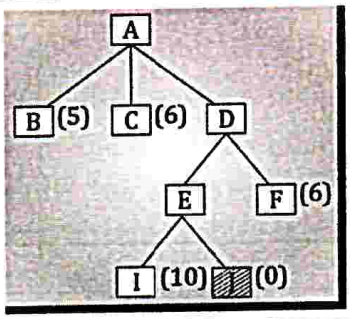
Problem : 2

Perform a BFS on the following tree.



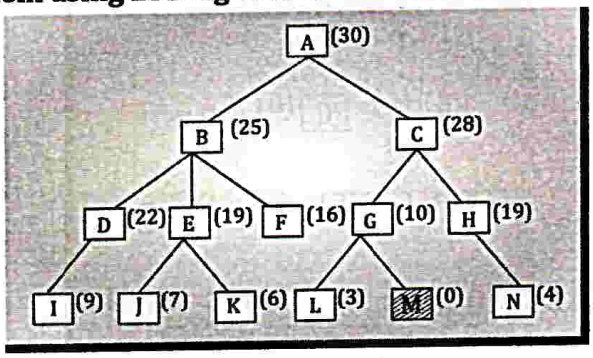
Solution :

Note : A is the initial node and
J is the goal node [since its heuristic value is 0].

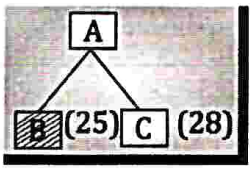
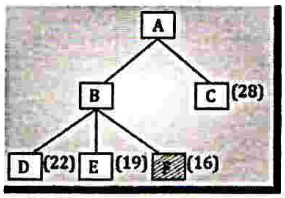
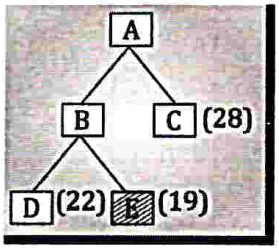
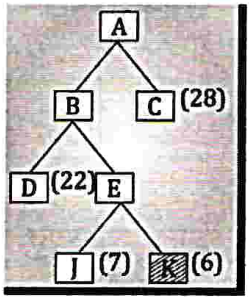
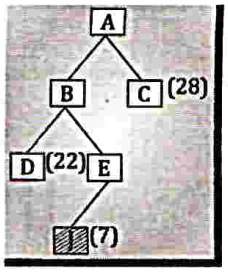
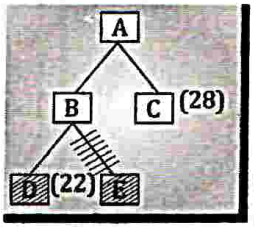
Step		Open node	Close node
1.		[A]	-
2.		[D,B,C]	[A]
3.		[E,B,C,F]	[A,D]
4.		[J,B,C,F,I]	[A,D,E]
			[A,D,E, J]

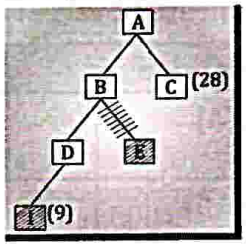
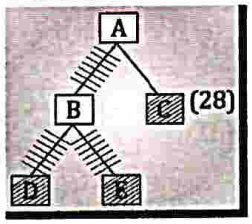
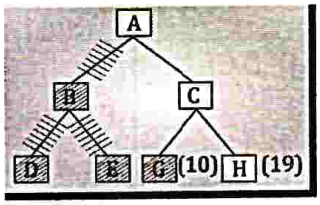
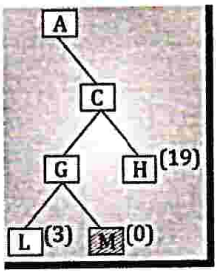
Problem : 3

Solve the following problem using BFS algorithm



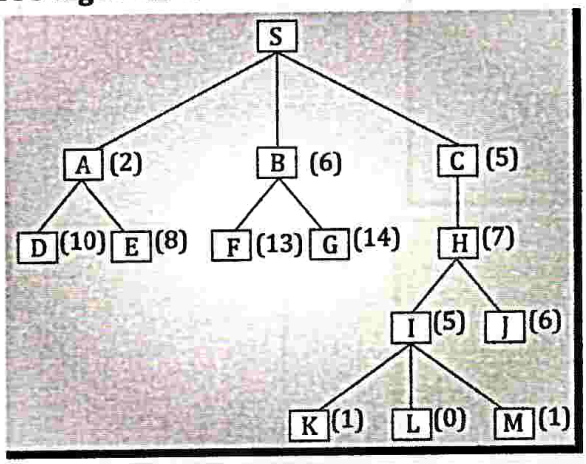
Note : A is the initial node and M is the goal node [since its heuristic value is 0].

Steps		Open Node	Close Node
1.		[A]	-
2.		[B,C]	[A]
3.		[F,E,D,C]	[A,B]
4.		[E,D,C]	[A,B,F] F has no children, so remove F. [A,B].
5.		[K,J,D,C]	[A,B,E]
6.		[J,D,C]	[A,B,E,K] K has no children so remove k [A,B,E]
7.		[D,C]	[A,B,E,J] J has no children so remove J. [A,B,E] E is discovered.

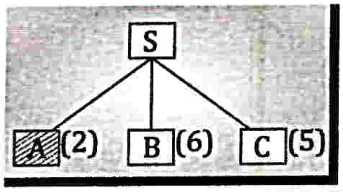
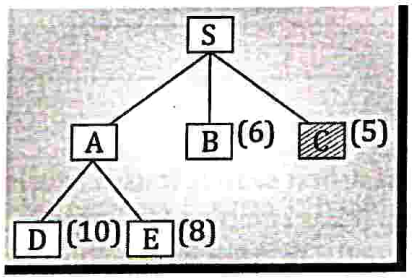
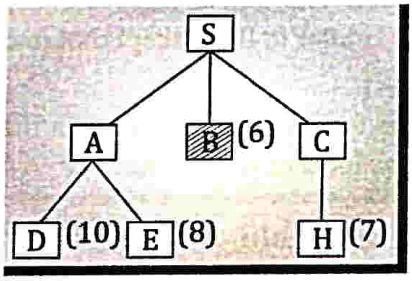
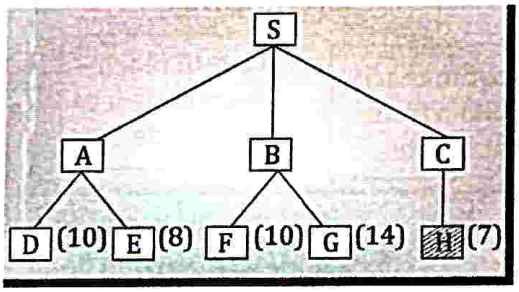
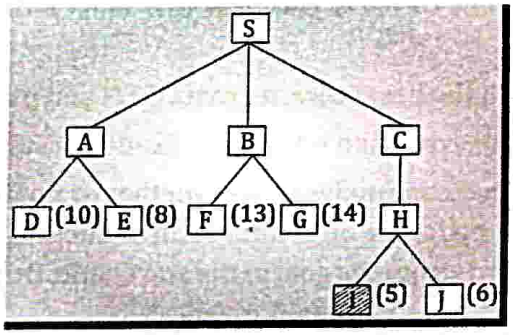
8.		[I,C]	[A,B,E,D]
9.		[C]	[A,B,E,D,I] I has no children, so remove I. [A,B,E,D] D is discovered B is also discovered
10.		[G,H]	[A,B,E,D,C]
11.		[M,L,H]	[A,B,E,D,C,G]
			[A,B,E,D,C,G,M]

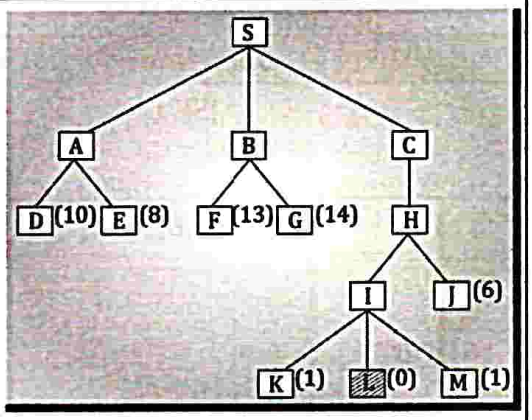
Problem : 4

Solve the following using BFS algorithm.



Note : S is the initial node and L is the goal node [since its heuristic value is 0].

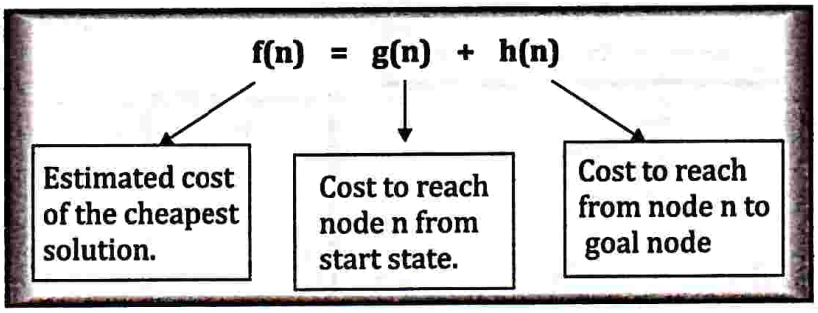
Step		Open Node	Close Node
1.		[S]	-
2.		[A,C,B]	[S]
3.		[C, B, E, D]	[S,A]
4.		[B, H, E, D]	[S, A, C]
5.		[H, E, D, F, G]	[S, A, C, B]
6.		[I, J, E, D, F, G]	[S, A, C, B, H]

7.		[L, K, M, J, E, D, F, G]	[S, A, C, B, H, I]
			[S, A, C, B, H, I, L]

Informed Search : A* Search Algorithm

A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a fitness number.



Algorithm of A* search:

1. The 1st step is to define the OPEN list with a single node, the starting node.
2. The 2nd step is to check whether or not OPEN is empty. If it is empty, then the algorithm returns failure and exits.
3. The 3rd step is to remove the node with the best score, n, from OPEN and place it in CLOSED.
4. The 4th step "expands" the node n, where expansion is the identification of successor nodes of n.
5. The 5th step then checks each of the successor nodes to see whether or not one of them is the goal node. If any successor is the goal node, the algorithm returns success and the solution, which consists of a path traced backwards from the goal to the start node. Otherwise, proceeds to the sixth step.

6. In 6th step, for every successor node, the algorithm applies the evaluation function, f, to it, then checks to see if the node has been in either OPEN or CLOSED. If the node has not been in either, it gets added to OPEN.
7. Finally, the 7th step establishes a looping structure by sending the algorithm back to the 2nd step. This loop will only be broken if the algorithm returns success in step 5 or failure in step 2.

Advantages of A* Search Algorithm

1. A* search algorithm is the best algorithm than other search algorithms.
2. A* search algorithm is optimal and complete.
3. This algorithm can solve very complex problems.

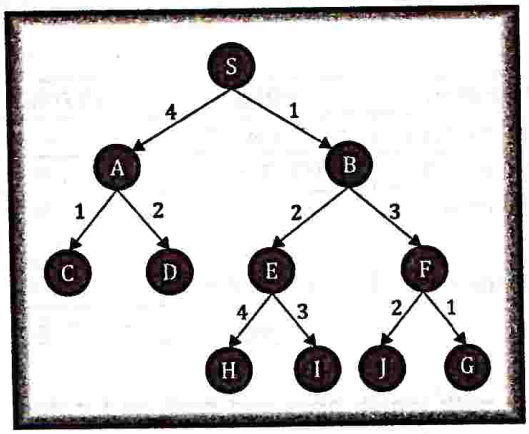
Disadvantages of A* Search Algorithm

1. It does not always produce the shortest path as it mostly based on heuristics and approximation.
2. A* search algorithm has some complexity issues.
3. The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Examples

In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula **f(n)= g(n) + h(n)**, where g(n) is the cost to reach any node from start state.

Here we will use OPEN and CLOSED list.



S: Initial state, G: goal.

Table shows the heuristic estimates:

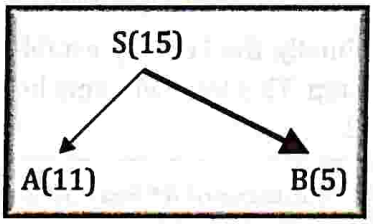
node	h(n)	node	h(n)	node	h(n)
A	11	E	4	I, J	3
B	5	F	2	S	15
C, D	9	H	7	G	0

Initialization : S (15)

Expand the nodes of S and add S into CLOSED

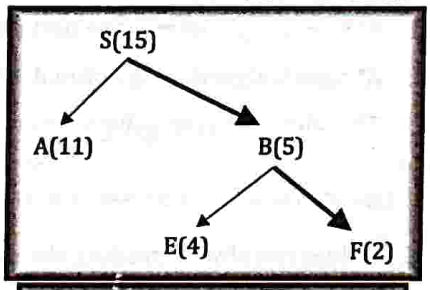
Iteration 1:

1. Add the successors of node s into OPEN and compute f(n)
2. Choose the most promising node (min h(n))
3. Remove the node from OPEN and add it into the CLOSED



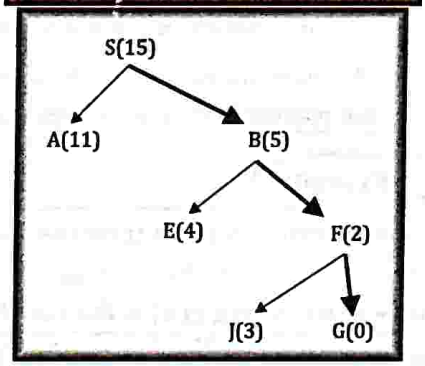
Iteration 2:

1. Add the successors of node B into OPEN and compute f(n)
2. Choose the most promising node (min h(n))
3. Remove the node from OPEN and add it into the CLOSED



Iteration 3:

1. Add the successors of node F into OPEN and compute f(n)
2. Choose the most promising node (min h(n))
3. Remove the node from OPEN and add it into the CLOSED



The shortest path form S -> G is S -> B -> F -> G

Total cost: 1+3+1=4

Iteration	OPEN	CLOSED
Initialization	A, B	S
Iteration 1	A	S, B
Iteration 2	F, E, A	S, B
	E, A	S, B, F
Iteration 3	J, G, E, A	S, B, F
	J, E, A	**S, B, F, G**

Points to remember:

- A* algorithm returns the path which occurred first, and it does not search for all remaining paths.
- The efficiency of A* algorithm depends on the quality of heuristic.
- A* algorithm expands all nodes which satisfy the condition f(n)

Complete: A* algorithm is complete as long as:

- Branching factor is finite.
- Cost at every action is fixed.

Optimal: A* search algorithm is optimal if it follows below two conditions:

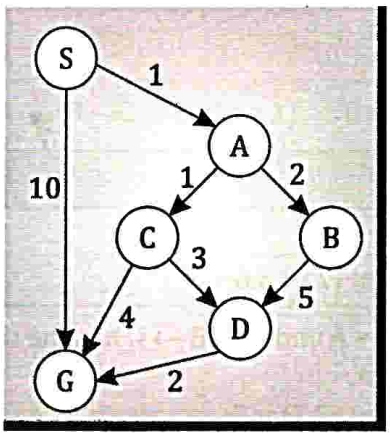
- **Admissible:** the first condition requires for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
- **Consistency:** Second required condition is consistency for only A* graph-search.

If the heuristic function is admissible, then A* tree search will always find the least cost path.

Time Complexity: The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.

Space Complexity: The space complexity of A* search algorithm is O(b^d)

Problem : 1



Heuristic values	
S	5
A	3
B	4
C	2
D	6
G	0

Using A* algorithm find the most cost – effective path from s to G.

Solution :

Step	Path	f' = g + h'	
1.	S → A S → G	1 + 3 = 4 10 + 0 = 10	[Select the least f'] mini (4,10) = 4
2.	S →A→B S→A→C	(1 + 2) + 4 = 7 (1 + 1) + 2 = 4	Mini (10,7,4) = 4
3.	S→A→C→D S→A→C→G	(1 + 1 + 3) + 6 = 11 (1 + 1 + 4) + 0 = 6	 G is the goal node.

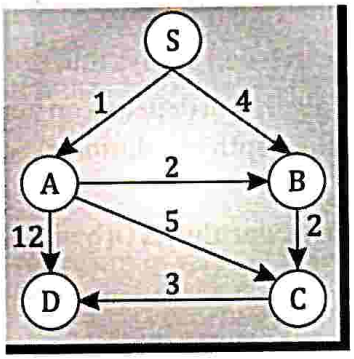
∴ The cost-effective path is

S→A→C→G

And the cost is 6

Problem : 2

Using A* algorithm find the optimal path.



Heuristic values	
S	7
A	6
B	2
C	1
D	0

Solution :

Since f' of D is 0, D is the goal node.

Step	Path	f' = g + h'	
1.	S → A S → B	1 + 6 = 7 4 + 2 = 6	Mini (7,6) = 6
2.	S → B → C	(4 + 2) + 1 = 7	S → A and S → B → C has the same least cost. Mini (7,7) = 7 Selected the first 7 [S → A]
3.	S → A → B S → A → C S → A → D	(1 + 2) + 2 = 5 (1 + 5) + 1 = 7 (1 + 12) + 0 = 13	Mini (7,5,7,13) = 5
4.	S → A → B → C	(1 + 2 + 2) + 1 = 6	Mini (7,7,13,6) = 6
5.	S → A → B → C → D	(1 + 2 + 2 + 3) + 0 =8	D is the goal node

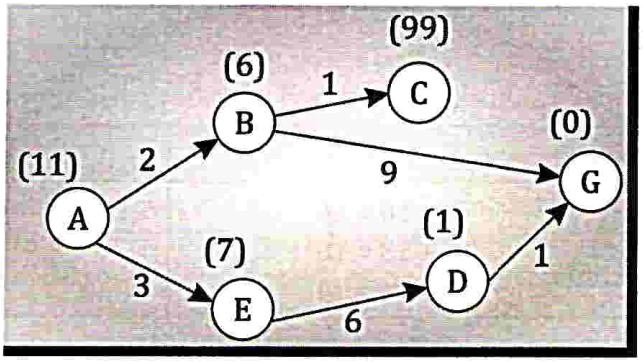
∴ The optional path is

S → A → B → C → D

And the minimum cost is 8.

Problem : 3

Using A* algorithm find the optional path for the following graph. A is the starting node and G is the goal node.



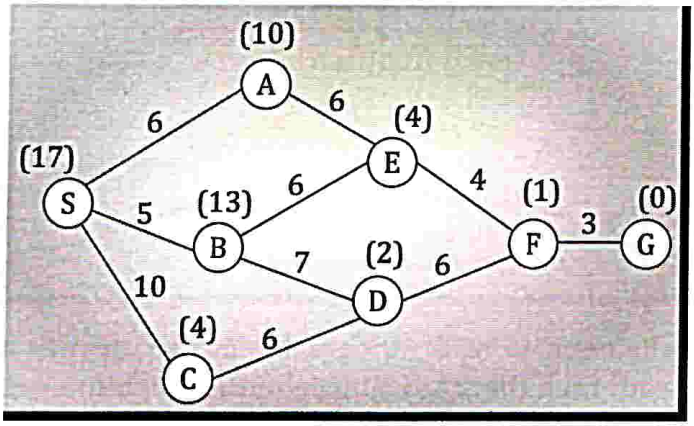
Solution :

Step	Path	f' = g + h'	
1.	A → B A → E	2 + 6 = 8 3 + 7 = 10	Mini (8,10) = 8
2.	A → B → C A → B → G	(2 + 1) + 99 = 102 (2 + 9) + 0 = 11	Mini (10,102,11) = 10 A → B → G has the least cost but it is more than the cost A → E. Therefore we explore A → E path.
3.	A → E → D	(3 + 6) + 1 = 10	Mini (102, 11, 10) = 10
4.	A → E → D → G	(3 + 6 + 1) + 0 = 10	G is the goal node

∴ The optimal path is A → E → D → G and
Minimum cost = 10.

Problem : 4

Perform A* algorithm on the following graph.



Solution :

Goal node is G

Step	Path	f = g + h'	
1.	S → A S → B S → C	6 + 10 = 16 5 + 13 = 18 10 + 4 = 14	
2.	S → C → D	(10 + 6) + 2 = 18	Minimum value is 16 for path S → A.
3.	S → A → E	(6 + 6) + 4 = 16	
4.	S → A → E → F S → A → E → B	(6 + 6 + 4) + 1 = 17 (6 + 6+ 6) + 13 = 31	
5.	S → A → E → F → G	(6 + 6 + 4 + 3) + 0 = 19	

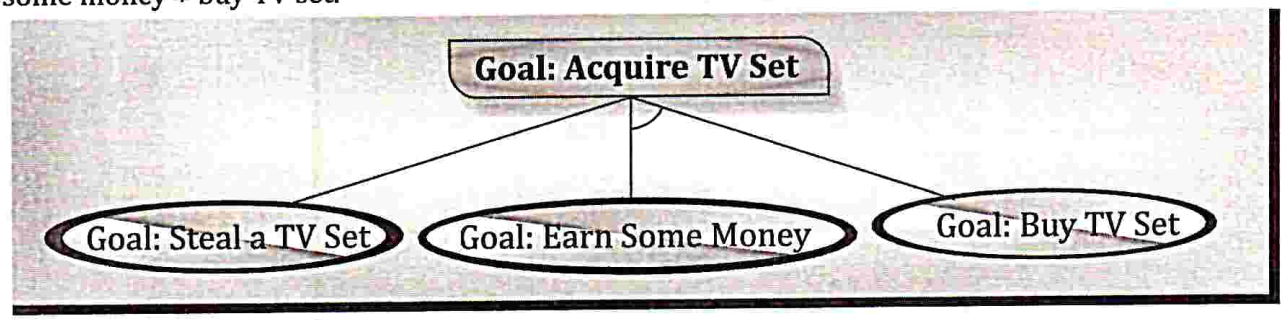
∴ The path is

S → A → E → F → G

3.3 AND-OR graphs

The AND-OR graph (or tree), is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems, all of which must then be solved. This decomposition, or reduction, generates AND arcs. One AND arc may point to any number of successor node all of which must be solved in order for the arc to point to a solution. Just as in an OR graph, several arcs may emerge from a single node, indicating a variety of ways in which the original problem might be solved. This is why the structure is called AND-OR graph.

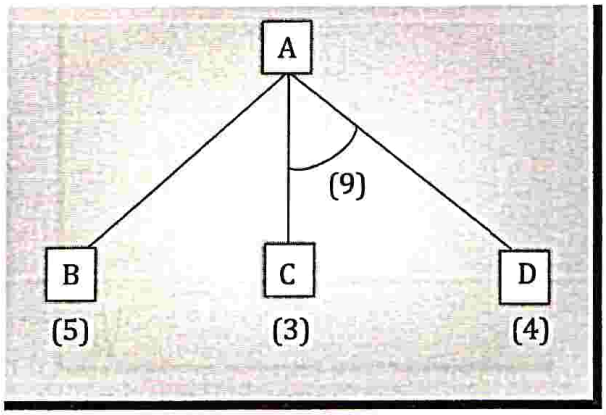
An example of an AND-OR graph is given in Figure 1.3. AND arcs are indicated with a line connecting all the components. In order to acquire TV set either earn some money and buy TV set or Steal a TV. Here the complete problem (Acquire TV set) is dived into subproblems; either steal a TV set or earn some money + buy TV set.



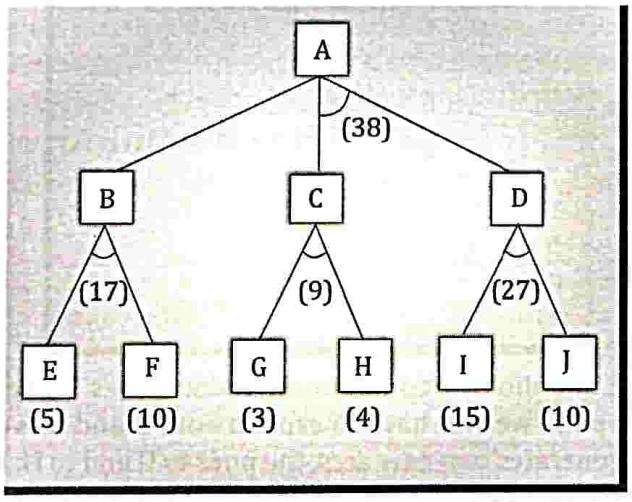
An algorithm to find a solution in an AND - OR graph must handle AND area appropriately. This algorithm should find a path from the starting node of the graph to a set of nodes representing solution states. Notice that it may be necessary to get to more than one solution since each arm of an AND arc must lead to its own solution node. A* algorithm cannot search AND - OR graphs efficiently therefore AO* Algorithm is used to handle AND-OR graphs.

To see why our best first search algorithm is not adequate for searching AND-OR graphs, consider below Figure.

The top node, A, has been expanded, producing two arcs, one leading to B and one leading to C and D. The numbers at each node represent the value of f at that node. We assume, for simplicity, that every operation has a uniform cost, so each arc with a single successor has a cost of 1 and each AND arc with multiple successors has also a cost of 1 for each of its components. If we look just at the nodes and choose for expansion the one with the lowest f value, we must select C. But using the information now available, it would be better to explore the path going through B since to use C we must also use D, for a total cost of 9 (3+4+1+1) compared to the cost of 6 (5+ 1) that we get by going through B. The problem is that the choice of which node to expand next must depend not only on the f' value of that node but also on whether that node is part of the current best path from the initial node.

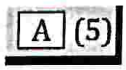


Consider below figure. The most promising node is G with an f' value of 3. It is even part of the most promising arc G-H, with a total cost of 9 (3+4+1+1). But that arc is not part of the current best path, since to use it we must also use the arc I-J with a cost of 27 (15+10+1+1). The path from A, through B to E and F is better with a total cost of 18 (17 + 1). So we should not expand G next; instead we should examine either E or F.



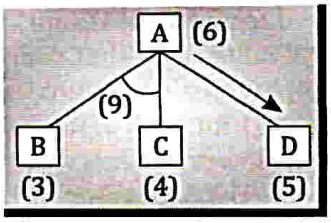
Let us perform a search operation on another AND-OR graph

- Start with the starting node



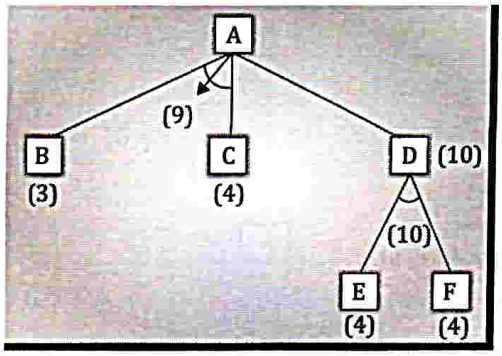
A is the only node, so it is at the end of the current best path.

- A is expanded, yielding nodes B, C, and D.



The arc to D is labeled as the most promising one emerging from A, since it costs 6 (5 + 1) compared to B and C, which costs 9(3+1+4+1).

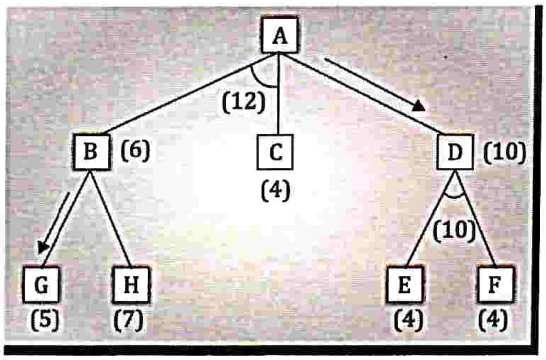
- Node D is chosen for expansion.



This process produces one new arc, the AND arc to E and F, with a combined cost estimate of 10(4 + 1+ 4 +1).

So we update the f' value of D to 10.

- Going back one more level, we see that the AND arc B-C better than the arc to D, so it is labeled as the current best path.



We traversed from A and discovered the unexpanded nodes B and C. If we are going to find a solution along this path, we will have to expand both B and C eventually, so let's choose to explore B first. This generates two new arcs, the ones to G and to H. Propagating their f' values backward, we update f' of B to 6 (5 + 1) . This requires updating the cost of the AND arc B-C to 12 (6+1+4+1).

- After doing that, the arc to D is again the better path from A, so we record it as the current best path and either node E or node F will be chosen for expansion. This process continues until either a solution is found or all paths have led to dead ends, indicating that there is no solution.

Informed Search : AO* Search Algorithm

The Depth first search and Breadth first search given earlier for OR trees or graphs can be easily adopted by AND-OR graph. The AO* method divides any given difficult problem into a smaller group of problems that are then resolved using the AND-OR graph concept. AND OR graphs are specialized graphs that are used in problems that can be divided into smaller problems. 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.

Like A* algorithm here we will use two arrays and one heuristic function.

- OPEN: It contains the nodes that have been traversed but yet not been marked solvable or unsolvable.
- CLOSE: It contains the nodes that have already been processed.
- h(n): The distance from the current node to the goal node.

The evaluation function in AO* looks like this:

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

f(n) = Actual cost + Estimated cost

here,

f(n) = The actual cost of traversal.

g(n) = the cost from the initial node to the current node.

h(n) = estimated cost from the current node to the goal state.

Algorithm

Step 1 : Place the starting node into OPEN.

Step 2 : Compute the most promising solution tree say T0.

Step 3 : Select a node n that is both on OPEN and a member of T0. Remove it from OPEN and place it in CLOSE

Step 4 : If n is the terminal goal node then levelled n as solved and levelled all the ancestors of n as solved. If the starting node is marked as solved then success and exit.

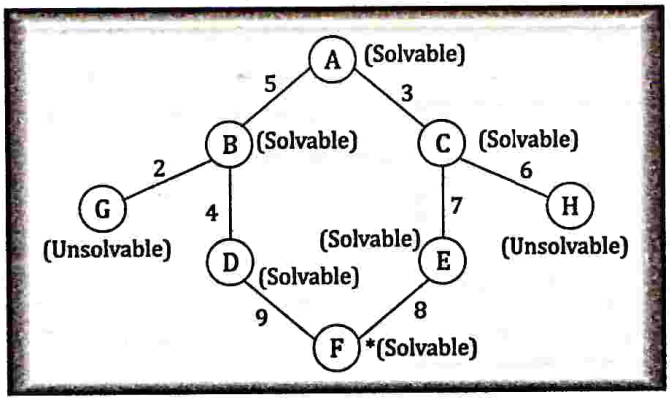
Step 5 : If n is not a solvable node, then mark n as unsolvable. If starting node is marked as unsolvable, then return failure and exit.

Step 6 : Expand n. Find all its successors and find their h (n) value, push them into OPEN.

Step 7 : Return to Step 2.

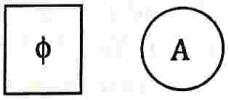
Step 8 : Exit.

Example :



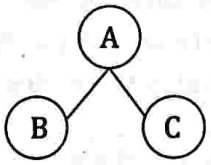
Step 1 : In the above graph, the solvable nodes are A, B, C, D, E, F and the unsolvable nodes are G, H. Take A as the starting node. So place A into OPEN.

i.e., OPEN = | A | CLOSE = (NULL) | ϕ |



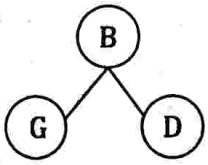
Step 2 : The Children of A are B and C which are solvable. So place them into OPEN and place A into the CLOSE.

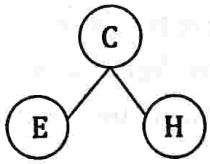
i.e., OPEN = | B | C | CLOSE = | A |



Step 3 : Now process the nodes B and C. The children of B and C are to be placed into OPNE. Also remove B and C from OPEN and place them into CLOSE.

So OPEN = | G | D | E | | c | A | B | C |

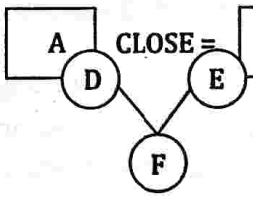




O -> indicates G and H are unsolvable

Step 4 : As the nodes G and H are unsolvable, so place them into CLOSE directly and process the nodes D and E.

i.e., OPEN = | A | CLOSE = | A | B | C | | $G^{(0)}$ | D | E | | $H^{(0)}$ |

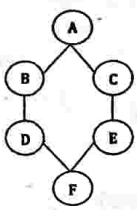


Step 5 : Now we have been reached at our goal state. So place F into CLOSE.

A	B	C		$G^{(0)}$	D	E		$H^{(0)}$	F

Step 6 :

AO* Graph



Advantages of AO* Star

1. It is an optimal algorithm.
2. If traverse according to the ordering of nodes.
3. It can be used for both OR and AND graph.

Disadvantages of AO* Star

1. Sometimes for unsolvable nodes, it can't find the optimal path.
2. Its complexity is than other algorithms.

Difference between the A* Algorithm and AO* algorithm

- A* algorithm and AO* algorithm both works on the best first search.
- They are both informed search and works on given heuristics values.
- A* always gives the optimal solution but AO* doesn't guarantee to give the optimal solution.
- Once AO* got a solution doesn't explore all possible paths but A* explores all paths.
- When compared to the A* algorithm, the AO* algorithm uses less memory.
- opposite to the A* algorithm, the AO* algorithm cannot go into an endless loop.

Means End Analysis in AI

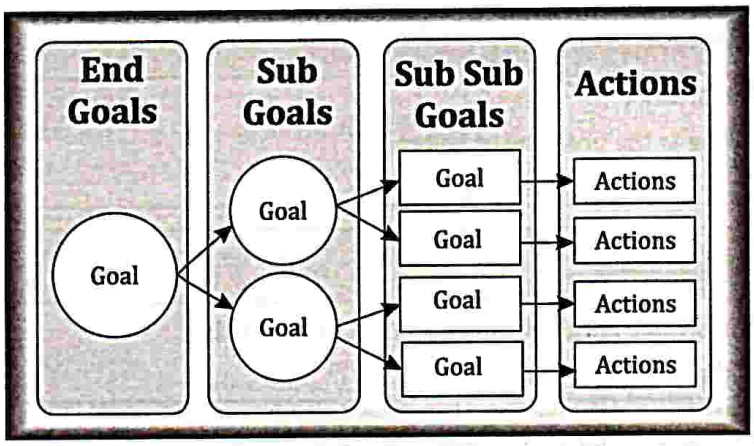
Means end analysis (MEA) is an important concept in artificial intelligence (AI) because it enhances problem resolution. MEA solves problems by defining the goal and establishing the right action plan. This technique is used in AI programs to limit search.

How MEA works

Means end analysis uses the following processes to achieve its objectives:

1. First, the system evaluates the current state to establish whether there is a problem. If a problem is identified, then it means that an action should be taken to correct it.
2. The second step involves defining the target or desired goal that needs to be achieved.
3. The target goal is split into sub-goals, that are further split into other smaller goals.
4. This step involves establishing the actions or operations that will be carried out to achieve the end state.
5. In this step, all the sub-goals are linked with corresponding executable actions (operations).
6. After that is done, intermediate steps are undertaken to solve the problems in the current state. The chosen operators will be applied to reduce the differences between the current state and the end state.
7. This step involves tracking all the changes made to the actual state. Changes are made until the target state is achieved.

The following image shows how the target goal is divided into sub-goals, that are then linked with executable actions.



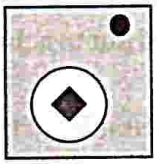
Algorithm steps for Means End Analysis

The following are the algorithmic steps for means end analysis:

1. Conduct a study to assess the status of the current state. This can be done at a macro or micro level.
2. Capture the problems in the current state and define the target state. This can also be done at a macro or micro level.
3. Make a comparison between the current state and the end state that you defined. If these states are the same, then perform no further action. This is an indication that the problem has been tackled. If the two states are not the same, then move to step 4.
4. Record the differences between the two states at the two aforementioned levels (macro and micro).
5. Transform these differences into adjustments to the current state.
6. Determine the right action for implementing the adjustments in step 5.
7. Execute the changes and compare the results with the target goal.
8. If there are still some differences between the current state and the target state, perform course correction until the end goal is achieved.

Example of problem-solving in Means End Analysis

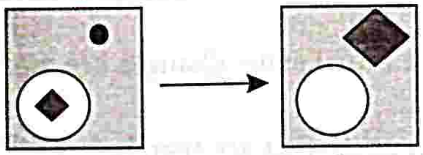
Let's assume that we have the following initial state.



Initial State

We want to apply the concept of means end analysis to establish whether there are any adjustments needed. The first step is to evaluate the initial state, and compare it with the end goal to establish whether there are any differences between the two states.

The following image shows a comparison between the initial state and the target state.



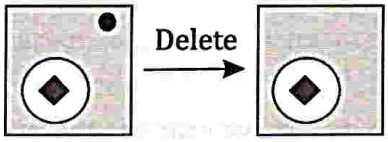
Initial State Goal State

The image above shows that there is a difference between the current state and the target state. This indicates that there is a need to make adjustments to the current state to reach the end goal.

The goal can be divided into sub-goals that are linked with executable actions or operations.

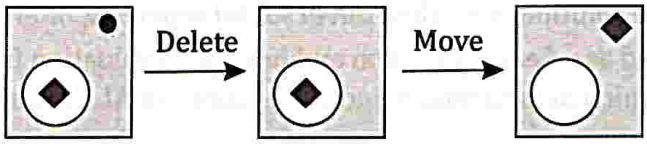
The following are the three operators that can be used to solve the problem.

1. **Delete operator:** The dot symbol at the top right corner in the initial state does not exist in the goal state. The dot symbol can be removed by applying the delete operator.



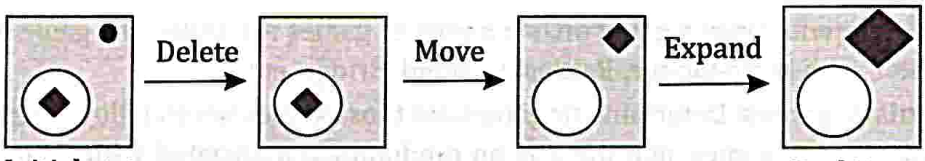
Initial state

2. **Move operator:** We will then compare the new state with the end state. The green diamond in the new state is inside the circle while the green diamond in the end state is at the top right corner. We will move this diamond symbol to the right position by applying the move operator.



Initial state

3. **Expand operator:** After evaluating the new state generated in step 2, we find that the diamond symbol is smaller than the one in the end state. We can increase the size of this symbol by applying the *expand operator*.



Initial state

Goal State

After applying the three operators above, we will find that the state in step 3 is the same as the end state. There are no differences between these two states, which means that the problem has been solved.

Applications of Means End Analysis

Means end analysis can be applied in the following fields:

- **Organizational Planning :**

  Means end analysis is used in organizations to facilitate general management. It helps organizational managers to conduct planning to achieve the objectives of the organization. The management reaches the desired goal by dividing the main goals into sub-goals that are linked with actionable tasks.

- **Business Transformation**

  This technique is used to implement transformation projects. If there are any desired changes in the current state of a business project, means end analysis is applied to establish the new processes to be implemented. The processes are split into sub-processes to enhance effective implementation.

- **Gap Analysis**

  Gap analysis is the comparison between the current performance and the required performance. Means end analysis is applied in this field to compare the existing technology and the desired technology in organizations. Various operations are applied to fill the existing gap in technology.

Adversarial Search

- Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us. A conflicting goal is given to the agents (multiagent). These agents compete with one another and try to defeat one another in order to win the game. Such conflicting goals give rise to the adversarial search.
- So, 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. Further games can be classified into :
- **Perfect Information:** A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.
- **Imperfect Information:** If in a game agents do not have all information about the game and not aware with what's going on, such type of games are called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.
- **Deterministic games:** Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.
- **Non-deterministic games:** Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

  Type of Games in AI :

Zero-Sum Game

- Zero-sum games are adversarial search which involves pure competition.
- In Zero-sum game each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent.
- One player of the game try to maximize one single value, while other player tries to minimize it.
- Each move by one player in the game is called as ply.
- Chess and tic-tac-toe are examples of a Zero-sum game.

Elements of Game Playing Search

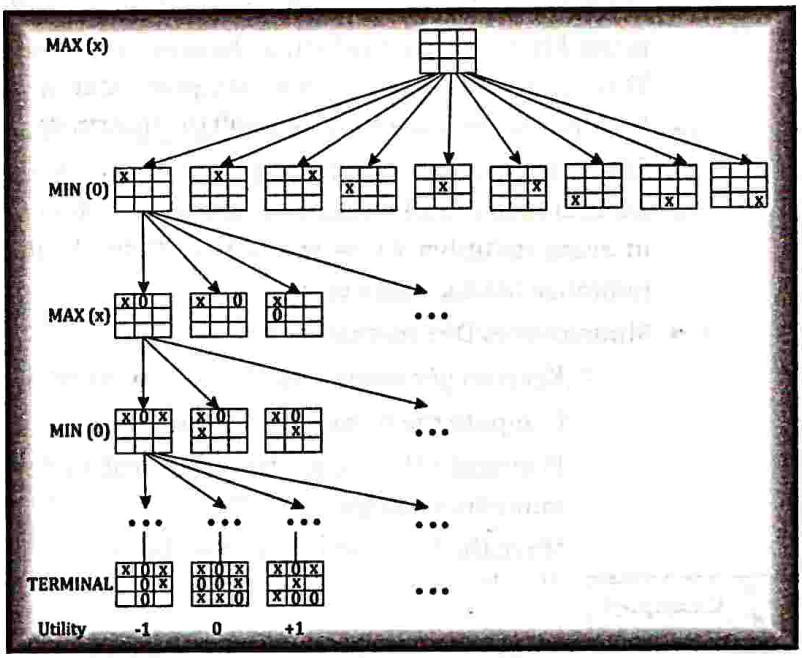
To play a game, we use a game tree to know all the possible choices and to pick the best one out. There are following elements of a game-playing:

- S0: It is the initial state from where a game begins.
- PLAYER (s): It defines which player is having the current turn to make a move in the state.
- ACTIONS (s): It defines the set of legal moves to be used in a state.
- RESULT (s, a): It is a transition model which defines the result of a move.
- TERMINAL-TEST (s): It defines that the game has ended and returns true.
- UTILITY (s,p): It defines the final value with which the game has ended. This function is also known as Objective function or Payoff function. The price which the winner will get i.e.
- (-1): If the PLAYER loses.
- (+1): If the PLAYER wins.
- (0): If there is a draw between the PLAYERS.

For example, in chess, tic-tac-toe, we have two or three possible outcomes. Either to win, to lose, or to draw the match with values +1,-1 or 0.

*"A **game tree** is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function."*

Let's understand the working of the elements with the help of a game tree designed for tic-tac-toe. Here, the node represents the game state and edges represent the moves taken by the players.



There are two players MAX and MIN.

Players have an alternate turn and start with MAX.

MAX maximizes the result of the game tree

MIN minimizes the result.

- INITIAL STATE (S0): The top node in the game-tree represents the initial state in the tree and shows all the possible choice to pick out one.
- PLAYER (s): There are two players, MAX and MIN. MAX begins the game by picking one best move and place X in the empty square box.
- ACTIONS (s): Both the players can make moves in the empty boxes chance by chance.
- RESULT (s, a): The moves made by MIN and MAX will decide the outcome of the game.

- TERMINAL-TEST(s): When all the empty boxes will be filled, it will be the terminating state of the game.
- UTILITY: At the end, we will get to know who wins: MAX or MIN, and accordingly, the price will be given to them.

3.4 Types of Algorithms in Adversarial Search

In a normal search, we follow a sequence of actions to reach the goal or to finish the game optimally. But in an adversarial search, the result depends on the players which will decide the result of the game. It is also obvious that the solution for the goal state will be an optimal solution because the player will try to win the game with the shortest path and under limited time.

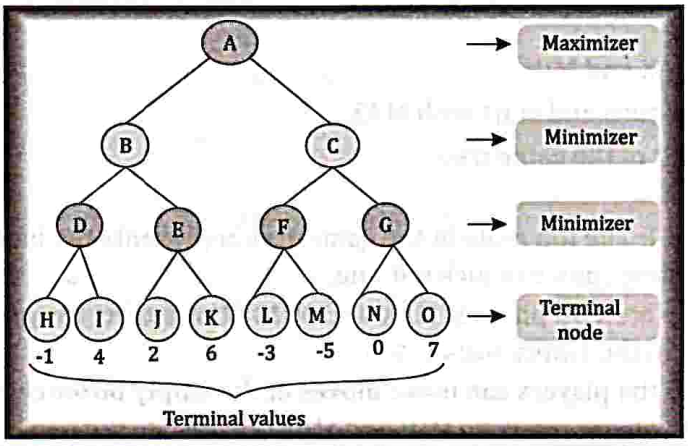
There are following types of adversarial search:

* Minmax Algorithm
* Alpha-beta Pruning.

3.4.1 Minmax Algorithm

- In artificial intelligence, minimax is a decision-making strategy under game theory, which is used to minimize the losing chances in a game and to maximize the winning chances. This strategy is also known as 'Minmax,' 'MM,' or 'Saddle point.'
- In minimax strategy, the result of the game or the utility value is generated by a heuristic function by propagating from the initial node to the root node. It follows the backtracking technique and backtracks to find the best choice. MAX will choose that path which will increase its utility value and MIN will choose the opposite path which could help it to minimize MAX's utility value.
- Minmax uses DFS method as follows :
  - Keep on generating the game tree/ search tree till a limit d.
  - Compute the move using a heuristic function.
  - Propagate the values from the leaf node till the current position following the minimax strategy.
  - Make the best move from the choices.

Examples



Step 1 : In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.

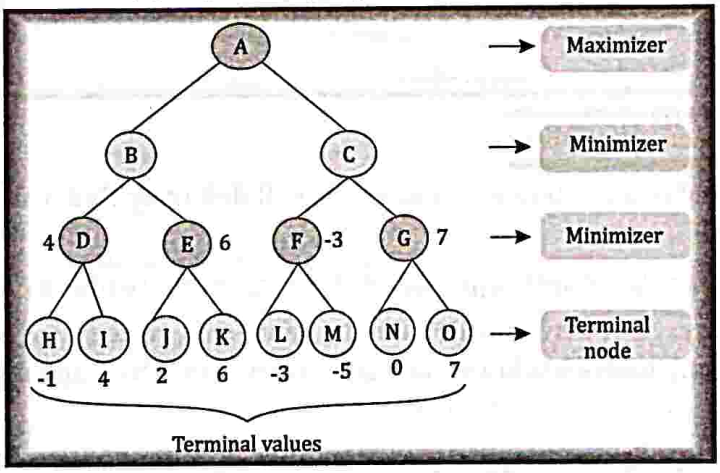
Step 2 : Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

For node D max(-1,- -∞) => max(-1,4)= 4

For Node E max(2, -∞) => max(2, 6)= 6

For Node F max(-3, -∞) => max(-3,-5) = -3

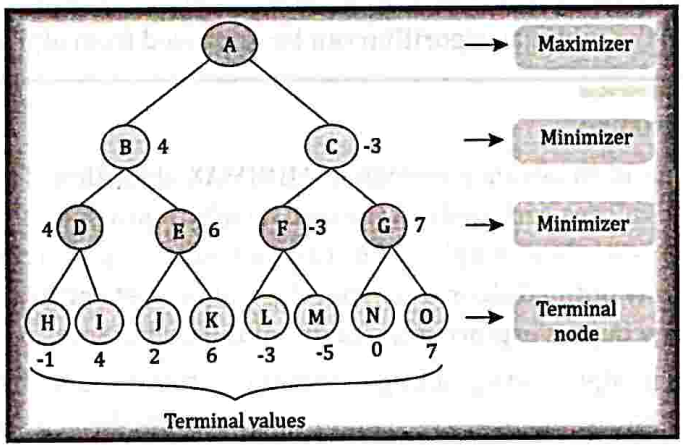
For node G max(0, -∞) = max(0, 7) = 7



Step 3 : In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

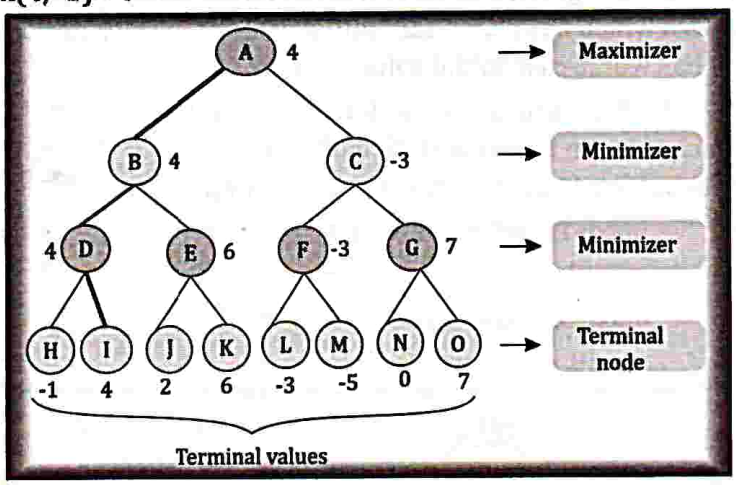
For node B= min(4,6) = 4

For node C= min (-3, 7) = -3



Step 4 : Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

For node A max(4, -3)= 4



Properties of Mini-Max Algorithm

- **Complete**- Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal**- Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity**- As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is O(bm), where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity**- Space complexity of Mini-max algorithm is also similar to DFS which is O(bm).

Limitation of the minimax Algorithm

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from alpha-beta pruning.

3.4.2 Alpha-Beta Pruning

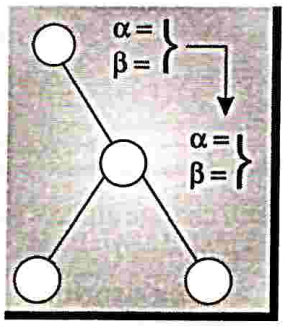
- Alpha-beta pruning is an advance version of MINIMAX algorithm. The drawback of minimax strategy is that it explores each node in the tree deeply to provide the best path among all the paths. This increases its time complexity. But as we know, the performance measure is the first consideration for any optimal algorithm. Therefore, alpha-beta pruning reduces this drawback of minimax strategy by less exploring the nodes of the search tree.
- The method used in alpha-beta pruning is that it cutoff the search by exploring less number of nodes. It makes the same moves as a minimax algorithm does, but it prunes the unwanted branches using the pruning technique (discussed in adversarial search). Alpha-beta pruning works on two threshold values, i.e., ? (alpha) and ? (beta).
    - ?: It is the best highest value, a MAX player can have. It is the lower bound, which represents negative infinity value.

- ?: It is the best lowest value, a MIN player can have. It is the upper bound which represents positive infinity.

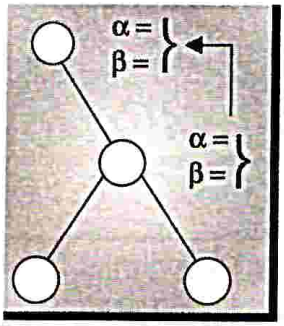
So, each MAX node has ?-value, which never decreases, and each MIN node has ?-value, which never increases.

Note:

- Initialize $\alpha = -\infty$ and beta $= \infty$ as the worst possible cases.
- Prune whenever $\alpha \geq \beta$. Therefore check whether $\alpha \geq \beta$ in every single node
- Never take any value upside the tree.



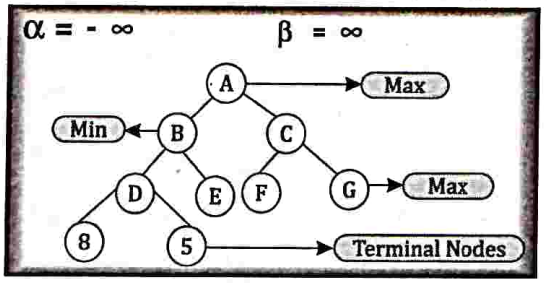
Here we are taking the value of alpha, beta from the root node and it is assigned to the node at the next level. This is possible whereas the following that is upside is not possible.



Working of Alpha Beta Pruning

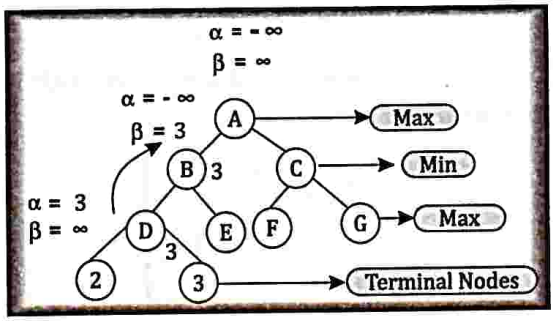
Let us consider a two-player search tree to understand further how Alpha-beta pruning in artificial intelligence works.

Step 1 : The Max player will start by traveling from node A to node B, where $\alpha = -\infty$ and $\beta = +\infty$, and delivering these alpha and beta values to node B, where = - and = + once again, and Node B transmitting the identical value to its offspring D.

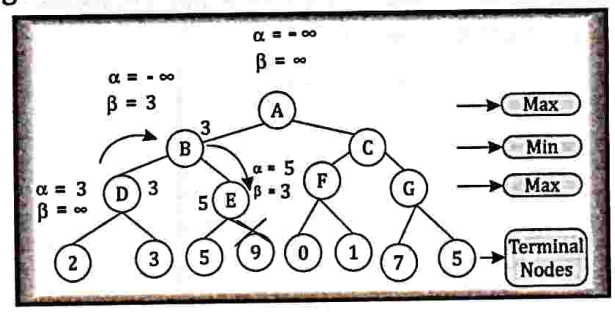


Step 2 : As Max's turn at Node D approaches, the value of α will be decided. When the value of α is compared to 2, then 3, the value at node D is max (2, 3) = 3. Hence, the node value is also 3.

Step 3 : The algorithm returns to node B, where the value of β will change since this is a turn of Min. Now β = +∞will be compared to the value of the available subsequent nodes, i.e., min (∞, 3) = 3, resulting in node B now α = -∞, and β = 3. In the next phase, the algorithm will visit the next successor of Node B, Node E, and pass the values of α = -∞ and β = 3.



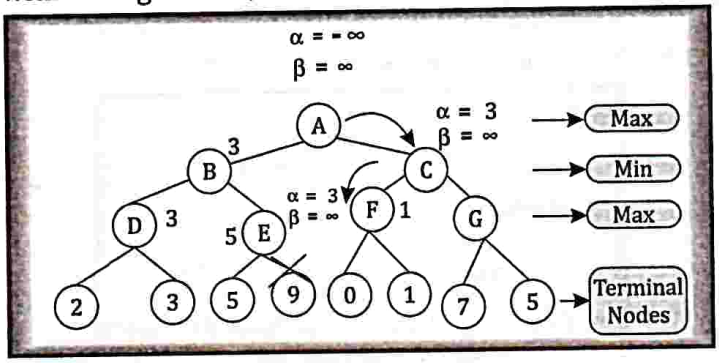
Step 4 : Max will take over at node E and change alpha's value. The current existing value of alpha will be compared to 5, resulting in max (-∞, 5) = 5, and at node E, where α>=β, the right successor of E will be pruned, and the algorithm will not traverse it, resulting in the value at node E being 5.



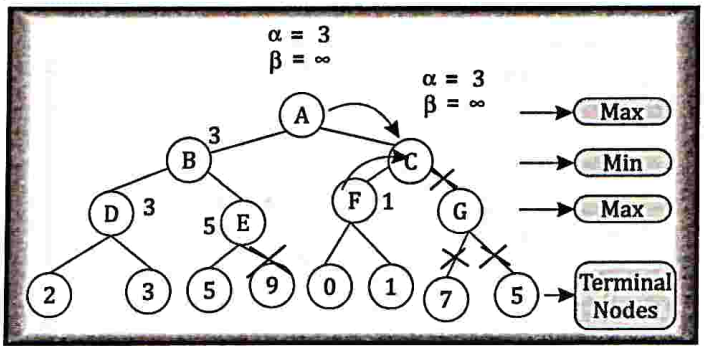
Step 5 : We now traverse the tree backward, from node B to node A. At node A, alpha will be converted to the greatest feasible value of 3, as max (-∞, 3)= 3, and β = +∞. These two values will now be passed on to Node C, A's right-hand successor.

At node C, the values and β = +∞ and α =3 will be passed on to node F, and node F will get the identical values.

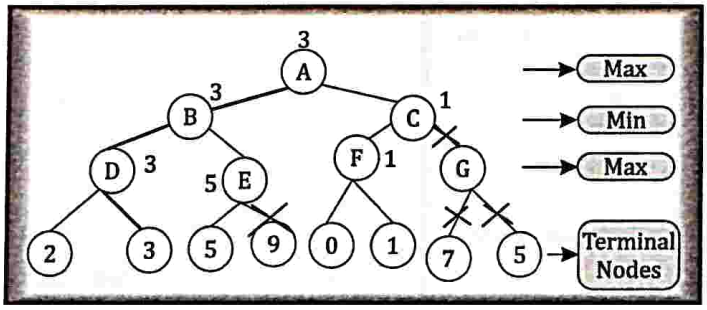
Step 6 : At node F, the value of α is compared with the left child 0, and max(3,0)= 3. It is then compared with the right child, which is 1, and max(3,1)= 3.



Step 7 : Node F sends the node value 1 to node C. The value of Beta is adjusted at C, $\alpha = 3$ and $\beta = +\infty$, and it is compared to 1, resulting in min $(\infty, 1) = 1$. Now, if $\alpha = 3$ and $\beta = 1$, the condition $\alpha >= \beta$ is met, the algorithm will prune the next child of C, which is G, rather than calculating the entire sub-tree G.



Step 8 : C now returns the value of 1 to A, with max (3, 1) = 3 being the optimum result for A. The final game tree is shown here, with nodes that have been calculated and nodes that have never been computed. As a result, in this case, the ideal value for the maximizer is 3.



Move Ordering in Alpha Beta Pruning

The sequence in which nodes are inspected determines the efficiency of alpha beta pruning. When it comes to alpha beta pruning in artificial intelligence, move ordering is crucial.

Move ordering are of two types in alpha beta pruning in artificial intelligence:

- **Worst Ordering**: In some circumstances of alpha beta pruning, the algorithm prunes none of the nodes and behaves like a conventional minimax algorithm. Because of the alpha and beta variables, this takes a long time and produces no valuable findings. In pruning, this is known as the Worst Ordering. In this situation, the optimal move is on the right side of the tree.
- **Ideal Ordering**: In some circumstances of alpha beta pruning in artificial intelligence, the algorithm prunes a large number of nodes. In pruning, this is referred to as ideal ordering. The optimal move is on the left side of the tree in this situation. We choose DFS Algorithm because it searches the left side of the tree first and then goes deep twice as fast as the minimax method.

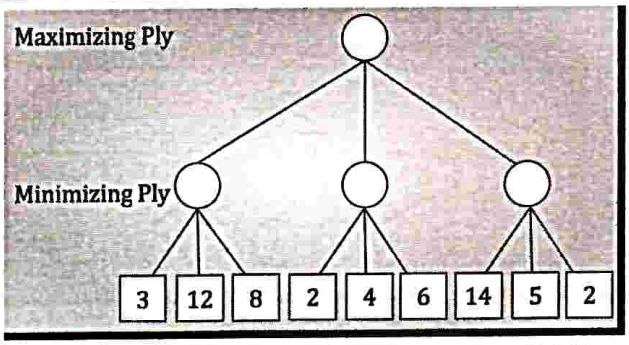
Rules to find Good Ordering

For finding the effective alpha-beta pruning ordering, we have to follow some rules:

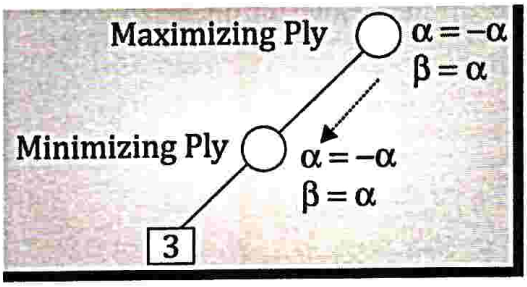
- In the tree, sort the nodes such that the better ones are checked first.
- The optimal move is made from the shallowest node.

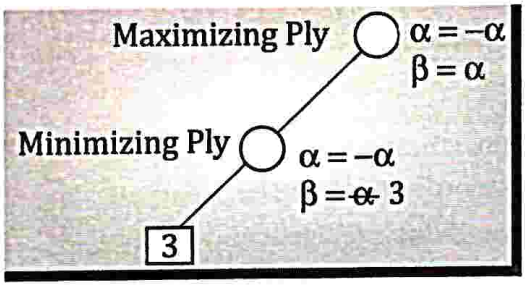
- We can keep track of the states since there's a chance they'll happen again.
- When deciding on the right step, make use of your subject expertise. For example, in chess, consider the following order: captures first, threats second, forward movements third, backward moves fourth.

Example : 1

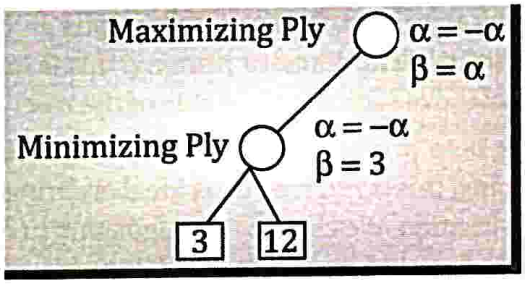


Initialize $\alpha = -\infty$ and $\beta = \infty$ for the root node and do depth first search until the first leaf.

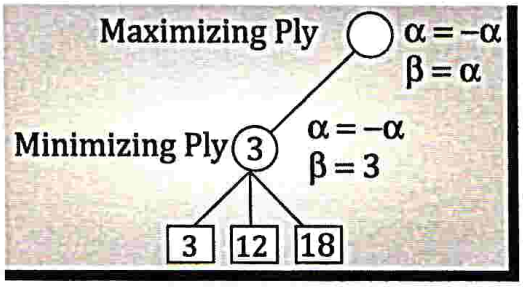




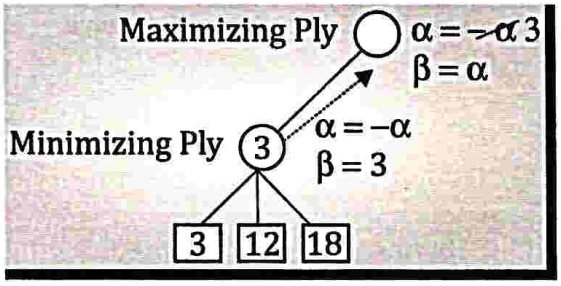
MIN updates β based on children



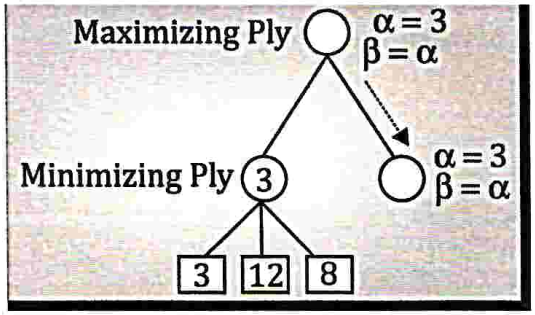
MIN updated β, based on children. No change



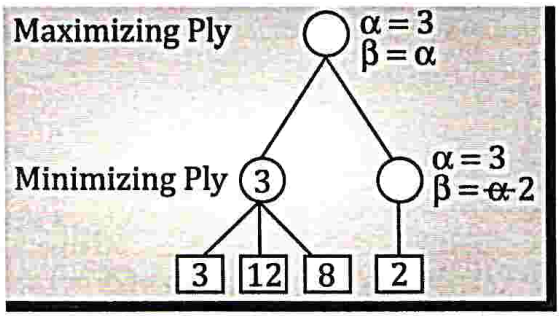
MIN updates β, based on children. No change



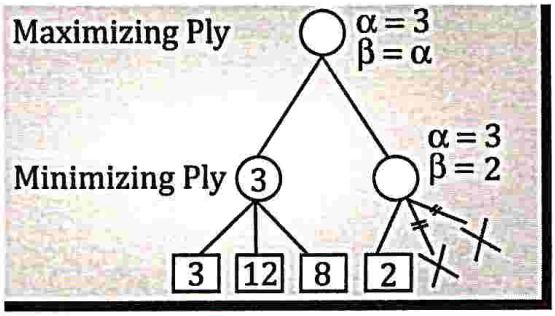
Max updates α, based on children. 3 is returned as node value



α, β passed to children

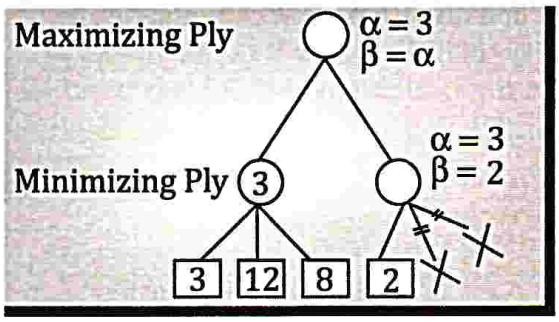


MIN updated β based on children



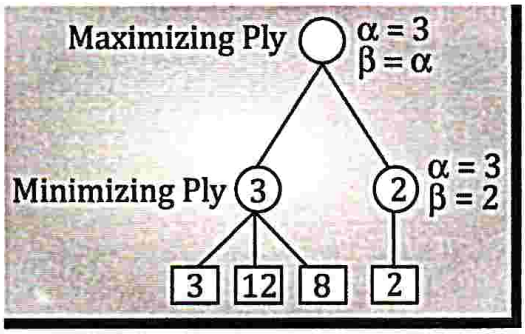
$\alpha \geq \beta$ (3 ≥ 2)

∴ Prune

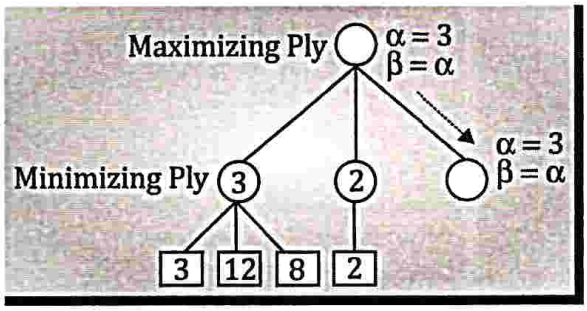


$\alpha \geq \beta$ (3 ≥ 2)

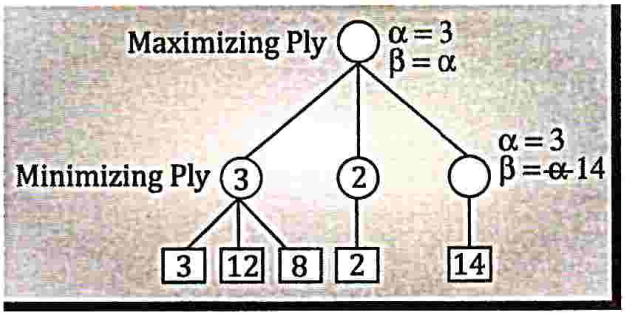
∴ Prune



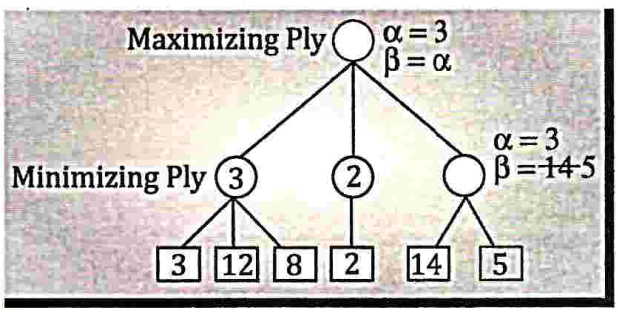
Max updates α, based on children. No change.



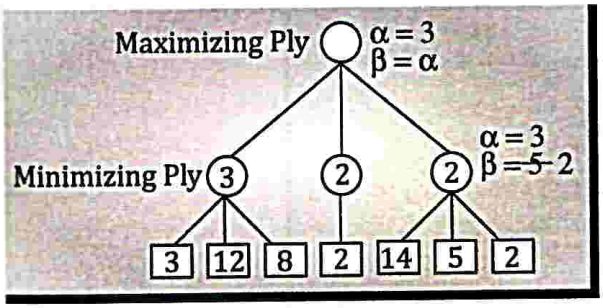
α, β is passed to children.



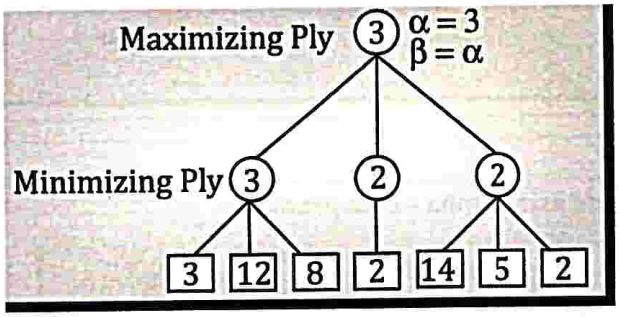
MIN updates β, based on children.



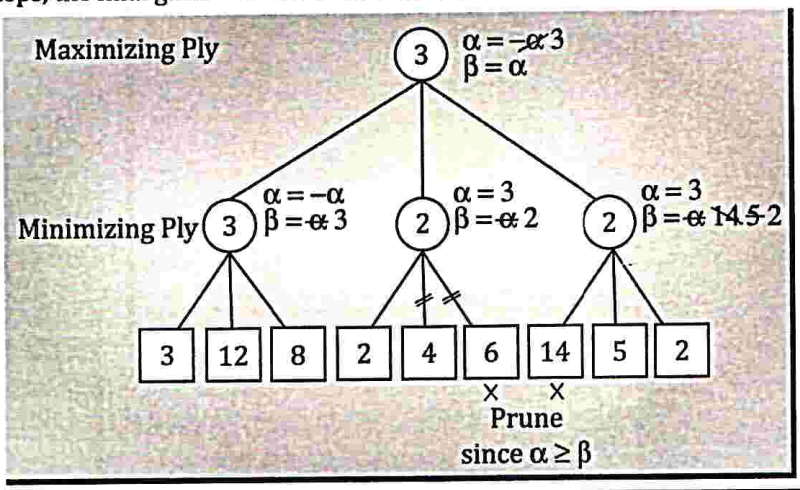
MIN updates β, based on children.



MIN updates β, based on children. 2 is returned as node value.

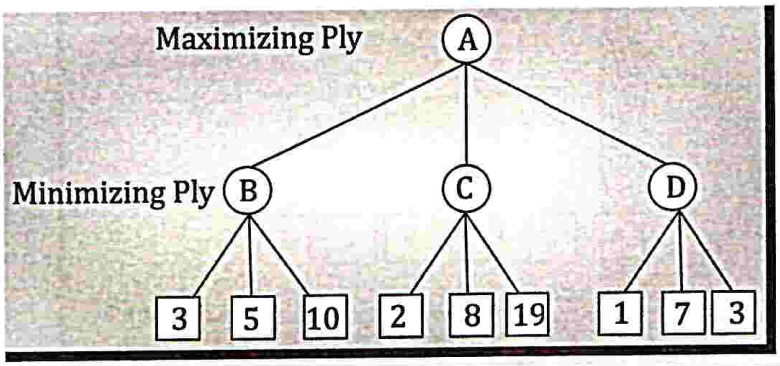


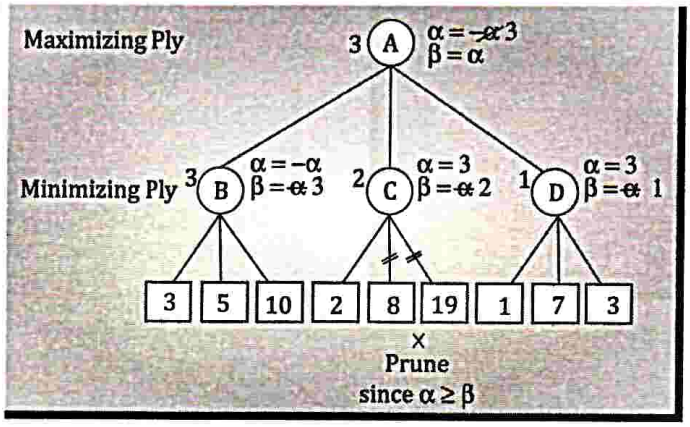
Combining all the steps, the final game tree looks as follows.



Example : 2

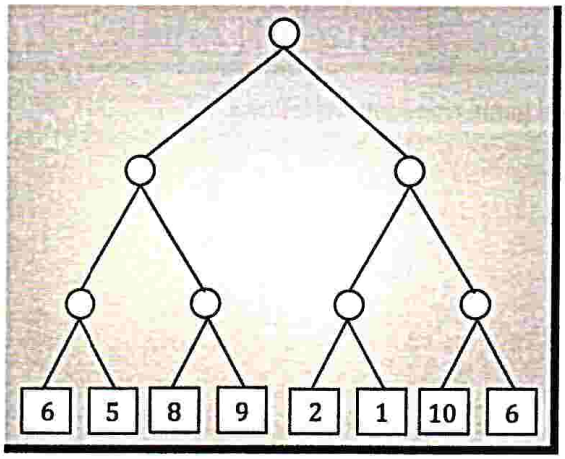
Solve the following game tree using alpha – beta pruning

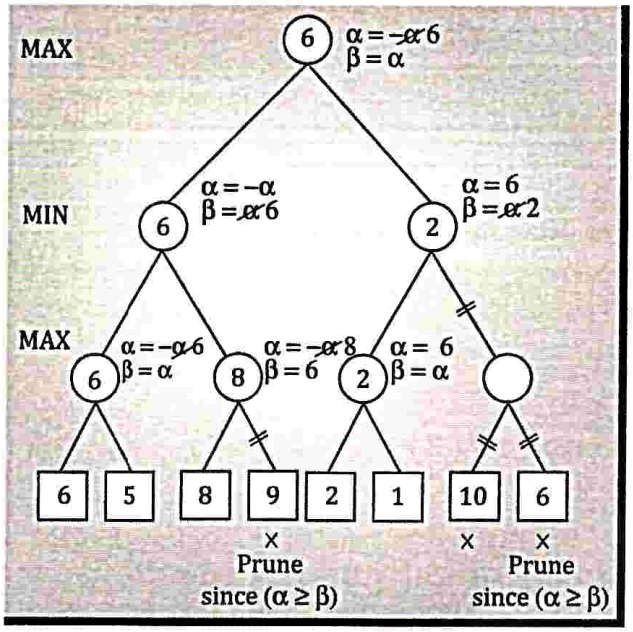




Example : 3

Solve the following game tree using alpha – beta pruning





Summary

* Problem-solving agent is a one type of goal-based agent, which uses atomic representation with no internal states visible to the problem-solving algorithms.
* The two most important steps in problem solving agent are : (a) Goal Formulation and (b) Problem Formulation.
* Basically there are two types of problem approaches (a) Toy Problem (Ex : 8 puzzle problem , 8 queens problem) and (b) Real world problem (Ex: Travel sales man problem, VLSI layout problem, Protein design, Robot navigation)
* A tree representation of search problem is called **Search tree**. The root of the search tree is the root node which is corresponding to the initial state.
* Properties of search algorithm are : (a) Completeness (b) Optimality (c) Time complexity (d) Space complexity
* Uninformed and Informed search are the two types of search algorithm.
* Uninformed search algorithms are : Breadth-first search, Uniform cost search, Depth-first search, Iterative deepening depth-first search, Bidirectional Search
* Informed search algorithms are : Greedy Best first search and A*search
* Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
* Time complexity of BFS $\mathbf{T(b) = 1+b^2+b^3+.....+ b^d= O(b^d)}$
* Depth-first search isa recursive algorithm for traversing a tree or graph data structure. It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
* Time complexity of DFS $\mathbf{T(n)= 1+ n^2+ n^3 +.....+ n^m=O(n^m)}$
* Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search.
* A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently
* AO* algorithm is a best first search algorithm. AO* algorithm uses the concept of AND-OR graphs to decompose any complex problem given into smaller set of problems which are further solved. AND-OR graphs are specialized graphs that are used in problems that can be broken down into sub problems where AND side of the graph represent a set of task that need to be done to achieve the main goal , whereas the or side of the graph represent the different ways of performing task to achieve the same main goal.
* Means end analysis (MEA) is an important concept in artificial intelligence (AI) because it enhances problem resolution. MEA solves problems by defining the goal and establishing the right action plan. This technique is used in AI programs to limit search.

* Applications of MEA are : (a) Organizational planning (b) Business Transformation (c) Gap analysis.
* , 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.
* Zero-sum games are adversarial search which involves pure competition. In Zero-sum game each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent.
* A game tree is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function
* In artificial intelligence, minimax is a decision-making strategy under game theory, which is used to minimize the losing chances in a game and to maximize the winning chances. This strategy is also known as 'Minmax,' 'MM,' or 'Saddle point.'
* Alpha-beta pruning is an advance version of MINIMAX algorithm. The drawback of minimax strategy is that it explores each node in the tree deeply to provide the best path among all the paths.

3.5 Review Questions

Short Answer Questions

1. Define problem solving agent and List the steps to be performed by problem solving agent.
2. List the two types of problem approaches.
3. Define search tree.
4. List the properties of search algorithm.
5. Write the time complexity of BFS.
6. Write the time complexity of DFS.
7. List any two advantages of BFS
8. List any two disadvantages of BFS
9. List any two advantages of DFS
10. List any two disadvantages of DFS
11. Define A* search
12. List any two advantages of A* search
13. List any two disadvantages of A* search
14. Define AO* search
15. List any two advantages of AO* search
16. List any two disadvantages of AO* search
17. List any 4 difference between A* & AO*
18. Define Mean End Analysis of AI.

19. Explain any two applications of Mean End Analysis of AI
20. Define Adversarial search / Games.
21. Define Zero-sum game.
22. Define game tree.
23. Define minmax algorithm.
24. List the properties of minmax algorithm.
25. Define Alpha-beta pruning.
26. List any four rules to find good ordering in Alpha-beta pruning.

Long Answer Questions

1. With a neat diagram, Explain 8 puzzle problem.
2. With a neat diagram, Explain 8 -queens problem.
3. Write a note on Real world problems.
4. With an example, Explain BFS.
5. With an example, Explain DFS.
6. With an example, Explain Best first search
7. With an example, Explain A* search
8. Write an algorithm of A* search
9. With an example, Explain AO* search
10. Write an algorithm of AO* search
11. With an example, Explain Mean End Analysis in AI
12. Write an algorithm for Mean End Analysis in AI
13. With a help of game tree, explain tic-tac-toe.
14. Explain minmax algorithm with an example.
15. Explain the working of Alpha-Beta pruning tree.

UNIT - II

CHAPTER 4

KNOWLEDGE BASED AGENTS

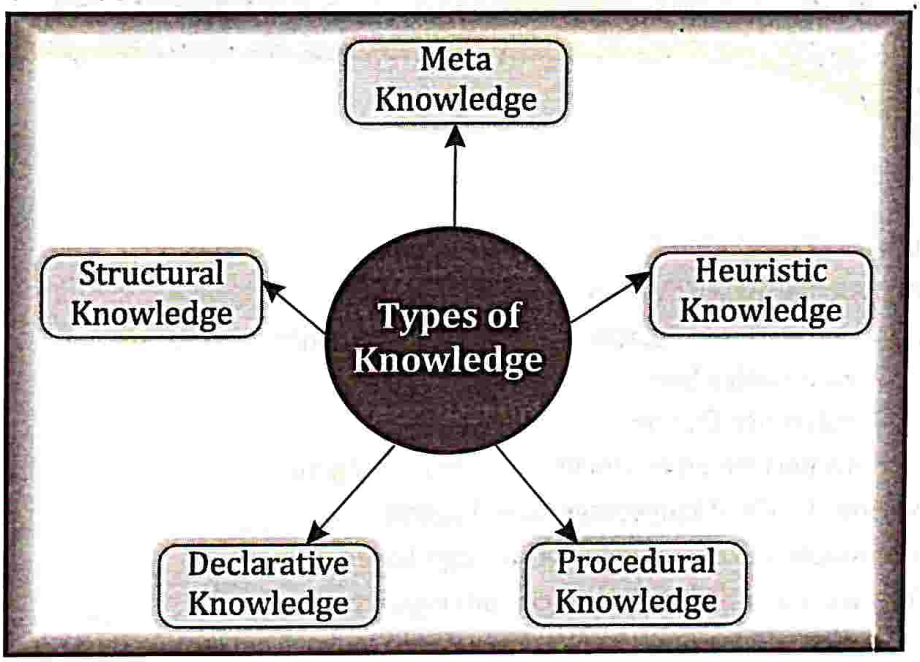
- What is Knowledge ?
- Knowledge-Based System
- Knowledge-Based System in Artificial Intelligence
  - Knowledge base
  - Inference Engine
- Actions performed by the knowledge base Agent
- Various levels of knowledge-based agent:
- Approaches to designing a knowledge-based agent:
- The Wumpus World in Artificial intelligence :
  - Knowledge base for Wumpus world in Artificial intelligence
- What is Logic?
- Propositional logic in Artificial intelligence
  - The Basic Idea of Propositional Logic
  - How Propositional Logic in Artificial Intelligence Represents Data to Machine ?
  - Logical Equivalence
- First order logical in Artificial Intelligence
  - Basic Elements of First-Order Logic
  - Rules of Inference in First-Order Logic
  - Unification in First-Order Logic
  - Resolution in First-Order-Logic
  - Inference Engine
  - Truth Maintenance System (TMS)
- Review Questions

4.1 What is Knowledge ?

Knowledge is the basic element for a human brain to know and understand the things logically. When a person becomes knowledgeable about something, he is able to do that thing in a better way. In AI, the agents which copy such an element of human beings are known as knowledge-based agents.

Types of Knowledge

There are mainly five types of knowledge.



Meta Knowledge: It is the information/knowledge about knowledge.

Heuristic Knowledge: It is the knowledge regarding a specific topic.

Procedural Knowledge: It gives information about achieving something.

Declarative Knowledge: It is the information which describes a particular object and its attributes.

Structural Knowledge: It describes the knowledge between the objects

4.2 Knowledge-Based System

A knowledge-based system is a system that uses artificial intelligence techniques to store and reason with knowledge. The knowledge is typically represented in the form of rules or facts, which can be used to draw conclusions or make decisions.

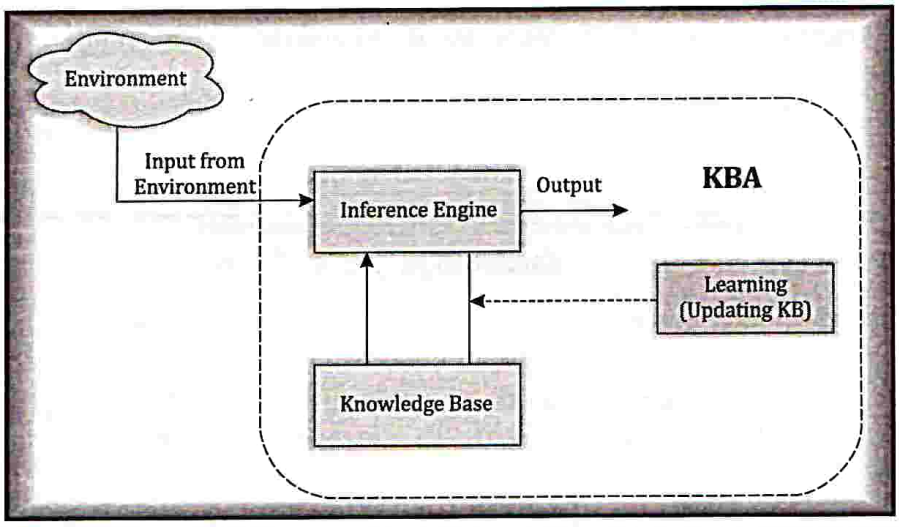
One of the key benefits of a knowledge-based system is that it can help to automate decision-making processes. For example, a knowledge-based system could be used to diagnose a medical condition, by reasoning over a set of rules that describe the symptoms and possible causes of the condition.

4.3 Knowledge-Based System in Artificial Intelligence

Knowledge-based agents represent searchable knowledge that can be reasoned. These agents maintain an internal state of knowledge, take decisions regarding it, update the data, and perform actions on this data based on the decision. Basically, they are intelligent and respond to stimuli like how humans react to different situations.

Examples

Based on the user's question (that behaves as the external stimuli), they provide an answer from their knowledge base (the data warehouse where they store basic knowledge) that provides a satisfactory answer to the user's question.



The above diagram is representing a generalized architecture for a knowledge-based agent. The knowledge-based agent (KBA) take input from the environment by perceiving the environment. The input is taken by the inference engine of the agent and which also communicate with KB to decide as per the knowledge store in KB. The learning element of KBA regularly updates the KB by learning new knowledge.

A knowledge-based system comprises of two distinguishable features which are:

- A Knowledge base
- An Inference Engine

4.3.1 Knowledge Base

A Knowledge base represents the actual facts which exist in the real world. It is the central component of a knowledge-based agent. It is a set of sentences which describes the information related to the world.

Note:

Here, a sentence is not an English language sentence, but it is represented in a language known as Knowledge representation language.

Why use a knowledge base?

Knowledge-base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.

4.3.2 Inference Engine

Inference means deriving new sentences from old. Inference system allows us to add a new sentence to the knowledge base. A sentence is a proposition about the world. Inference system applies logical rules to the KB to deduce new information.

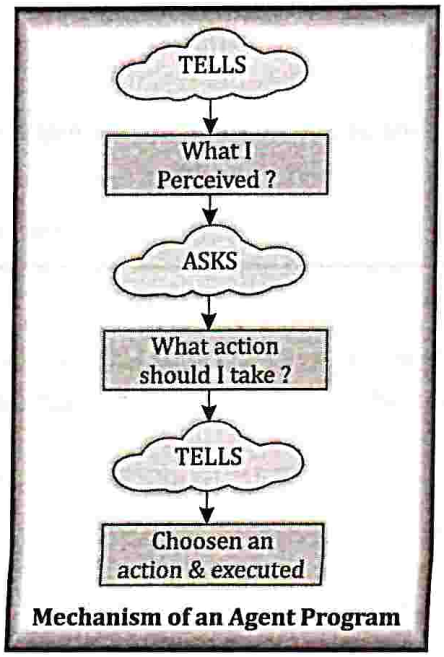
Inference system generates new facts so that an agent can update the KB. An inference system works mainly in two rules which are given as:

- Forward chaining
- Backward chaining (both are discussed in detail in 4.10.4)

4.4 Actions performed by the Knowledge base Agent

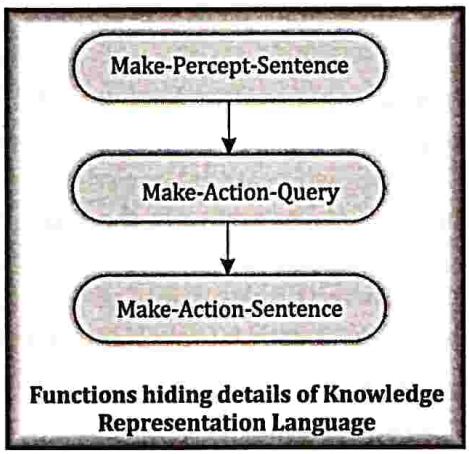
When there is a need to add/ update some new information or sentences in the knowledge-based system, we require an inference system. Also, to know what information is already known to the agent, we require the inference system. The technical words used for describing the mechanism of the inference system are: TELL and ASK. When the agent solves a problem, it calls the agent program each time. The agent program performs three things:

- It TELLS the knowledge base what it has perceived from the environment.
- It ASKS the knowledge base about the actions it should take?
- It TELLS the action which is chosen, and finally, the agent executes that action.



Mechanism of an Agent Program

The details of the knowledge representation language are abstracted under these three functions. These functions create an interface between the two main components of an intelligent agent, i.e., sensors and actuators.



Functions hiding details of Knowledge Representation Language

- **MAKE-PERCEPT-SENTENCE()**

  This function returns a sentence which tells the perceived information by the agent at a given time.
- **MAKE-ACTION-QUERY()**

  This function returns a sentence which tells what action the agent must take at the current time.
- **MAKE-ACTION-SENTENCE()**

  This function returns a sentence which tells an action is selected as well as executed.

4.5 Various levels of Knowledge-based Agent

A knowledge-based agent can be viewed at different levels which are given below:

- **Knowledge level :** Knowledge level is the first level of knowledge-based agent, and in this level, we need to specify what the agent knows, and what the agent goals are. With these specifications, we can fix its behaviour. For example, suppose an automated taxi agent needs to go from a station A to station B, and he knows the way from A to B, so this comes at the knowledge level.
- **Logical level:** At this level, we understand that how the knowledge representation of knowledge is stored. At this level, sentences are encoded into different logics. At the logical level, an encoding of knowledge into logical sentences occurs. At the logical level we can expect to the automated taxi agent to reach to the destination B.
- **Implementation level:** This is the physical representation of logic and knowledge. At the implementation level agent perform actions as per logical and knowledge level. At this level, an automated taxi agent actually implement his knowledge and logic so that he can reach to the destination.

4.6 Approaches to designing a Knowledge-based Agent

There are mainly two approaches to build a knowledge-based agent:

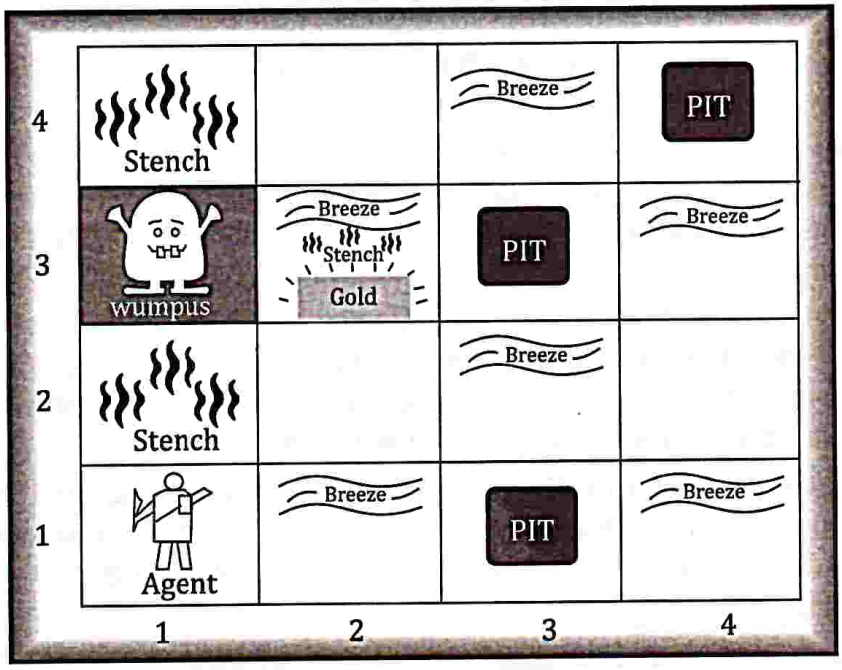
- **Declarative Approach:** In this beginning from an empty knowledge base, the agent can TELL sentences one after another till the agent has knowledge of how to work with its environment. This is known as the declarative approach. It stores required information in empty knowledge-based system.
- **Procedural Approach:** This converts required behaviours directly into program code in empty knowledge-based system. It is a contrast approach when compared to Declarative approach. In this by coding behaviour of system is designed .

4.7 The Wumpus World in Artificial Intelligence

The Wumpus world is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game Hunt the Wumpus by Gregory Yob in 1973.

The Wumpus world is a cave which has 4/4 rooms connected with passageways. So there are total 16 rooms which are connected with each other. We have a knowledge-based agent who will go forward in this world. The cave has a room with a beast which is called Wumpus, who eats anyone who enters the room. The Wumpus can be shot by the agent, but the agent has a single arrow. In the Wumpus world, there are some Pits rooms which are bottomless, and if agent falls in Pits, then he will be stuck there forever. The exciting thing with this cave is that in one room there is a possibility of finding a heap of gold. So the agent goal is to find the gold and climb out the cave without fallen into Pits or eaten by Wumpus. The agent will get a reward if he comes out with gold, and he will get a penalty if eaten by Wumpus or falls in the pit.

Following is a sample diagram for representing the Wumpus world. It is showing some rooms with Pits, one room with Wumpus and one agent at (1, 1) square location of the world.



There are also some components which can help the agent to navigate the cave. These components are given as follows:

1. The rooms adjacent to the Wumpus room are smelly, so that it would have some stench.
2. The room adjacent to PITs has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze.
3. There will be glitter in the room if and only if the room has gold.
4. The Wumpus can be killed by the agent if the agent is facing to it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.

PEAS description of Wumpus World

1. **Performance Measure:**
   - +1000 reward points if the agent comes out of the cave with the gold.
   - -1000 points penalty for being eaten by the Wumpus or falling into the pit.
   - -1 for each action, and -10 for using an arrow.
   - The game ends if either agent dies or came out of the cave.
2. **Environment:**
   - A 4*4 grid of rooms.
   - The agent initially in room square [1, 1], facing toward the right.
   - Location of Wumpus and gold are chosen randomly except the first square [1,1].
   - Each square of the cave can be a pit with probability 0.2 except the first square.
3. **Actuators:**
   - Left turn,
   - Right turn
   - Move forward
   - Grab
   - Release
   - Shoot
4. **Sensors:**
   - The agent will perceive the stench if he is in the room adjacent to the Wumpus. (Not diagonally).
   - The agent will perceive breeze if he is in the room directly adjacent to the Pit.
   - The agent will perceive the glitter in the room where the gold is present.
   - The agent will perceive the bump if he walks into a wall.
   - When the Wumpus is shot, it emits a horrible scream which can be perceived anywhere in the cave.
   - These precepts can be represented as five element list, in which we will have different indicators for each sensor.
   - Example if agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as: [Stench, Breeze, None, None, None].

The Wumpus World Properties

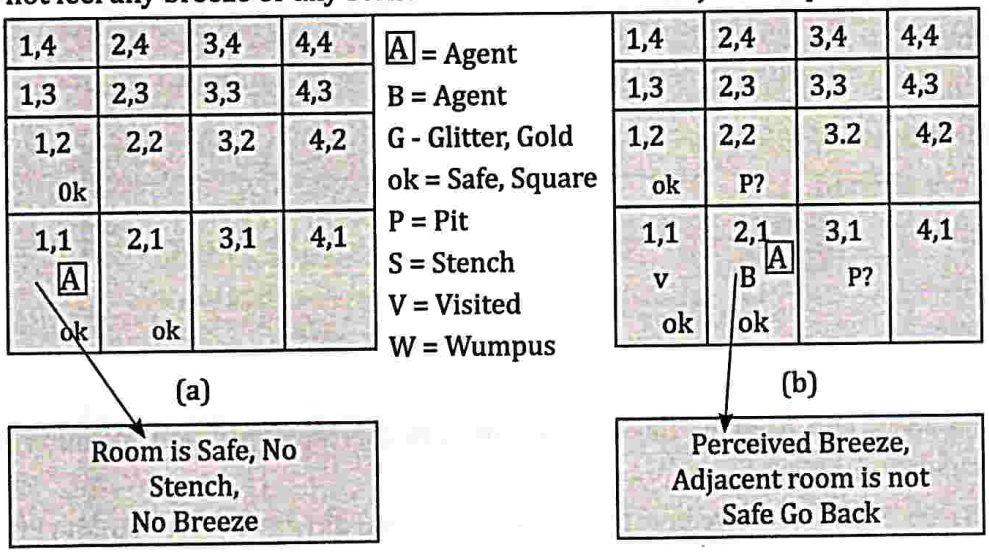
1. **Partially Observable:** The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
2. **Deterministic:** It is deterministic, as the result and outcome of the world are already known.
3. **Sequential:** The order is important, so it is sequential.

4. **Static:** It is static as Wumpus and Pits are not moving.
5. **Discrete:** The environment is discrete.
6. **One Agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

Exploring the Wumpus World

Now we will explore the Wumpus world and will determine how the agent will find its goal by applying logical reasoning.

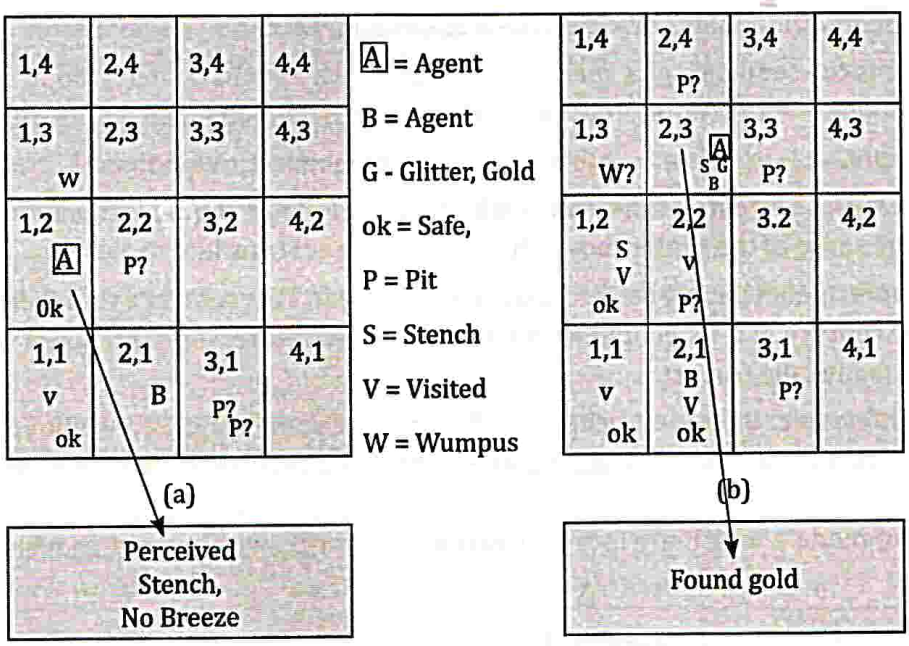
1. **Agent's First Step:** Initially, the agent is in the first room or on the square [1,1], and we already know that this room is safe for the agent, so to represent on the below diagram (a) that room is safe we will add symbol OK. Symbol A is used to represent agent, symbol B for the breeze, G for Glitter or gold, V for the visited room, P for pits, W for Wumpus. At Room [1,1] agent does not feel any breeze or any Stench which means the adjacent squares are also OK.



2. **Agent's Second Step:** Now agent needs to move forward, so it will either move to [1, 2], or [2,1]. Let's suppose agent moves to the room [2, 1], at this room agent perceives some breeze which means Pit is around this room. The pit can be in [3, 1], or [2,2], so we will add symbol P? to say that, is this Pit room?

   Now agent will stop and think and will not make any harmful move. The agent will go back to the [1, 1] room. The room [1,1], and [2,1] are visited by the agent, so we will use symbol V to represent the visited squares.

3. **Agent's Third Step:** At the third step, now agent will move to the room [1,2] which is OK. In the room [1,2] agent perceives a stench which means there must be a Wumpus nearby. But Wumpus cannot be in the room [1,1] as by rules of the game, and also not in [2,2] (Agent had not detected any stench when he was at [2,1]). Therefore agent infers that Wumpus is in the room [1,3], and in current state, there is no breeze which means in [2,2] there is no Pit and no Wumpus. So it is safe, and we will mark it OK, and the agent moves further in [2,2].



4. **Agent's Fourth Step:** At room [2,2], here no stench and no breezes present so let's suppose agent decides to move to [2,3]. At room [2,3] agent perceives glitter, so it should grab the gold and climb out of the cave.

4.7.1 Knowledge base for Wumpus World in Artificial Intelligence

Following is the Simple KB for wumpus world when an agent moves from room [1, 1], to room [2,1]:

$\neg W_{11}$	$\neg S_{11}$	$\neg P_{11}$	$\neg B_{11}$	$\neg G_{11}$	$\neg V_{11}$	OK_{11}
$\neg W_{12}$	-----	$\neg P_{12}$	-----	-----	$\neg V_{12}$	OK_{12}
$\neg W_{21}$	$\neg S_{21}$	$\neg P_{21}$	$\neg B_{21}$	$\neg G_{21}$	$\neg V_{21}$	OK_{21}

- Here in the first row, we have mentioned propositional variables for room[1,1], which is showing that room does not have wumpus(¬ W11), no stench (¬S11), no Pit(¬P11), no breeze(¬B11), no gold (¬G11), visited (V11), and the room is Safe(OK11).
- In the second row, we have mentioned propositional variables for room [1,2], which is showing that there is no wumpus, stench and breeze are unknown as an agent has not visited room [1,2], no Pit, not visited yet, and the room is safe.
- In the third row we have mentioned propositional variable for room[2,1], which is showing that there is no wumpus(¬ W21), no stench (¬S21), no Pit (¬P21), Perceives breeze(B21), no glitter(¬G21), visited (V21), and room is safe (OK21).

4.8 What is Logic?

Logic is the key behind any knowledge. It allows a person to filter the necessary information from the bulk and draw a conclusion. In artificial intelligence, the representation of knowledge is done via logics.

There are three main components of logic, which are as follows:

- **Syntax:** It is the sequence of a specific language which should be followed in order to form a sentence. Syntax is the representation of a language. Every language has its own syntax.

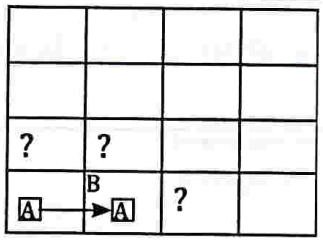
  For example, ax2+bx+c is a well-formed syntax of a quadratic equation.

- **Semantics:** The sentence or the syntax which a logic follows should be meaningful. Semantics defines the sense of the sentence which relates to the real world.

  For example, Indian people celebrate Diwali every year. This sentence represents the true fact about the country and its people who are Indians. Therefore, the sentence is syntactically as well as semantically correct.

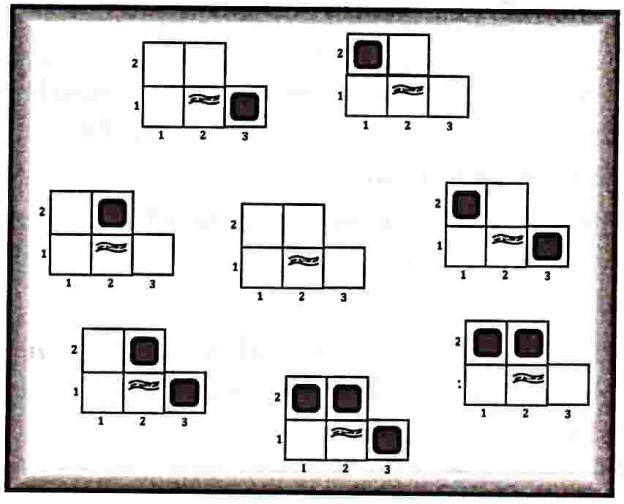
- **Logical Inference:** Inference means to infer or draw some conclusions about some fact or a problem. Logical inference is thinking all the possible reasons which could lead to a proper result. Inference algorithms are used to perform logical inference. Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

Example **In Wumpus World**

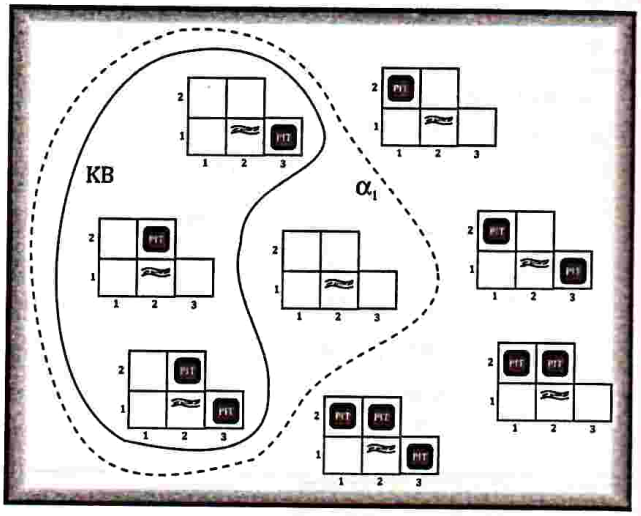


- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models for all ?, assuming only pits
- 3 Boolean choices Ô⇒ 8 possible models

Possible Wumpus Models



Entailment



KB = wumpus-world rules + observations

α1 = "[1,2] is safe", KB ⊨ α1, proved by model checking

4.9 Propositional Logic in Artificial Intelligence

4.9.1 The Basic Idea of Propositional Logic

Proposition means sentences. Propositional logic applies the Boolean logic to convert our real-world data into a format that is readable to the computer. For instance, if we say 'It is hot and humid today', the machine won't understand. But if we can create propositional logic for this sentence, then, we can make the machine-read, and interpret our message.

Example

Earth is round', the output for this proposition is TRUE. If we say, 'Earth is square', then the output is FALSE. Propositional logic applies to those sentences where the output can only be either TRUE or FALSE. But if we refer to the sentence like 'Some children are lazy' then here we have two possible outputs. This preposition is TRUE for those children who are lazy, but it is FALSE for those children who are not lazy. So, for such sentences/ propositions where two or more outputs are possible, propositional logic doesn't apply.

Definition : Propositional Logic

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.

4.9.2 How Propositional Logic in Artificial Intelligence Represents Data to Machine ?

There are two types of prepositions, **atomic** and **complex**, that can be represented through propositional logic.

1. **Atomic** means a single preposition like 'the sky is blue', 'hot days are humid', water is liquid, etc. It is made up of only one proposition sign. These are the sentences that must be true or untrue in order to pass.
2. **Complex prepositions** are those, which have been formed by connecting one, two, or more sentences. In propositional logic, there are five symbols(***logical connectives***) to create the syntax to represent the connection of two or more sentences.

   Example P1 : It is raining today P2 : Street is wet P=, P1 ∧ P2 ; It is raining today and street is wet.

   Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

   The proposition symbols P, Q etc are sentences

   - If P is a sentence, ¬P is a sentence (negation)
   - If P and Q are sentences, P ∧ Q is a sentence (conjunction)
   - If P and Q are sentences, P ∨ Q is a sentence (disjunction)

- If P and Q are sentences, P⇒ Q is a sentence (implication)
- If P and Q are sentences, P ⇔ Q is a sentence (biconditional)

➢ **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.

P	¬ P
True	False
False	True

➢ **Conjunction:** A sentence which has ∧ connective such as, P ∧ Q is called a conjunction.
Example: Rohan is intelligent and hardworking. It can be written as,
P= Rohan is intelligent,
Q= Rohan is hardworking. → P∧ Q.

P	Q	P ∧ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

➢ **Disjunction:** A sentence which has ∨ connective, such as P ∨ Q. is called disjunction, where P and Q are the propositions.
Example: "Ritika is a doctor or Engineer",
Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as P ∨ Q.

P	Q	P ∨ Q
True	True	**True**
False	True	**True**
True	False	**True**
False	False	**False**

➢ **Implication:** A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as
Example : If it is raining, then the street is wet.
Let P= It is raining, and Q= Street is wet, so it is represented as P → Q

P	Q	P→ Q
True	True	**True**
True	False	**False**
False	True	**True**
False	False	**True**

➢ **Biconditional:** A sentence such as P⇔ Q is a Biconditional sentence, example If I am breathing, then I am alive.

P = I am breathing, Q = I am alive, it can be represented as P ⇔ Q.

P	Q	P ⇔ Q
True	True	**True**
False	False	**False**
False	True	**False**
False	False	**True**

Precedence of Connectives

Just like arithmetic operators, there is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem. Following is the list of the precedence order for operators:

Precedence	Operators
First Precedence	Parenthesis
Second Precedence	Negation
Third Precedence	Conjunction(AND)
Fourth Precedence	Disjunction(OR)
Fifth Precedence	Implication
Six Precedence	Biconditional

Applying Connectives in Wumpus world sentences:

- Let $P_{i,j}$ be true if there is a pit in [i, j]
  - observation $R_1 : \neg P_{1,1}$
- Let Bi,j be true if there is a breeze in [i, j].
- "Pits cause breezes in adjacent squares"
  - rule $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - rule $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
  - observation $R_4 : \neg B_{1,1}$
  - observation $R_5 : B_{2,1}$
- What can we infer about $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, etc.?

Truth table for Inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	false	false	false	false	false	false	true	true	true	true	false	false
false	false	false	false	false	false	true	true	true	false	true	false	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	true	false	false	false	false	false	true	true	false	true	true	false
false	true	false	false	false	false	true	true	true	true	true	true	true

false	true	false	false	false	true	false	true	true	true	true	true	true
false	true	false	false	false	true	true	true	true	true	true	true	true
false	true	false	false	true	false	false	true	false	false	true	true	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	true	true	true	true	true	true	false	true	true	false	true	false

- Enumerate rows (different assignments to symbols $P_{i,j}$)
- Check if rules are satisfied (R_i)
- Valid model (KB) if all rules satisfied

4.9.3 Logical Equivalence

Logical equivalence is one of the features of propositional logic. Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.

Let's take two propositions A and B, so for logical equivalence, we can write it as A⇔B. In below truth table we can see that column for ¬A∨ B and A→B, are identical hence A is Equivalent to B

A	B	¬ A	¬ A ∨ B	A→ B
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Properties of Operators:

- **Commutativity:**

  P∧ Q= Q ∧ P, or

  P ∨ Q = Q ∨ P.

- **Associativity:**

  (P ∧ Q) ∧ R= P ∧ (Q ∧ R),

  (P ∨ Q) ∨ R= P ∨ (Q ∨ R)

- **Identity element:**

  P ∧ True = P,

  P ∨ True= True.

- **Distributive:**

  P∧ (Q ∨ R) = (P ∧ Q) ∨ (P ∧ R).

  P ∨ (Q ∧ R) = (P ∨ Q) ∧ (P ∨ R).

- **DE Morgan's Law:**

  ¬ (P ∧ Q) = (¬P) ∨ (¬Q)

  ¬ (P ∨ Q) = (¬ P) ∧ (¬Q).

- ***Double-negation elimination***:

  ¬ (¬P) = P.

Limitations of Propositional logic

1. We cannot represent relations like ALL, some, or none with propositional logic.

   Example:

   - All the girls are intelligent.
   - Some apples are sweet.
2. Propositional logic has limited expressive power.
3. In propositional logic, we cannot describe statements in terms of their properties or logical relationships.

4.10 First order logical in Artificial Intelligence

- First-Order Logic (FOL) is a logical way of allowing us to use statements to express relationships between things. It goes beyond fundamental logic by allowing us to talk about individual items by using variables and quantifiers. Based on what we already know, First-Order Logic helps us discover new knowledge or facts about a family. First-Order Logic is widely used in the field of Artificial Intelligence as a critical tool for tackling real-world issues.
- First-Order Logic, also known as First-Order Predicate Calculus or First-Order Predicate Logic, adds quantifiers, variables, and predicates to propositional logic.

  Imagine having a magical language that allows you to explain and understand things in a very structured manner. In Artificial Intelligence, this magical language is known as First-Order Logic.

  Simple statements may be used for conveying information about items and their characteristics. In this language. You may say, "All dogs bark," or "Every bird can fly." These statements serve as building blocks for understanding how things function in the real world.

  First-Order Logic also enables you to discuss the relationships between objects. You can say something like, "If you study hard, then you will get good marks," or "If it's sunny, then I will not go for a movie". These statements can help you in making logical relationships between various types of information.

  We may use FOL to connect all of these statements and infer new information.

  For example, if we know that all dogs bark and come across a new dog, we may assume that this animal will bark similarly.

Examples

Consider the following simple family scenario:

Define the Predicates:

Parent(x, y): Indicates that x is the parent of y.

Statements addressing the family:

Parent(Ram, Riya): Ram is Riya's father.

Parent (Riya, Aarav): Riya is Aarav's mother.

By using First Order Logic, we can conclude.
Parent(Ram, Aarav):
Because Ram is the father of Riya (Parent(Ram, Riya)) and Riya is the mother of Peter (Parent(Riya, Aarav)), we can infer that **Ram is also the father of Aarav.**

In this brief example, we used First-Order Logic to conclude that Ram is Aarav's father based on the statements provided.

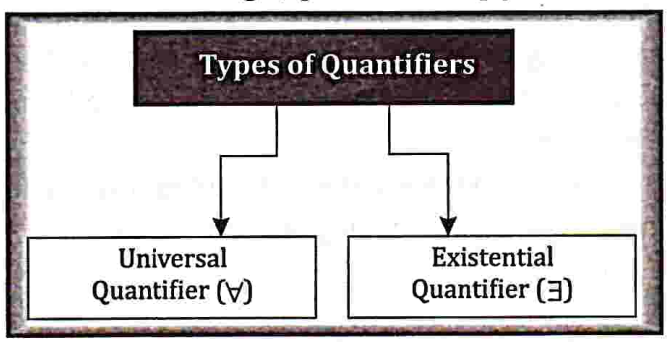
4.10.1 Basic Elements of First-Order Logic

The following table furnishes the basic elements of First Order Logic in Artificial Intelligence.

Element	Example	Meaning
Constant	1,2,A, John, Mumbai, Cat...	Values that can not be changed
Variables	x, y, z, a, b, ...	Can take up any value and can also change
Predicates	Brother, Father, > , ...	Defines a relationship between its input terms
Function	Sqrt, LeftLegOf, ...	Computes a defined relation of input term
Connectives	^ , v , ¬, =>, ⇔	Used to form complex sentences using atomic sentences
Equality	==	Relational operator that checks equality
Quantifier	∀, ∃	Imposes a quantity on the respective variable.

Quantifiers in First-Order Logic

- In First-Order Logic, quantifiers allow us to discuss collections of objects, qualities, or relationships as a group. They help us in expressing our thoughts on whether particular statements apply to all members of a group or apply to particular members that fit a specified requirement.
- Assume you're discussing a large group of items, such as all the animals at a zoo or all the students in a class. In First-Order Logic, quantifiers help you talk about these groupings.



1. **Universal Quantifier (∀):** The Universal Quantifier is a logic concept that allows you to make statements about each and every individual member in a group. It's equivalent to stating "for all" or "for every". It states a condition that holds true for every member of a set or collection in mathematical and logical terms.
   Statement: "In every classroom, all students have notebooks."
   Meaning: This statement conveys that within every classroom, each and every student has a notebook.

2. **Existential Quantifier (∃):** The Existential Quantifier (∃) is a key idea in logic that allows you to say that at least one member in a group or domain meets a certain condition. In other words, it's the same as stating "there is" or "there exists" in daily speech. It states the presence of at least one member who fits a specific requirement in logical terms.

   For Example, "There is a garden in the neighbourhood with colourful flowers."

   Consider a neighbourhood that has a variety of gardens. When we apply the Existential Quantifier (∃), we mean that among all of these gardens, at least one stands out because of its gorgeous flowers. We may not know which garden it is, but we are certain that it exists.

4.10.2 Rules of Inference in First-Order Logic

Let's discuss the rules that are to be followed in First-Order Logic:

1. **Modus Ponens :** If you have a statement, "P implies Q" (P → Q), and you also know that "P" is true, then you can correctly conclude that "Q" is true.

   Examples

   Given: If it's raining (P), then I'll take an umbrella (Q).
   Statement 1: It's raining (P).
   Conclusion: I'll take an umbrella (Q).

2. **Modus Tollens :** If you have a statement "P implies Q", and you also know that "Q" is not true (¬Q), then you can infer that "P" is not true (¬P).

   Examples

   Given: If I study hard (P), then I'll pass the exam (Q).
   Given: I didn't pass the exam (¬Q).
   Conclusion: I didn't study hard (¬P).

3. **Universal Instantiation :** If you have a statement that says something is true for all instances, like "For all x, P(x)," and you have a specific individual "a," you can conclude that "P(a)" is true.

   Examples

   Given: All dogs (∀x Dog(x)) bark (P(x)).
   Conclusion: Fido barks (Dog(Fido) → Bark(Fido)).

4. **Universal Generalization :** If you have a statement that's true for a specific individual "a," like "P(a)," you can generalise it to say that it's true for all instances, or "For all x, P(x)."

   Examples

   Statement: "Riya aced the Science test."
   Conclusion: "All students as good as Riya aced the Science test."

   Explanation: Starting with the Statement that "Riya aced the Science test," we can apply Universal Generalization to make a broader statement. The conclusion "All students as good as Riya aced the Science test" depicts that any student who possesses the same level of skill as Riya also aced the Science test. This demonstrates how Universal Generalization allows us to generalise from a specific case to a more general statement that applies to a group with similar attributes.

5. **Existential Instantiation :** If you have a statement that says something exists, like "There exists an x such that P(x)," and you introduce a new variable "a," you can conclude that "P(a)" is true.

Examples

Given Statement: "At least one student in the class speaks German."

Inference: "Jerry is a student in the class who speaks German."

The initial statement indicates that there's a student in the class who knows how to speak German. Applying Existential Instantiation, we can pinpoint a specific individual, Jerry, who fulfils this condition. This showcases how Existential Instantiation helps us identify particular instances that satisfy the requirement mentioned in the given statement.

6. **Existential Generalisation :** If you have a statement that's true for a specific individual "a," like "P(a)," you can generalise it to say that something exists for all instances, or "There exists an x such that P(x)."

Examples

Given Statement: "One student scored the highest marks in the exam."

Statement: "There exists a student who scored the highest marks in the exam."

Explanation: Let's say there are 20 students in the class. The given statement asserts that among these 20 students, there's at least one who achieved the highest marks.

By using Existential Generalisation, we can conclude that out of the entire group of students, there exists at least one student who indeed secured the top marks in the exam.

4.10.3 Unification in First-Order Logic

Unification in First-Order Logic deals with finding a common substitution for variables in different terms to make them match. The goal of unification is to make two expressions identical by assigning values to variables in a way that preserves their meanings. It involves substituting terms for variables in a way that the resulting expressions match.

Conditions for Unification

1. The predicate symbol must be the same.
2. The number of arguments in both expressions must be identical.
3. If two similar variables are present in the same expression, then unification fails.

Pseudocode of Unification

```
function unify(expression1, expression2):
    if expression1 is the same as expression2:
        return "SUCCESS"
    if expression1 is a variable:
        return substitute(expression1, expression2)
    if expression2 is a variable:
        return substitute(expression2, expression1)
```

```
if both expressions are terms and not equal:
    return "FAILURE"
if both expressions are functions or predicates with matching name and arity:
for each corresponding argument (arg1, arg2):
    result <- unify(arg1, arg2)
    if result is "FAILURE":
        return "FAILURE"
return "SUCCESS"
```

function substitute(variable, expression):

```
 if variable occurs in expression:
    return "FAILURE"
else:
    return {variable / expression}
```

Explanation: The provided pseudocode describes the process of unification in First-Order Logic using simplified code. It checks for identical expressions, handles cases involving variables, terms, functions, and predicates, and ensures consistency in variable substitutions. The substitute function handles variable replacements and checks for valid substitutions.

Examples

Consider the following phrases, which must be unified:
"Eats(x, Apple)" is an expression.
"Eats(Riya, y)" is an expression.
The following is an explanation of the unification process:

Comparison

Expression A ("Eats(x, Apple)") is compared to Expression B ("Eats(Riya, y)"

Substitution Variable

We can see that Expression A's first parameter is a variable "x," and the second argument is a constant "Apple."

The first parameter in Expression B is a constant "Riya," while the second argument is a variable "y."

Unifying Variables

We unify the variable "x" in Expression A with "Riya" in Expression B. This results in the substitution: x = Riya.

We also unify the variable "y" in Expression B with the constant "Apple." This gives us the substitution: y = Apple.

Applying Substitutions

After substitutions, Expression A becomes "Eats(Riya, Apple)."

Expression B remains the same: "Eats(Riya, Apple)."

Unified Expression

Both expressions are now identical: "Eats(John, Apple)."

Unification is successful, and the unified expression is "Eats(Riya, Apple)."

4.10.4 Resolution in First-Order-Logic

In First-Order Logic (FOL), resolution is a fundamental inference method for theorem proving and logical reasoning. It's very beneficial for showing the truth of statements through contradiction. The resolution principle helps in the generation of new clauses from old ones, resulting in the resolution of difficult logical issues.

Steps for Resolution

1. ***Conversion to Clausal Form***: The statements in FOL are transformed into clausal form, where each statement becomes a collection of literals (positive or negated atomic formulas) joined together by logical disjunction (OR).
2. ***Application of Negation Normal Form (NNF):*** If necessary, statements are converted into Negation Normal Form, ensuring negations only relate to atomic formulas.
3. ***Resolution Process: Selection of Clause Pair:*** Two clauses are chosen for resolution. These clauses generally contain a pair of complementary literals (one positive, one negated).

   Literal Resolution: Complementary literals are identified within the selected clauses.

   Resolution Step: By merging the chosen clauses and removing the resolved literals, a new clause is created. This new clause is a disjunction of the remaining literals.
4. ***Iterative Iteration:*** The resolution rule is iteratively applied to yield new clauses. If an empty clause (contradiction) emerges, the original statements are shown to be inconsistent, thus substantiating the theorem.

Example

Consider the following situation with a family and their relationships. Based on the facts provided, we wish to prove that "John is Sarah's grandfather."

Given facts:

John is Alice's father.

Sarah's mother is Alice.

Goal: Demonstrate that "John is Sarah's grandfather."

Steps to Resolve

- **Clausal Form Conversion:**

  Fact 1: "Father(John, Alice)" transforms into "Father(John, Alice) True.

  Fact 2: "Mother(Alice, Sarah)" transforms into "Mother(Alice, Sarah) True.

- **Process of Resolution:**

  Choose "Father(John, Alice) True" and "Mother(Alice, Sarah) True."

  Determine the complementary literals "Father(John, Alice)" and "Father(John, Alice)."

The resolved clause is "True True," which is shortened to "True."

Conclusion:

The resolution method returns "True," indicating that the objective "John is the grandfather of Sarah" is correct based on the information provided.

Difference between First order logic and Propositional logic

Basis	First-Order Logic (FOL)	Propositional Logic
Definition	FOL is a formal logic that extends propositional logic by allowing variables. quantifiers, functions, and relations. It enables the representation of complex relationship and structured knowledge.	Propositional logic is a formal logic that deals with propositions, where propositions are atomic statements that can be either true or false. It focuses on truth values and logical connectives.
Variables	FOL uses variables to represent varying entities, object, and properties. It introduces the concept of quantified variables.	Propositional logic lacks variables; it only deals with atomic propositions.
Quantifiers	FOL employs quantifiers, such as universal ($\forall$) and existential ($\exists$), to express generalities and existence over variables.	Propositional logic lacks quantifiers; it cannot express statements about all or some entities.
Predicates and Functions	FOL accommodates predicates (relations) and functions to model complex properties and relationship among entities.	Propositional logic does not have predicates or functions; it deals with atomic propositions only.
Expressive Power	FOL is more expressive due to its ability to represent intricate relationships, quantified statements, and complex structures.	Propositional logic less expressive and is limited to representing simple propositions and truth values.
Deductive Reasoning	FOL support deductive reasoning about objects, properties, and relations. It offers more sophisticated inference methods like resolution.	Propositional logic allows for deduction but has simpler inference methods like modus ponens.
Applications	FOL is widely used in AI, robotics, databases, natural language processing, and knowledge representation due to its ability to model complex scenarios and relationships.	Propositional logic finds application in basis logic puzzles, automated reasoning, and fundamental knowledge representation.

4.10.5 Inference Engine

1. **Forward Chaining :** Forward chaining is a data-driven reasoning approach used by inference engines. It starts with given facts and applies rules to derive new conclusions or facts from them. The engine keeps applying the rules until it reaches a conclusion or cannot apply any more rules. It is based on logical prediction methodology. One of its examples is the prediction of trends in the stock market.

Characteristics

- It is a bottom-up approach.
- It starts with the initial state and progresses toward the end state to reach a conclusion.
- It is a data-driven approach which utilises given data.
- It derives conclusions without any need for explicit guidance.

Examples **Given below are some facts and some rules**

Facts:

1. Sohail is a coder.
2. Sohail studies computer science.

Rules:

1. If a person is a coder, it means he enrolled in a college.
2. If a person is enrolled in a college and studies computer science, it means he learned DSA.

Now, we will use forward chaining to derive a conclusion based on given facts and rules.

Step 1: If a person is a coder, it means he enrolled in a college.

- FOL: $\forall x$ (coder(x) $\rightarrow$ Enrolled_IN(x, college))
- Applying rule 1 to the fact: Enrolled_IN(Sohail, college)

Step 2: If a person is enrolled in a college and studied computer science, it means he learned DSA.

- FOL: $\forall x,y$ (Enrolled_In(x, college) $\wedge$ studies(x, computer science) $\rightarrow$ Learned_DSA(x))
- Applying rule 2 to the fact: Learned_DSA(Sohail)

Final conclusion: Learned_DSA(Sohail)

In the above example, we used forward chaining to derive a conclusion based on given facts and rules. We converted facts and rules into FOL and reached the final conclusion.

Note:

The FOL (First Order Logic) representation helps the inference engine to reach a conclusion effectively.

Advantages of Inference Engine

It is a great way to reach a conclusion based on available data.

Forward chaining can provide more information from limited data.

It is suited for expert systems where an application needs more control and monitoring.

It should be applied when there is a very few numbers of initial states available.

Disadvantages of Inference Engine

The inference engine generates information without knowing which data is more relevant and useful to reach the final result.

The inference engine may fire many unnecessary rules to reach the goal state.

The user may need to enter a lot of information to reach the goal state.

It can result in a high cost for the chaining process.

2. Backward Chaining

Backward chaining is also called backward reasoning. In this technique, the inference engine starts with the goal and works backwards to find evidence that supports the goal. In simple words, it works from the goal state and reaches the initial state.

Characteristics

- It is a top-down approach.
- It uses a depth-first search strategy.
- It is a goal-driven approach, as we start from the goal state and reach the initial state.

It divides objectives into sub-goals to validate the facts.

Examples **Given below are some facts and rules from which we would derive the conclusion.**

Fact:

1. Sohail is a coder.
2. Sohail studies computer science.

Rules:

1. If a person is a coder, it means they enrolled in a college.
2. If a person is enrolled in a college and studies computer science, it means they learned DSA.
3. **Goal:** Did Sohail learn DSA or not?

Now, we will use backward chaining to check whether Sohail learns DSA or not.

Step 1: If a person is enrolled in a college and studies computer science, it means they learned DSA. (Rule 1)

- FOL: $\forall x,y$ (Enrolled_In(x, college) $\wedge$ studies(x, computer_science) $\rightarrow$ Learned(x, DSA))

Now, we need to check if Sohail is enrolled in a college and studies computer science.

Step 2: If a person is a coder, it means they enrolled in a college. (Rule 2)

- FOL: $\forall x$ (coder(x) $\rightarrow$ Enrolled_In(x, college))

Step 3: We need to check if Sohail is a coder and studies computer science.

- Based on fact 1, Sohail is a coder.
- Based on fact 2, Sohail studies computer science.

From the above, we can conclude that Sohail is enrolled in a college (from Rule 1) and studies computer science. Therefore, using the backward chaining, we can conclude that Sohail learned DSA.

Advantages of Backward Chaining

- In backward chaining, the process terminates once the fact is verified.
- It is a faster process as it only analyses and verifies facts.
- It only considers the relevant part of the data and eliminates unnecessary information.
- It is very effective in solving problems like debugging and diagnosing.

Disadvantages of Backward Chaining

- In backward chaining, the goal should be known before starting the process.
- Backward chaining is very difficult and complex to implement.
- It can only generate a limited number of outcomes.
- It only tests for the required rules and neglects the others.

Forward Chaining vs Backward Chaining

Forward Chaining	Backward Chaining
In forward chaining, the decision is taken based on given data.	In backward chaining, the process starts from the goal state and reaches the initial state.
It is a data-driven technique	It is a goal-driven technique.
It is a bottom-up approach	It is a top-down approach
It uses a breadth-first search strategy	It uses a depth-first search strategy.
Its only goal is to reach a conclusion	Its goal is to validate the facts
It is a slow process	It is a fast process.
It operates in the forward direction (initial state to goal state)	It operates in the backward direction (goal state to initial state)
It may include multiple ASK questions from the information source.	Backward chaining includes fewer ASK questions compared to forward chaining.

4.10.6 Truth Maintenance System (TMS)

- The world of artificial intelligence is continuously evolving, and one of the essential components that have emerged in AI research is the Truth Maintenance System (TMS).
- A Truth Maintenance System is a knowledge representation and reasoning tool that assists AI systems in maintaining and updating their beliefs according to the available evidence.
- A TMS is designed to manage inconsistencies and contradictions, allowing AI systems to reason with incomplete or uncertain information.
- It achieves this by tracking dependencies between beliefs and assumptions, enabling the system to make informed decisions based on the current state of knowledge.

Types of Truth Maintenance Systems

- **Justification-based TMS :** A JTMS is a TMS that represents knowledge in the form of justifications. Each justification consists of a set of premises and a conclusion. When the premises of a justification are satisfied, the conclusion becomes a valid belief. A JTMS maintains consistency by ensuring that conflicting beliefs do not coexist.
- **Assumption-based TMS :** An ATMS, on the other hand, represents knowledge in terms of assumptions and their consequences. It focuses on exploring alternative sets of assumptions and their corresponding belief states. This approach is particularly useful in scenarios where multiple explanations or solutions are possible.

TMS Components

A typical TMS consists of three primary components:

1. **Nodes** : Nodes represent the beliefs or assertions in a TMS. They can be either true or false, depending on the available evidence and assumptions.
2. **Justifications :** Justifications are the links between nodes, representing the reasoning behind a belief. They contain a set of premises and a conclusion. When all premises are true, the conclusion is also considered true.
3. **Inference Procedures :** Inference procedures are algorithms that manipulate nodes and justifications to update the belief state of the TMS. They are responsible for maintaining consistency and resolving conflicts between beliefs.

TMS Applications in Artificial Intelligence

TMS has been employed in various AI applications, including:

- **Expert Systems** : Expert systems are AI programs that emulate human experts' reasoning and decision-making processes. TMS helps maintain the consistency of the knowledge base, allowing the system to reason with incomplete or uncertain information.
- **Planning Systems** : In planning systems, TMS assists in managing alternative plans and their assumptions, enabling the AI system to choose the most suitable plan based on the available information.
- **Diagnosis Systems :** TMS plays a crucial role in diagnostic systems, as it helps manage multiple hypotheses and their corresponding evidence. This allows the AI system to provide accurate diagnoses based on the available data.

Benefits of Using a Truth Maintenance System

There are several benefits to incorporating a TMS into an AI system:

- **Consistency Management**: TMS ensures that the AI system's beliefs remain consistent and free from contradictions.
- **Handling Uncertainty**: TMS allows AI systems to reason with incomplete or uncertain information, making them more robust and adaptable.
- **Belief Revision**: TMS enables AI systems to update their beliefs based on new evidence or changes in assumptions, ensuring that the system remains relevant and up-to-date.
- **Efficient Reasoning**: TMS can improve the efficiency of the reasoning process by keeping track of dependencies between beliefs and assumptions, which can help avoid unnecessary computations.

Challenges of Truth Maintenance Systems

Despite their advantages, TMS also faces some challenges:

- **Scalability**: As the size of the knowledge base grows, maintaining consistency and updating beliefs can become computationally expensive, making it challenging to scale TMS to larger AI systems.

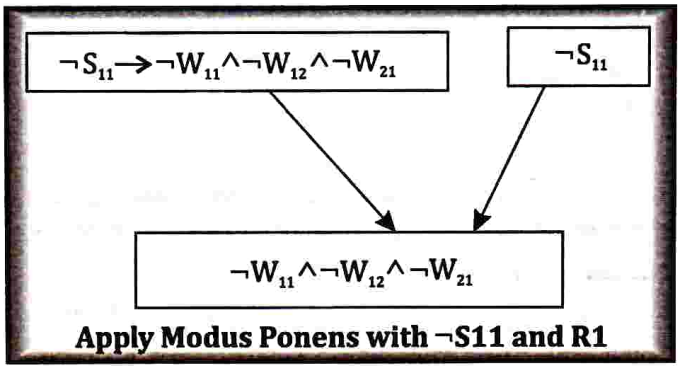
- **Complexity**: Managing dependencies between beliefs and assumptions can be complex, especially when dealing with multiple, conflicting beliefs.
- **Integration with Other AI Techniques**: Combining TMS with other AI techniques, such as machine learning or natural language processing, can be challenging due to their different knowledge representation and reasoning mechanisms.

Wumpus problem using Inference Rules

Prove that Wumpus is in the room (1, 3)

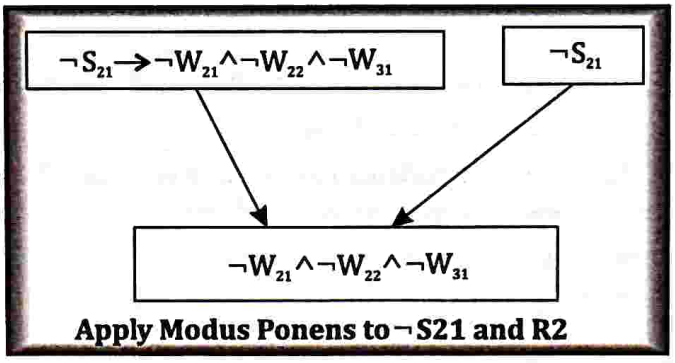
We can prove that wumpus is in the room (1, 3) using propositional rules which we have derived for the wumpus world and using inference rule.

- **Apply Modus Ponens** with ¬S11 and R1: At first we will apply MP rule with R1 which is ¬S11 → ¬W11^ ¬W12 ^ ¬W21, and ¬S11 which will give this output ¬ W11 ^ W12 ^ W12



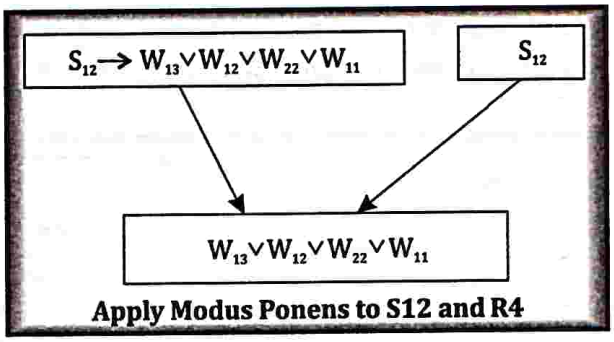
Apply Modus Ponens with ¬S11 and R1

- **Apply And-Elimination Rule**: After we apply And-elimination rule to ¬W11 ∧ ¬W12 ∧ ¬W21, we will see three statements: ¬W11, ¬W12, and ¬W21.
- **Apply Modus Ponens to ¬S21, and R2**: We will now apply Modus Ponens to ¬S21 and R2 which is ¬S21 → ¬W21 ∧¬ W22 ∧ ¬W31, which will give the Output as ¬W21 ∧ ¬W22 ∧ ¬W31

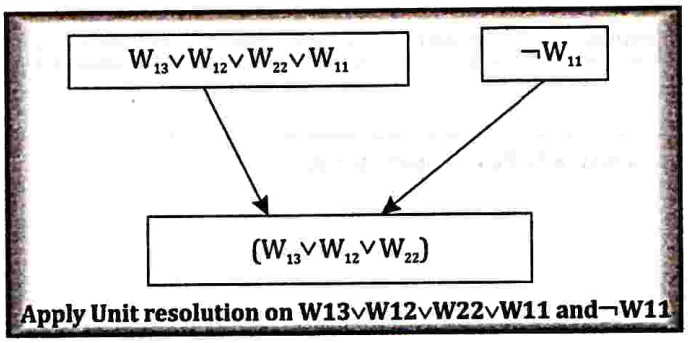


Apply Modus Ponens to ¬S21 and R2

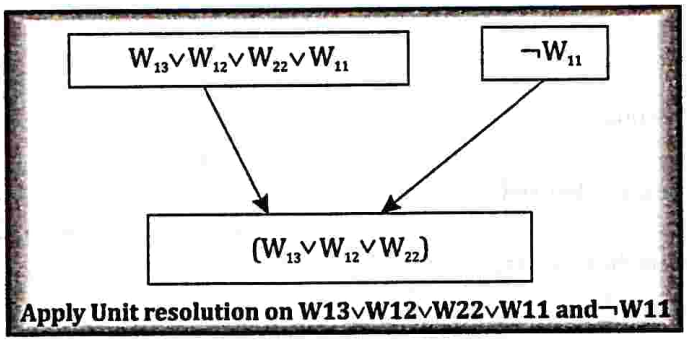
- **Apply And -Elimination rule**: Again we will now apply And-elimination rule to ¬ W21 ∧ ¬ W22 ∧¬ W31, We will see three statements:¬W21, ¬W22, and ¬W31.
- **Apply MP to S12 and R4:** Apply Modus Ponens to S12 and R4 which is S12 → W13 ∨. W12 ∨. W22 ∨. W11, we will get the output as W13 ∨ W12 ∨ W22 ∨ W11.



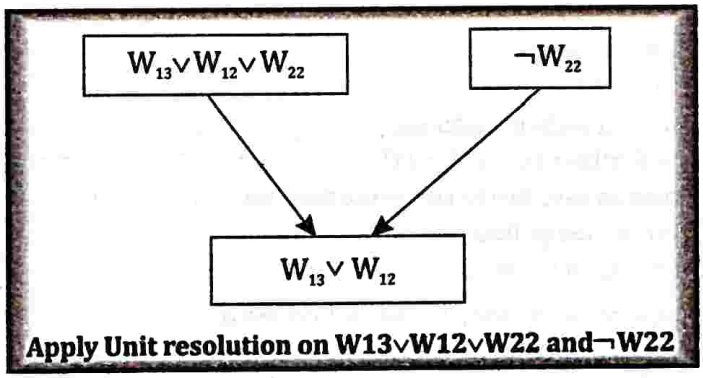
- **Apply Unit resolution on W13 ∨ W12 ∨ W22 ∨ W11 and ¬W11** : After applying Unit resolution formula on W13 ∨ W12 ∨ W22 ∨ W11 and ¬W11 we will see W13 ∨ W12 ∨ W22



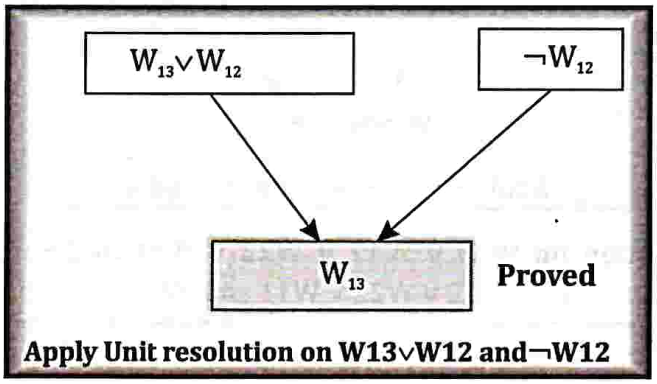
- **Apply Unit resolution on W13 ∨ W12 ∨ W22 ∨ W11 and ¬W11** : After applying Unit resolution formula on W13 ∨ W12 ∨ W22 ∨ W11 and ¬W11 we will see W13 ∨ W12 ∨ W22.



- **Apply Unit resolution on W13 ∨ W12 ∨ W22 and ¬W22** : After applying Unit resolution on W13 ∨ W12 ∨ W22, and ¬W22, we will get W13 ∨ W12 as output.



- **Apply Unit Resolution on W13 ∨ W12 and ¬W12 :** After Applying Unit resolution on W13 ∨ W12 and ¬W12, we will see W13 as an output, therefore, it is proved that the Wumpus is in the room [1, 3].



Example : 1

Represent the following statements in FOPL /Symbolic form.

1. Caesar was a ruler

   $\underbrace{\text{Ruler}}_{\text{Predicate}}\underbrace{(\text{CAESAR})}_{\text{constant}}$

2. Everyone loves everyone

   $\forall x \forall y \text{ Loves } \underbrace{(x,y)}_{\text{quantified variables}}$

3. Everyone loves someone.
   $\forall x \exists y$ loves (x, y)
4. Everyone loves himself or her self
   $\forall x$ Loves (x, x)
5. Everyone loves him or her parents
   $\forall x$ Loves (x, parents (x))
   ↓
   Function
6. There is someone who loves every one
   $\exists x \forall y$ Loves (x, y)
7. Everything with feathers is a bird
   $\forall x$ [with -feathers (x) → Bird (x)]
8. Man is a two leged animal without feathers.
   $\forall x$ [Two legs (x) ∧ ~ Feathers (x) → Man (x)
9. If a person cannot receive an idea, then he cannot use that idea.
   $\forall p$ [~ Receive (p, IDEA) → ~ use (p, IDEA)].
10. All Romans were either loyal to caesar or hated him.
    $\forall x$ [Roman (x) → Loyalto (x, CAESAR) V Hate (x, CAESAR)]

These symbolic forms are called as well formed formulas (wff's)

Example : 2

Invalid wffs.

father-of (Loves (x)) ↓ ↓ function Predicate	-	A function cannot have a predicate as argument.
Man (~JOHN) ↓ Constant	-	Constants should not be negated. Only the predicates can be negated but not the terms.
$\forall_p P(x) \rightarrow Q(x)$	-	Predicate symbols should not be quantified
Married (Man, Women) ↓ Predicates	-	Predicates cannot have predicates as arguments.

Syntax:

wffs are defined recursevely as follows.

An atomic formula is a wff.

If P and Q are wffs,

then $\sim P$

$p \wedge Q$

$p \vee Q$

$p \rightarrow Q$

$p \leftrightarrow Q$

$\forall x\, p(x)$

and $\exists\, x\, p(x)$

are wffs.

wffs are formed only by applying the above rules a finite number of times.

Semantics of FOPL

Semantics or meaning of a wff is the assignment of truth values (true/ false) to the predicate symbols.

Properties of wff is the same as PL.

Equivalence laws of FOPL

All the equivalence laws used in PL is applicable to FOPL. In addition FOPL has the following.

$\forall x\, F(x) \vee G = \forall x\, (F(x) \vee G)$

$\exists\, x\, F(x) \vee G = \exists\, x\, (F(x) \vee G)$

$\forall x\, F(x) \wedge G = \forall x\, (Fx) \wedge G\,)$

$\exists\, x\, F(x) \wedge G = \exists\, x\, (F(x) \wedge G)$

$\forall x\, F(x) \vee \forall x\, G(x) = \forall x\, (F(x)\ V\ G(x))$

$\exists x\, F(x) \vee \exists x\ G(x) = \exists x\, (F(x)\ VG(x))$

$\sim (\forall x)\, F(x)$

$= \exists x\, (\sim F(x))$

$\sim (\exists x)\, F(x)$

$= \forall x\, (\sim F(x))$

Example : 3

For the following set of sentences write wffs in predicate logic.

1. Marcus was a man.
2. Marcus was a Pompeian
3. All Pompeians were Romans
4. Caesar was a ruler.
5. All Romans were either loyal to Caesar or hated him.
6. Everyone is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.
9. Men only try to assassinate rulers they are not loyal to
10. All Pompeians died in 79AD

The facts described by these sentences can be represented as a set of wff's in predicate logic as follows:

1. Marcus was a Man.
   Man(Marcus)
2. Marcus was a Pompeian.
   Pompeian(Marcus)
3. All Pompeians were Romans.
   $\forall x$: Pompeian(x)$\rightarrow$ Roman(x)
4. Caesar was a ruler.
   Ruler(Caesar)
5. All Romans were either loyal to Caesar or hated him.
   $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) V hate(x, Caesar)
6. Everyone is loyal to someone.
   $\forall x\, \exists y$: loyalto(x,y)
7. People only try to assassinate rulers they are not loyal to.
   $\forall x\, \forall y$: person(x) ^ ruler(y) ^ tryassassinate(x,y) $\rightarrow \neg$ loyalto(x,y)
8. Marcus tried to assassinate Caesar.
   tryassassinate(Marcus, Caesar)
9. Men only try to assassinate rulers they are not loyal to
   $\forall w$: $\forall u$:(Man(w) ^ Ruler (u) ^ tryassassinate(w, u) $\rightarrow \neg$ loyalto(w,u)
10. All Pompeians died in 79AD
    $\forall x$:(Pompeian(x)$\rightarrow$ Died (x, 79AD))

Example : 4

Convert all the given statements into first order predicate logic.

a. John likes all kind of food.
b. Apple and vegetable are food
c. Anything anyone eats and not killed is food.
d. Anil eats peanuts and still alive
e. Harry eats everything that Anil eats.
f. John likes peanuts.

a. $\forall x$: food (x) $\rightarrow$ likes (john, x)
b. food (Apple) $\wedge$ food (vegetables)
c. $\forall x \forall y$: eats (x, y) $\wedge \neg$ killed (x) $\rightarrow$ food (y)
d. eats (Anil, Peanuts) $\wedge$ alive (Anil)
e. $\forall x$: eats (Anil, x) $\rightarrow$ eats (Harry, x)
f. likes (John, Peanuts)

Example : 5

Write wffs in predicate logic for the following

1. Everybody loves Raymond.
2. Everybody loves somebody.
3. There is somebody whom everybody loves.
4. There is somebody who Raymond doesn't love.
5. There is somebody whom no one loves.

Let Variables x,y denote people and L(x,y) denotes "x loves y".

Therefore the translation is as follows

1. $\forall x$:L(x, Raymond)
2. $\forall x \exists y$:L(x,y)
3. $\exists y \forall x$:L(x,y)
4. $\exists y$:$\neg$L(Raymond,y)
5. $\exists y \forall x$: $\neg$L(x,y)

Example : 6

Consider the following

1. Policeman goes wherever thief goes
2. Thief is at bank

Represent these statements in FOPL.

1. $\forall x$:(At(Thief, x)) → At(policeman, x))
2. At(Thief, Bank)

Example : 7

Consider the following facts

1. The members of the Elm St. Bridge Club are Joe, Sally, Bill and Ellen.
2. Joe is married to Sally.
3. Bill is Ellen's brother.
4. The spouse of every married person in the club is also in the club.
5. The last meeting of the club was at Joe's house.

Represent these facts in FOPL.

1. Member(Joe, SBC) ∧ Member(Sally, SBC) ∧ Member(Bill, SBC) ∧ Member(Ellen, SBC)
2. Married(Joe) → Husband(Joe, Sally)
3. Brother(Bill, Ellen)
4. $\forall x \forall y$: (Married(x) ^Member(x, SBC)) → Husband(x,y) ∧ Present(y, SBC)
5. LastMeeting(SBC, JoeHouse)

Example : 8

1. All people who are not poor and are smart are happy.
2. Those people who read are not stupid.
3. John can read and is wealthy.
4. Happy people have exciting lives.

1. $(\forall x)[\{\sim$ poor (x) ∧ smart (x)$\}$ → happy (x)]
2. $(\forall y)$[read (y)→ stupid(y)]
3. read (Join) ∧ ~ Poor (John)
4. $(\forall y)$ [happy (z) → exciting (z)]

Example : 9

1. Mary loves everyone.
2. Mary loves everyone.
3. No one talks.
4. Everyone loves himself.
5. Everyone loves everyone.
6. Everyone loves everyone except himself.
7. Every student smiles.
8. Every student except George smiles.
9. Everyone walks or talks.
10. Every student walks or talks
11. Every student who walks talks.
12. Every student who loves Mary is happy.
13. Every boy who loves Mary hates every boy who Mary loves.
14. Every boy who loves Mary hates every other boy who Mary loves.

1. ∀x: love (Mary, x)
2. ∀x: (person(x) → love (Mary, x))
3. ∀x:¬talk(x)
4. ∀x: love (x, x)
5. ∀x:∀y love (x, y)
6. ∀x:∀y(¬ x = y → love (x, y))
7. ∀x: (student(x) → smile(x))
8. ∀x: ((student(x) → (x ≠ George ↔ smile(x)))
9. ∀x: (walk (x) ∨ talk (x))
10. ∀x: (student(x) → (walk (x) ∨ talk (x)))
11. ∀x: ((student(x) ∧ walk (x)) → talk (x)))
12. ∀x: ((student(x) ∧ love (x, Mary)) → happy (x)))
13. ∀x:((boy(x) ∧ love (x, Mary)) → ∀y((boy(y) ∧ love(Mary, y))→ hate (x,y)))
14. ∀x:((boy(x) ∧ love (x, Mary)) → ∀y((boy(y) ∧ love(Mary, y) ∧ y ≠ x) → hate (x,y)))

Summary

- Knowledge is the basic element for a human brain to know and understand the things logically. When a person becomes knowledgeable about something, he is able to do that thing in a better way. In AI, the agents which copy such an element of human beings are known as knowledge-based agents.
- **Types of Knowledge :** Meta Knowledge, Heuristic Knowledge, Procedural Knowledge, Declarative Knowledge, Structural Knowledge
- **Knowledge-based agents** represent searchable knowledge that can be reasoned. These agents maintain an internal state of knowledge, take decisions regarding it, update the data, and perform actions on this data based on the decision. Basically, they are intelligent and respond to stimuli like how humans react to different situations
- A knowledge-based system comprises of two **distinguishable features** they are
  A Knowledge base & An Inference Engine
- Knowledge-base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.
- **Inference system** generates new facts so that an agent can update the KB. An inference system works mainly in two rules which are given as: Forward chaining & Backward chaining
- Various level of Knowledge based agent : Knowledge level, Logical level & Implementation level.
- Declarative approach & Procedural approach are the two approaches to design a knowledge based agent.
- The three major components of logic : Syntax, Semantic & Logical inference.
- **Propositional logic** (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.
- **Atomic** means a single preposition like 'the sky is blue', 'hot days are humid', water is liquid, etc. It is made up of only one proposition sign. These are the sentences that must be true or untrue in order to pass.

- **Complex prepositions** are those, which have been formed by connecting one, two, or more sentences. In propositional logic, there are five symbols(***logical connectives***) to create the syntax to represent the connection of two or more sentences.
- ***Logical connectives*** are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives
- The proposition symbols P, Q etc are sentences
  - If P is a sentence, ¬P is a sentence (negation)
  - If P and Q are sentences, P ∧ Q is a sentence (conjunction)
  - If P and Q are sentences, P ∨ Q is a sentence (disjunction)
  - If P and Q are sentences, P⇒ Q is a sentence (implication)
  - If P and Q are sentences, P ⇔ Q is a sentence (biconditional)
- **Logical equivalence** is one of the features of propositional logic. Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.
- **First-Order Logic** (FOL) is a logical way of allowing us to use statements to express relationships between things. It is also known as First-Order Predicate Calculus or First-Order Predicate Logic, adds quantifiers, variables, and predicates to propositional logic.
- The basic elements of First Order Logic in Artificial Intelligence- Constants, Variables, Predicates, Function, Connectives, Equality & Quantifiers.
- In First-Order Logic, **quantifiers** allow us to discuss collections of objects, qualities, or relationships as a group. They help us in expressing our thoughts on whether particular statements apply to all members of a group or apply to particular members that fit a specified requirement.
- The **Universal Quantifier (∀)** is a logic concept that allows you to make statements about each and every individual member in a group. It's equivalent to stating "for all" or "for every". It states a condition that holds true for every member of a set or collection in mathematical and logical terms.
- The **Existential Quantifier (∃)** is a key idea in logic that allows you to say that at least one member in a group or domain meets a certain condition. In other words, it's the same as stating "there is" or "there exists" in daily speech. It states the presence of at least one member who fits a specific requirement in logical terms
- Modus ponens, Modus Tollens, Universal Instantiation, Universal Generalization, Existential Instantiation, Existential Generalisation - **Rules of Inference in FOL**
- **Unification** in First-Order Logic deals with finding a common substitution for variables in different terms to make them match. The goal of unification is to make two expressions identical by assigning values to variables in a way that preserves their meanings. It involves substituting terms for variables in a way that the resulting expressions match.
- In First-Order Logic (FOL), **resolution** is a fundamental inference method for theorem proving and logical reasoning. It's very beneficial for showing the truth of statements through contradiction. The resolution principle helps in the generation of new clauses from old ones, resulting in the resolution of difficult logical issues.

- **Forward chaining** is a data-driven reasoning approach used by inference engines. It starts with given facts and applies rules to derive new conclusions or facts from them. The engine keeps applying the rules until it reaches a conclusion or cannot apply any more rules. It is based on logical prediction methodology. One of its examples is the prediction of trends in the stock market.
- **Backward chaining** is also called backward reasoning. In this technique, the inference engine starts with the goal and works backwards to find evidence that supports the goal. In simple words, it works from the goal state and reaches the initial state.
- A **Truth Maintenance System** is a knowledge representation and reasoning tool that assists AI systems in maintaining and updating their beliefs according to the available evidence. A TMS is designed to manage inconsistencies and contradictions, allowing AI systems to reason with incomplete or uncertain information.

4.11 Review Questions

Short Answer Questions

1. List different types of Knowledge
2. Define Knowledge base system in AI.
3. Why use a Knowledge base ?
4. Define Inference Engine. List the two rules of Inference engine.
5. List the two approaches to build Knowledge base system.
6. Define Propositional logic.
7. List two types of Propositional logic.
8. List all Propositional logical connectives.
9. Define implication with truth table.
10. Define disjunction with truth table.
11. Define Logic equivalence.
12. State any two limitations of Propositional logic.
13. Define FOL in AI.
14. What are Quantifiers in FOL?
15. Define Universal Quantifier?
16. Define Existential Quantifier?
17. List the rules of inference in FOL.
18. Define Unification in FOL.
19. Define Resolution in FOL.
20. State any two difference between FOL & Propositional logic.
21. Define forward chaining.

22. Define backward chaining
23. State any two advantages of forward chaining
24. State any two advantages of backward chaining
25. State any two disadvantages of forward chaining
26. State any two disadvantages of backward chaining
27. List any two difference between forward and backward chaining
28. Define TMS (Truth Maintenance System).
29. List applications of TMS in AI
30. List any two challenges of TMS in AI

Long Answer Questions

1. With a neat diagram, Explain the Knowledge-Based System in AI.
2. With diagrammatic representation, Illustrate Wumpus World as an example for Knowledge based agent and determine how the agent will find its goal by applying logical reasoning.
3. How Propositional Logic in Artificial Intelligence Represents Data to Machine ? Explain all logical connectives with truth table.
4. With an example, Explain FOL in AI.
5. Write a note on Quantifiers in FOL.
6. With an example, Explain rules of inference in FOL.
7. Define Unification in FOL. Write the pseudocode of Unification.
8. With an example, Explain Resolution in FOL.
9. Differentiate between First order logic and Propositional logic.
10. With an example, Explain forward chaining.
11. With an example, Explain backward chaining.
12. Write a note on Truth Maintenance System.

UNIT - II

CHAPTER 5

KNOWLEDGE IN LEARNING

- What is Learning ?
- Types of Learning :
  - Rote Learning
  - Learning by taking Advice :
  - Learning in Problem Solving
    - Learning by Parameter Adjustment
    - Learning with Macro-operations
    - Learning by Chunking
    - The Utility Problem
  - Learning from examples (Induction Learning)
    - Winston's Learning Program
    - Decision Trees
- Review Questions

5.1 What is Learning ?

Most often heard criticisms of AI is that machines cannot be called intelligent until they are able to learn to do new things and adapt to new situations, rather than simply doing as they are told to do.

Some critics of AI have been saying that computers cannot learn!

Definitions of Learning: changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.

- Learning is a an area of AI that focusses on processes of self-improvement.
- Information processes that improve their performance or enlarge their knowledge bases are said to learn.

5.2 Types of Learning

Many approaches have been taken to attempt to provide a machine with learning capabilities. This is because learning tasks cover a wide range of phenomenon :

- Simple storing of computed information or ***rote learning*** is the most basic learning activity.
- Many computer programs i,e database systems can be said to learn in this sense although most people would not call such ***simple storage learning***.
- Another way we learn if through taking advice from others. ***Advice taking*** is similar to rote learning, but high-level advice may not be in a form simple enough for a program to use directly in problem solving.
- People also learn through their own ***problem-solving*** experience.
- ***Learning from examples*** : we often learn to classify thing in the world without being given explicit rules.
- Learning from examples usually involves a teacher who help us classify things by correcting us when we are wrong.

5.2.1 Rote Learning

The meaning of rote in 'rote learning' itself means learning by repetition. che. It differs from other forms of learning in that it doesn't require the learner to carefully think about something, and is rather dependent on the act of repetition itself.

Even though complete and holistic learning is not dependent on rote learning techniques alone, they do allow students to quickly recall basic facts and laws and master foundational knowledge of a topic in students. Some examples of rote learning in schools can be found in the following:

- Repeating words to instil them in your vocabulary.
- Learning scales in music.
- Memorizing the periodic table.
- Learning the basic laws and formulae in physics and sundry sciences.
- Learning statutes and cases in law.

Having to memorize the basic facts and principles of a field is an important prerequisite to later analyse and study them. This is where rote learning techniques come in handy and allow you to remember the building blocks of concepts without having to dive deep into them.

Rote Learning Techniques

Rote learning techniques are aplenty, and they all require time and effort in repetition. The more you repeat for longer periods, the easier it will be to recall. Even if you only have a few hours to memorize something, the following rote learning techniques will help you remember quickly:

- **Read it Aloud** - Read the text out loud with understanding. You can even try it before a mirror, ask a friend to listen to you, or read it out just under your breath. You can experiment with how slow or fast you want to read, how expressive you want to be, and internalize the rhythm of the text. Auditory learners will greatly benefit from this rote learning technique.
- **Write it Down** - Writing down the text information after reading is one of the best rote learning techniques. Doing so will help identify difficult passages and areas that need more practice. If you're preparing for a written exam, this kinaesthetic rote technique will serve as a rehearsal and commit the information for easy retrieval.
- **Sing it Out** - There's a reason why songs commit to memory easier than text that is spoken simply and without any variation in pitch. So try putting the text that you want to learn with a melody that you like. Or, if you're feeling creative, come up with your own catchy tune to remember the text.
- **Visualize** - Humans are visual creatures and our brains are wired to remember things better with images. For every line and connected phrase, come up with ways to visualize it and remember it. The memory palace can be a useful trick for such rote learning techniques.
- **Free Association** - Free association is one of the more interesting rote learning techniques, and a very useful way of remembering things quickly, especially if they are too messy for the traditional rote learning techniques. The main idea of this method is to combine new information with what you already know in a fun and personal way. For instance, if you're learning the 'Circle of Fifths' in music, you can associate each note to the numbers on the clock, one for each of the 12 notes in music. You are free to form your own associations as you see fit, as long as it helps you to recall the information.

Advantages of Rote Learning Techniques

Rote learning is considered useful for a variety of reasons. Here are a few:

1. Rote learning requires very little analysis.
2. With rote, one can remember just about anything over time and repetition.
3. Rote learning allows one to recall information wholly, and even to retain it for life.
4. Rote learning makes it easier for people to score who find it difficult to understand or master reading and maths concepts.
5. Rote learning can help improve short-term memory.

Disadvantages of Rote Learning Techniques

On the other hand, there are a few drawbacks of rote learning that you need to be aware of as well.

1. The repetitive nature of rote learning can become dull.
2. One can easily lose focus while rote learning.
3. Rote learning is not holistic.
4. There is no connection between new and old information with rote learning.
5. Rote learning doesn't lead to a deeper understanding of the information.

Example : Samuel's Checkers Program employed Rote Learning - A Case Study

Samuel is most known within the AI community for his ground breaking work in computer checkers in 1959, and seminal research on machine learning, beginning in 1949. He believed teaching computers to play games was very fruitful for developing tactics appropriate to general problems, and he chose checkers as it is relatively simple though has a depth of strategy. The main driver of the machine was a search tree of the board positions reachable from the current state. Since he had only a very limited amount of available computer memory, Samuel implemented what is now called alpha-beta pruning. Instead of searching each path until it came to the game's conclusion, Samuel developed a scoring function based on the position of the board at any given time. This function tried to measure the chance of winning for each side at the given position. It took into account such things as the number of pieces on each side, the number of kings, and the proximity of pieces to being "kinged". The program chose its move based on a minimax strategy, meaning it made the move that optimized the value of this function, assuming that the opponent was trying to optimize the value of the same function from its point of view.

Samuel also designed various mechanisms by which his program could become better. In what he called rote learning, the program remembered every position it had already seen, along with the terminal value of the reward function. This technique effectively extended the search depth at each of these positions. Samuel's later programs re-evaluated the reward function based on input from professional games. He also had it play thousands of games against itself as another way of learning. With all of this work, Samuel's program reached a respectable amateur status, and was the first to play any board game at this high a level. He continued to work on checkers until the mid-1970s, at which point his program achieved sufficient skill to challenge a respectable amateur.

5.2.2 Learning by taking Advice

- This is a simple form of learning requires more inference than rote learning. In this type of learning, a programmer writes a program to give some instructions to perform a task to the computer. Once it is learned (i.e. programmed), the system will be able to do new things. Also, there can be several sources for taking advice such as humans(experts), internet etc.
- This type of learning has a more necessity of inference than rote learning. As the stored knowledge in knowledge base gets transformed into an operational form, the reliability of the knowledge source is always taken into consideration.
- The programs shall operationalize the advice by turning it into a single or multiple expressions that contain concepts and actions that the program can use while under execution. This ability to operationalize knowledge is very critical for learning. This is also an important aspect of Explanation Based Learning (EBL)

First Operational Operationalizer (FOO)

FOO (First Operational Operationaliser), for example, is a learning system which is used to learn the game of Heart

Hearts

- Game played as a series of tricks.
- One player - who has the lead - plays a card.
- Other players follow in turn and play a card.
- The player must follow suit.
- If he cannot he play any of his cards.
- The player who plays the highest value card wins the trick and the lead.
- The winning player takes the cards played in the trick.
- The aim is to avoid taking points. Each heart counts as one point the queen of spades is worth 13 points.
- The winner is the person that after all tricks have been played has the lowest points score.

Hearts is a game of partial information with no known algorithm for winning.

Although the possible situations are numerous general advice can be given such as:

- Avoid taking points.
- Do not lead a high card in suit in which an opponent is void.
- If an opponent has the queen of spades try to flush it.

In order to receive advice a human must convert into a FOO representation (LISP clause)

- FOO operationalises the advice by translating it into expressions it can use in the game. It can UNFOLD avoid and then trick o give:

```
(achieve (not (during
        (scenario
         (each p1 (players) (play-card p1))
         (take-trick (trick-winner)))
         (take-points me))))
```

- However the advice is still not operational since it depends on the outcome of trick which is generally not known. Therefore FOO uses case analysis (on the during expression) to determine which steps could case one to take points. Step 1 is ruled out and step 2's take-points is UNFOLDED:

```
(achieve (not (exists c1 (cards-played)
         (exists c2 (point-cards)
         (during (take (trick-winner) c1)
              (take me c2))))))
```

- FOO now has to decide: Under what conditions does (take me c2) occur during (take (trick-winner) c1).
- A technique, called partial matching, hypothesises that points will be taken if me = trick-winner and c2 = c1. We can reduce our expression to:

  (achieve (not (and (have-points(card-played))

  (= (trick-winner) me))))

  This not quite enough a this means Do not win trick that has points. We do not know who the trick-winner is, also we have not said anything about how to play in a trick that has point led in the suit. After a few more steps to achieve this FOO comes up with:

  (achieve (>= (and (in-suit-led(card-of me))

  (possible (trick-has-points)))

  (low(card-of me)))
- FOO had an initial knowledge base that was made up of:
  - basic domain concepts such as trick, hand, deck suits, avoid, win etc.
  - Rules and behavioural constraints -- general rules of the game.
  - Heuristics as to how to UNFOLD.
- FOO has 2 basic shortcomings:
  - It lacks a control structure that could apply operationalisation automatically.

5.2.3 Learning in Problem Solving

When the program does not learn from advice, it can learn by generalizing from its own experiences.

- Learning by parameter adjustment
- Learning with macro-operators
- Learning by chunking
- The unity problem

5.2.3.1 Learning by Parameter Adjustment

- Here the learning system relies on evaluation procedure that combines information from several sources into a single summary static. For example, the factors such as demand and production capacity may be combined into a single score to indicate the chance for increase of production. But it is difficult to know a priori how much weight should be attached to each factor. The correct weight can be found by taking some estimate of the correct settings and then allow the program modify its settings based on its experience. Features that appear to be good predictors of overall success will have their weights increases, while those that do not will have their weights decreased. This type of learning systems is useful when little knowledge is available.
- In game programs, for example, the factors such as piece advantage and mobility are combined into a single score to decide whether a particular board position is desirable. This single score

is nothing but a knowledge which the program gathered by means of calculation. Programs do this in their static evaluation functions, in which a variety of factors are combined into a single score. This function as a polynomial form is given below:

$$c_1t_1 + c_2t_2 + c_3t_3 + ...$$

The t terms are the values of the features that contribute to the evaluation. The c terms are the coefficients or weights that are attached to each of these values. As learning progresses, the c values will change. In designing programs it is often difficult to decide on the exact value to give each weight initially. So the basic idea of idea of parameter adjustment is to:

- Start with some estimate of the correct weight settings.
- Modify the weight in the program on the basis of accumulated experiences.
- Features that appear to be good predictors will have their weights increased and bad ones will be decreased

Important Factors

- When should the value of a coefficient be increased and when should it be decreased
  - The coefficients of terms that predicted the final outcome accurately should be increased, while the coefficients of poor predictors should be decreased.
  - The problem of appropriately assigning responsibility to each of the steps that led to a single outcome is known as credit assignment system.
- By how much should be value be changed.
  - Learning procedure is a variety of hill-climbing.

This method is very useful in situations where very little additional knowledge is available or in programs in which it is combined with more knowledge intensive methods.

5.2.3.2 Learning with Macro-operations

- Sequence of actions that can be treated as a whole are called macro-operators. Once a problem is solved, the learning component takes the computed plan and stores it as a macro-operator. The preconditions are the initial conditions of the problem just solved, and its post conditions correspond to the goal just achieved.
  - The problem solver efficiently uses the knowledge base it gained from its previous experiences. By generalizing macro-operators the problem solver can even solve different problems .Generalization is done by replacing all the constants in the macro-operators with variables.

The STRIPS, for example, is a planning algorithm that employed macro-operators in it's learning phase.

- It builds a macro operator MACROP, that contains preconditions, postconditions and the sequence of actions.
- The macro operator will be used in the future operation. The set of problems for which macro-operators are critical are exactly those problems with non-serializable sub goals.(working on one subgoal will necessarily interfere with the previous solution to another subgoal).

- One macro operator can produce a small global change in the world, even though the individual operators that make it up produce many undesirable local changes. Domain specific knowledge we need can be learnt in the form of macro operators
- Consider a blocks world example in which ON(C,B) and ON(A,TABLE) are true. STRIPS can achieve ON(A,B) in four steps:
  - ✓ UNSTACK(C,B)
  - ✓ PUTDOWN(C)
  - ✓ PICKUP(A)
  - ✓ STACK(A,B)

  STRIPS now builds a macro-operator MACROP with preconditions ON(C,B), ON(A,TABLE), postconditions ON(A,B), ON(C,TABLE) and the four steps as its Body

5.2.3.3 Learning by Chunking

- Chunking is similar to learning with macro-operators. Generally, it is used by problem solver systems that make use of production systems.
- A production system consists of a set of rules that are in if-then form. That is given a particular situation, what are the actions to be performed. For example, if it is raining then take umbrella.
- Production system also contains knowledge base, control strategy and a rule applier. To solve a problem, a system will compare the present situation with the left hand side of the rules. If there is a match then the system will perform the actions described in the right hand side of the corresponding rule.
- Problem solvers solve problems by applying the rules. Some of these rules may be more useful than others and the results are stored as a chunk.
- Chunking can be used to learn general search control knowledge. Several chunks may encode a single macro-operator and one chunk may participate in a number of macro sequences
- Chunks learned in the beginning of problem solving, may be used in the later stage. The system keeps the chunk to use it in solving other problems.
- **Soar** is a general cognitive architecture for developing intelligent systems. Soar requires knowledge to solve various problems. It acquires knowledge using chunking mechanism. The system learns reflexively when impasses have been resolved. An impasse arises when the system does not have sufficient knowledge. Consequently, Soar chooses a new problem space (set of states and the operators that manipulate the states) in a bid to resolve the impasse. While resolving the impasse, the individual steps of the task plan are grouped into larger steps known as chunks. The chunks decrease the problem space search and so increase the efficiency of performing the task.
- In Soar, the knowledge is stored in long-term memory. Soar uses the chunking mechanism to create productions that are stored in long-term memory. A chunk is nothing but a large production that does the work of an entire sequence of smaller ones. The productions have a set of conditions or patterns to be matched to working memory which consists of current goals,

problem spaces, states and operators and a set of actions to perform when the production fires. Chunks are generalized before storing. When the same impasse occurs again, the chunks so collected can be used to resolve it

5.2.3.4 The Utility Problem

- The utility problem in learning systems occurs when knowledge learned in an attempt to improve a system's performance degrades it instead. The problem appears in many AI systems, but it is most familiar in speedup learning.
- Speedup learning systems are designed to improve their performance by learning control rules which guide their problem-solving performance. These systems often exhibit the undesirable property of actually slowing down if they are allowed to learn in an unrestricted fashion.
- Each individual control rule is guaranteed to have a positive utility (improve performance) but, in concert, they have a negative utility (degrade performance).
- One of the causes of the utility problem is the serial nature of current hardware. The more control rules that speedup learning systems acquire, the longer it takes for the system to test them on each cycle.
- One solution to the utility problem is to design a parallel memory system to eliminate the increase in match cost. his approach moves the matching problem away from the central processor and into the memory of the system. These so-called active memories allow memory search to occur in "nearly constant-time" in the number of data items, relying on the memory for fast, simple inference and reminding.
- PRODIGY program maintains a utility measure for each control rule. This measure takes into account the average savings provided by the rule, the frequency of its application and the cost of matching it.
- If a proposed rule has a negative utility, it is discarded or forgotten. If not, it is placed in long term memory with the other rules. It is then monitored during subsequent problem solving.
- Empirical experiments have demonstrated the effectiveness of keeping only those control rules with high utility. Such utility considerations apply to a wide range of learning problems

5.2.4 Learning from examples (Induction Learning)

- Induction learning is carried out on the basis of supervised learning. In this learning process, a general rule is induced by the system from a set of observed instance. However, class definitions can be constructed with the help of a classification method.
- ***Classification*** is a process of assigning to a particular input, to the name of the class to which it belongs to. Classification is important component in many problem solving tasks. But often classification is embedded inside another operation.
- For example: If: the current goal is to get from place A to place B and there is a wall separating two places then look for a Doorway in the wall and through it. To use this rule successfully, the system's matching routine must be able to identify an object as a wall. Without this the rule can never be invoked. Then to apply the rule, the system must be able to recognize the doorway.

- Before classification is done , the classes it will use must be defined . This can be done in variety of ways:
  - Isolate a set of features that are relevent to task domain. Define each class by some values of these features. Example: for weather predictions the parameters can be of rainfall, sunny, cloudy
  - Define a class as a structure composed of those features. For example if the task is to identify animals, the body of each type of animal can be stored as structure and various features like color, length of a neck can be represented. The task of constructing class definitions is called ***concept learning or Induction.***

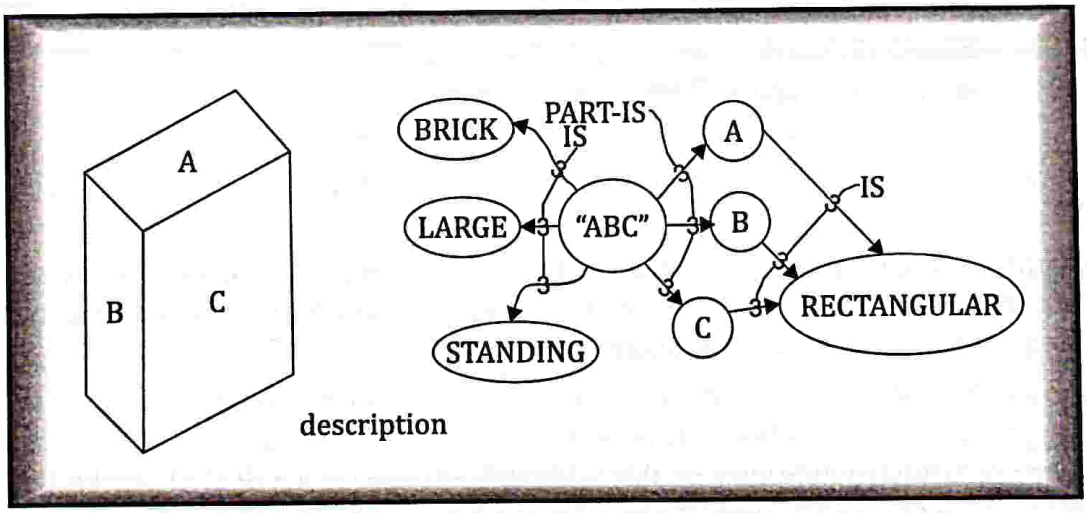
5.2.4.1 Winston's Learning Program

Winston's doctoral thesis at MIT entitled "Learning Structural Descriptions from Examples" (1970) was a major step towards a clarification of how concepts involving complex structural relationships might be learned.

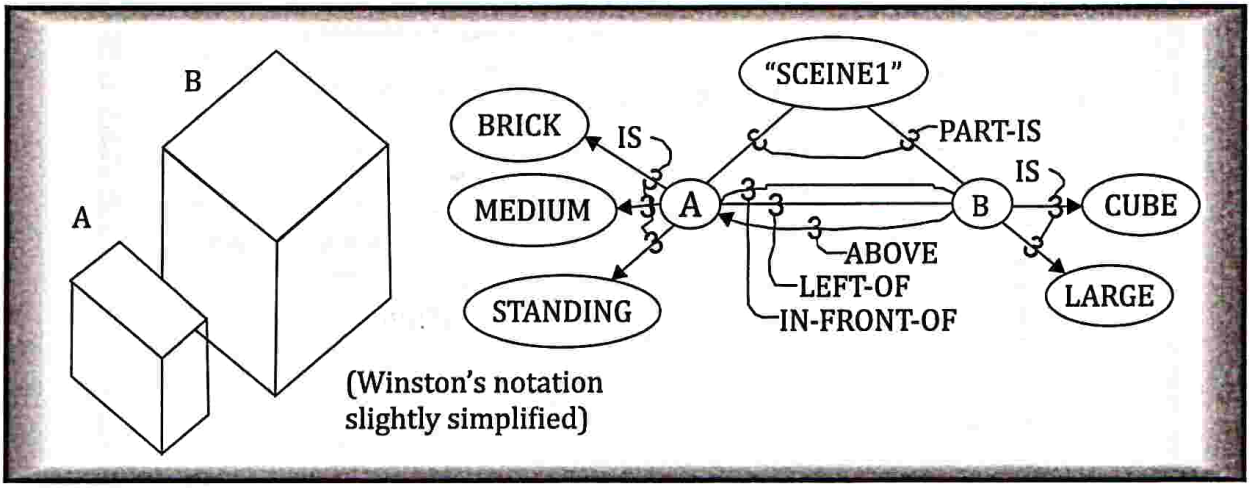
His program is presented with line drawings of scenes containing children's toy blocks, such as bricks, cubes, pyramids, and wedges. The program forms descriptive networks for these scenes, which shows the properties and relationships of the objects appearing in them. Using these structural descriptions, the program can learn structural concepts such as "pedestal", "arch" or "arcade" on the basis of examples and counterexamples of the concepts.

Winston's program uses Guzman's algorithm to determine the bodies in a scene; it then determines which edges belong to which object and fills in partially occluded edges. Then it infers the types of objects (brick, wedge, etc.) from the shapes and adjacency relationships of the viable faces. The sizes and orientation are then readily available.

The description in the following example is in the form of a "semantic network". The nodes are particular things (such as the object "ABC", and its faces "A", "B", and "C") or general concepts (such as BRICK, LARGE, etc.) and the edges are relations between things and/or general concepts (e.g., PART-IS is a relation which holds between a thing and its parts).

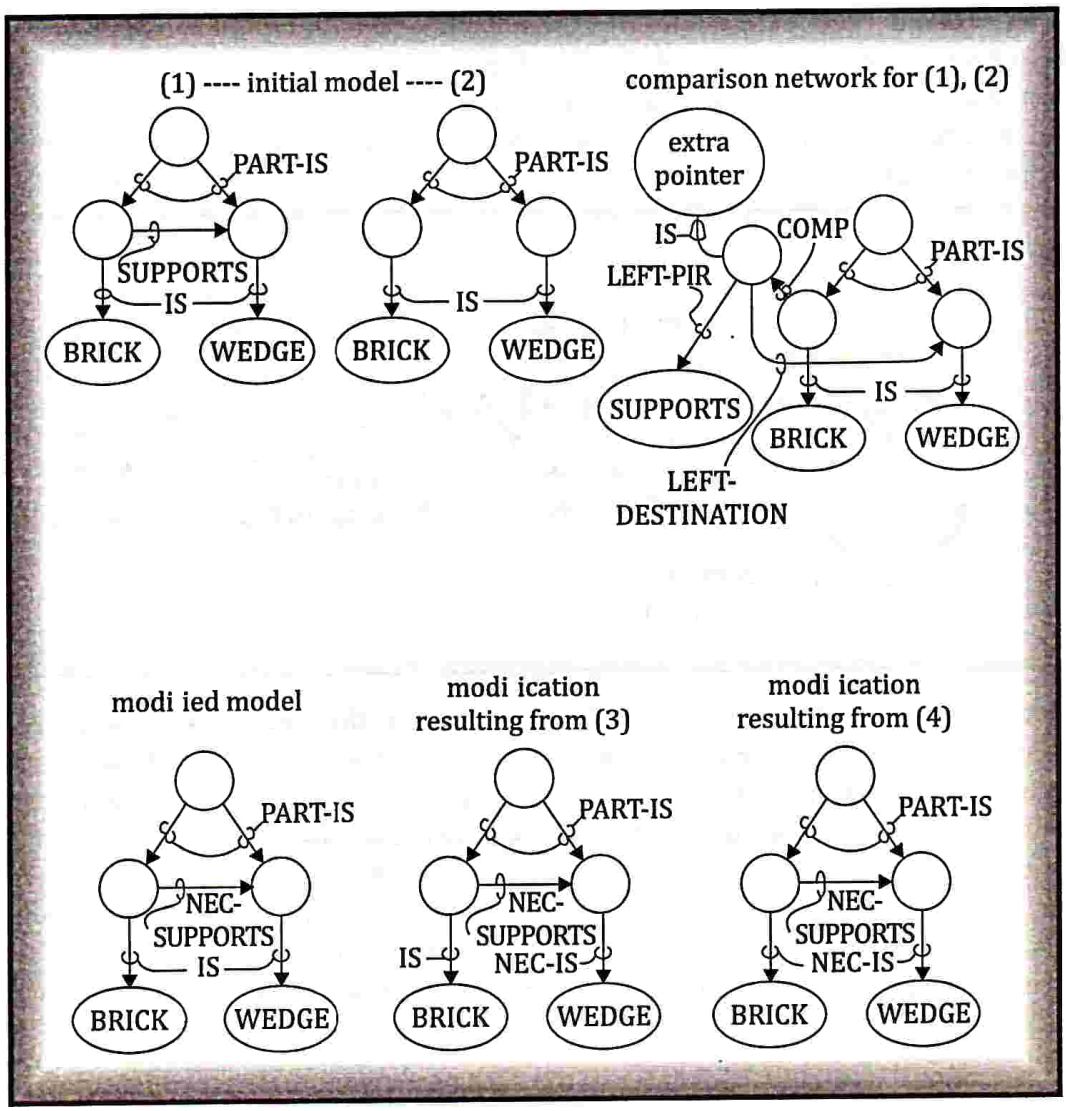


Next various heuristic routines are applied to the scene to obtain relationships between bodies, especially support relationships and also relative position (above, behind, to the left of). The information thus obtained is again represented in descriptive network form. The final scene description includes these relations as well as the overall attributes of the objects determined earlier (brick, wedge, etc.) but not the finer details such as component faces and their shape.



The initial model formed by the system is simply a description of the first true instance of the concept. The model is generalised for subsequent examples so that it will accept any new instances and reject non-instances (near misses). So that the system doesn't have too difficult a time determining which features of a non-instance disqualify it, the non-instances are required to be fairly close to true instances. Each modification of the model is made by generalising a comparison network for the current model and the given instance or non-instance. This comparison network describes the similarities and differences between the model and the new example. The descriptions of scenes (1) and (2), their comparison network, and subsequent modifications of the model are shown for the "house" sequence below.

Note the absence of the SUPPORTS pointer in the second network. Basically this is the difference described in the comparison network, which says that there is an extra pointer in the 'left' network, labelled SUPPORTS and with destination "the node for the wedge". The modified model says there is necessarily a support relation between the two parts of the scene. This modified model will reject (2) as an instance of a house. Similarly (3) causes a reinforcement of the "wedge" property of the supported object by the necessary operator, and (4) causes a reinforcement of the "brick" property of the supporting object. Just as certain features of the model can be reinforced through counterexamples, others can be relaxed when true instances are presented which differ from the model. For example, if a sample house were presented with a pyramidal rather than a "wedge" shaped roof, the NEC IS WEDGE requirement for the roof would be relaxed and replaced by the pair of alternatives MAY BE WEDGE and MAY BE PYRAMID. Now if the counterexample with a brick shaped roof were again presented, the requirement NEC-IS NOT-BRICK would be added to the existing alternatives.



The program can be faulted on the many ad-hoc decisions it embodies, both in analysing scenes and in the learning process. Also there are logical inadequacies in the network formalism (disjunction and quantification can not be represented). Nevertheless, the program incorporates theoretically interesting ideas (the network formalism and representation and the algorithm for comparison networks) and exhibits nontrivial learning behaviour.

Method for Determining Object Properties and Relations

- Object Properties
  - (a) edges belonging to an object
    1. - exterior edges belong to object if bounded by ob ject
    2. - at object-object boundaries, T verticies are used to assign edges to objects
    3. - bottom collinear line segments are connected
  - (b) shape of faces - triangle, quadrilateral, hexagon, octagon, etc.
  - (c) size of an object - tiny (.5% of visual area), small, medium (1.5-5% of visual area), large, huge (35-100% of visual area)

(d) orientation - standing, lying (applicable to bricks only)

- Object Relations

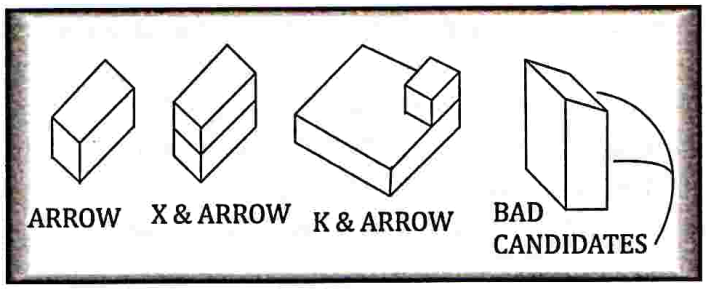
(a) left-of, right-of: A is to the left-of B is centre of area of A is left-of centre of area of B and the rightmost part of A is left-of the rightmost part of B.

(b) above, below (similar to (a))

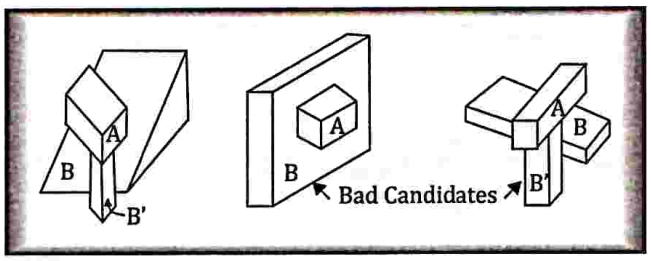
(c) supports, supported-by: first find base line of objects:

if the lower end and an interior line of an object lies at an exterior vertex, the exterior edges radiating from that vertex are candidate base lines.

Example



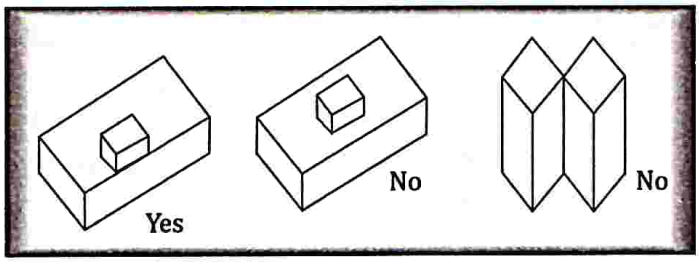
Eliminate both candidates if one candidate is more vertical than the interior line concerned. There are various refinements, e.g., extension of base lines through matched T's. Now if any object B is unbounded by a base line of an object A, B is a possible support of A. Examples



Eliminate "vertical slides" (using vertical edges in supposed supporting surface) and eliminate mismatched-height cases.

(d) in front-of, behind: an object bounded by a line belonging to another object and not a base line is behind that other object; there are many refinements.

(e) marries - common face common edge. Examples



X, K, and T verticies common to 2 or more objects are used to determine the "marries" property

5.2.4.2 Decision Trees

- ID3 and C4.5 are algorithms introduced by Quinlan for inducing Classification Models, also called **Decision Trees**, from data.
- We are given a set of records. Each record has the same structure, consisting of a number of attribute/value pairs. One of these attributes represents the category of the record. The problem is to determine a decision tree that on the basis of answers to questions about the non-category attributes predicts correctly the value of the category attribute. Usually the category attribute takes only the values {true, false}, or {success, failure}, or something equivalent. In any case, one of its values will mean failure.
- For example, we may have the results of measurements taken by experts on some widgets. For each widget we know what is the value for each measurement and what was decided, if to pass, scrap, or repair it. That is, we have a record with as non categorical attributes the measurements, and as categorical attribute the disposition for the widget.

Here is a more detailed example. We are dealing with records reporting on weather conditions for playing golf. The categorical attribute specifies whether or not to Play. The non-categorical attributes are:

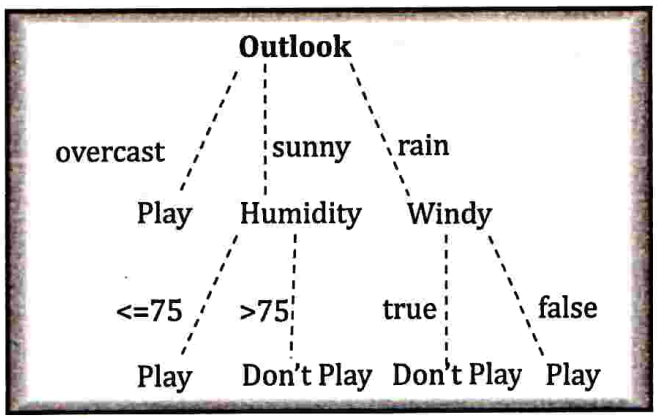
ATTRIBUTE	POSSIBLE VALUES
outlook	sunny, overcast, rain
temperature	continuous
humidity	continuous
windy	true, false

and the training data is:

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	fasle	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

Notice that in this example two of the attributes have continuous ranges, Temperature and Humidity. ID3 does not directly deal with such cases, though below we examine how it can be extended to do so. A decision tree is important not because it summarizes what we know, i.e. the training set, but because we hope it will classify correctly new cases. Thus when building classification models one should have both training data to build the model and test data to verify how well it actually works.

We obtain the following decision tree:



The ID3 Algorithm

The ID3 algorithm is used to build a decision tree, given a set of non-categorical attributes C1, C2, .., Cn, the categorical attribute C, and a training set T of records.

function ID3 (R: a set of non-categorical attributes,
C: the categorical attribute,
S: a training set) returns a decision tree;

begin

- If S is empty, return a single node with value Failure;
- If S consists of records all with the same value for the categorical attribute, return a single node with that value;
- If R is empty, then return a single node with as value the most frequent of the values of the categorical attribute that are found in records of S; [note that then there will be errors, that is, records that will be improperly classified];

Let D be the attribute with largest Gain(D,S) among attributes in R;

Let {dj| j=1,2, .., m} be the values of attribute D;

Let {Sj| j=1,2, .., m} be the subsets of S consisting respectively of records with value dj for attribute D;

Return a tree with root labeled D and arcs labeled d1, d2, .., dm going respectively to the trees ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm);

end ID3;

Summary

- Learning is an area of AI that focusses on processes of self-improvement.
- Rote learning, Learning by advice, Learning from Problem solving , Learning from examples are the different approaches of learning.
- ***Rote learning*** : The process of repeating something over and over engages the short-term memory and allows us to quickly remember basic things like facts, dates, names, multiplication tables, etc.
- ***Samuel's checker program*** employed rote learning. It uses ***scoring function*** for board positions reachable from the current state. The program remembered every position it had already seen, along with the terminal value of the reward function.
- ***Learning by advice*** : In this type of learning, a programmer writes a program to give some instructions to perform a task to the computer. Once it is learned (i.e. programmed), the system will be able to do new things. Also, there can be several sources for taking advice such as humans(experts), internet etc.
- FOO (First Operational Operationaliser), for example, is a learning system which is used to learn the game of Heart
- FOO had an initial knowledge base that was made up of:
  - basic domain concepts such as trick, hand, deck suits, avoid, win etc.
  - Rules and behavioural constraints -- general rules of the game.
  - Heuristics as to how to UNFOLD.
- When the program does not learn from advice, it can learn by generalizing from its own experiences.
  - Learning by parameter adjustment
  - Learning with macro-operators
  - Learning by chunking
  - The unity problem
- ***Learning by parameter adjustment*** : the learning system relies on evaluation procedure that combines information from several sources into a single summary static

Learning with Macro Operations

Sequence of actions that can be treated as a whole are called macro-operators. Once a problem is solved, the learning component takes the computed plan and stores it as a macro-operator. The preconditions are the initial conditions of the problem just solved, and its post conditions correspond to the goal just achieved.

- ***Learning by Chunking*** is similar to learning with macro-operators. Generally, it is used by problem solver systems that make use of production systems.
- ***The utility problem*** in learning systems occurs when knowledge learned in an attempt to improve a system's performance degrades it instead. The problem appears in many AI systems, but it is most familiar in speedup learning.

- ***Classification*** is a process of assigning to a particular input, to the name of the class to which it belongs to. Classification is important component in many problem solving tasks. But often classification is embedded inside another operation.
- ***Induction learning*** is carried out on the basis of supervised learning. In this learning process, a general rule is induced by the system from a set of observed instance. However, class definitions can be constructed with the help of a classification method.
- ***Winston's program*** uses Guzman's algorithm to determine the bodies in a scene; it then determines which edges belong to which object and fills in partially occluded edges. Then it infers the types of objects (brick, wedge, etc.) from the shapes and adjacency relationships of the viable faces. The sizes and orientation are then readily available.

5.3 Review Questions

Short Answer Questions

1. What is learning ?
2. List different types of learning.
3. Define Rote learning.
4. List any two advantages of rote learning.
5. List any two disadvantages of rote learning.
6. What is Chunking ?
7. Define Utility problem.
8. Define classification.
9. Define Induction learning.
10. Define Decision trees.

Long Answer Questions

1. Briefly explain Samuel's Checker program to illustrate the concept of rote learning.
2. Briefly explain Learning by taking advice with an example.
3. Briefly explain Learning by parameter adjustment with an example.
4. Briefly explain Learning with macro operations with an example.
5. Briefly explain Learning by chunking with an example.
6. Briefly explain Winston's Learning Program.
7. Write a note on Decision trees.

UNIT - III

CHAPTER 6

INTRODUCTION TO PLANNING

- What is Planning?
  - Types of Planning
- The Blocks World Problem
- Components of Planning System
- What is Uncertainty in AI?
  - Reasons for Uncertainty in Artificial Intelligence
- Nonmonotonic Logics
- Probabilistic Reasoning in AI - A way to deal with Uncertainty
  - What is Bayes Theorem in AI?
  - Challenges to probabilistic approaches:
- Fuzzy Logic
  - Architecture
  - Membership function :
  - Advantages of Fuzzy Logic System
  - Disadvantages of Fuzzy Logic Systems
  - Application
- Review Questions

6.1 What is Planning?

Planning in the context of artificial intelligence (AI) refers to the process of determining a sequence of actions or decisions to achieve a specific goal or solve a problem. It involves generating a plan, which is a structured representation of the intended course of action, considering the initial state, desired goal, available resources, and constraints:

1. **Initial State:**
   - The initial state defines the starting conditions or the current state of the world or system.
   - It includes information about the environment, objects, their properties, and their relationships.
   - The initial state serves as the foundation for the planning process, determining the context from which the plan will be developed.
2. **Goal:**
   - The goal specifies the desired state or the outcome that the planning system aims to achieve.
   - It represents the condition or set of conditions that the plan should satisfy or accomplish.
   - The goal provides the direction and purpose for the planning process, guiding the selection of actions and decisions.
3. **Actions:**
   - Actions represent the executable steps or operations that can be taken to modify the state of the system.
   - Each action typically has preconditions that must be satisfied for the action to be applicable or executable.
   - Actions also have effects, which describe the changes they induce on the state of the system once executed.
   - The set of available actions and their properties define the action space or the range of possible operations in the planning process.
4. **Transition Model:**
   - The transition model defines the dynamics of the system, specifying how the state changes when an action is applied.
   - It describes the relationship between the current state, the selected action, and the resulting new state.
   - The transition model is used to simulate the consequences of actions and predict the resulting state in the planning process.
5. **Search Algorithm:**
   - The search algorithm determines the strategy for exploring the space of possible plans or sequences of actions.
   - It systematically explores the search space to find a plan that satisfies the goal while considering constraints and optimizing certain criteria (e.g., plan length, cost).

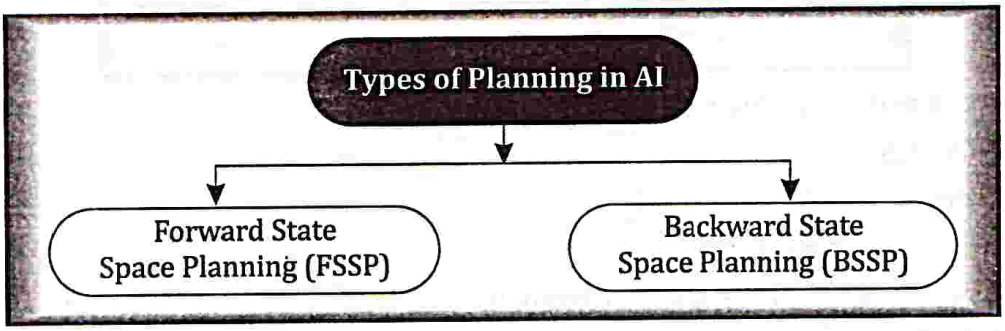
- Common search algorithms used in planning include depth-first search, breadth-first search, A* search, and heuristic search.

6. **Plan Representation:**
   - A plan is the output of the planning system and represents the sequence of actions to be executed.
   - Plans can be represented in various formats, such as action sequences, decision trees, or graphs.
   - The plan representation should be structured, unambiguous, and executable to guide the execution phase.
7. Plan Execution and Monitoring:
   - Once a plan is generated, it needs to be executed in the real or simulated environment.
   - During execution, the actual state of the system is monitored and compared to the expected states described in the plan.
   - Deviations or unexpected events may require plan adaptation or replanning to handle dynamic environments.

Planning systems find applications in diverse domains, including robotics, logistics, scheduling, resource allocation, and autonomous systems. They enable AI systems to autonomously reason about actions, make decisions, and generate effective plans to achieve desired goals in complex and uncertain environments.

6.1.1 Types of Planning

we have Forward State Space Planning (FSSP) and Backward State Space Planning (BSSP) at the basic level.



- **Forward State Space Planning (FSSP)**

  FSSP behaves in the same way as forwarding state-space search. It says that given an initial state S in any domain, we perform some necessary actions and obtain a new state S' (which also contains some new terms), called a progression. It continues until we reach the target position. Action should be taken in this matter.

  Disadvantage: Large branching factor

  Advantage: The algorithm is Sound

- **Backward State Space Planning (BSSP)**

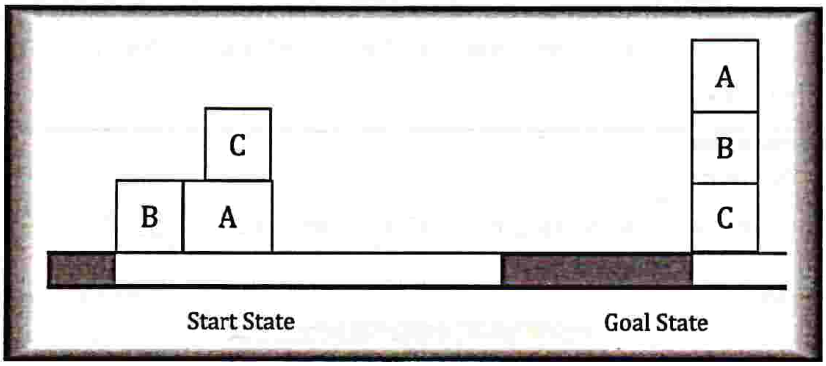
  BSSP behaves in a similar fashion like backward state space search. In this, we move from the goal state g towards sub-goal g' that is finding the previous action to be done to achieve that respective goal. This process is called regression (moving back to the previous goal or sub-goal). These sub-goals have to be checked for consistency as well. The actions have to be relevant in this case.

  Disadvantage: Not a sound algorithm (sometimes inconsistency can be found)

  Advantage: Small branching factor (very small compared to FSSP)

6.2 The Blocks World Problem

One of the most famous planning domains is known as the blocks world. This domain consists of a set of cube-shaped blocks sitting on a table.The blocks can be stacked, but only one block can fit directly on top of another. A robot arm can pick up a block and move it to another position, either on the table or on top of another block. The arm can pick up only one block at a time, so it cannot pick up a block that has another one on it The goal will always be to build one or more stacks of blocks, specified in terms of what blocks are on top of what other blocks. For example, a. goal might be to get block A on B and block B on C.



The actions it can perform include

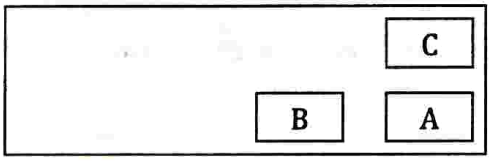
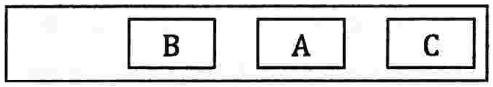
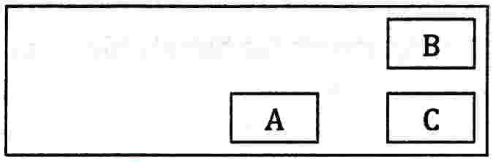
- UNSTACK (A,B)
  - ➢ remove block A from block B
  - ➢ the arm must be empty
  - ➢ block A must have no blocks on top of it.
- **STACK (A,B)**
  - ➢ put block A from block B
  - ➢ the arm must already be holding A
  - ➢ the surface of B must be clear
- PICKUP(A)
  - ➢ pick up block A from the table
  - ➢ the arm must be empty and there must be nothing on top of A

- PUTDOWN(A)
  - ➢ put block A on the table.
  - ➢ the arm must have holding block A

Predicates can be used :	Ontable(A)	Block A is on the table
	On(A,B)	Block A is on Block B
	Clear(A)	nothing stands on A
	Armempty	robot hand is empty
	Holding(A)	robot hand holds A

Logical notations can be used :

1. [∃x : HOLDING(x)] → ARMEMPTY
   If the arm is holding anything, then it is not empty.
2. ∀x : ONTABLE(x) -→ ∃y : ON(x,y)
   If a block is on the table, then it is not also on another block.
3. ∀x : [∃y : ON(y,x)] → CLEAR(x)
   Any block with no blocks on it is clear

Init (Ontable (A) ∧ Ontable (B) ∧ On(C,A) **Goal** (On(A,B) ∧ On(B , C))	
Action (a) Move block C on the Table • Holding(C) • Ontable (C) • Arm empty • Clear (A)	
(b) Move block B on block C • Ontable (B) • Holding (B) • On (B,C) • Armempty	
(c) Move block A on block B • Ontable (A) • Holding (A) • On (A, B) • Armempty	

6.3 Components of Planning System

In problem solving systems, it is necessary to perform each of the following functions :

- Choose the best rule for applying the next rule based on the best available heuristics.
- Apply the chosen rule for computing the new problem state.
- Detect when a solution has been found.
- Detect dead ends so that they can be abandoned and the system's effort is directed in more fruitful directions.
- Detect when an almost correct solution has been found.

Choose the best rule for applying the next rule based on the best available heuristics.

- Finding the difference between the current and goal states.
- Choose the best rules that reduce these differences most effectively.
  - ➢ Example – Means – End analysis
    - ♦ Wishing to travel by car to visit a friend
      - ✈ First -> fill up the car with fuel.
      - ✈ If no car -> acquire one.
    - ♦ Largest difference must be tackled first.

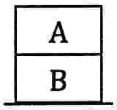
Apply the chosen rule for computing the new problem state.

- In simple systems, applying rules is easy. Each rule simply specified the problem state that would result from its applications.
- In complex systems, we must be able to deal with rules that specify only a small part of the complete problem state.
- One way is to describe, for each action, each of the changes it makes to the state description.
- A number of approaches have been used for this task.

Green's Approach

- Basically, a given state was described by a set of predicates representing the facts that were true in that state. Each state was represented explicitly as part of the predicate.

  Green' Approach for Simple block world description



$ON(A, B, S_0) \wedge ON\ TABLE(B, S_0) \wedge CLEAR\ (A, S_0)$

This figure show the state called S_0 of a simple blocks world problems could be represented.

- If wanted to **UNSTACK(A,B)** , the operation is expressed as :

$$[CLEAR(x, S) \wedge ON(x, y, S)] \rightarrow [HOLDING(x, Do(UNSTACK(x, y), S)) \wedge CLEAR(y, Do(UNSTACK(x, y), S))]$$

- x , y – any blocks
- S - any state
- Do() – specifies that a new state result from the given action.

- The result of applying this to state S_0 to give S_1 by applying UNSTACK is

HOLDING(A, S1) ^ CLEAR(B, S1)

- One problem with this approach is, B is still on the table.
- This needs to be encoded into frame axioms that describe components of the state not affected by the operator.
- So, it must be,

ONTABLE(Z, S) → ONTABLE(Z, Do(UNSTACK(x, y), s))

- If want to color the blocks, an axiom used is,

COLOR(x, c, s) → COLOR(x, c, Do(UNSTACK(y, z), s))

STRIPS Approach

STRIPS → Stanford Research Institute Planning System

- A mechanism that doesn't require a large number of explicit frame axiom
- Basically each operator has 3 lists of predicates associated with it.
- Each operation is described by a,
  - List of new predicates that become **TRUE** called **ADD**
  - List of old predicates that become **FALSE** called **DELETE**
  - List contains those predicates that must be true for the operator to be applied called as **PRECONDITION**
  - Anything not in these lists is assumed to be unaffected by the operation.

STRIPS style operators that correspondence to the block world operations are :

STACK(x,y)

P : CLEAR(Y) ∧ HOLDING(X) (precondition)

D : CLEAR(Y) ∧ HOLDING(X) (delete)

A : ARMEMPTY ∧ ON(x,y) (add)

UNSTACK(x,y)

P : ON(x,y) ∧ CLEAR(x) ∧ ARMEMPTY (precondition)

D : ON(x,y) ∧ ARMEMPTY (delete)

A : HOLDING(x) ∧ CLEAR(Y) (add)

PICKUP(x)

P : CLEAR(x) ∧ ONTABLE(x) ∧ ARMEMPTY (precondition)

D : ONTABLE(x) ∧ ARMEMPTY (delete)

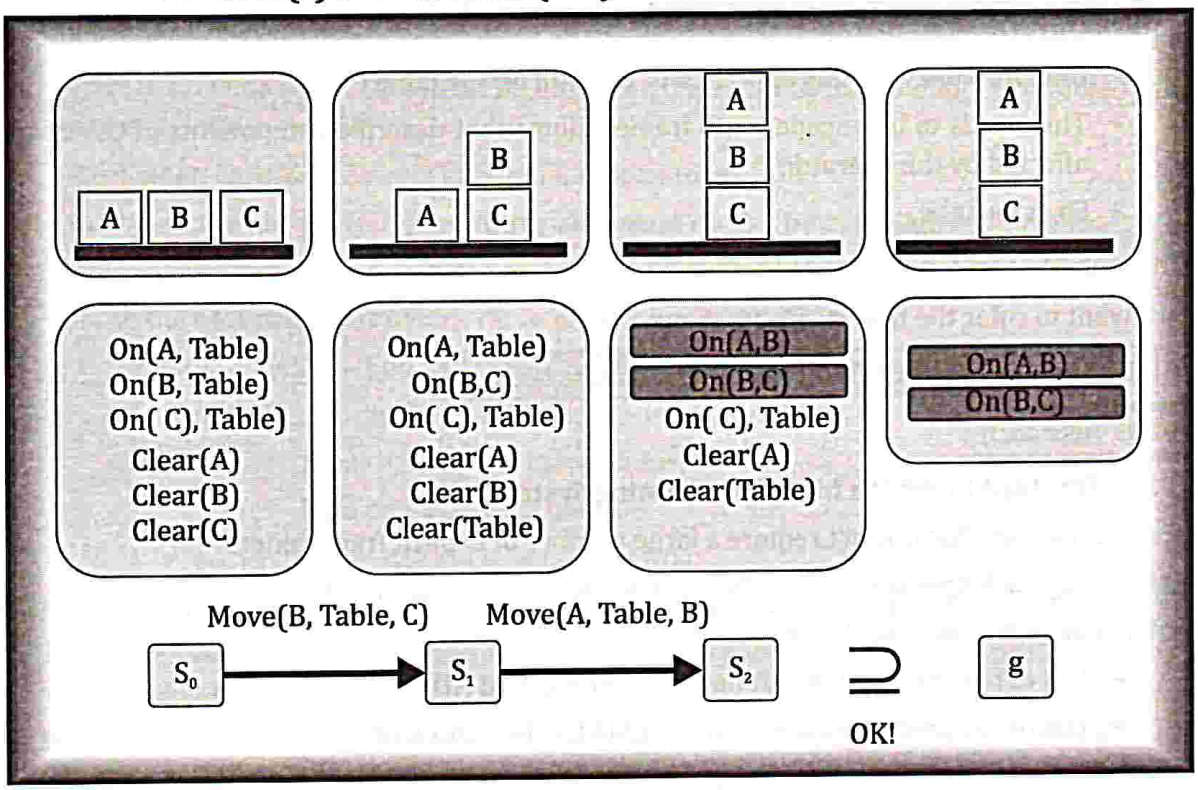
A : HOLDING(x) (add)

PUTDOWN(x)

P : HOLDING(x) (precondition)

D : HOLDING(x) (delete)

A : ONTABLE(x) ∧ ARMEMPTY (add)



Detect when a solution has been found.

- A planning system will succeed in finding a solution to a problem only when it has found a sequence of operators that transforms the initial problem state into the goal state. But how could a system will know it has found the solution. This could be done by comparing the problem`s goal state with the current state if it matches it we could straightaway say that a solution state has been reached. But on the other hand if the entire states are not represented explicitly but rather described in term of some properties then detecting a solution becomes more complex. This could be solved depending on the way the state descriptions are represented.
- For any representational scheme that is used it must be possible to reason with representation to discover whether one matches the other or not. Then the corresponding reasoning mechanism could be used to discover when a solution has been found.
- Predicate Logic serves as the basis for many of the planning system as a representation technique. Due to its deductive mechanism the researchers often find it more appealing.

Detecting Dead Ends :

- A Planning system might sometime could land into dead-ends when searching for a sequence of operators to solve a particular problem. The same reasoning mechanism that is used for reasoning can also be used for detecting the dead ends.

- If the search process is reasoning forward from the initial state, it can prune any paths that leads to a state from which the goal state cannot be reached.
- If the search process is reasoning backwards from the goal state it can also terminate a path either because it is sure that the initial state cannot be reached or because little progress is being made.
- In reasoning backward each goals is decomposed into subgoals where each of them in turn may lead to a set of additional subgoals. Sometime it may be easy to identify that there is no way that all the sub-goals in a given set can be satisfied at once. For example the robot arm cannot be both empty and holding a block. If any path that is attempting to make both of those goals true can be simultaneously pruned immediately.

Detect when an almost correct solution has been found

- While solving a nearly decomposable problem the sub solutions when combined may not yield a complete solution . So when such situation arises there are certain approaches to handle it. The simplest is just to throw out the solution, look for another one, and hope that is is better. This approach looks simple but it may lead to a great deal of wasted effort.
- A slightly better approach is to compared the desired solution and derived solution and if there is a difference then the problem solving system can be called again and asked to find a way of eliminating this new difference. The first solution could be combined with the second one to form a solution to the original problem.
- An even better way to patch up an almost correct solution is to appeal to specific knowledge about what went wrong and then apply a direct patch.
- A still better way to patch up incomplete solutions is not really to patch them up at but rather to leave them incompletely specified until the last possible moment. Then when as much information as possible is available, complete the specification in such a way that no conflict arise. This approach is called as Least Commitment Strategy.

6.4 What is Uncertainty in AI?

- When we talk about perceiving information from the environment, then the main problem that arises is that there is always some uncertainty is our observations. This is because the world is an enormous entity and the surroundings that we take under study is not always well defined. So, there needs to be an estimation taken for getting to any conclusion.
- Human being face this uncertainty daily that too many times. But still, they manage to take successful decisions. This is because humans have strong estimating and decision making power and their brains function in such a way that every time such a situation arises, the alternative with the maximum positive output is chosen. But, the artificial agents are not able to take proper decisions while working in such an environment. This is because, if the information available to them is not accurate, then they cannot choose the right decision from their knowledge base.

Uncertainty Example

Taking a real life example, while buying vegetables, humans can easily distinguish between the different kinds of vegetables by their color, sizes, textures, etc. But there is uncertainty in making the right choices here, because the vegetables may not be exactly the same as described. Some of them may be distorted, some may vary than the usual size and there can be many such variations. But in spite of all of them, humans do not face any problem in situations like these.

But, this same thing becomes a hurdle when the decision is to be made by a computer based agent. Because, we can feed the agent by the information that how the vegetables look, but still we cannot accurately define the exact shape and size of each of them, because all of them have some variations.

So, as a solution to this, only the basic information is provided to the agent. Based on this very information, the agent has to make certain estimates to find out which vegetable is kept in front of it.

Not only in this agent, but in designing almost every AI based agent, this strategy is followed. So, there should be a proper method so that the agent can make certain estimations by itself without any help or input from human beings.

6.4.1 Reasons for Uncertainty in Artificial Intelligence

The following are the reasons for uncertainty in Artificial Intelligence:

1. **Partially Observable Environment :** The entire environment is not always in reach of the agent. There are some parts of the environment which are out of the reach of the agent and hence they are left unobserved. So, the decisions that the agent makes do not include the information from these areas and hence, the result drawn may vary form the actual case.
2. **Dynamic Environment :** As we all know that the environment is dynamic, i.e. there are always some changes that keep taking place in the environment. So, the decision or calculations made at any instant may not be the same after some time due to the changes that have occurred in the surroundings by that time. So, if the observations made at any instance are considered later, then there can be an ambiguity in the decision making.
3. **Incomplete Knowledge of the Agent :** If the agent has incomplete knowledge or insufficient knowledge about anything, then it cannot produce correct results because the agent itself does not know about the situation and the way in which the situation is to be handled.
4. **Inaccessible Areas in the Environment :** There are areas in the environment which are observable, but not in reach of the agent to access. In such situations. The observation made is correct, but the as an agent cannot act on these parts of the environment, these parts will remain unchanged by the actions of the agent. This will not affect the current decision but can affect the estimations made by the agent in the future.

6.5 Nonmonotonic Logics

- A reasoning system is ***monotonic*** if the truthfulness of a conclusion does not change when new information is added to the system — the set of theorem can only monotonically grows when new axioms are added. In contrast, in a system doing ***non-monotonic*** reasoning the set of conclusions may either grow or shrink when new information is obtained.

- Nonmonotonic logics are used to formalize plausible reasoning, such as the following inference step:

  Birds typically fly.

  Tweety is a bird.

  Tweety (presumably) flies.

  Such reasoning is characteristic of commonsense reasoning, where default rules are applied when case-specific information is not available.

  The conclusion of nonmonotonic argument may turn out to be wrong. For example, if Tweety is a penguin, it is incorrect to conclude that Tweety flies. Nonmonotonic reasoning often requires **jumping to a conclusion and subsequently retracting that conclusion** as further information becomes available.

- All systems of nonmonotonic reasoning are concerned with the issue of consistency. Inconsistency is resolved by removing the relevant conclusion(s) derived previously by default rules. Simply speaking, the truth value of propositions in a nonmonotonic logic can be classified into the following types:
  1. facts that are definitely true, such as "Tweety is a bird"
  2. default rules that are normally true, such as "Birds fly"
  3. tentative conclusions that are presumably true, such as "Tweety flies"

  When an inconsistency is recognized, only the truth value of the last type is changed.

  A related issue is belief revision. Revising a knowledge base often follows the principle of minimal change: one conserves as much information as possible.

- One approach towards this problem is truth maintenance system, in which a "justification" for each proposition is stored, so that when some propositions are rejected, some others may need to be removed, too.

  Major problems in these approaches:

  - conflicts in defaults
  - computational expense: to maintain the consistency in a huge knowledge base is hard, if not impossible

Advantages of Non-monotonic Reasoning
1. For real-world systems such as Robot navigation, we can use non-monotonic reasoning.
2. In Non-monotonic reasoning, we can choose probabilistic facts or can make assumptions.

Disadvantages of of Non-monotonic Reasoning
1. In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.
2. It cannot be used for theorem proving.

6.6 Probabilistic Reasoning in AI - A way to deal with Uncertainty

- we know that there are many cases where the answer to the problem is neither completely true nor completely false. For example, the statement- "Student will pass in the board exams". We cannot say anything about a student's result before the results are declared. However, we can draw some predictions based on the student's past performances in academics.
- In these types of situations, probabilistic theory can help us give an estimate of how much an event is likely to occur or happen? In this theory, we find the probabilities of all the alternatives that are possible in any experiment. The sum of all these probabilities for an experiment is always because all these events/alternatives can happen only within this experiment.
- Basic idea: to use probability theory to represent and process uncertainty. In probabilistic reasoning, the truth value of a proposition is extended from {0, 1} to [0, 1], with binary logic as its special case.
- Justification: though no conclusion is absolutely true, the one with the highest probability is preferred. Under certain assumptions, probability theory gives the optimum solutions.

  To extend the basic Boolean connectives to probability functions:

  - negation: $P(\neg A) = 1 - P(A)$
  - conjunction: $P(A \wedge B) = P(A) * P(B)$ if A and B are independent of each other
  - disjunction: $P(A \vee B) = P(A) + P(B)$ if A and B never happen at the same time

  Furthermore, the conditional probability of B given A is $P(B|A) = P(B \wedge A) / P(A)$, from which Bayes' Theorem is derived, and it is often used to update a system's belief according to new information: $P(H|E) = P(E|H) * P(H) / P(E)$.

6.6.1 What is Bayes Theorem in AI?

Bayes theorem is a method to find the probability of an event whose occurrence is dependent on some other event's occurrence. In simple words, using the Bayes theorem, we can find the conditional probability of any event. Bayes theorem, given by Reverend Thomas Bayes and thus named after him

Bayes Theorem Mathematical Equation

The Bayes Theorem, also known as Bayes law or Bayes equation is a mathematical equation which is given as follows:

$$P(B) = \frac{P(B|A)\,P(A)}{P(B)}$$ Where, A and B are events and $P(B) \neq 0$.

Here,

- P(A|B): Conditional probability of occurrence of event A when event B has already occurred.
- P(B|A): Conditional probability of occurrence of event B when event A has already occurred.
- P(A): Probability of occurrence of event A alone without any dependence on other events.
- P(B): Probability of occurrence of event B alone without any dependence on other events.

Derivation of Bayes Theorem

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

Similarly

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

Putting the value of P (A^B) in equation (1), we get

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

Which is our required Bayes equation.

It should be noted that in the Bayesian equation, we need not find the probability of both the events occurring simultaneously, i.e. P(A^B). We can simply calculate the conditional probability of an event if we know the conditional probability of the event on which it is dependent and the individual probabilities of both the events without any dependency on each other.

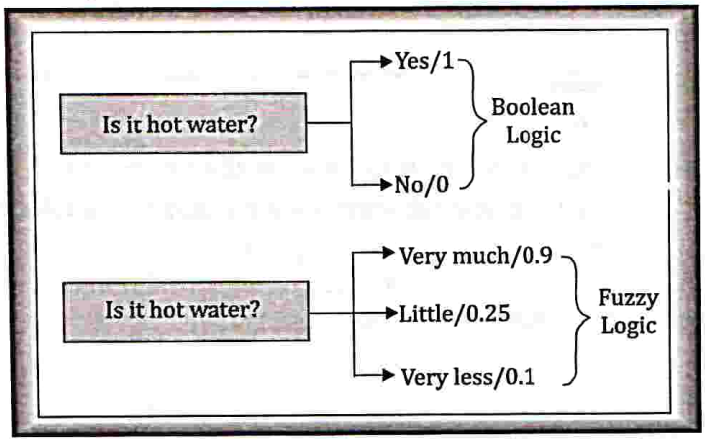
Bayes Theorem is applicable only in those experiments where we have only two events. It is not applicable to the cases where the number of events is more than two.

6.6.2 Challenges to probabilistic approaches:

- unknown probability values
- inconsistent probability assignments
- computational expense

6.7 Fuzzy Logic

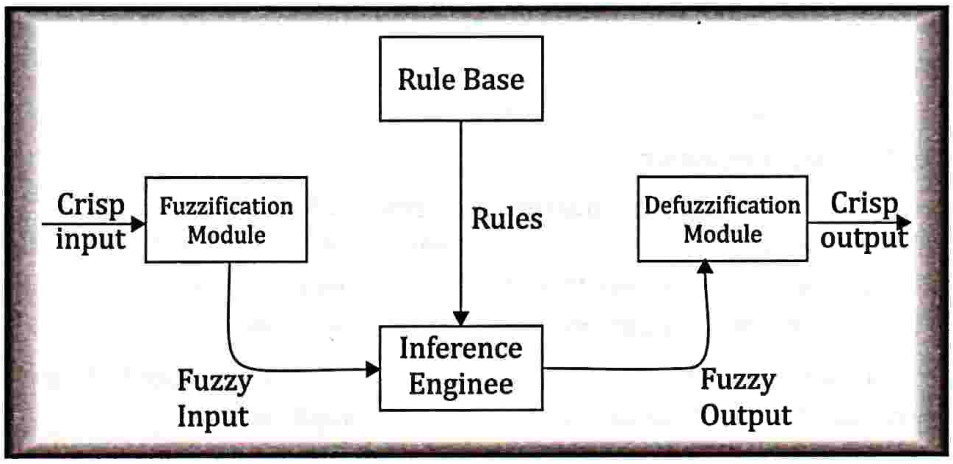
- Fuzzy logic is a generalization of classical logic, and reflects the impression of human language and reasoning. Examples of fuzzy concepts: "young", "furniture", "most", "cloudy", and so on.
- According to fuzzy logic, whether an instance belongs to a concept is usually not a matter of "yes/no", but a matter of degree. Fuzzy logic uses a degree of membership, which is a real number in [0, 1]. Example :



- In summary, Fuzzy Logic is a mathematical method for representing vagueness and uncertainty in decision-making, it allows for partial truths, and it is used in a wide range of applications. It is based on the concept of membership function and the implementation is done using Fuzzy rules.

6.7.1 Architecture

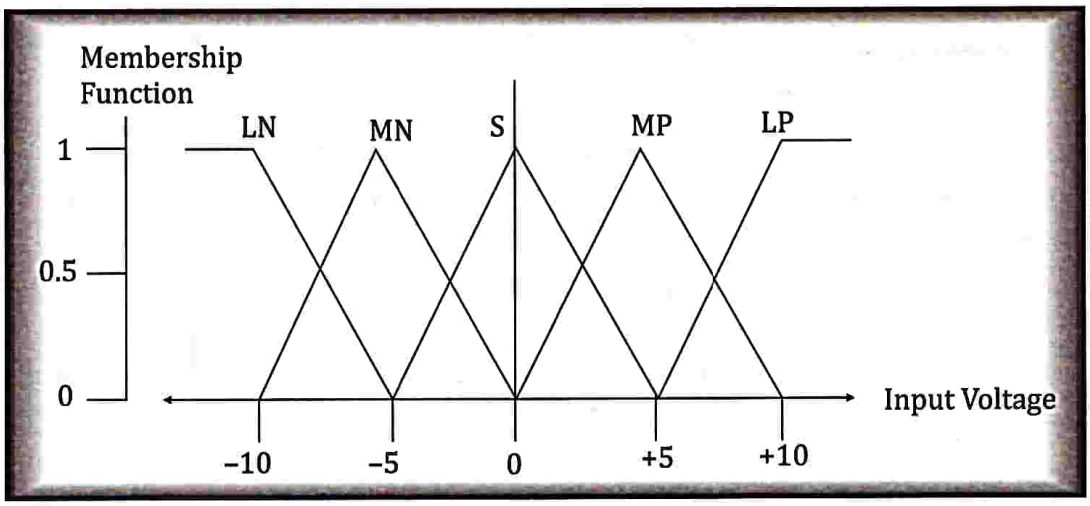
Its Architecture contains four parts :



- **RULE BASE:** It contains the set of rules and the IF-THEN conditions provided by the experts to govern the decision-making system, on the basis of linguistic information. Recent developments in fuzzy theory offer several effective methods for the design and tuning of fuzzy controllers. Most of these developments reduce the number of fuzzy rules.
- **FUZZIFICATION:** It is used to convert inputs i.e. crisp numbers into fuzzy sets. Crisp inputs are basically the exact inputs measured by sensors and passed into the control system for processing, such as temperature, pressure, rpm's, etc.
- **INFERENCE ENGINE:** It determines the matching degree of the current fuzzy input with respect to each rule and decides which rules are to be fired according to the input field. Next, the fired rules are combined to form the control actions.
- **DEFUZZIFICATION:** It is used to convert the fuzzy sets obtained by the inference engine into a crisp value. There are several defuzzification methods available and the best-suited one is used with a specific expert system to reduce the error.

6.7.2 Membership function :

The membership function is a function which represents the graph of fuzzy sets, and allows users to quantify the linguistic term. It is a graph which is used for mapping each element of x to the value between 0 and 1. There can be multiple membership functions applicable to fuzzify a numerical value. Simple membership functions are used as the complex functions do not add precision in the output. The membership functions for LP, MP, S, MN, and LN are:



The triangular membership function shapes are most common among various other membership function shapes. Here, the input to 5-level fuzzifier varies from -10 volts to +10 volts. Hence the corresponding output also changes.

6.7.3 Advantages of Fuzzy Logic System

Advantages of Fuzzy Logic System

1. This system can work with any type of inputs whether it is imprecise, distorted or noisy input information.
2. The construction of Fuzzy Logic Systems is easy and understandable.
3. Fuzzy logic comes with mathematical concepts of set theory and the reasoning of that is quite simple.
4. It provides a very efficient solution to complex problems in all fields of life as it resembles human reasoning and decision-making.
5. The algorithms can be described with little data, so little memory is required.

6.7.4 Disadvantages of Fuzzy Logic Systems

Disadvantages of Fuzzy Logic Systems

1. Many researchers proposed different ways to solve a given problem through fuzzy logic which leads to ambiguity. There is no systematic approach to solve a given problem through fuzzy logic.
2. Proof of its characteristics is difficult or impossible in most cases because every time we do not get a mathematical description of our approach.
3. As fuzzy logic works on precise as well as imprecise data so most of the time accuracy is compromised.

6.7.5 Application

- It is used in the aerospace field for altitude control of spacecraft and satellites.
- It has been used in the automotive system for speed control, traffic control.
- It is used for decision-making support systems and personal evaluation in the large company business.
- It has application in the chemical industry for controlling the pH, drying, chemical distillation process.
- Fuzzy logic is used in Natural language processing and various intensive applications in Artificial Intelligence.
- Fuzzy logic is extensively used in modern control systems such as expert systems.
- Fuzzy Logic is used with Neural Networks as it mimics how a person would make decisions, only much faster. It is done by Aggregation of data and changing it into more meaningful data by forming partial truths as Fuzzy sets.

Summary

- **Planning** in the context of artificial intelligence (AI) refers to the process of determining a sequence of actions or decisions to achieve a specific goal or solve a problem.
- Different **stages of planning** are : Initial State, Goal, Actions, Transition Model, Search Algorithm, Plan Representation & Plan Execution and Monitoring.
- Types of planning : **Forward State Space Planning (FSSP) & Backward State Space Planning (BSSP)**
- **Forward State Space Planning (FSSP) :** FSSP behaves in the same way as forwarding state-space search. It says that given an initial state S in any domain, we perform some necessary actions and obtain a new state S' (which also contains some new terms), called a progression
- **Backward State Space Planning (BSSP) :** BSSP behaves in a similar fashion like backward state space search. In this, we move from the goal state g towards sub-goal g' that is finding the previous action to be done to achieve that respective goal. This process is called regression
- One of the most famous planning domains is known as the **blocks world**. This domain consists of a set of cube-shaped blocks sitting on a table. The blocks can be stacked, but only one block can fit directly on top of another. A robot arm can pick up a block and move it to another position, either on the table or on top of another block. The arm can pick up only one block at a time, so it cannot pick up a block that has another one on it The goal will always be to build one or more stacks of blocks, specified in terms of what blocks are on top of what other blocks. For example, a. goal might be to get block A on B and block B on C.
- **Components of planning system:**
  - Choose the best rule for applying the next rule based on the best available heuristics.
  - Apply the chosen rule for computing the new problem state.
  - Detect when a solution has been found.

- Detect dead ends so that they can be abandoned and the system's effort is directed in more fruitful directions.
- Detect when an almost correct solution has been found.

- Two important approaches of problem solving are : Green's Approach & STRIPS Approach.
- **Green's Approach :**Basically, a given state was described by a set of predicates representing the facts that were true in that state. Each state was represented explicitly as part of the predicate.
- **STRIPS** is a planning algorithm that was developed by Stanford AI Lab in the early 1970s. STRIPS is an acronym for "STanford Research Institute Planning System". The STRIPS algorithm works by breaking down a planning problem into a series of smaller sub-problems, each of which can be solved independently. The algorithm then combines the solutions to the sub-problems to find a solution to the overall problem.
- **Uncertainty in AI :** When we talk about perceiving information from the environment, then the main problem that arises is that there is always some uncertainty is our observations. This is because the world is an enormous entity and the surroundings that we take under study is not always well defined. So, there needs to be an estimation taken for getting to any conclusion.
- Reasons for uncertainty in AI : Partially observable environment, Dynamic Environment. Incomplete knowledge of the agent, Inaccessible areas in the environment.
- A reasoning system is ***monotonic*** if the truthfulness of a conclusion does not change when new information is added to the system — the set of theorem can only monotonically grows when new axioms are added. In contrast, in a system doing ***non-monotonic*** reasoning the set of conclusions may either grow or shrink when new information is obtained.
- **Probabilistic Reasoning in AI -** In probabilistic reasoning, probabilistic theory can help us give an estimate of how much an event is likely to occur or happen? In this theory, we find the probabilities of all the alternatives that are possible in any experiment. The sum of all these probabilities for an experiment is always because all these events/alternatives can happen only within this experiment.
- The truth value of a proposition is extended from {0, 1} to [0, 1], with binary logic as its special case. ,
- **Bayes theorem** is a method to find the probability of an event whose occurrence is dependent on some other event's occurrence. In simple words, using the Bayes theorem, we can find the conditional probability of any event. Bayes theorem, given by Reverend Thomas Bayes and thus named after him
- **Fuzzy Logic** is a mathematical method for representing vagueness and uncertainty in decision-making, it allows for partial truths, and it is used in a wide range of applications. It is based on the concept of membership function and the implementation is done using Fuzzy rules.
- The **membership function** is a function which represents the graph of fuzzy sets, and allows users to quantify the linguistic term. It is a graph which is used for mapping each element of x to the value between 0 and 1.

6.8 Review Questions

Short Answer Questions

1. Define planning in context of AI.
2. List the stages of Planning in AI.
3. Briefly define two types of Planning in AI.
4. List the components of Planning system in AI.
5. Define STRIDE approach of Planning system in AI.
6. What is Uncertainty in AI?
7. List any 4 reasons for Uncertainty in AI.
8. Define Nonmonotonic logics.
9. List any two advantages and disadvantages of Nonmonotonic logics.
10. Justify- "Probabilistic Reasoning in AI - A way to deal with Uncertainty"
11. Define Bayes Theorem in AI.
12. List any two challenges to probabilistic approaches.
13. Define Fuzzy logic.
14. Define member functions.
15. List any two advantages of Fuzzy logic.
16. List any two disadvantages of Fuzzy logic.
17. Name any 4 applications of Fuzzy logic.

Long Answer Questions

1. Explain different stages of planning in AI. Differentiate between FSSP & BSSP.
2. With a neat diagram, Explain The block world problem.
3. Define Green's Approach. Explain Green's Approach for Simple block world problem.
4. Define STRIPS approach. Explain STRIPS style operators that correspondence to the block world operations.
5. With an example explain Uncertainty in Artificial Intelligence. Explain any four reasons.
6. Write a note on Nonmonotonic logics with advantages & disadvantages.
7. What is Bayes Theorem in AI? Explain.
8. Define Fuzzy logic. Explain its architecture.

UNIT - III

CHAPTER 7

ROBOTICS

- Robot Defined
  - Types of Robots
  - Components of Robots
  - Laws of Robotics
  - Applications of Robotics
  - Advantages
  - Disadvantages
  - Robot Kinematics
- Computer Vision
  - What is Computer Vision?
  - Computer Vision Techniques
- Review Questions

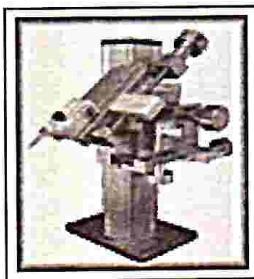
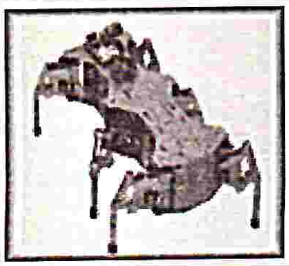
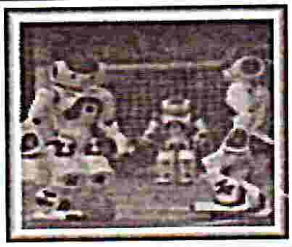
7.1 Robot Defined

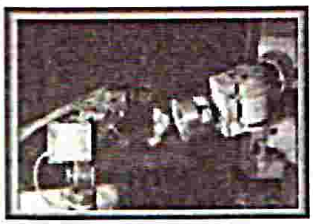
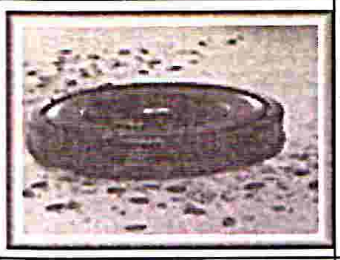
- Word robot was coined by a Czech novelist Karel Capek in 1920 play titled Rassum's Universal Robot (RUR). Robot in Czech means worker or servant.
- According to Robot Institute of America, 1979 - Robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks:

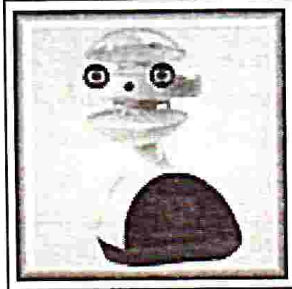
Definition : Robots

Robotics is a sub-domain of engineering and science that includes mechanical engineering, electrical engineering, computer science, and others. This branch deals with the design, construction, use to control robots, sensory feedback and information processing.

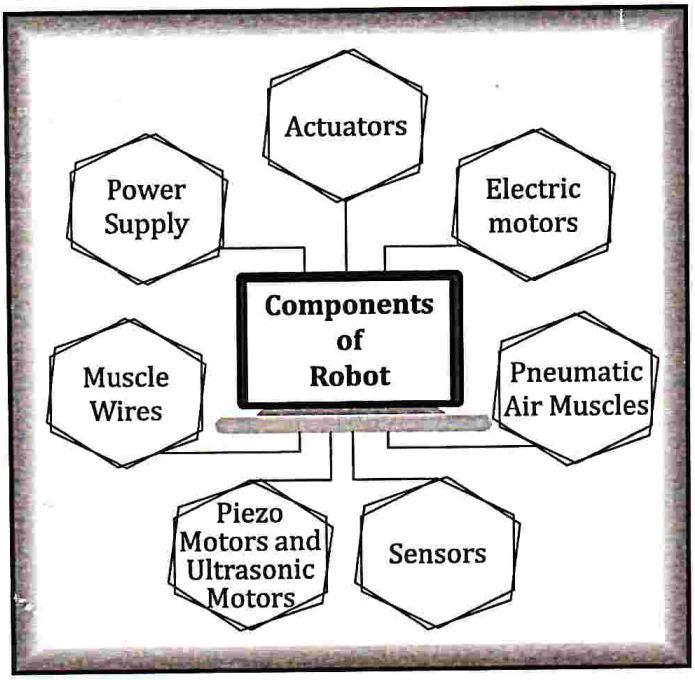
7.1.1 Types of Robots

	**Manipulator :** In robotics, a manipulator is a device used to manipulate materials without direct physical contact by the operator. The applications were originally for dealing with radioactive or biohazardous materials, using robotic arms, or they were used in inaccessible places. In more recent developments they have been used in diverse range of applications including welding automation, robotic surgery and in space.
	**Legged Robots :** Legged robots are a type of mobile robot which use articulated limbs, such as leg mechanisms, to provide locomotion. They are more versatile than wheeled robots and can traverse many different terrains, though these advantages require increased complexity and power consumption. Legged robots often imitate legged animals, such as humans or insects, in an example of biomimicry
	**Wheeled Robots :** Wheeled robots can change their positions with the help of their wheels, Wheeled robot motion can be achieved easily & its cost is pretty low, control of wheeled movement is easier, So, wheeled robots are one of the most frequently seen robots, Single wheeled robots, mobile ball robots, two-wheeled robots, three-wheeled robots, four-wheeled robots & multi-wheeled robots are examples of wheeled robots.
	**Autonomous Robots :** Autonomous robots are self-supported. They use a program that provides them the opportunity to decide the action to perform depending on their surroundings. Using artificial intelligence these robots often learn new behavior. hey start with a short routine and adapt this routine to be more successful in a task they perform. Hence, the most successful routine will be repeated.

	**Industrial Robots :** Industrial robots perform same tasks repeatedly without ever moving. These robots are working in industries in which there is requirement of performing dull and repeated tasks suitable for robot. An industrial robot never tired, it will perform their works day and night without ever complaining.
	**Remote Controlled Robots :** Remote controlled robot used for performing complicated and undetermined tasks that autonomous robot cannot perform due to uncertainty of operation. Complicated tasks are best performed by human beings with real brainpower. Therefore a person can guide a robot by using remote. Using remote controlled operation human can perform dangerous tasks without being at the spot where the tasks are performed. Example : NASA robot designed to explore volcanoes via remote control.
	**Aquatic Robots** aren't afraid of getting wet. They are used to gather environmental data about the world's oceans, perform surveillance missions, and inspect and repair infrastructure. Some of them float on the surface of water, while others dive to extreme depths.
	**Consumer Robots** are robots you can buy and use just for fun or to help you with tasks and chores. One of the most famous consumer robots is Roomba, which vacuums your floor autonomously.
	**Delivery Robots** transport items like food, groceries, and medical supplies from one point to another. They use cameras, GPS, and other sensors to travel autonomously, carrying their cargo in secure compartments. Starship robots drive on streets and sidewalks to bring packages to people›s homes.
	**Drones** are flying robots that let you capture data and images from an elevated vantage point. Drones come in a variety of sizes and shapes. A common design, which uses four rotors to fly, is called a quadrotor or quadcopter.

	**Educational Robots** include a variety of hands-on robotics modules and kits. You can find them in classrooms, STEM programs, and homes. Popular models include programmable robots like Cubelets, Dash and Dot, and Root.
	**Exoskeletons** are wearable robotic suits equipped with electric motors that help move the user's body. Some powered exoskeletons can even give the wearer superhuman strength. They work using sensors that detect when the user wants to move-to get up or walk, for example. The suit then activates the motors in a way that enables the desired motion
	**Humanoid Robots** have a mechanical body with arms, legs, and a head like that of a person's, and they can often walk and manipulate objects much like we do. The robot's body typically has a machine-like appearance, as is the case with humanoids like the friendly robot ambassador Asimo from Honda and the athletic, agile Atlas from Boston Dynamics.

7.1.2 Components of Robots



- **Actuators:** Actuators are the devices that are responsible for moving and controlling a system or machine. It helps to achieve physical movements by converting energy like electrical, hydraulic and air, etc. Actuators can create linear as well as rotary motion.
- **Power Supply:** It is an electrical device that supplies electrical power to an electrical load. The primary function of the power supply is to convert electrical current to power the load.
- **Electric Motors:** These are the devices that convert electrical energy into mechanical energy and are required for the rotational motion of the machines.
- **Pneumatic Air Muscles:** Air Muscles are soft pneumatic devices that are ideally best fitted for robotics. They can contract and extend and operate by pressurized air filling a pneumatic bladder. Whenever air is introduced, it can contract up to 40%.
- **Muscles wire:** These are made up of nickel-titanium alloy called Nitinol and are very thin in shape. It can also extend and contract when a specific amount of heat and electric current is supplied into it. Also, it can be formed and bent into different shapes when it is in its martensitic form. They can contract by 5% when electrical current passes through them.
- **Piezo Motors and Ultrasonic Motors:** Piezoelectric motors or Piezo motors are the electrical devices that receive an electric signal and apply a directional force to an opposing ceramic plate. It helps a robot to move in the desired direction. These are the best suited electrical motors for industrial robots.
- **Sensor:** They provide the ability like see, hear, touch and movement like humans. Sensors are the devices or machines which help to detect the events or changes in the environment and send data to the computer processor. These devices are usually equipped with other electronic devices. Similar to human organs, the electrical sensor also plays a crucial role in Artificial Intelligence & robotics. AI algorithms control robots by sensing the environment, and it provides real-time information to computer processors.

7.1.3 Laws of Robotics

Asimov proposed three "Laws of Robotics " and later added the "zeroth law"

- Law 0: A robot may not injure humanity or through inaction, allow humanity to come to harm
- Law 1: A robot may not injure a human being or through inaction, allow a human being to come to harm, unless this would violate a higher order law
- Law 2: A robot must obey orders given to it by human beings, except where such orders would conflict with a higher order law
- Law 3: A robot must protect its own existence as long as such protection does not conflict with a higher order law

7.1.4 Applications of Robotics

Robotics have different application areas. Some of the important applications domains of robotics are as follows:

- **Robotics in defence sectors:** The defence sector is undoubtedly the one of the main parts of any country. Each country wants their defence system to be strong. Robots help to approach

inaccessible and dangerous zone during war. DRDO has developed a robot named Daksh to destroy life-threatening objects safely. They help soldiers to remain safe and deployed by the military in combat scenarios. Besides combat support, robots are ***also deployed in anti-submarine operations, fire support, battle damage management, strike missions, and laying machines.***

- **Robotics in Medical sectors**: Robots also help in various medical fields such as laparoscopy, neurosurgery, orthopaedic surgery, disinfecting rooms, dispensing medication, and various other medical domains.
- **Robotics in Industrial Sector:** Robots are used in various industrial manufacturing industries such as cutting, welding, assembly, disassembly, pick and place for printed circuit boards, packaging & labelling, palletizing, product inspection & testing, colour coating, drilling, polishing and handling the materials.

  Moreover, Robotics technology increases productivity and profitability and reduces human efforts, resulting from lower physical strain and injury. The industrial robot has some important advantages, which are as follows:

  - Accuracy
  - Flexibility
  - Reduced labour charge
  - Low noise operation
  - Fewer production damages
  - Increased productivity rate.
- **Robotics in Entertainment**: Over the last decade, use of robots is continuously getting increased in entertainment areas. Robots are being employed in entertainment sector, such as movies, animation, games and cartoons. Robots are very helpful where repetitive actions are required. A camera-wielding robot helps shoot a movie scene as many times as needed without getting tired and frustrated. A big-name Disney has launched hundreds of robots for the film industry.
- **Robots in the mining industry**: Robotics is very helpful for various mining applications such as robotic dozing, excavation and haulage, robotic mapping & surveying, robotic drilling and explosive handling, etc. A mining robot can solely navigate flooded passages and use cameras and other sensors to detect valuable minerals. Further, robots also help in excavation to detect gases and other materials and keep humans safe from harm and injuries. The robot rock climbers are used for space exploration, and underwater drones are used for ocean exploration.

7.1.5 Advantages of Robotics

Advantages of Robotics

1. ***Increased Efficiency***: Robots can work 24/7 without getting tired, leading to increased productivity and efficiency.
2. ***Improved Accuracy***: Robots are capable of performing tasks with high precision and accuracy, reducing errors and improving quality.

3. ***Increased Safety***: Robots can perform tasks that are dangerous for humans, improving overall safety in the workplace.
4. ***Reduced Labor Costs***: The use of robots can lead to reduced labor costs, as robots can perform tasks more cheaply than human workers.

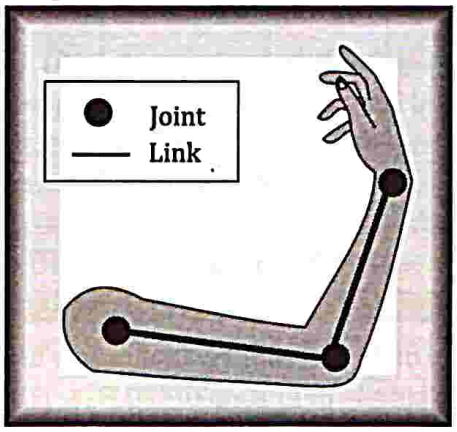
7.1.6 Disadvantages of Robotics

Disadvantages of Robotics

1. **Initial Cost:** Implementing and maintaining a robotics system can be expensive, especially for small and medium-sized businesses.
2. **Job Losses:** The increased use of robots may result in job losses for human workers, particularly in industries where manual labor is prevalent.
3. **Limited Capabilities:** Robots are still limited in their capabilities compared to human workers and may not be able to perform tasks requiring dexterity or creativity.
4. **Maintenance Costs:** Robots require regular maintenance and repair, which can be time-consuming and expensive.

7.1.7 Robot Kinematics

- Robot kinematics refers to the study of the motion of robotic systems without considering the forces that cause the motion. It focuses on determining the position and orientation of a robot's end effector, such as a robotic arm, based on the joint angles of the robot. By analyzing the kinematics of a robot, engineers can understand how the robot moves and interacts with its environment.
- **Important terminologies :**
  - A **link** is defined as a single part which can be a resistant body or a combination of resistant bodies having inflexible connections and having a relative motion with respect to other parts of the machine. A link is also known as a kinematic link or element.



There are different division of link in robot.

* **Rigid link:** In this type of link, there will not be any deformation while transmitting the motion. For example, the industrial robotic arm is having rigid links, there will not be any deformation while moving the arm.

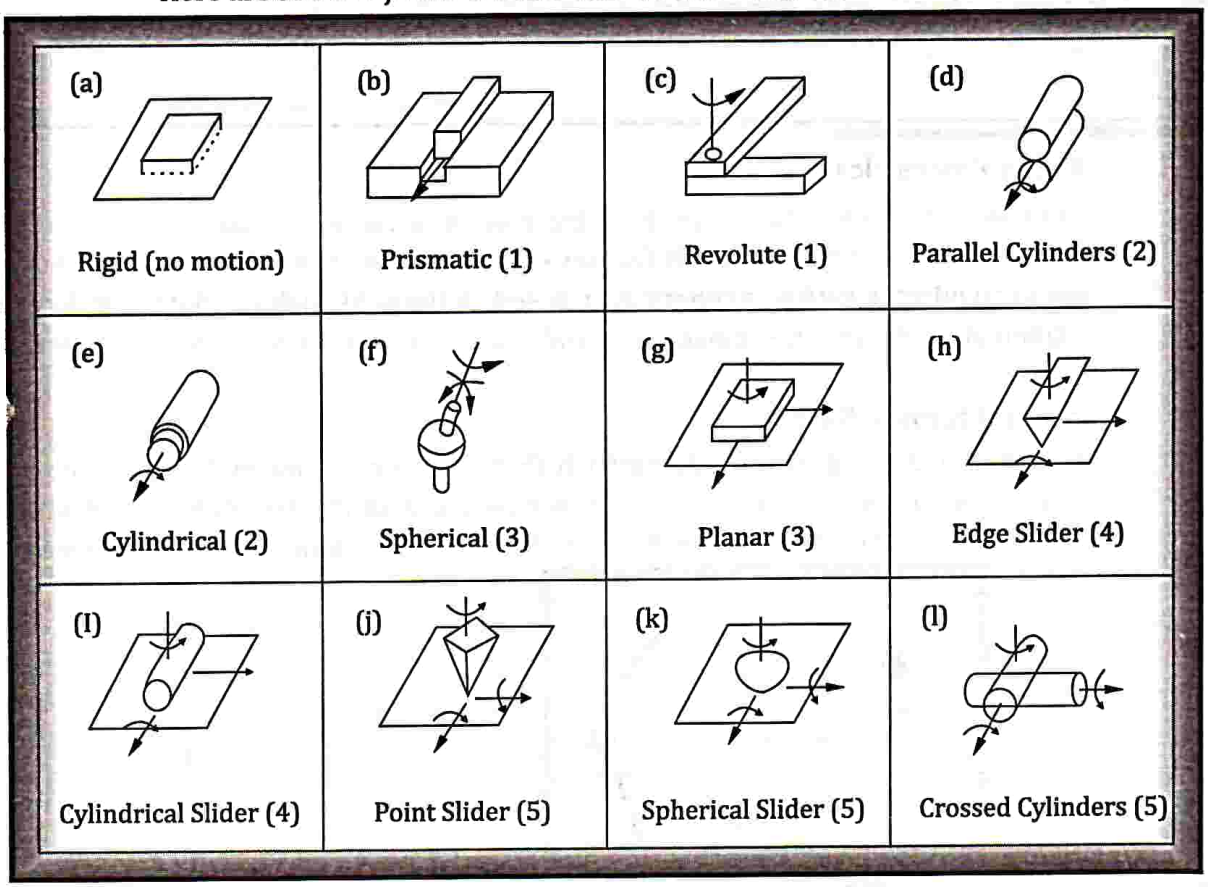
* **Flexible link:** In this type of link, there will be a partial deformation while transmitting the motion. One of the examples of flexible links is belt drives.
* **Fluid link:** In this type of link, motion is transmitted with the help of fluid pressure. Hydraulic actuators, brakes are an example of a fluid link.

- A **joint** is a connection between two or more links, which allows some motion, or potential motion, between the connected links. Joints are also called Kinematic pair."

  There are different classification of joints. Here is the main classification of joints based on.

  - Type of contact between links
  - Type of relative motion
  - Nature of constraint or Types of closure

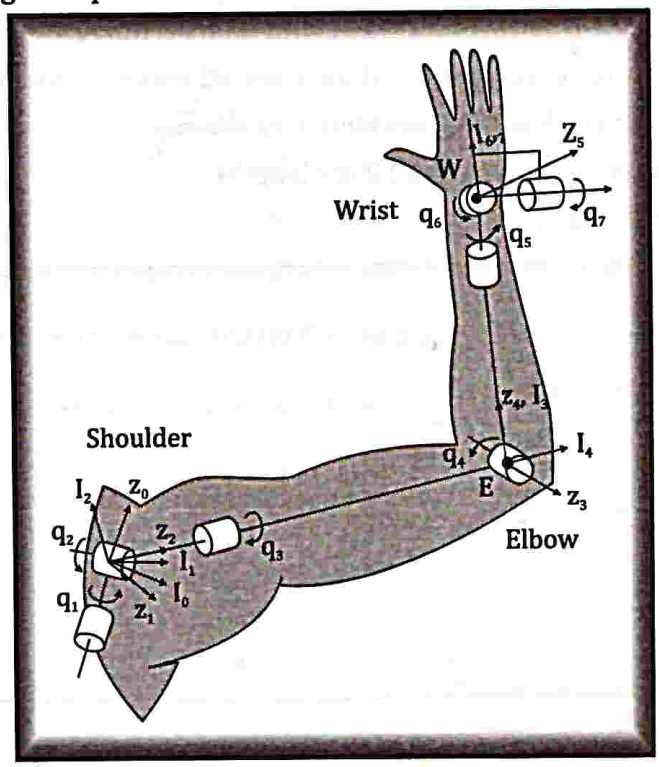
  Here are some of joints based on above classification.



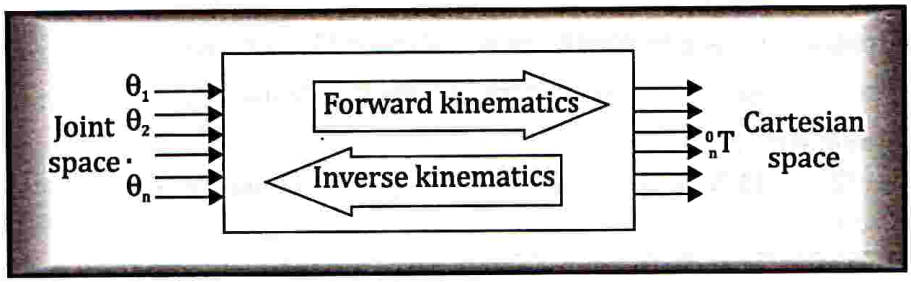
- The **Degree of Freedom** (D.O.F) is one of the parameters commonly using to mention the motion capability of a robot. Here is a simple definition of D.O.F.

  D.O.F is defined as the way in which a robot or machine can move. The number of degrees of freedom is equal to the total number of independent displacement or aspects of motion.

The following example shows the D.O.F of a human arm. We can move the arm in 7 D.O.F.

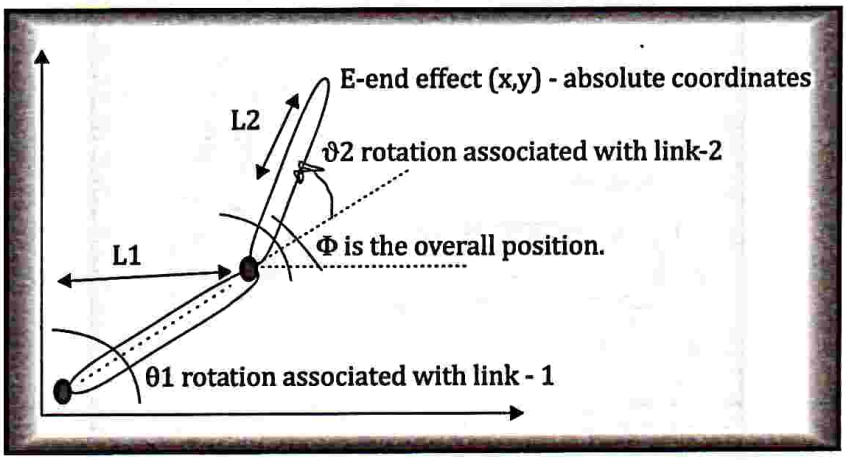


- In robot kinematics, there are two main components: forward kinematics and inverse kinematics.



- **Forward Kinematics :** Forward kinematics involves determining the position and orientation of the end effector of a robot given the joint angles. It answers the question, "Where is the end effector located in the robot's workspace?" This information is crucial for tasks such as trajectory planning and motion control. To calculate the forward kinematics, engineers use kinematic equations and the Denavit-Hartenberg parameters. These parameters describe the relationship between adjacent robot links and joints, allowing for the transformation of joint angles into Cartesian coordinates.
- **Inverse kinematics**, on the other hand, involves finding the joint angles required to achieve a desired position and orientation of the end effector. The rotation matrix is crucial in this process as it helps us determine the orientation of the end effector based on the given position. By decomposing the rotation matrix, we can extract the joint angles needed to achieve the desired orientation.

- **Example : Geometrically approach for Kinematic Analysis of 2-DOF**
  - Let say a 2-DOF robotic system with link-1 & link-2.
  - $\vartheta 1$ rotation associated with link-1 and $\vartheta 2$ rotation associated with link-2.
  - ϑ in anti-clock will be considered as +Ve.
  - Let E-> end effect. L1 and L2 are lengths.
  - Φ is the overall position.



From the above figure :

Joint space → $\vartheta 1$& $\vartheta 2$

World space → Φ , (x,y)

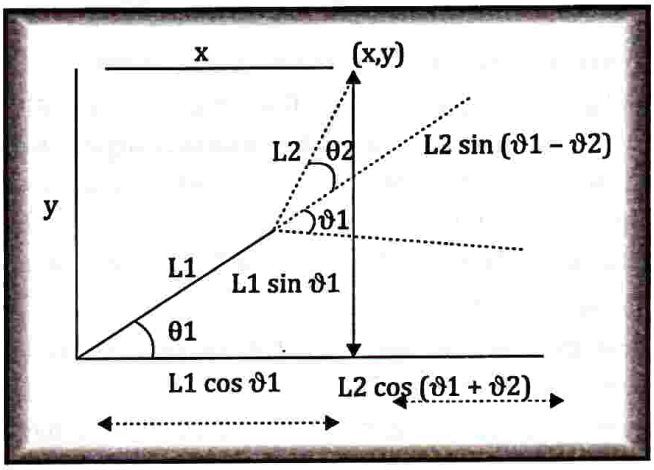
Conversion from Joint space to World space → Forward kinematics

Conversion from World space to Joint space → Inverse kinematics

Forward kinematics :

Given : $\vartheta 1$ & $\vartheta 2$ 11 & l2

To find : Φ = ? x = ? y = ?



Considering:

C1 → cos $\vartheta 1$

S1 → sin $\vartheta 1$

C2 → cos $\vartheta 2$

S2 → sin $\vartheta 2$

C12 → cos ($\vartheta 1$ + $\vartheta 2$)

S12 → sin ($\vartheta 1$ + $\vartheta 2$)

x = L1C1 + L2C12

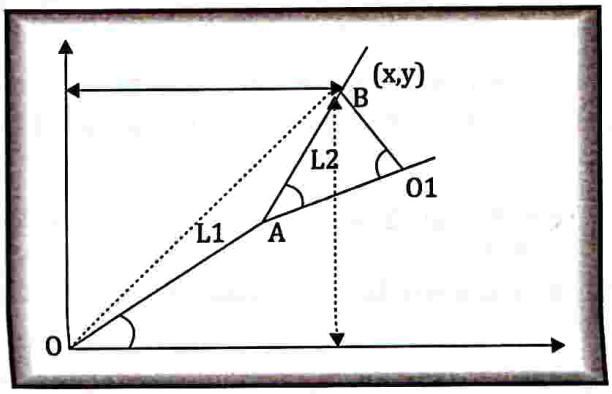
y = L2S2 + L2S12

Φ = $\vartheta 1$ + $\vartheta 2$

Inverse kinematics :

Given : Φ, (x ,y),11 & l2

To find : ϑ1 = ? , ϑ2 = ?



Inverse kinematics can be obtained by squaring x & y as:

$x^2 + y^2 = OB^2$

$OB^2 = O^1B^2 + OO^1$

$= (l_2S_2)^2 + *1_1 + 1_2C_2)^2$

$x^2 + y^2 = l_2^2 S_2^2 + l_1^2 + l_2^2 C_2^2 + 2l_1 l_2 C_2$

$2l_1l_2C_2 = x^2 + y^2 - l_1^2 - l_2^2$

$C2 = |\text{Cos } \vartheta 2 = \left(x^2 + y^2 - l_1^2 - l_2^2\right)/\left(2l_1l_2\right)$

$\vartheta 2 = \text{Cos}^{-1}(C2)$

$\vartheta 1 = \Phi - \vartheta 2$

Note:

Inverse kinematics may not have unique solution. It may have different configuration (say we may have two solution for the given problem). Hence we define we need under solution or above solution.

7.2 Computer Vision

7.2.1 What is Computer Vision?

- Computer vision is one of the most important fields of artificial intelligence (AI) and computer science engineering that makes computer systems capable of extracting meaningful information from visual data like videos and images. Further, it also helps to take appropriate actions and make recommendations based on the extracted information.
- Some most popular applications of computer vision:
  - **Facial recognition**: Computer vision has enabled machines to detect face images of people to verify their identity. Initially, the machines are given input data images in which computer vision algorithms detect facial features and compare them with databases of fake profiles. Popular social media platforms like Facebook also use facial recognition to detect and tag users. Further, various government spy agencies are employing this feature to identify criminals in video feeds.
  - **Healthcare and Medicine**: Computer vision has played an important role in the healthcare and medicine industry. Traditional approaches for evaluating cancerous tumors are time-consuming and have less accurate predictions, whereas computer vision technology provides faster and more accurate chemotherapy response assessments; doctors can identify cancer patients who need faster surgery with life-saving precision.

- **Self-driving vehicles:** Computer vision technology has also contributed to its role in self-driving vehicles to make sense of their surroundings by capturing video from different angles around the car and then introducing it into the software. This helps to detect other cars and objects, read traffic signals, pedestrian paths, etc., and safely drive its passengers to their destination.
- **Optical character recognition** helps us extract printed or handwritten text from visual data such as images. Further, it also enables us to extract text from documents like invoices, bills, articles, etc.
- **Machine inspection:** Computer vision is vital in providing an image-based automatic inspection. It detects a machine's defects, features, and functional flaws, determines inspection goals, chooses lighting and material-handling techniques, and other irregularities in manufactured products.
- **Retail (e.g., automated checkouts):** Computer vision is also being implemented in the retail industries to track products, shelves, wages, record product movements into the store, etc. This AI-based computer vision technique automatically charges the customer for the marked products upon checkout from the retail stores.
- **3D model building:** 3D model building or 3D modeling is a technique to generate a 3D digital representation of any object or surface using the software. In this field also, computer vision plays its role in constructing 3D computer models from existing objects. Furthermore, 3D modeling has a variety of applications in various places, such as Robotics, Autonomous driving, 3D tracking, 3D scene reconstruction, and AR/VR.
- **Medical imaging:** Computer vision helps medical professionals make better decisions regarding treating patients by developing visualization of specific body parts such as organs and tissues. It helps them get more accurate diagnoses and a better patient care system. E.g., Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) scanner to diagnose pathologies or guide medical interventions such as surgical planning or for research purposes.
- **Automotive safety:** Computer vision has added an important safety feature in automotive industries. E.g., if a vehicle is taught to detect objects and dangers, it could prevent an accident and save thousands of lives and property.
- **Surveillance:** It is one of computer vision technology's most important and beneficial use cases. Nowadays, CCTV cameras are almost fitted in every place, such as streets, roads, highways, shops, stores, etc., to spot various doubtful or criminal activities. It helps provide live footage of public places to identify suspicious behavior, identify dangerous objects, and prevent crimes by maintaining law and order.
- **Fingerprint recognition and biometrics:** Computer vision technology detects fingerprints and biometrics to validate a user's identity. Biometrics deals with recognizing persons based on physiological characteristics, such as the face, fingerprint, vascular pattern, or iris, and behavioral traits, such as gait or speech. It combines Computer Vision with knowledge of human physiology and behavior.

7.2.2 Computer Vision Techniques

1. **Image Classification :** Image classification is the simplest technique of Computer Vision. The main aim of image classification is to classify the image into one or more different categories. Image classifier basically takes an image as input and tells about different objects present in that image, such as a person, dog, tree, etc. However, it would not give you other more information about the image data, such as how many persons are there, tree colour, item positions, etc., and for this, we need to go for any other CV technique.

   Step-by-Step process :

   a. *Pre-processing – This step improves image data by eliminating undesired deformities and enhancing specific key aspects of the picture so that Computer Vision models can operate with this better data.*
   b. Data cleaning-sometimes called data cleansing-is an important step in preparing your data for training your model, as inaccuracies in data lead to inaccuracies in the image classification model.
   c. Object detection: locating objects within the image set - This is the process of locating an object, which entails segmenting the picture and determining the location of the object.
   d. Object recognition and training: labeling located images - Deep Learning algorithms discover patterns in the picture and characteristics that may be unique to a certain label. The model learns from this dataset and becomes more accurate in the future.
   e. Object classification: your model is ready to classify your images - This is the final step in the process-you've built an AI model that classifies fashion images by several different criteria.
   f. Connecting to an AI workflow - After completing this process, you can now connect your image classifying AI model to an AI workflow. This defines the input-where new data comes from, and output-what happens once the data has been classified. For example, data could come from new stock intake and output could be to add the data to a Google sheet.

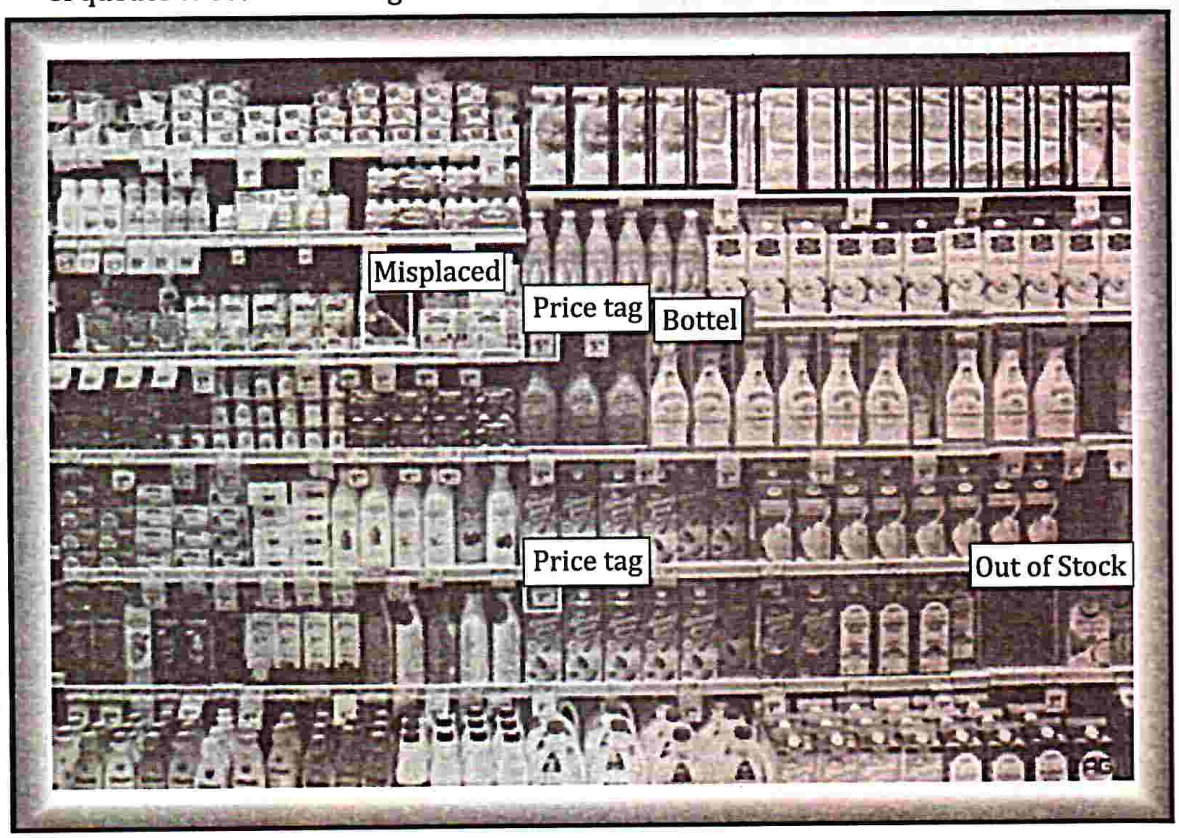
   Image classification is basically of two types, Binary classification and multi-class classification. As the name suggests, binary image classification looks for a single class in the given image and provides results based on if the image has that object or not. For example, we can achieve superhuman performance in detecting skin cancer in humans by training an AI system on both images that have skin cancer and images that do not have skin cancer.

2. **Object Detection :** Object detection is another popular technique of computer vision that can be performed after Image classification or which uses image classification to detect the objects in visual data. It is basically used to recognize the objects within the boundary boxes and find the class of the objects in the image. Object detection makes use of deep learning and machine learning technology to generate useful results.

   Object detection has several applications, including object tracking, retrieval, video surveillance, image captioning, etc. A variety of techniques can be used to perform object detection, which includes R-CNN, YOLO v2, etc.

Object detection use cases & applications :

- **Object detection in retail -** Strategically placed people counting systems throughout multiple retail stores are used to gather information about how customers spend their time and customer footfall. AI-based customer analysis to detect and track customers with cameras helps to understand customer interaction and experience, optimise the store layout, and make operations more efficient. A widespread use case is the detection of queues to reduce waiting time in retail stores.



- **Autonomous driving**: Self-driving cars depend on object detection to recognise pedestrians, traffic signs, other vehicles, and more. For example, Tesla's Autopilot AI heavily utilises object detection to perceive environmental and surrounding threats, such as oncoming vehicles or obstacles.
- **Video surveillance:** Because state-of-the-art object detection techniques can accurately identify and track multiple instances of a given object in a scene, these techniques naturally lend themselves to automated video surveillance systems. For example, object detection models can track multiple people at once, in real-time, as they move through a given scene or across video frames. From retail stores to industrial factory floors, this kind of granular tracking could provide invaluable insights into security, worker performance and safety, retail foot traffic, and more. Example of object detection in video analytics for people detection in dangerous areas using CCTV cameras.

- **Vehicle detection with AI in transportation**: Object recognition is used to detect and count vehicles for traffic analysis or to detect cars that stop in dangerous areas, such as crossroads or highways.
- **Animal detection in agriculture**: Object detection is used in agriculture for tasks such as counting, animal monitoring, and evaluation of the quality of agricultural products. Damaged produce can be detected while processing using machine learning algorithms.
- **Medical feature detection in healthcare**: Object detection has allowed for many breakthroughs in the medical community. Because medical diagnostics rely heavily on studying images, scans, and photographs, object detection involving CT and MRI scans has become extremely useful for diagnosing diseases, for example, with ML algorithms for tumour detection.

3. **Semantic Segmentation :** Semantic Segmentation is not only about detecting the classes in an image as image classification. Instead, it classifies each pixel of an image to specify what objects it has. It tries to determine the role of each pixel in the image. It basically classifies pixelS in a particular category without differentiating the object instances. Or we can say it classifies similar objects as a single class from the pixel levels. For example, if an image contains two dogs, then semantic segmentation will put both the dogs under the same label. It tries to understand the role of each pixel in an image.
4. **Instance Segmentation :** Instance segmentation can classify the objects in an image at pixel level as similar to semantic segmentation but with a more advanced level. It means Instance Segmentation can classify similar types of objects into different categories. For example, if visual consists of various cars, then with semantic segmentation, we can tell that there are multiple cars, but with instance segmentation, we can label them according to their colour, shape, etc.

5. **Panoptic Segmentation :** Panoptic Segmentation is one of the most powerful computer vision techniques as it combines the Instance and Semantic Segmentation techniques. It means with Panoptic Segmentation, you can classify image objects at pixel levels and can also identify separate instances of that class.
6. **Keypoint Detection :** Keypoint detection tries to detect some key points in an image to give more details about a class of objects. It basically detects people and localizes their key points. There are mainly two keypoint detection areas, which are Body Keypoint Detection and Facial Keypoint Detection. For example, Facial keypoint detection includes detecting key parts of the human face such as the nose, eyes, corners, eyebrows, etc. Keypoint detection mainly has applications, including face detection, pose detection, etc.
7. **Person Segmentation :** Person segmentation is a type of image segmentation technique which is used to separate the person from the background within an image. It can be used after the pose estimation, as with this, we can closely identify the exact location of the person in the image as well as the pose of that person.
8. **Depth Perception :** Depth perception is a computer vision technique that provides the visual ability to machines to estimate the 3D depth/distance of an object from the source. Depth Perception has wide applications, including the Reconstruction of objects in Augmented Reality, Robotics, self-driving cars, etc. LiDAR(Lights Detection and Ranging) is one of the popular techniques that is used for in-depth perception. With the help of laser beams, it measures the relative distance of an object by illuminating it with laser light and then measuring the reflections using sensors.
9. **Image Captioning :** Image captioning, as the name suggests, is about giving a suitable caption to the image that can describe the image. It makes use of neural networks, where when we input an image, then it generates a caption for that image that can easily describe the image. It is not only the task of Computer vision but also an NLP task.
10. **3D Object Reconstruction :** As the name suggests, 3D object reconstruction is a technique that can extract 3D objects from a 2D image. Currently, it is a much-developing field of computer vision, and it can be done in different ways for different objects. On this technique, one of the most successful papers is PiFuHD, which tells about 3D human digitization.

Summary

- **Robotics** is a sub-domain of engineering and science that includes mechanical engineering, electrical engineering, computer science, and others. This branch deals with the design, construction, use to control robots, sensory feedback and information processing.
- **Different types** of Robots : Manipulator, Legged robots, Wheeled robots, Autonomous robots, Industrial robots, Remote controlled robots, Aquatic robots, Consumer robots, Delivery robots, Drones, Educational robots, Exoskeletons, Humanoid.
- **Components** of Robots – Actuator, Electric motor, Power supply, Pneumatic Air Muscles, Muscles wire, Piezo Motors and Ultrasonic Motors, Sensor
- Asimov proposed three "Laws of Robotics " and later added the "zeroth law"
- **Applications** of Robotics in various sectors such as – Defence, Medical, Industry, Entertainment, mining industry.

- **Advantages** : Increased efficiency, Improved Accuracy, Increased Safety , Reduced Labor Costs
- **Disadvantages** : Initial Cost, Job Losses, Limited Capabilities, Maintenance Costs
- **Robot kinematics** refers to the study of the motion of robotic systems without considering the forces that cause the motion. It focuses on determining the position and orientation of a robot's end effector, such as a robotic arm, based on the joint angles of the robot. By analyzing the kinematics of a robot, engineers can understand how the robot moves and interacts with its environment.
- A **link** is defined as a single part which can be a resistant body or a combination of resistant bodies having inflexible connections and having a relative motion with respect to other parts of the machine. A link is also known as a kinematic link or element.
- A **joint** is a connection between two or more links, which allows some motion, or potential motion, between the connected links. Joints are also called Kinematic pair."
- The **Degree of Freedom** (D.O.F) is defined as the way in which a robot or machine can move. The number of degrees of freedom is equal to the total number of independent displacement or aspects of motion
- **Forward kinematics** involves determining the position and orientation of the end effector of a robot given the joint angles. It answers the question, "Where is the end effector located in the robot's workspace?" This information is crucial for tasks such as trajectory planning and motion control
- **Inverse kinematics**, on the other hand, involves finding the joint angles required to achieve a desired position and orientation of the end effector. The rotation matrix is crucial in this process as it helps us determine the orientation of the end effector based on the given position
- **Computer vision** is one of the most important fields of artificial intelligence (AI) and computer science engineering that makes computer systems capable of extracting meaningful information from visual data like videos and images. Further, it also helps to take appropriate actions and make recommendations based on the extracted information.
- **Applications of Computer Vision** : Facial recognition, Healthcare and Medicine, Self-driving vehicles, Optical character recognition, Machine inspection, Retail (e.g., automated checkouts), 3D model building, Medical imaging, Automotive safety, Surveillance, Fingerprint recognition and biometrics
- **Image classification** is the simplest technique of Computer Vision. The main aim of image classification is to classify the image into one or more different categories. Image classifier basically takes an image as input and tells about different objects present in that image, such as a person, dog, tree, etc.
- **Object detection** is another popular technique of computer vision that can be performed after Image classification or which uses image classification to detect the objects in visual data. It is basically used to recognize the objects within the boundary boxes and find the class of the objects in the image. Object detection makes use of deep learning and machine learning technology to generate useful results.
- **Other techniques of Computer Vision :** Semantic Segmentation, Instance Segmentation, Panoptic Segmentation, Keypoint Detection, Person Segmentation, Depth Perception, Image Captioning and 3D Object Reconstruction

7.3 Review Questions

Short Answer Questions

1. What is Robotic Engineering ?
2. List any 4 types of Robots.
3. List the components of Robots.
4. List the laws of robotics.
5. Explain any one application area of Robotics.
6. List the advantages of Robotic.
7. List the disadvantages of Robotics
8. Define Robot Kinematics.
9. Define link & join.
10. What do you mean by Degree of Freedom (D.O.F)?
11. Differentiate between Forward Kinematics & Inverse Kinematics.
12. Define Computer Vision.
13. List any four application areas of Computer vision.
14. What is Image classification?
15. Define Object detection.

Long Answer Questions

1. With a neat diagram, explain the different components of robots.
2. Explain any 4 application areas of Robotics.
3. With an example, explain forward kinematics & inverse kinematics for 2-DOF. Use Geometrical approach.
4. Explain any 4 application areas of computer vision.
5. What do you mean by Image classification? Explain the step-by-step process.
6. Define object detection. Explain any 4 application areas of object detection.
7. Write a note on :
    a. Semantic Segmentation
    b. Key point detection.
    c. Depth Perception
    d. Image Captioning

❖❖❖❖

UNIT - IV

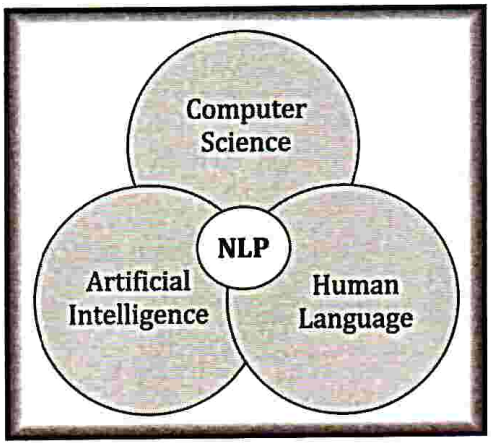
CHAPTER 8

NATURAL LANGUAGE PROCESSING

- Introduction
  - Time line of NLP
  - Components of NLP
  - Ambiguity in NLP
  - Steps to build an NLP pipeline
  - Advantages of NLP
  - Disadvantages of NLP
  - Applications of NLP
- Five Phases of NLP
  - Lexical or Morphological Analysis
  - Syntax Analysis or Parsing
  - Semantic Analysis
  - Discourse Integration
  - Pragmatic Analysis
- Review Questions

8.1 Introduction

- Natural language processing (NLP) is a subfield of Artificial Intelligence (AI). This is a widely used technology for personal assistants that are used in various business fields/areas. This technology works on the speech provided by the user breaks it down for proper understanding and processes it accordingly.
- Natural Language Processing (NLP) is a field that combines computer science, linguistics, and machine learning to study how computers and humans communicate in natural language. The goal of NLP is for computers to be able to interpret and generate human language. This not only improves the efficiency of work done by humans but also helps in interacting with the machine. NLP bridges the gap of interaction between humans and electronic devices.



8.1.1 Time line of NLP

1. **1940** - The Natural Languages Processing started in the year 1940s.
2. **1948**- In the Year 1948, the first recognisable NLP application was introduced in Birkbeck College, London.
3. **1950**s-In the Year 1950s, there was a conflicting view between linguistics and computer science. Now, Chomsky developed his first book syntactic structures and claimed that language is generative in nature.
4. **(1960-1980)** - Flavored with Artificial Intelligence (AI). Some of the key developments are :
    - **Augmented Transition Networks (ATN)** - is a finite state machine that is capable of recognizing regular languages.
    - **Case Grammar** uses languages such as English to express the relationship between nouns and verbs by using the preposition.
        - **SHRDLU** - helps users to communicate with the computer and moving objects. It can handle instructions such as "pick up the green boll" and also answer the questions like "What is inside the black box."
        - **LUNAR** is the classic example of a Natural Language database interface system that is used ATNs and Woods' Procedural Semantics.

5. **1980 – Current** - Till the year 1980, natural language processing systems were based on complex sets of hand-written rules. After 1980, NLP introduced machine learning algorithms for language processing.

8.1.2 Components of NLP

There are two components of NLP

1. **Natural Language Understanding (NLU)** - Natural Language understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

   NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

   NLU involves the following tasks -

   It is used to map the given input into useful representation.

   It is used to analyze different aspects of the language.

2. **Natural Language Generation (NLG)** - Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

NLU	NLG
NLU is the process of reading and interpreting language	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs	It produces constructing natural language outputs from non-linguistic inputs.

8.1.3 Ambiguity in NLP

NLU is naturally harder than NLG tasks. There are lot of ambiguity while learning or trying to interpret a language. NLP has the following types of ambiguities:

1. **Lexical Ambiguity :** The ambiguity of a single word is called lexical ambiguity. For example, treating the word silver as a noun, an adjective, or a verb.
2. **Syntactic Ambiguity :** This kind of ambiguity occurs when a sentence is parsed in different ways. For example, the sentence "The man saw the girl with the telescope". It is ambiguous whether the man saw the girl carrying a telescope or he saw her through his telescope.
3. **Semantic Ambiguity :** This kind of ambiguity occurs when the meaning of the words themselves can be misinterpreted. In other words, semantic ambiguity happens when a sentence contains an ambiguous word or phrase. For example, the sentence "The car hit the pole while it was moving" is having semantic ambiguity because the interpretations can be "The car, while moving, hit the pole" and "The car hit the pole while the pole was moving".
4. **Anaphoric Ambiguity :** This kind of ambiguity arises due to the use of anaphora entities in discourse. For example, the horse ran up the hill. It was very steep. It soon got tired. Here, the anaphoric reference of "it" in two situations cause ambiguity.

5. **Pragmatic ambiguity** : Such kind of ambiguity refers to the situation where the context of a phrase gives it multiple interpretations. In simple words, we can say that pragmatic ambiguity arises when the statement is not specific. For example, the sentence "I like you too" can have multiple interpretations like I like you (just like you like me), I like you (just like someone else dose).

8.1.4 Steps to build an NLP pipeline

Step1: Sentence Segmentation

Sentence Segment is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences.

Example: Consider the following paragraph -

Independence Day is one of the important festivals for every Indian citizen. It is celebrated on the 15th of August each year ever since India got independence from the British rule. The day celebrates independence in the true sense.

Sentence Segment produces the following result:

1. "Independence Day is one of the important festivals for every Indian citizen."
2. "It is celebrated on the 15th of August each year ever since India got independence from the British rule."
3. "This day celebrates independence in the true sense."

Step2: Word Tokenization:

Word Tokenizer is used to break the sentence into separate words or tokens.

Example:

JavaTrainSoft offers Corporate Training, Summer Training, Online Training, and Winter Training.

Word Tokenizer generates the following result:

"JavaTrainSoft", "offers", "Corporate", "Training", "Summer", "Training", "Online", "Training", "and", "Winter", "Training", "."

Step3: Stemming:

Stemming is used to normalize words into its base form or root form. For example, celebrates, celebrated and celebrating, all these words are originated with a single root word "celebrate." The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

For Example, intelligence, intelligent, and intelligently, all these words are originated with a single root word "intelligen." In English, the word "intelligen" do not have any meaning.

Step 4: Lemmatization:

Lemmatization is quite similar to the Stamming. It is used to group different inflected forms of the word, called Lemma. The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

Example:

In lemmatization, the words intelligence, intelligent, and intelligently has a root word intelligent, which has a meaning.

Step 5: Identifying Stop Words:

In English, there are a lot of words that appear very frequently like "is", "and", "the", and "a". NLP pipelines will flag these words as stop words. Stop words might be filtered out before doing any statistical analysis.

Step 6: Dependency Parsing:

Dependency Parsing is used to find that how all the words in the sentence are related to each other.

Step 7: POS tags:

POS stands for parts of speech, which includes Noun, verb, adverb, and Adjective. It indicates that how a word functions with its meaning as well as grammatically within the sentences. A word has one or more parts of speech based on the context in which it is used.

Example:

"Google" something on the Internet.

In the above example, Google is used as a verb, although it is a proper noun.

Step 8: Named Entity Recognition (NER)

Named Entity Recognition (NER) is the process of detecting the named entity such as person name, movie name, organization name, or location.

Example:

Steve Jobs introduced iPhone at the Macworld Conference in San Francisco, California.

Step 9: Chunking

Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

8.1.5 Advantages of NLP

Advantages of NLP

1. **Improved Human-Computer Interaction:** A specific advantage of NLP is that it enables computers to understand and process human language. This makes it easier for people to interact with their computers and vice versa. NLP results in more efficient and effective human-computer interaction and communication.
2. **Time-Efficiency and Cost-Effectiveness:** Language-based task automation is one of the applications of NLP. Examples of these tasks include text and speech processing, morphological and syntactic analyses, and lexical and relational semantics. This can result in time and cost savings for individuals and organizations.
3. **Data Generation and Content Creation:** Another advantage of natural language processing is that it aids in the generation of data and even in the creation of new and original content such as texts and images based on text-based training datasets and using natural language as command input.

4. **Professional and Business Applications:** NLP can also benefit individuals. It can improve and optimize sales and after-sales services through chatbots. It can aid professionals with their tasks as demonstrated through generative AI products such as ChatGPT from OpenAI or writing tools such as Grammarly.

   Advances Artificial Intelligence: Remember that NLP is one of the main goals and fields of artificial intelligence. Developments in natural language processing mark further developments in AI. It is also important to note that advancing large language models are critical to advancing artificial intelligence applications.

8.1.6 Disadvantages of NLP

Disadvantages of NLP

1. **Possible Issues with Context and Meanings:** One of the more specific limitations of NLP is its limited understanding of context and meanings. A particular model may not always understand the nuances of human language. It may not be able to identify sarcasm and idioms. This can lead to errors, inaccuracies, or irrelevance.
2. **Biased Results from Biased Training Data:** It is important to note that the quality of an NLP model depends on its training data. A dataset containing biases or inaccuracies would result in this particular model producing biased or inaccurate results. This can further result in controversial outcomes such as prejudicial claims.
3. **Issues with Rare or Out-of-Vocabulary Words:** NLP models and their applications may struggle to process certain words such as jargons and slangs that are not included in its training data. This can lead to unreliable outcomes of specific NLP tasks such as text classification and named entity recognition.
4. **Technical and Computational Requirements:** Another disadvantage of natural language processing is that its high-level applications depend on large language models which require high computational power. It is impossible for an individual or small organization with limited resources to deploy n-house NLP capabilities.
5. **Possible Ethical Concerns and Legal Issues:** Developing an NLP model requires using data. Some of these data are obtained from the personal or private data of individuals and organizations. The deployment of NLP applications raises concerns over data ownership, privacy rights, and intellectual property infringement, among others.

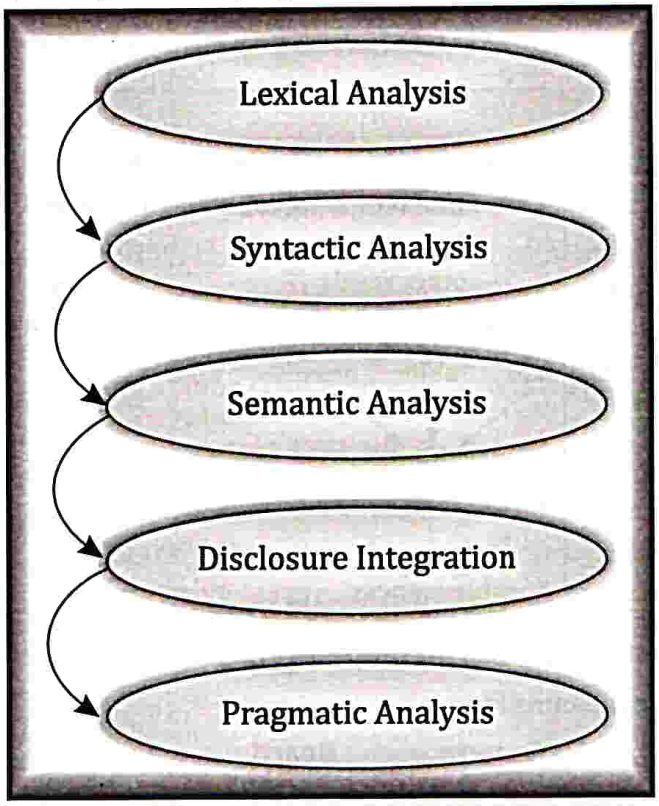
8.1.7 Applications of NLP

1. **Question Answering :** NLP can be seen in action by using Google Search or Siri Services. A major use of NLP is to make search engines understand the meaning of what we are asking and generate natural language in return to give us the answers.
2. **Spam Filters:** One of the most irritating things about email is spam. Gmail uses natural language processing (NLP) to discern which emails are legitimate and which are spam. These spam filters look at the text in all the emails you receive and try to figure out what it means to see if it's spam or not.

3. **Algorithmic Trading:** Algorithmic trading is used for predicting stock market conditions. Using NLP, this technology examines news headlines about companies and stocks and attempts to comprehend their meaning in order to determine if you should buy, sell, or hold certain stocks.
4. **Summarizing Information:** On the internet, there is a lot of information, and a lot of it comes in the form of long documents or articles. NLP is used to decipher the meaning of the data and then provides shorter summaries of the data so that humans can comprehend it more quickly.

8.2 Five Phases of NLP

Natural Language Processing is separated into five primary stages or phases, starting with simple word processing and progressing to identifying complicated phrase meanings.

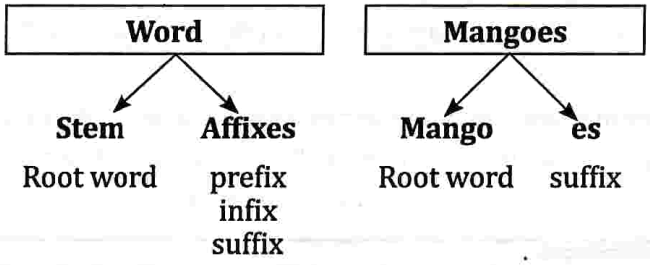


8.2.1 Lexical or Morphological Analysis

Lexical or Morphological Analysis is the initial step in NLP. It entails recognizing and analysing word structures. The collection of words and phrases in a language is referred to as the lexicon. Lexical analysis is the process of breaking down a text file into paragraphs, phrases, and words. The source code is scanned as a stream of characters and converted into intelligible lexemes.

It searches for morphemes, which are the smallest units of a word. The lexical analysis identifies the relationship between these morphemes and transforms the word into its root form. The word's probable parts of speech (POS) are also assigned by a lexical analyzer.

Examples



Steps to design Morphological Parser:

- Lexicon
- Morphotactic
- Orthographic Rules

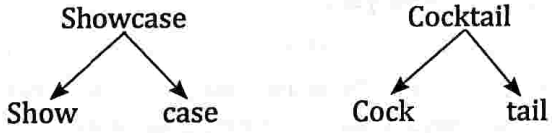
Lexicon	Morphotactic	Orthographic rules
• Stores basic information about a word • Word is stem or affix • If stem then, whether a Verb stem or Noun stem. • If affix then, whether it is prefix, infix or suffix	• Set of Rules to make decision • Decides a word appears before, after or in between other words • For ex: use able ness → useableness — Valid, obey rules able use ness → ableuseness — Invalid, disobey rules	Set of rules used to decide spelling changes. Lady + s -→ Ladys (invalid meaning) Lady + s → Ladies (y & s changes to ies to give a valid word – change in spelling) Kinfe + s -> Kinves

Thus Morphology means study of word/making of word

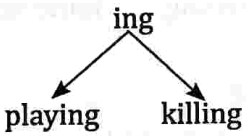
Some words has their own meaning for eg:

Camera Board Pen

Some words are there by which when divided into different words, those new words have their own meaning for ex :



Some words are which does not have their own meaning but when they are combined with other words, they become meaningful for ex:



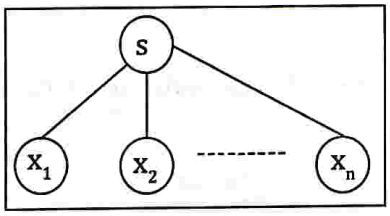
Thus, the detailed study of making of word is called as Morphological Analysis

Some more examples :

Input	Morphological	Explanation
Cats	Cat + N + PL	Root word(cat) + Noun + Plural
Geese	Goose + N + PL	Root word(goose) + Noun + Plural
Caught	Catch + V + PAST	Root word(catch) + Verb + Past speech
Merging	Merge + V + Pre_part	Root word(merge) + Verg + Present speech

8.2.2 Syntax Analysis or Parsing

1. Syntactic or Syntax analysis is a technique for checking grammar, arranging words, and displaying relationships between them. It entails examining the syntax of the words in the phrase and arranging them in a way that demonstrates the relationship between them.
2. Syntax analysis guarantees that the structure of a particular piece of text is proper. In this sense, syntactic analysis or parsing may be defined as the process of analysing the strings of symbols in natural language conforming to the rules of formal grammar.
3. For example, New York goes to John. This sentence New York goes to John is rejected by the Syntactic Analyzer as it makes no sense.
4. It also builds a data structure generally in the form of parse tree or abstract syntax tree or other hierarchical structure. ***Parse tree*** is defined as the graphical depiction of a derivation. The start symbol of derivation serves as the root of the parse tree. In every parse tree, the leaf nodes are terminals and interior nodes are non-terminals. A property of parse tree is that in-order traversal will produce the original input string.
5. Representation
   - Root vertex – Must be labeled by the start symbol.
   - Vertex – Labeled by a non-terminal symbol.
   - Leaves – Labeled by a terminal symbol or ε.



6. Grammar is very essential and important to describe the syntactic structure of well-formed programs. In 1956, Noam Chomsky has proposed different model of grammar which is effective for writing computer languages. Among various types, CFG (Context Free Grammar) is used.

7. A context-free grammar (CFG) consisting of a finite set of grammar rules is a quadruple (N, T, P, S) where
   - N is a set of non-terminal symbols.
   - T is a set of terminals where N ∩ T = NULL.
   - P is a set of rules, P: N → (N ∪ T)*, i.e., the left-hand side of the production rule P does have any right context or left context.
   - S is the start symbol

Examples

The grammar ({A}, {a, b, c}, P, A), P : A → aA, A → abc.

Non-terminal — terminal — Valid production rules — start symbol

Examples

Ram ate the delicious cake

Ram read the interesting book

S: Sentence → SB VP OB

SB : Subject → PN

VP : Verb Pharse → ADV V | V

OB : Object → the S1

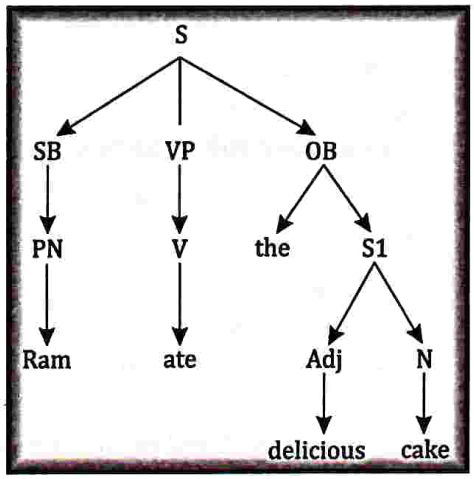
S1 : Subset of S → ADJ | N

PN : Proper Noun → Ram

ADJ : Adjective → interesting | delicious

N : Noun → book | cake

V : verb → read | ate



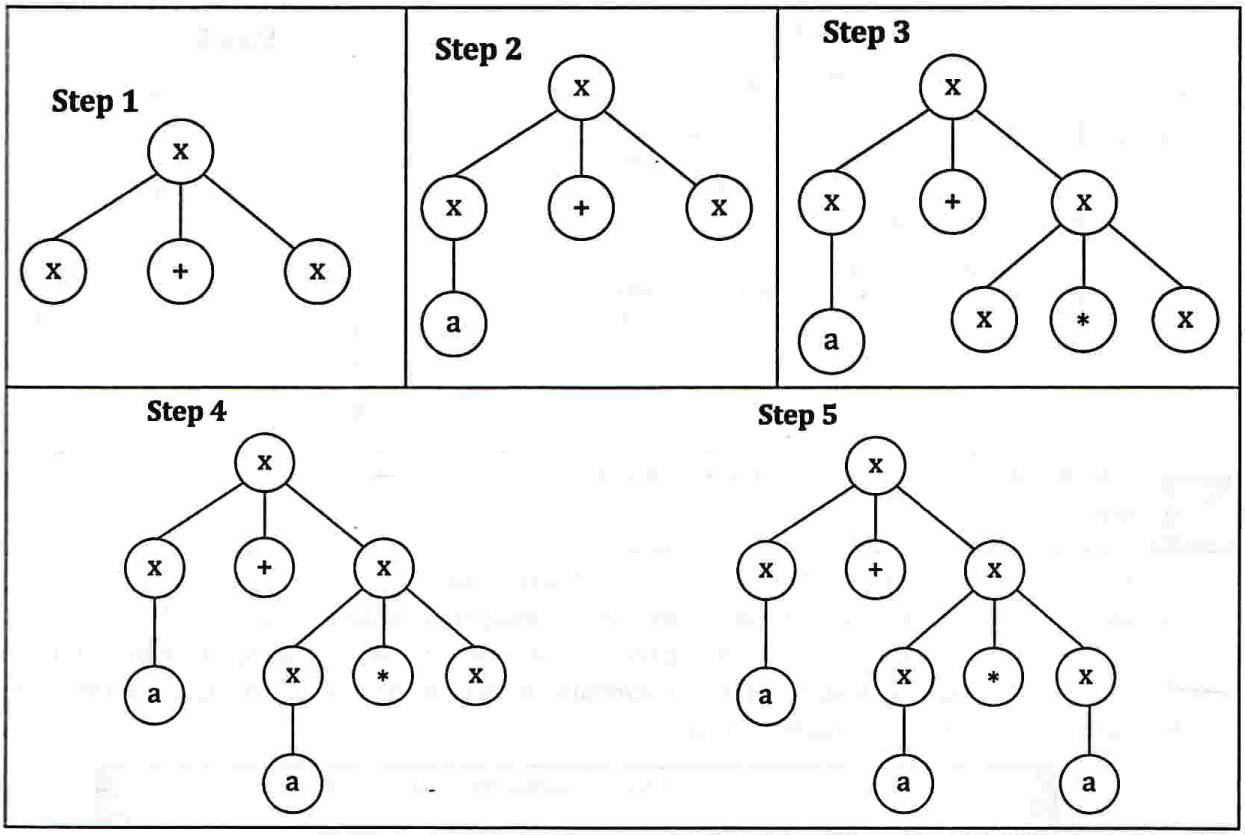
8. **Types of Derivation** : The two types of derivations, which can be used to decide which non-terminal to be replaced with production rule.
   - **Left-most Derivation :** In the left-most derivation, the sentential form of an input is scanned and replaced from the left to the right. The sentential form in this case is called the left-sentential form.

Examples **Let any set of production rules in a CFG be**

X → X+X | X*X |X| a over an alphabet {a}.

The leftmost derivation for the string "a+a*a" may be –

X → X+X → a+X → a + X*X → a+a*X → a+a*a



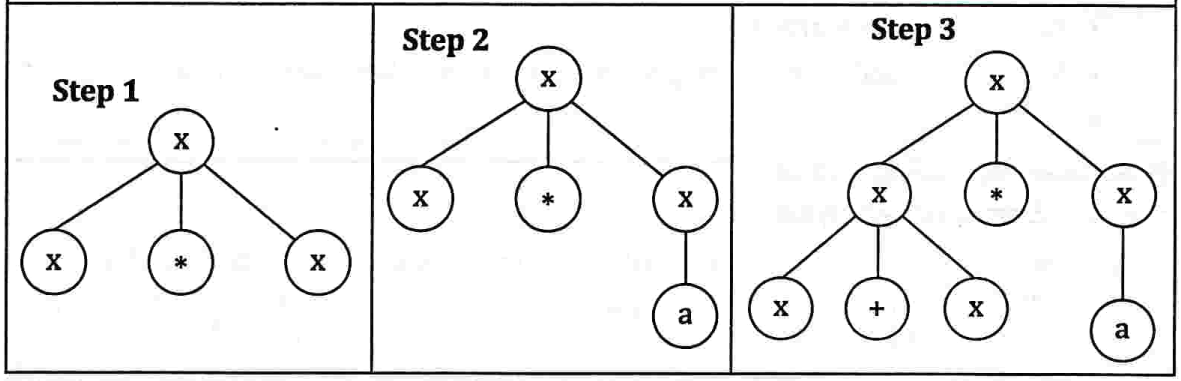
- ***Right-most Derivation*** : In the right-most derivation, the sentential form of an input is scanned and replaced from the right to the leftt. The sentential form in this case is called the right-sentential form.

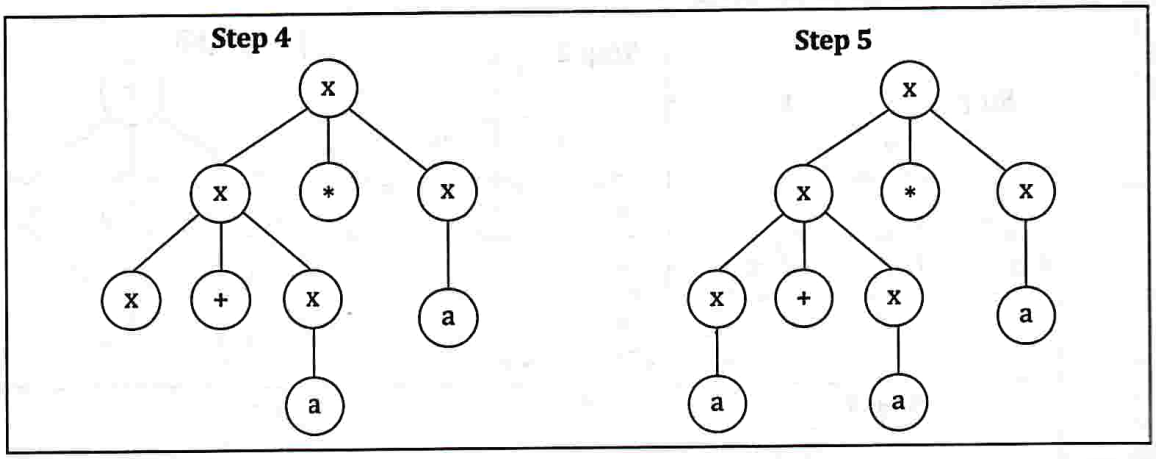
Examples **Let any set of production rules in a CFG be**

X → X+X | X*X |X| a over an alphabet {a}.

The rightmost derivation for the string "a+a*a" may be –

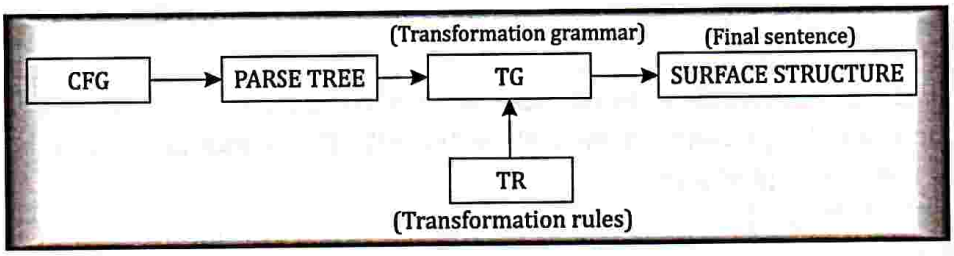
X → X*X → X*a → X+X*a → X+a*a → a+a*a





Note:

"Chomsky's grammar is a 'generative grammar of the transformational type. 'By that he means that it makes explicit the rules for generating new sentences, not for analyzing existing sentences; the rules themselves provide the analysis. And he means that among the rules are those for transforming one type of sentence into another (affirmative into negative, simple into compound or complex, and so forth); the transformations make the relationships among such sentences clear."



- Using parse tree is generated
- Expand parse tree to get surface structure
- In order to expand parse tree we need TG
- Parse tree is a deep structure
- To attach tags and restructuring grammar TR is used (tokens are reordered)
- Surface structure is the final output.

8.2.3 Semantic Analysis

1. Semantic analysis is the process of looking for meaning in a statement. It concentrates mostly on the literal meaning of words, phrases, and sentences is the main focus. It also deals with putting words together to form sentences. It extracts the text's exact meaning or dictionary definition. The meaning of the text is examined. It is accomplished by mapping the task domain's syntactic structures and objects.

   Take the following sentence for example: "The guava ate an apple." The line is syntactically valid, yet it is illogical because guavas cannot eat.

2. **Parts of Semantic Analysis:**

Semantic Analysis of Natural Language can be classified into two broad parts:

- **Lexical Semantic Analysis:** Lexical Semantic Analysis involves understanding the meaning of each word of the text individually. It basically refers to fetching the dictionary meaning that a word in the text is deputed to carry.
- **Compositional Semantics Analysis:** Although knowing the meaning of each word of the text is essential, it is not sufficient to completely understand the meaning of the text. For example, consider the following two sentences:

  Sentence 1: Students love GeeksforGeeks.

  Sentence 2: GeeksforGeeks loves Students.

  Although both these sentences 1 and 2 use the same set of root words {student, love, geeksforgeeks}, they convey entirely different meanings. Hence, under Compositional Semantics Analysis, we try to understand how combinations of individual words form the meaning of the text.

3. **Tasks involved in Semantic Analysis:**

In order to understand the meaning of a sentence, the following are the major processes involved in Semantic Analysis:

- Word Sense Disambiguation
- Relationship Extraction

1. **Word Sense Disambiguation :** In Natural Language, the meaning of a word may vary as per its usage in sentences and the context of the text. Word Sense Disambiguation involves interpreting the meaning of a word based upon the context of its occurrence in a text.

   For example, the word 'Bark' may mean 'the sound made by a dog' or 'the outermost layer of a tree.' Likewise, the word 'rock' may mean 'a stone' or 'a genre of music' – hence, the accurate meaning of the word is highly dependent upon its context and usage in the text.

   Thus, the ability of a machine to overcome the ambiguity involved in identifying the meaning of a word based on its usage and context is called Word Sense Disambiguation.

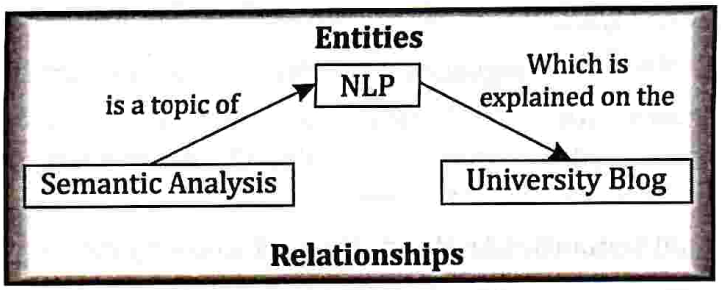
2. **Relationship Extraction:** Another important task involved in Semantic Analysis is Relationship Extracting. It involves firstly identifying various entities present in the sentence and then extracting the relationships between those entities.

   For example, consider the following sentence:

   Semantic Analysis is a topic of NLP which is explained on the university blog. The entities involved in this text, along with their relationships, are shown below.

   Semantic Analysis is a topic of **NLP** which is explained on the **university blog**

   [Entity] [Entity] [Entity]



4. Elements of Semantic Analysis:

Some of the critical elements of Semantic Analysis that must be scrutinized and taken into account while processing Natural Language are:

- **Hyponymy:** Hyponymys refers to a term that is an instance of a generic term. They can be understood by taking class-object as an analogy. For example: 'Color' is a hypernymy while 'grey', 'blue', 'red', etc, are its hyponyms.
- **Homonymy:** Homonymy refers to two or more lexical terms with the same spellings but completely distinct in meaning. For example: 'Rose' might mean 'the past form of rise' or 'a flower', – same spelling but different meanings; hence, 'rose' is a homonymy.
- **Synonymy:** When two or more lexical terms that might be spelt distinctly have the same or similar meaning, they are called Synonymy. For example: (Job, Occupation), (Large, Big), (Stop, Halt).
- **Antonymy:** Antonymy refers to a pair of lexical terms that have contrasting meanings – they are symmetric to a semantic axis. For example: (Day, Night), (Hot, Cold), (Large, Small).
- **Polysemy:** Polysemy refers to lexical terms that have the same spelling but multiple closely related meanings. It differs from homonymy because the meanings of the terms need not be closely related in the case of homonymy. For example: 'man' may mean 'the human species' or 'a male human' or 'an adult male human' – since all these different meanings bear a close association, the lexical term 'man' is a polysemy.
- **Meronomy:** Meronomy refers to a relationship wherein one lexical term is a constituent of some larger entity. For example: 'Wheel' is a meronym of 'Automobile'

5. Meaning Representation:

While, as humans, it is pretty simple for us to understand the meaning of textual information, it is not so in the case of machines. Thus, machines tend to represent the text in specific formats in order to interpret its meaning. This formal structure that is used to understand the meaning of a text is called **meaning representation**.

In order to accomplish Meaning Representation in Semantic Analysis, it is vital to understand the building units of such representations. The basic units of semantic systems are explained below:

- **Entity:** An entity refers to a particular unit or individual in specific such as a person or a location. For example BangaloreUniversity, Delhi, etc.

- **Concept:** A Concept may be understood as a generalization of entities. It refers to a broad class of individual units. For example Learning Portals, City, Students.
- **Relations:** Relations help establish relationships between various entities and concepts. For example: 'Bangalorelearninghub is a Learning Portal', 'Delhi is a City.', etc.
- **Predicate:** Predicates represent the verb structures of the sentences.

In Meaning Representation, we employ these basic units to represent textual information.

Approaches to Meaning Representations:

Now that we are familiar with the basic understanding of Meaning Representations, here are some of the most popular approaches to meaning representation:

- First-order predicate logic (FOPL)
- Semantic Nets
- Frames
- Conceptual dependency (CD)
- Rule-based architecture
- Case Grammar
- Conceptual Graphs

6. Semantic Analysis Techniques:

Based upon the end goal one is trying to accomplish, Semantic Analysis can be used in various ways. Two of the most common Semantic Analysis techniques are:

- ***Text Classification*** : In-Text Classification, our aim is to label the text according to the insights we intend to gain from the textual data.

Examples

- **In Sentiment Analysis,** we try to label the text with the prominent emotion they convey. It is highly beneficial when analyzing customer reviews for improvement.
- **In Topic Classification**, we try to categories our text into some predefined categories. For example: Identifying whether a research paper is of Physics, Chemistry or Maths
- **In Intent Classification**, we try to determine the intent behind a text message. For example: Identifying whether an e-mail received at customer care service is a query, complaint or request.

- ***Text Extraction :*** In-Text Extraction, we aim at obtaining specific information from our text.

Examples

- **In Keyword Extraction**, we try to obtain the essential words that define the entire document.
- **In Entity Extraction**, we try to obtain all the entities involved in a document.

7. Significance of Semantics Analysis:

Semantics Analysis is a crucial part of Natural Language Processing (NLP). In the ever-expanding era of textual information, it is important for organizations to draw insights from such data to fuel businesses. Semantic Analysis helps machines interpret the meaning of texts and extract useful information, thus providing invaluable data while reducing manual efforts.

Besides, Semantics Analysis is also widely employed to facilitate the processes of automated answering systems such as chatbots – that answer user queries without any human interventions.

8.2.4 Discourse Integration

1. The most difficult problem of AI is to process the natural language by computers or in other words natural language processing is the most difficult problem of artificial intelligence. If we talk about the major problems in NLP, then one of the major problems in NLP is discourse processing – building theories and models of how utterances stick together to form coherent discourse. Actually, the language always consists of collocated, structured and coherent groups of sentences rather than isolated and unrelated sentences like movies. These coherent groups of sentences are referred to as **discourse**.
2. The term "**discourse integration**" refers to a feeling of context. The meaning of any sentence is determined by the meaning of the sentence immediately preceding it. In addition, it establishes the meaning of the sentence that follows. The sentences that come before it play a role in discourse integration. That is to say, that statement or word is dependent on the preceding sentence or words. It's the same with the use of proper nouns and pronouns.

   For example, Billy bought it.

   The word "it" in the above sentence is dependent on the preceding discourse context. We can see that the "it" does not make sense in this statement. In fact, it refers to anything we don't understand. That is nothing more than the fact that the word "it" is dependent on the preceding sentence, which is not provided. So, once we've learned about "it," we'll be able to simply locate the reference. Discourse is concerned with the impact of a prior sentence on the current sentence.
3. **Lexical repetition:** Lexical repetition is a way to find the structure in a discourse, but it does not satisfy the requirement of being coherent discourse. To achieve the coherent discourse, we must focus on coherence relations in specific. As we know that coherence relation defines the possible connection between utterances in a discourse. Hebb has proposed such kind of relations as follows –

   We are taking two terms S0 and S1 to represent the meaning of the two related sentences –

   - **Result :** It infers that the state asserted by term S0 could cause the state asserted by S1. For example, two statements show the relationship result: Ram was caught in the fire. His skin burned.
   - **Explanation :** It infers that the state asserted by S1 could cause the state asserted by S0. For example, two statements show the relationship – Ram fought with Shyam's friend. He was drunk.
   - **Parallel:** It infers p(a1,a2,...) from assertion of S0 and p(b1,b2,...) from assertion S1. Here ai and bi are similar for all i. For example, two statements are parallel – Ram wanted car. Shyam wanted money.

- **Elaboration :** It infers the same proposition P from both the assertions – S0 and S1 For example, two statements show the relation elaboration: Ram was from Chandigarh. Shyam was from Kerala.
- **Occasion :** It happens when a change of state can be inferred from the assertion of S0, final state of which can be inferred from S1 and vice-versa. For example, the two statements show the relation occasion: Ram picked up the book. He gave it to Shyam.

4. Building Hierarchical Discourse Structure

The coherence of entire discourse can also be considered by hierarchical structure between coherence relations. For example, the following passage can be represented as hierarchical structure –

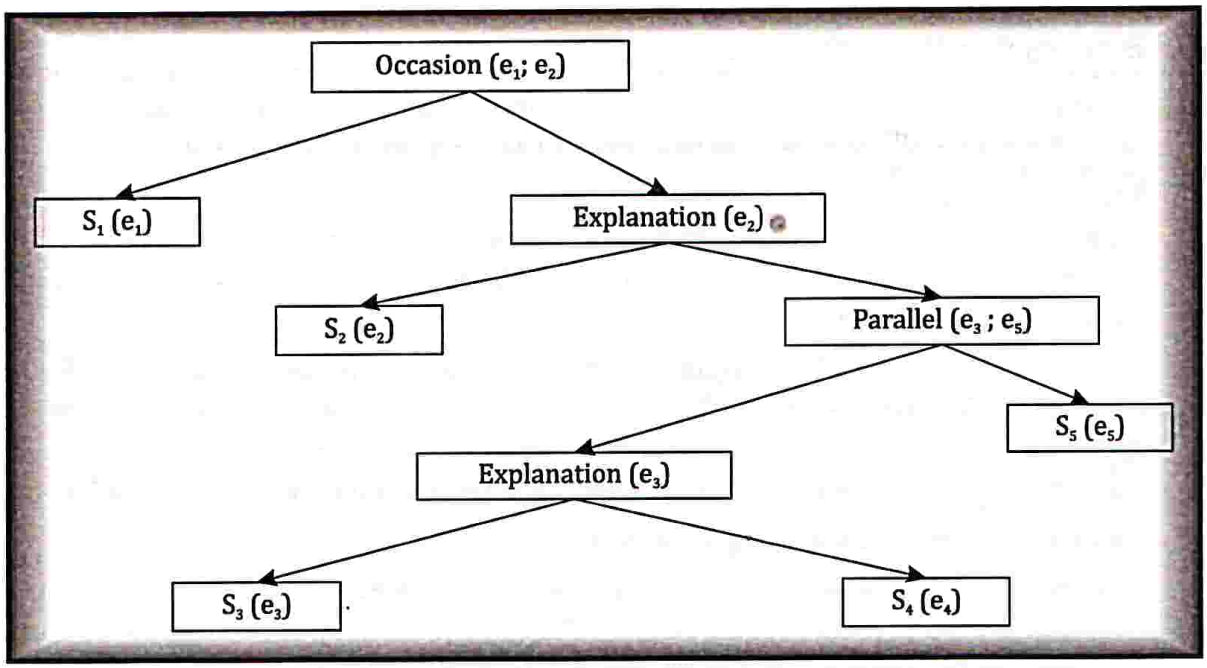
S1 – Ram went to the bank to deposit money.

S2 – He then took a train to Shyam's cloth shop.

S3 – He wanted to buy some clothes.

S4 – He do not have new clothes for party.

S5 – He also wanted to talk to Shyam regarding his health



5. Discourse Integration in NLP Applications

- Discourse Integration has several practical applications in NLP. One of the most important applications is in Sentiment Analysis. Sentiment Analysis is the process of analysing the emotional tone of a text. By integrating discourse information, sentiment analysis algorithms can better understand the underlying sentiment of a text. For example, in the sentence "The product is good, but the customer service is terrible," the use of the word "but" signals a change in sentiment. A sentiment analysis algorithm that can identify such changes can provide more accurate results.

- Discourse Integration is also important in Question Answering Systems. Question Answering Systems are designed to answer natural language questions. By integrating discourse information, these systems can better understand the context of a question and provide more accurate answers. For example, in the question "What is the capital of France?" the use of the word "France" signals that the answer should be a geographical location.

8.2.5 Pragmatic Analysis

1. The fifth and final phase of NLP is pragmatic analysis. The overall communicative and social content, as well as its impact on interpretation, are the focus of pragmatic analysis.
2. Pragmatic Analysis uses a set of rules that describe cooperative dialogues to help you find the intended result. It covers things like word repetition, who said what to whom, and so on. It comprehends how people communicate with one another, the context in which they converse, and a variety of other factors. It refers to the process of abstracting or extracting the meaning of a situation's use of language. It translates the given text using the knowledge gathered in the preceding stages. "Switch on the TV" when used in a sentence, is an order or request to switch the TV on.
3. Pragmatic analysis is crucial in NLP because it can help computers understand the meaning of a text beyond the surface level. It can help computers identify sarcasm, irony, metaphors, and other figurative language devices that are commonly used in natural language.

   For example, consider the following sentence: "I am so excited to attend the meeting tomorrow." A computer program that uses only syntactic analysis may interpret this sentence as a statement of fact. However, a program that uses pragmatic analysis can understand that the speaker is expressing their enthusiasm for the meeting.
4. Pragmatic analysis can also help computers understand the meaning of negation. *For example, consider the following sentence: "The movie was not bad."* A computer program that uses only syntactic analysis may interpret this sentence as a positive statement. However, a program that uses pragmatic analysis can understand that the speaker means that the movie was good.
5. **Challenges of Pragmatic Analysis in NLP**
   - Pragmatic analysis is a complex process that involves analysing the context in which a text is used. It is challenging because the context can vary depending on various factors, such as the speaker's intention, the audience, the setting, and the cultural background. Therefore, developing an NLP system that uses pragmatic analysis requires a significant amount of data and a sophisticated analysis technique.
   - Another challenge of pragmatic analysis is that it can be computationally expensive. Pragmatic analysis involves analysing the context of a text, which can be a time-consuming process. Therefore, developing an NLP system that uses pragmatic analysis requires a high-performance computing system that can analyse large amounts of data quickly.

In conclusion, pragmatic analysis is a crucial aspect of NLP that can help overcome some of the limitations of NLP. It enables computers to understand the meaning of a text beyond the surface level,

leading to improved accuracy and better performance of NLP systems. Developing an NLP system that uses pragmatic analysis requires a significant amount of data and a sophisticated analysis technique, but the benefits are well worth the effort. As the amount of data available continues to increase, the need for accurate NLP systems that use pragmatic analysis will only continue to grow.

Summary

- Natural Language Processing (NLP) is a field that combines computer science, linguistics, and machine learning to study how computers and humans communicate in natural language.
- There are two components of NLP- (a) **Natural Language Understanding (NLU)** and **Natural Language Generation (NLG)**
- There are lot of ambiguity while learning or trying to interpret a language. Few of them are : **Lexical Ambiguity, Syntactic Ambiguity, Semantic Ambiguity, Anaphoric Ambiguity, Pragmatic ambiguity**
- **Sentence Segmentation -** is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences.
- **Word Tokenization -** is used to break the sentence into separate words or tokens
- **Stemming** is used to normalize words into its base form or root form.
- **Lemmatization** is quite similar to the Stamming. It is used to group different inflected forms of the word, called Lemma.
- Dependency Parsing is used to find that how all the words in the sentence are related to each other.
- **POS** stands for parts of speech, which includes Noun, verb, adverb, and Adjective. It indicates that how a word functions with its meaning as well as grammatically within the sentences. A word has one or more parts of speech based on the context in which it is used.
- Named Entity Recognition (**NER**) is the process of detecting the named entity such as person name, movie name, organization name, or location.
- **Chunking** is used to collect the individual piece of information and grouping them into bigger pieces of sentences.
- Natural Language Processing is separated into five primary stages or phases- Lexical Analysis, Syntactic Analysis , Semantic Analysis, Disclosure Integration, Pragmatic Analysis.
- **Lexical or Morphological Analysis** is the initial step in NLP. It entails recognizing and analysing word structures. The collection of words and phrases in a language is referred to as the lexicon.
- **Syntactic or Syntax analysis** is a technique for checking grammar, arranging words, and displaying relationships between them. It entails examining the syntax of the words in the phrase and arranging them in a way that demonstrates the relationship between them.
- ***Parse tree*** is defined as the graphical depiction of a derivation. The start symbol of derivation serves as the root of the parse tree. In every parse tree, the leaf nodes are terminals and interior nodes are non-terminals.

- A **context-free grammar** (CFG) consisting of a finite set of grammar rules is a quadruple (N, T, P, S) where
  - N is a set of non-terminal symbols.
  - T is a set of terminals where N ∩ T = NULL.
  - P is a set of rules, P: N → (N ∪ T)*, i.e., the left-hand side of the production rule P does have any right context or left context.
  - S is the start symbol
- **Semantic analysis** is the process of looking for meaning in a statement. It concentrates mostly on the literal meaning of words, phrases, and sentences is the main focus. It also deals with putting words together to form sentences. It extracts the text's exact meaning or dictionary definition. The meaning of the text is examined. It is accomplished by mapping the task domain's syntactic structures and objects.
- **Lexical Semantic Analysis** and **Compositional Semantic Analysis** are the two broad parts of semantic analysis.
- In Natural Language, the meaning of a word may vary as per its usage in sentences and the context of the text. **Word Sense Disambiguation** involves interpreting the meaning of a word based upon the context of its occurrence in a text.
- Another important task involved in Semantic Analysis is **Relationship Extracting.** It involves firstly identifying various entities present in the sentence and then extracting the relationships between those entities.
- Elements of Semantic Analysis are – Hyponymy, Homonymy, Synonymy, Antonymy, Polysemy, Meronomy
- Machines tend to represent the text in specific formats in order to interpret its meaning. This formal structure that is used to understand the meaning of a text is called **meaning representation.**
- Some of the popular approach of meaning representations are : First-order predicate logic (FOPL), Semantic Nets , Frames , Conceptual dependency (CD) , Rule-based architecture, Case Grammar , Conceptual Graphs
- Semantic Analysis techniques - ***Text Classification and Text Extraction***
- The term "**discourse integration**" refers to a feeling of context. The meaning of any sentence is determined by the meaning of the sentence immediately preceding it. In addition, it establishes the meaning of the sentence that follows. The sentences that come before it play a role in discourse integration. That is to say, that statement or word is dependent on the preceding sentence or words. It's the same with the use of proper nouns and pronouns.
- **Lexical repetition** is a way to find the structure in a discourse, but it does not satisfy the requirement of being coherent discourse. To achieve the coherent discourse, we must focus on coherence relations in specific.
- Pragmatic Analysis uses a set of rules that describe cooperative dialogues to help you find the intended result. It covers things like word repetition, who said what to whom, and so on. It comprehends how people communicate with one another, the context in which they converse, and a variety of other factors

8.3 Review Questions

Short Answer Questions

1. Define NLP.
2. Differentiate between NLU and NLG
3. List any four ambiguity in NLP.
4. What do you mean by Lemmatization in NLP?
5. What do you mean by chunking in NLP?
6. List any four advantages of NLP.
7. List any four disadvantages of NLP.
8. List any four applications of NLP.
9. List all the 5 phases of NLP.
10. Define Lexical Analysis.
11. Define Syntax Analysis.
12. Define Parse tree in NLP.
13. Define CFG in NLP.
14. Define Semantic Analysis.
15. List different parts of Semantic Analysis.
16. What do you mean by Word Sense Disambiguation.
17. List any four the elements of Semantic Analysis.
18. What do you mean by Meaning representation?
19. List any four approaches of Meaning representation.
20. What do you mean by discourse integration?
21. Define Lexical repetition in discourse.
22. Define Pragmatic Analysis.

Long Answer Questions

1. Explain the different types ambiguity in NLP.
2. With suitable example, explain how to build a NLP pipeline?
3. Explain any two advantages and two disadvantages of NLP.
4. Explain the applications of NLP.
5. With an example, Explain Lexical or Morphological Analysis.
6. With an example, Explain the construction of CFG & parse tree of Syntax analysis.

7. With an example, Explain Left most and Right most derivation. Construct parse tree.
8. Write a note on Word Sense Disambiguation in semantic analysis
9. With an example, Explain relation extraction in semantic analysis.
10. Briefly explain different elements of semantic analysis.
11. With an example, Explain how to build hierarchical disclosure structure.
12. Explain the challenges of Pragmatic analysis in NLP

UNIT IV

EXPERT SYSTEM

9

- Introduction
- Architecture of Expert System
- Characteristics of an Expert System
- Advantages of Expert System
- Limitations of Expert System
- Applications of Expert System
- Building an Expert System
  - Case-1 : Medical Expert system on Wilson's Disease identification
  - Case-2 : Internet-based expert system
  - Case-3 : Mobile phone-based e-health system
  - Case-4 : Web Based Expert Systems: A Web Engineering Application Category
- Review Questions

9.1 Introduction

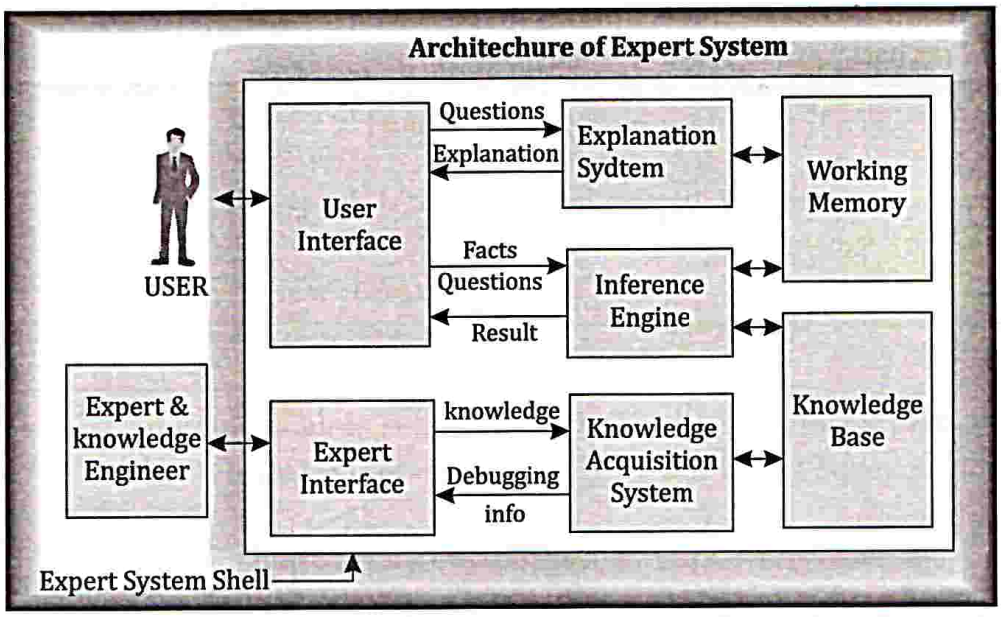
1. An ever increasing demand for communication drives the popularity of AI based data analytic tools. Such AI tools have application in numerous areas such as analytics, medicine, etc. **Expert Systems** are computer programs that are derived from a branch of computer science research called Artificial Intelligence (AI). AI's scientific goal is to understand intelligence by building computer programs that exhibit intelligent behaviour. AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called **knowledge-based or expert systems.**
2. Often, the term expert systems is reserved for programs whose knowledge base contains the knowledge used by human experts, in contrast to knowledge gathered from textbooks or non-experts. The area of human intellectual endeavour to be captured in an expert system is called the task domain. Task refers to some goal-oriented, problem-solving activity. Domain refers to the area within which the task is being performed. Typical tasks are diagnosis, planning, scheduling, configuration and design.
3. **Some of the popular examples of expert systems :**
   - **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.
   - **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.
   - **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.
   - **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.
   - **R1/XCON** – It could select specific software to generate a computer system wished by the user.

9.2 Architecture of Expert System

1. **User Interface:**

   With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.
2. **Explanation Systems:**

   These systems are put into place to supply the information that helps in clarifying the problem domain and the structure. This has multiple use cases not only in the field of expert systems but otherwise as well.



3. Inference Engine:

The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user. With the help of an inference engine, the system extracts the knowledge from the knowledge base.

There are two types of inference engine:

- ***Deterministic Inference engine***: The conclusions drawn from this type of inference engine are assumed to be true. It is based on facts and rules.

  For example , what would be produced when an acid combine with base.

  The inference is sure to be Salt or Water.

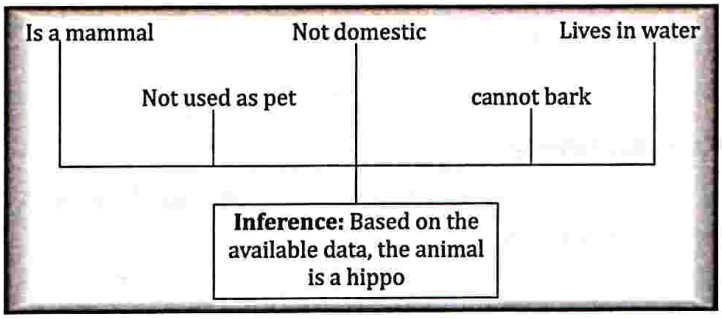
  This is deterministic since it is known fact that acid combine with a base to produce salt and water.

- ***Probabilistic Inference engine***: This type of inference engine contains uncertainty in conclusions, and based on the probability.

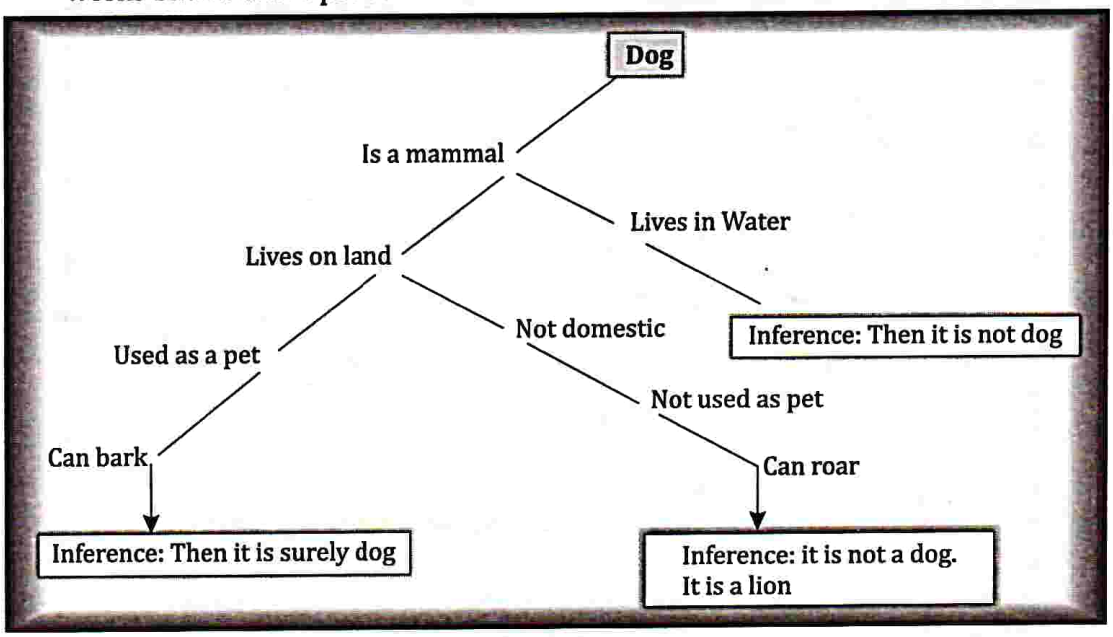
  For example, if there are 70 students in the class talking a mathematics exams, how many of these students will pass exams? We can't give a certain answer.

  Inference engine uses the below modes to derive the solutions:

- ***Forward Chaining***: It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.



- ***Backward Chaining***: It is a backward reasoning method that starts from the goal and works backward to prove the known facts.



4. **Memory Units:**

   It is the storage for the raw data, which is used as an input for models to train and function. The important aspect here is the variety of methodologies and techniques used to store the data for immediate access when required.

5. **Knowledge Base:**

   The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.

   It is similar to a database that contains information and rules of a particular domain or subject. One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Object :
Lion
Attributes :
Is a mammal
Lives on land
Is not domestic
Not used as a pet
Can roar

- **Components of Knowledge Base**
  - Factual Knowledge: The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

- Heuristic Knowledge: This knowledge is based on practice, the ability to guess, evaluation, and experiences.
- Knowledge Representation: It is used to formalize the knowledge stored in the knowledge base using the If-else rules.
- Knowledge Acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

9.3 Characteristics of an Expert System

1. **No memory Limitations**: It can store as much data as required and can memorize it at the time of its application. But for human experts, there are some limitations to memorize all things at every time.
2. **High Efficiency**: If the knowledge base is updated with the correct knowledge, then it provides a highly efficient output, which may not be possible for a human.
3. **Expertise in a domain**: There are lots of human experts in each domain, and they all have different skills, different experiences, and different skills, so it is not easy to get a final output for the query. But if we put the knowledge gained from human experts into the expert system, then it provides an efficient output by mixing all the facts and knowledge
4. **Not affected by emotions**: These systems are not affected by human emotions such as fatigue, anger, depression, anxiety, etc.. Hence the performance remains constant.
5. **High security**: These systems provide high security to resolve any query.
6. **Considers all the facts:** To respond to any query, it checks and considers all the available facts and provides the result accordingly. But it is possible that a human expert may not consider some facts due to any reason.
7. **Regular updates improve the performance:** If there is an issue in the result provided by the expert systems, we can improve the performance of the system by updating the knowledge base.

9.4 Advantages of Expert System

Advantages of Expert System

1. Availability: Expert systems are available easily due to mass production software.
2. Cheaper: The cost of providing expertise is not expensive.
3. Reduced danger: They can be used in any risky environments where humans cannot work with.
4. Permanence: The knowledge will last long indefinitely.
5. Multiple expertise: It can be designed to have knowledge of many experts.
6. Explanation: They are capable of explaining in detail the reasoning that led to a conclusion.

7. Fast response: They can respond at great speed due to the inherent advantages of computers over humans.
8. Unemotional and response at all times: Unlike humans, they do not get tense, fatigue or panic and work steadily during emergency situations.
9. Improved management of uncertainty and im-proved consistency in decisions: In case of uncertainty, more precise results are obtained.

9.5 Limitations of Expert System

Disadvantages of Expert System

1. They are not able to learn from the mistakes.
2. They cannot creatively come with new solutions for the issues.
3. It's not easily achievable to mimic the exact knowledge of an Expert in Computer Programs

9.6 Applications of Expert System

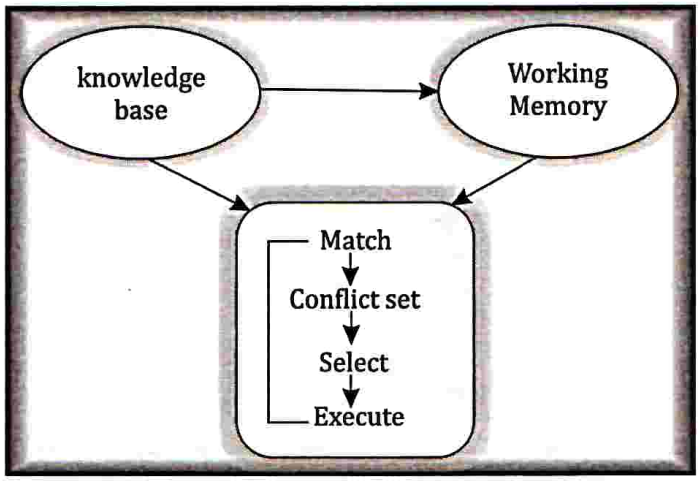
- **In designing and manufacturing domain:**
  It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.
- **In the knowledge domain:**
  These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.
- **In the finance domain**
  In the finance industries, it is used to detect any type of possible fraud, suspicious activity, and advise bankers that if they should provide loans for business or not.
- **In the diagnosis and troubleshooting of devices**
  In medical diagnosis, the ES system is used, and it was the first area where these systems were used.
- **Planning and Scheduling**
  The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

9.7 Building an Expert System

The process that goes into the building of an expert system is as follows:

- First, you need to begin by determining the characteristics of the Expert Systems and the requirements of the problem at hand. Doing this gives you a thorough understanding of the problem.
- The second step is to use the expertise of the domain experts and the knowledge engineers to create and define a structured problem statement.

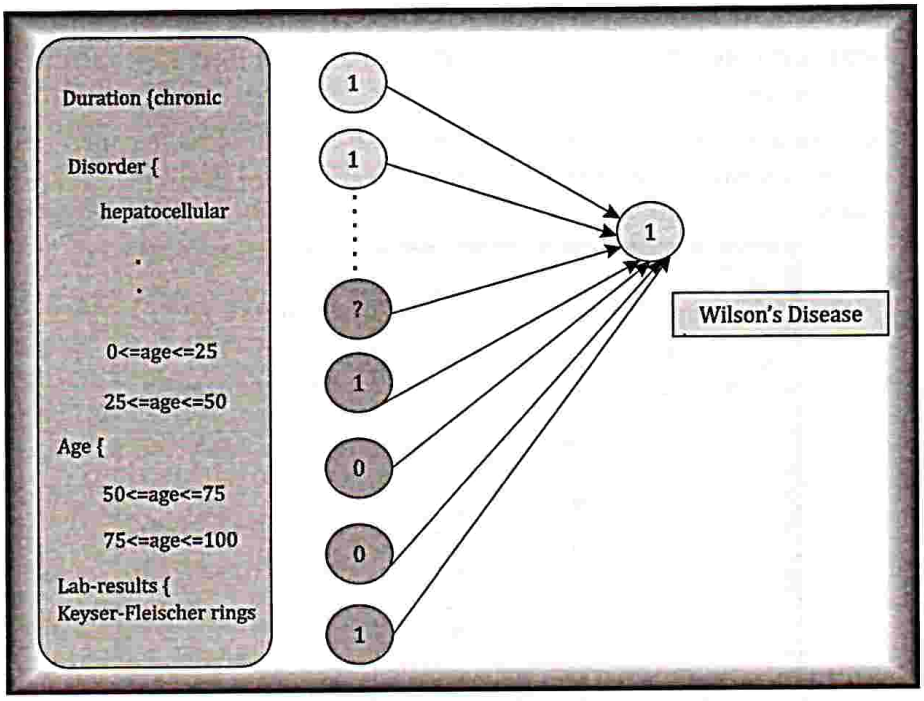
- Next, the knowledge engineers help convert the knowledge packages into high-level, easy-to-understand verbiage. The inference engine and a structured approach to the solution are designed at this point.
- Lastly, the knowledge experts will ensure how these raw knowledge sources can be integrated with the reasoning system, and they assess the segregation of the explanations that would help solve the problem.



9.7.1 Case-1 : Medical Expert system on Wilson's Disease identification

- Wilson's disease (WD) is a rare autosomal recessive disorder of hepatocellular copper deposition. The diagnostic approach to patients with WD may be challenging and is based on a complex set of clinical findings that derive from patient history, physical examination, as well as laboratory and imaging testing. No single examination can unequivocally confirm or exclude the disease. Timely identification of signs and symptoms using novel biomarkers and modern diagnostic tools may help to reduce treatment delays and improve patient prognosis.
- The knowledgebase of this expert system consist of various IF–THEN rules that are related with diagnosing illnesses to achieve the more appropriate treatment and this system can be implemented as a three layered-neural network.
- The neural network consists of cells that correspond to symptoms in the input layer, the diseases are presented in the intermediate or hidden layer and neural cells (nodes) for the treatments defined in the output layer. Training patterns consist of 0's (lack of knowledge about presence or absence of the disease), 1's(presence of the disease), and -1's (absence of the disease).

IF
duration (x, chronic) **and**
disorder (x, hepatocellular) **and**
age (x, <25) **and**
lab-results (x, Kayser–Fleischer rings)
THEN
diagnostic (x, Wilson's disease)



The general topology for this connectionist ES consists of a two-layer neural network and is shown in Figure . In this structure, the diagnosis of the related disease is activated only certain symptomatic situations are identified as true ("1").At the same time, this system is trained and tested by one of the supervised learning algorithm such as ***backpropagation algorithm.***

However, we should note that not all expert system problems are suitable for a neural network approach. The most suitable problems are those that seek to classify inputs into a small number of groups.

9.7.2 Case-2 : Internet-based Expert System

Computer-assisted advisement programs

This case describes an Internet-based expert system found at www.MyMajors.com, which provides advice to high school students or college freshmen who are seeking assistance in selecting a potential major.

- **The Expert System:**

  MyMajors conducts a 15–20 min interview with the student in a manner that emulates the process some academic advisors would use in working with a student to identify an appropriate major. The key difference, in this respect, from other, more psychological, aptitude tests is that the system begins by focusing on the students' experiences with high school courses (what they took or did not take, what they did well in or did not do well in, and what they enjoyed

- **Knowledge Acquisition:**

  To acquire knowledge the author worked closely with university coordinators of academic advisement who counselled incoming and enrolled undeclared majors. These individuals have training in counselling, have on-the-job experience, and have been exposed to the experiences of advisors and other counsellors. The author interviewed many advisors in colleges and

departments who have expertise regarding specific majors and in relation to who succeeds and fails in these curriculums.

- **Evaluating the Expert System:**

  There are two primary measures of effectiveness that can be used to assess expert systems: verification and validation. Verification of an expert system is conducted by determining whether the rules being used by the experts are consistent with the software's implementation of those rules. Verification is difficult since few firm rules exist in this form of counselling.

- **Potential Limitations:**

  Students using the system are advised to avoid using the expert system as their sole source of advice. An important decision pertaining to a student's future should involve parents and school personnel. Even though assistance from others is valuable, many of the potential users of the system are without any advising services.

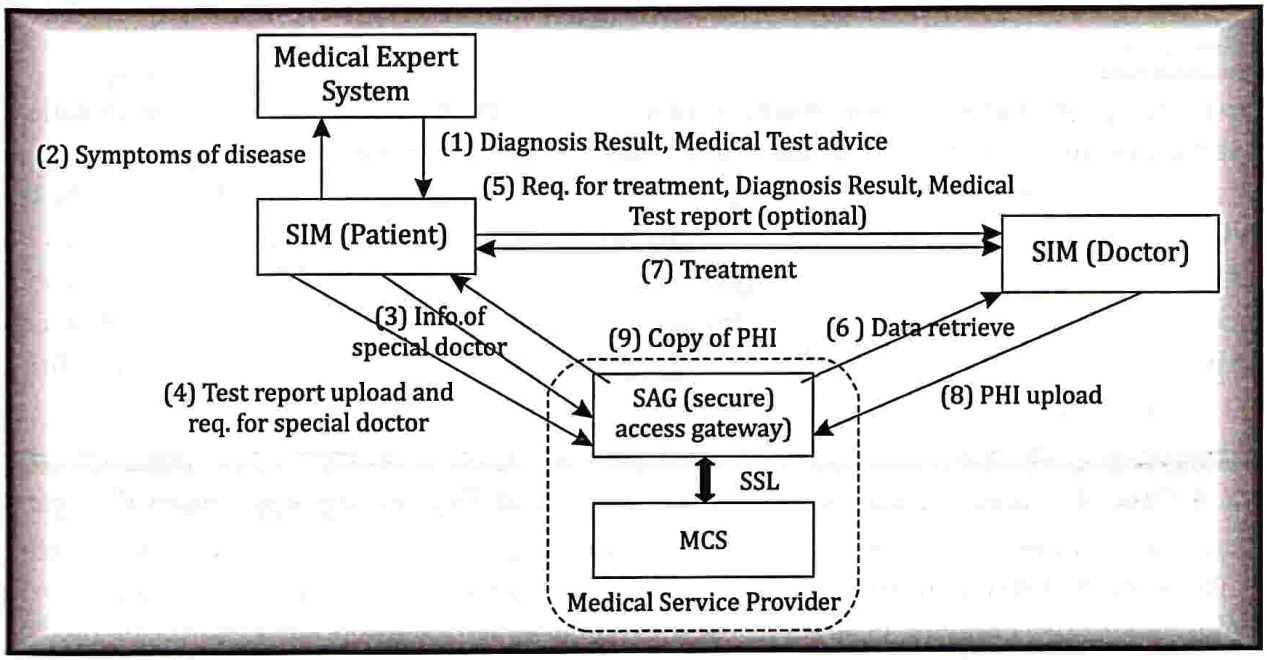
Conclusions

The process of advising students is an uncertain one. Despite this uncertainty, students must select majors. A system such as MyMajors can effectively recommend majors for a student at a level close to that of counsellors given the same exposure to the student's background. Since an advisor may still be needed to provide follow-up counselling, the expert system offers an effective means of collecting base information from the student and of giving the student options to consider before meeting...

9.7.3 Case-3 : Mobile Phone-based e-health System

The mobile phone-based e-health system is proposed to provide an end-to-end secure communication between the medical service provider and patients using mobile phone.

The mobile phone-based e-health system is proposed to provide an end-to-end secure communication between the medical service provider and patients using mobile phone.



The workflow of this system architecture is as follows:

Step 1: Initially, a patient provides his disease symptoms to the medical expert system. The expert system performs an investigation on the patient's conditions and may ask for more symptoms of disease.

Step 2: The medical expert system diagnoses the patient's symptoms and provides the diagnostic result and medical test advice to the patient.

Step 3: Patient completes the medical test and sends an SMS request to MCS for a specialised doctor and to upload the diagnostic result and medical test report.

Step 4: On behalf of MCS, SAG performs the mutual authentication, allows the patient to upload diagnostic result and medical test report, searches the information of a doctor with the patient's specified specialisation from the MCS and sends the identity and mobile no of the specialised doctor to the patient.

Step 5: The patient sends treatments request to the specialised doctor chosen by the MCS and may send his diagnostic result provided by the medical expert system and the medical test report through SMS to the specialised doctor on his requirement.

Step 6: The doctor retrieves the diagnostic result and medical test report from the MCS (or sent by the patient) and starts treatment.

Step 7: The patient consults with the specialised doctor and responses on doctor's query through mobile phone conversation.

Step 8: After completion of treatment, the doctor generates the patient's PHI and uploads it to MCS.

Step 9: MCS stores the PHI and sends a copy of the uploaded PHI to the patient.

Step 10: In future treatment of the particular patient, Steps 1–9 will be followed and the doctor may retrieve the patient's previous PHI stored at MCS

Conclusion

The mobile phone based solution in e-health system is to provide quick health services to patients especially residing in remote locations. Both the doctors and patients are benefited in the proposed scheme as the former saves precious time not by consulting physically and the later gets the best treatment from the specialised doctors without visiting to the doctor. The access of patient's PHI data from MCS is protected and controlled through cryptographic authentication, encryption, nonrepudiation etc. The best quality treatment in less time for the patients in remote locations and the use of mobile phone and MCS are strongly supported in the proposed scheme. It is more efficient and able to preserve the regulations of the HIPAA.

9.7.4 Case-4 : Web Based Expert Systems: A Web Engineering Application Category

Many expert systems have been developed since the mid of 1960's and significant experience is available on their development, which unfortunately it is not enough, when it comes to develop a Web based expert system. In that case, it is essential to have appropriate engineering methods

and processes in order to design and develop a Web application that is something more than just a classical expert system. Some attributes that distinguish Web based expert system development process from traditional expert systems are the growth of their requirements, because of the rapid Web evolution, and the need of continuous upgrade and change of their knowledge base and their information content respectively. For that reason Web based expert system development processes may necessitate the use of several scientific principals like requirements engineering, knowledge engineering, expert system programming, Wed programming, graphic and database design, network security etc.

Web engineering is a new discipline, which recognizes that Web applications, during its developing process, needs the extra processes mentioned above. Therefore, Web engineering attempts to model the knowledge and experience which gained from the development of complex Web systems, in order to deliver processes that outline the various activities and steps of Web systems development. Analogous processes must be defined for Web based expert systems development. However, this did not mentioned clearly by the Web engineering experts, which are mainly focused to applications that do not utilize expert system technology. In this case, the new defined processes -LOMA Web based expert system is considered which are well analysed processes of traditional expert systems development.

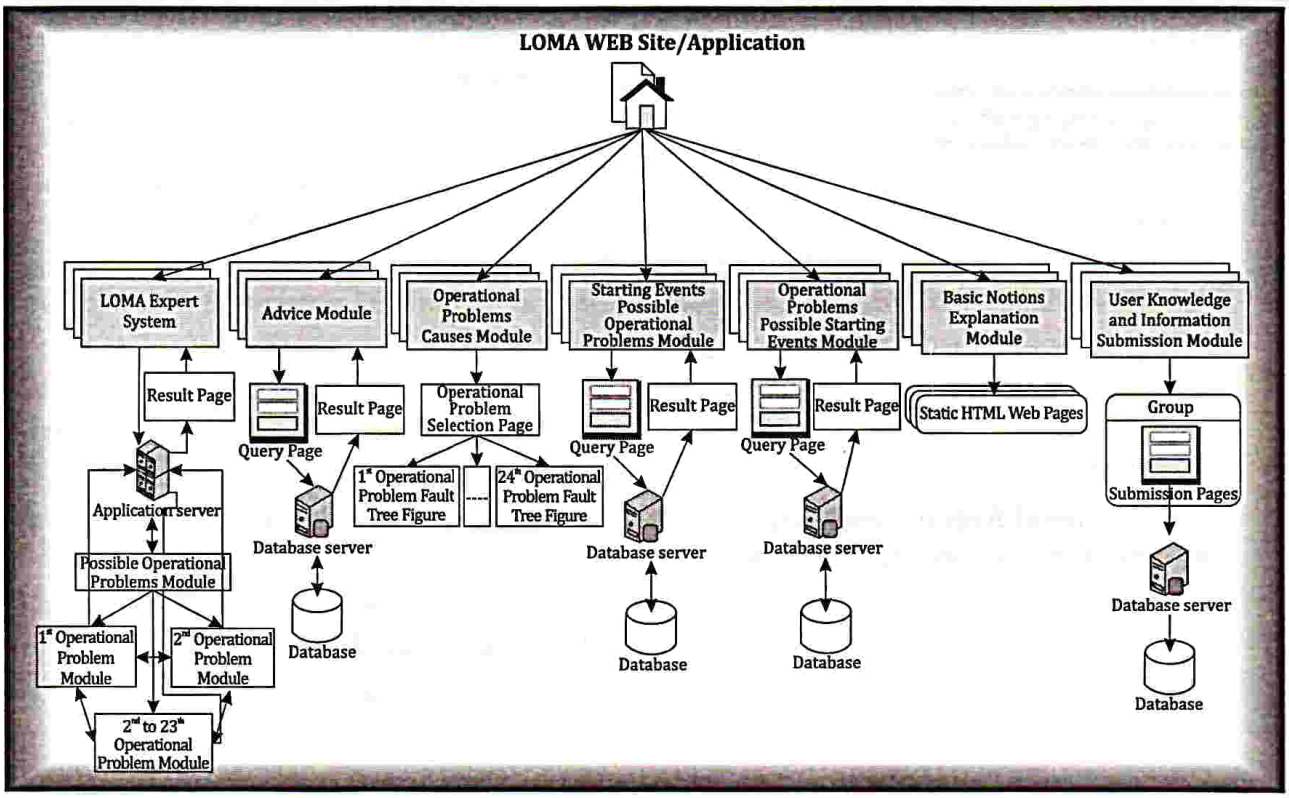
Functional Requirements

- ***1st Functional Requirement***: The Web application must provide the means that will allow to users to make queries regarding the direct extraction of a specific problem advice, without activating the Web based expert system, which at the beginning asks the user to describe the landfill working conditions, then estimates the operation problem occurrence possibility and afterwards provides the corresponding advises solutions.
- ***2nd Functional Requirement***: Users would like to know and understand the reasoning of the expert system and based on which formula it calculates the operational problem occurrence possibility.
- ***3rd Functional Requirement***: Especially landfill managers must have access to information that can provide answers to the following questions:
  1. If during the operation of a landfill a specific event (e.g. strong wind, burning loads, heavy rain, etc.) occurs, which problems/accidents can be triggered?
  2. By which event or combination of events one operational problem/accident can be triggered?
- ***4th Functional Requirement***: Users would appreciate the fact that the Web application is providing information regarding the:
  1. Knowledge acquisition process.
  2. Tools that were used.
  3. Basic notions of the Web application (e.g. what is expert system, knowledge acquisition, etc.).

- ***5th Functional Requirement***: The Web application must provide the means that will allow to users to make comments and to submit their proposals and experiences. Specifically, domain experts proposed to have the opportunity to describe how an operation problem/accident occurred in a landfill and based to their description the developer of the system to be able to update the knowledge base of the Web expert/knowledge based system.

The architecture of LOMA's web site/application is displayed in Figure. It consists of the following modules:

1. LOMA expert system.
2. Advice module.
3. Operational problems causes module.
4. Starting events – possible operational problems module.
5. Operational problems – possible starting events module.
6. Basic notions explanation module.
7. User knowledge and information submission module.



1. **LOMA Expert System**: It is a module of the Web expert/knowledge based system.

   LOMA expert system consists of 24 modules. Each one can analyse a specific operational problem. Moreover, LOMA consists of one extra module. This extra module is activated when the expert system is triggered. This extra module asks the user to describe what is happening in a landfill and based on his description displays a list of possible to occur landfill operational problems. The user can choose one operational problem for further analysis. When the user

makes his selection the first module activates the corresponding to the selected operational problem, among the 24 alternatives, module. This module can estimate the occurrence possibility of the operational problem and can provide advice for the problem prevention or for the minimization of its consequences. This module is the main application of the Web knowledge based system.

2. **Advice Module**: Using this module the user can view the acquired/stored advice, which are related to a specific landfill operation problem/accident. This module is very useful in the case where a landfill operation problem/accident has occurred and the landfill manager wants to lessen the impact of its consequences. This module consists of a database, a query Web page and a query result Web page. ***This module satisfies the 1st functional requirement*** .
3. **Operational Problems Causes Module**: This module uses graphical representations to describe to users how landfill operational problems causes are related with each other and with the corresponding operational problems. Thus, users can understand the reasoning of LOMA expert system. Moreover, explains how a specific operational problem can act as a cause to another operational problem. The module consists of 24 image files (i.e. jpeg files). Each image file display a fault tree, which provides a graphical description of the relation between a landfill operational problem (a top event) and its possible causes (basic events) via AND – OR logic gates. ***This module satisfies the 2nd functional requirement.***
4. **Starting Events** – Possible Operational Problems Module: This module informs the user of the operational problem that can be triggered if a specific event (e.g. heavy rain, landfill compactor out of duty, incoming burning loads etc.) occur during landfill operations. These events are called starting events. One starting event can trigger more than one operational problems/ accidents. This module consists of a database, a query Web page and a query result Web page. ***This module satisfies the 3rd functional requirement*** .
5. **Operational Problems** - Possible Starting Events Module: One operational problem can be triggered by several starting events. This module provides answers to the following question: By which starting events a specific landfill operational problem can be triggered? It consists of a database, a query Web page and a query result Web page. ***This module satisfies the 3rd functional requirement***
6. **Basic Notions Explanation Module**: The main goal of this module is to explain and inform any Web user about the basic notions of LOMA Web system/application (eg. expert systems, knowledge acquisition, fault tree analysis etc) and its developing processes/steps. It consists of static Web pages since its stored information is not expected to change often. ***This module satisfies the 4th functional requirement.***
7. **User Knowledge and Information Submission Module**: This module provides the means by which Web users can submit their knowledge and useful information to the Web system developer so that he can update LOMA's expert system. ***This module satisfies the 5^{h} functional requirement.***

Conclusion:

The Web based expert systems LOMA can be considered as a Web engineering application category that has been developed, is a proof of the development of complex Web systems, in order to deliver processes that outline the various activities and steps of Web systems development.

Summary

- **Expert Systems** are computer programs that are derived from a branch of computer science research called Artificial Intelligence (AI). AI's scientific goal is to understand intelligence by building computer programs that exhibit intelligent behaviour.
- AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called **knowledge-based or expert systems.**
- **Some of the popular examples of expert systems are :** DENDRAL, MYCIN, PXDES, CaDeT, R1/XCON
- The three major components of Expert systems are : User Interface, Inference Engine, Knowledge base.
- **User Interface** : it is an interface that helps a non-expert user to communicate with the expert system to find a solution
- **Explanation System** : These systems are put into place to supply the information that helps in clarifying the problem domain and the structure
- The **inference engine** is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.
- The two types of inference engine are : **Deterministic Inference engine and Probabilistic Inference engine.**
- **Deterministic Inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on facts and rules
- **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.
- The two ways to derive solution from inference engine are : **Forward chaining and Backward chaining.**
- **Forward Chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- **Backward Chaining:** It is a backward reasoning method that starts from the goal and works backward to prove the known facts
- **Memory units :** It is the storage for the raw data, which is used as an input for models to train and function.
- The **knowledgebase** is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.
- **Factual Knowledge** and **Heuristic Knowledge** are two components of knowledge base.
- **Knowledge Representation:** It is used to formalize the knowledge stored in the knowledge base using the If-else rules

- **Knowledge Acquisitions:** It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.
- **Characteristics of an Expert System :** (a) No memory Limitations (b) High Efficiency (c) Expertise in a domain (d) Not affected by emotions (e) Considers all the facts (f) Regular updates improve the performance
- **Advantages of Expert System :** (a) Availability (b) Cheaper (c) Reduced danger (d) Permanence (e)Multiple expertise (f) Explanation (g) Fast response (h) Unemotional and response at all times (i) Improved management of uncertainty and im-proved consistency in decisions
- **Limitations of Expert System :**
  - They are not able to learn from the mistakes.
  - They cannot creatively come with new solutions for the issues.
  - It's not easily achievable to mimic the exact knowledge of an Expert in Computer Programs
- Applications of Expert System
  - In designing and manufacturing domain
  - In the knowledge domain
  - In the finance domain
  - In the diagnosis and troubleshooting of devices
  - Planning and Scheduling

9.8 Review Questions

Short Answer Questions

1. Define Expert system.
2. List any four popular examples of Expert systems.
3. Define Inference engine.
4. Explain the two types of inference engine.
5. Explain the two modes to derive solution in inference engine.
6. What do you mean by Knowledge base?
7. List any two characteristics of Expert system.
8. List any two advantages of Expert system.
9. List any two limitations of expert system.
10. List any two applications of expert system.

Long Answer Questions

1. With neat diagram explain the architecture of Expert system.
2. With neat diagram differentiate between Forward chaining and Backward chaining in Inference engine.
3. Explain any 5 characteristics of Expert system.
4. Explain any 5 applications of Expert system.
5. With neat diagram, briefly explain Medical Expert system on Wilson's Disease identification.
6. With neat diagram, briefly explain mobile phone-based e-health system
7. Explain LOMA expert system.
8. Explain internet based expert system.
9. With any one of the case study, Explain expert system in detail.

UNIT - IV

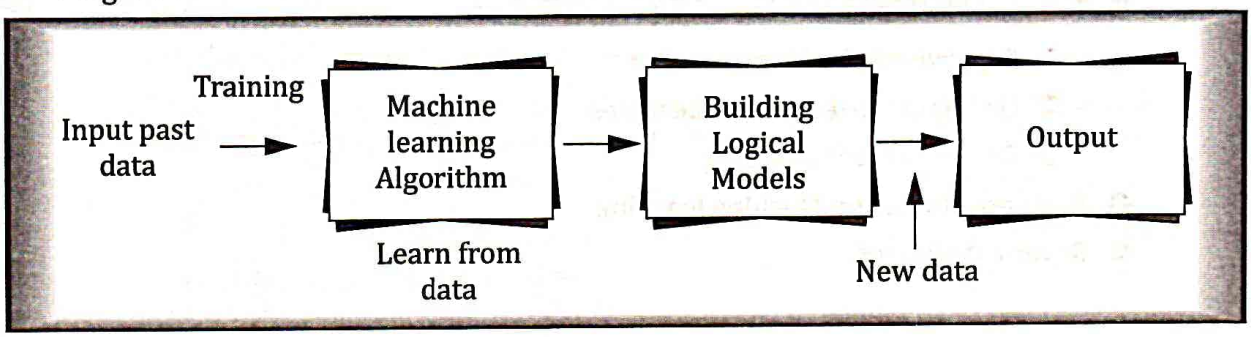
CHAPTER 10

MACHINE LEARNING

- Introduction
- Difference between Machine Learning and Traditional Programming
- Machine Learning Life Cycle
- Types of Machine Learning
  - Supervised Machine Learning
  - Un Supervised Machine Learning:
  - Reinforcement Learning
- Few case studies on Machine learning
- Review Questions

10.1 Introduction

- From translation apps to autonomous vehicles, all powers with Machine Learning. It offers a way to solve problems and answer complex questions. It is basically a process of training a piece of software called an algorithm or model, to make useful predictions from data.
- In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of **Machine Learning.**
- A subset of artificial intelligence known as **machine learning** focuses primarily on the creation of algorithms that enable a computer to independently learn from data and previous experiences. ***Arthur Samuel*** first used the term "machine learning" in 1959.
- Let's say we have a complex problem in which we need to make predictions. Instead of writing code, we just need to feed the data to generic algorithms, which build the logic based on the data and predict the output. Our perspective on the issue has changed as a result of machine learning. The Machine Learning algorithm's operation is depicted in the following block diagram:



10.2 Difference between Machine Learning and Traditional Programming

Machine Learning	Traditional Programming
Machine Learning is a subset of artificial intelligence(AI) that focus on learning from data to develop an algorithm that can be used to make a prediction.	In traditional programming, rule-based code is written by the developers depending on the problem statements.
Machine Learning uses a data-driven approach, It is typically trained on historical data and then used to make predictions on new data.	Traditional programming is typically rule-based and deterministic. It hasn't self-learning features like Machine Learning and AI.
ML can find patterns and insights in large datasets that might be difficult for humans to discover.	Traditional programming is totally dependent on the intelligence of developers. So, it has very limited capability.

Machine Learning is the subset of AI. And Now it is used in various AI-based tasks like Chatbot Question answering, self-driven car., etc.	Traditional programming is often used to build applications and software systems that have specific functionality.

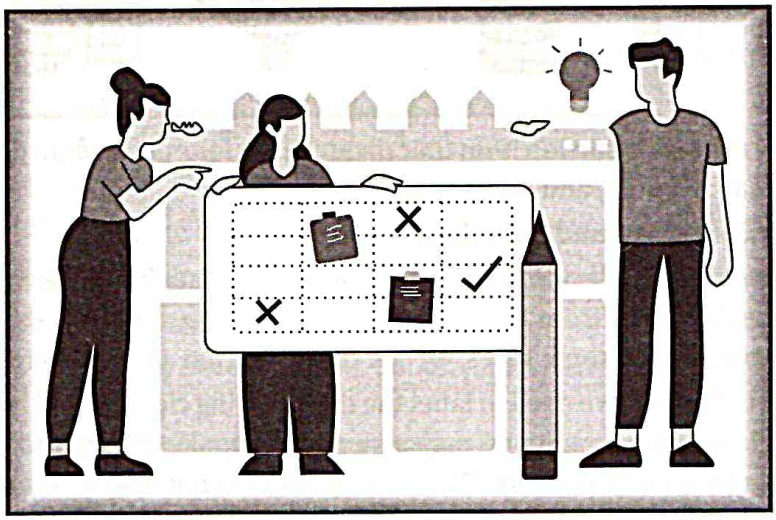
10.3 Machine Learning Life Cycle

The life cycle of a machine learning project involves a series of steps that include:

1. Planning
2. Data Preparation
3. Model Engineering
4. Model Evaluation
5. Model Deployment
6. Monitoring and Maintenance

Each phase in the machine learning cycle follows a quality assurance framework for constant improvement and maintenance by strictly following requirements and constraints.

1. **Planning :**



The planning phase involves assessing the scope, success metric, and feasibility of the ML application. You need to understand the business and how to use machine learning to improve the current process. For example: do we require machine learning? Can we achieve similar requests with simple programming?

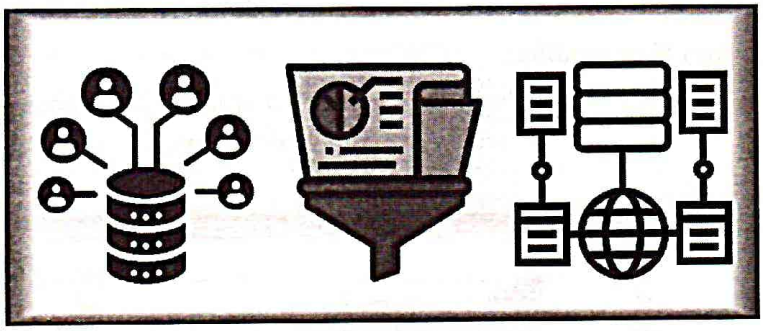
You also need to understand the cost-benefit analysis and how you will ship the solution in multiple phases. Furthermore, you need to define clear and measurable success metrics for business, machine learning models (Accuracy, F1 score, AUC), and economic (key performance indicators).

Finally, you need to create a feasibility report. It will consist of the information about:

- ***Availability of the data***: do we have enough data available to train the model? Can we get a constant supply of new and updated data? Can we use synthetic data to reduce the cost?

- ***Applicability***: will this solution solve the problem or improve the current process? Can we even use machine learning to solve this issue?
- ***Legal constraints***: do we have permission from the local government to implement this solution? Are we following an ethical way of collecting the data? What will be the impact of this application on society?
- ***Robustness and Scalability***: is this application robust enough? Is it scalable?
- ***Explainability:*** can we explain how the machine learning model is coming up with the results? Can we explain the deep neural networks' inner workings?
- ***Availability of resources***: do we have enough computing, storage, network, and human resources? Do we have qualified professionals?

2. Data Preparation



The data preparation section is further divided into four parts: data procurement and labelling, cleaning, management, and processing.

- **Data Collection and Labelling**

  We need first to decide how we will collect the data by gathering the internal data, open-source, buying it from the vendors, or generating synthetic data. Each method has pros and cons, and in some cases, we get the data from all four methodologies.

  After collection, we need to label the data. Buying cleaned and labelled data is not feasible for all companies, and you may also need to make changes to the data selection during the development process. That is why you cannot buy it in bulk and why the data can eventually be useless for the solution.

  The data collection and labelling require most of the company resources: money, time, professionals, subject matter experts, and legal agreements.

- **Data Cleaning**

  Next, we will clean the data by imputing missing values, analysing wrong-labelled data, removing outliers, and reducing the noise. You will create a data pipeline to automate this process and perform data quality verification.

- **Data Processing**

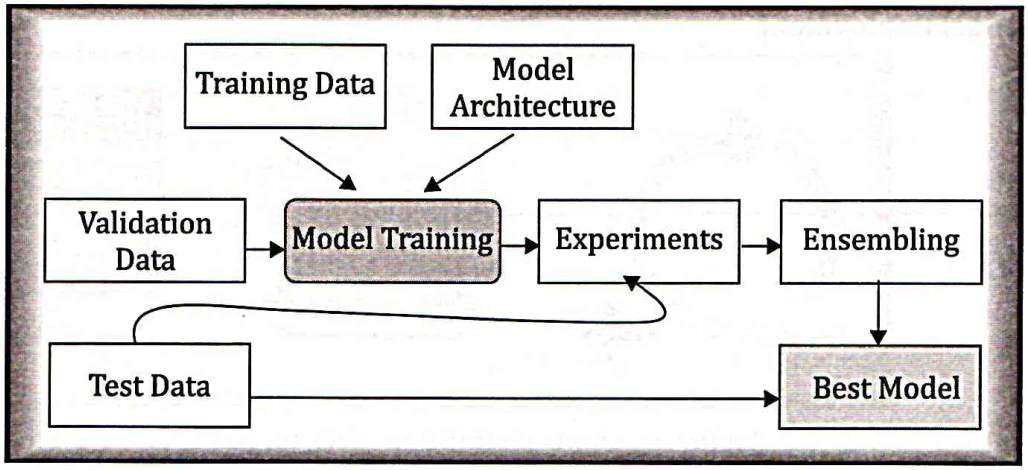
  The data processing stage involves feature selection, dealing with imbalanced classes, feature engineering, data augmentation, and normalizing and scaling the data.

  For reproducibility, we will store and version the metadata, data modelling, transformation pipelines, and feature stores.

- **Data Management**

  Finally, we will figure out data storage solutions, data versioning for reproducibility, storing metadata, and creating ETL pipelines. This part will ensure a constant data stream for model training.

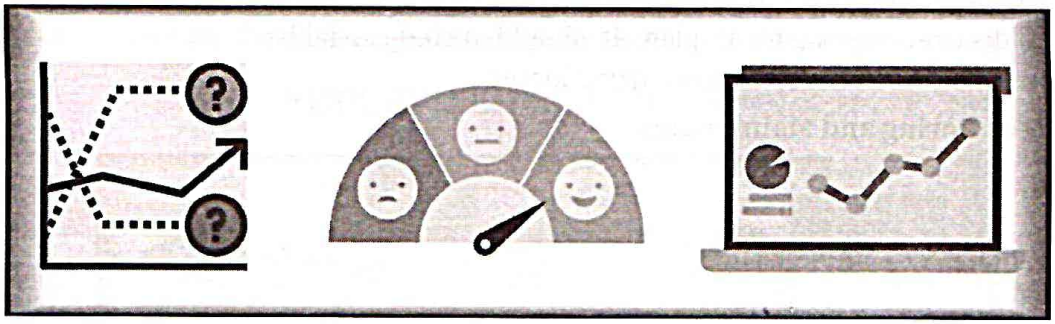
3. Model Engineering



In this phase, we will be using all the information from the planning phase to build and train a machine learning model. For example: tracking model metrics, ensuring scalability and robustness, and optimizing storage and compute resources.

1. Build effective model architecture by doing extensive research.
2. Defining model metrics.
3. Training and validating the model on the training and validation dataset.
4. Tracking experiments, metadata, features, code changes, and machine learning pipelines.
5. Performing model compression and ensembling.
6. Interpreting the results by incorporating domain knowledge experts.

4. Model Evaluation



Now that we have finalized the version of the model, it is time to test various metrics. Why? So that we can ensure that our model is ready for production.

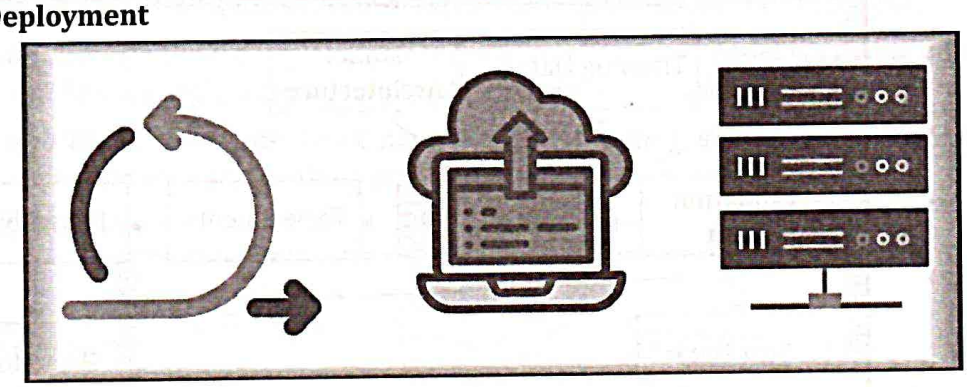
We will first test our model on a test dataset and make sure we involve subject matter experts to identify the error in the predictions.

We also need to ensure that we follow industrial, ethical, and legal frameworks for building AI solutions.

Furthermore, we will test our model for robustness on random and real-world data. Making sure that the model inferences fast enough to bring the value.

Finally, we will compare the results with the planned success metrics and decide on whether to deploy the model or not. In this phase, every process is recorded and versioned to maintain quality and reproducibility.

5. **Model Deployment**



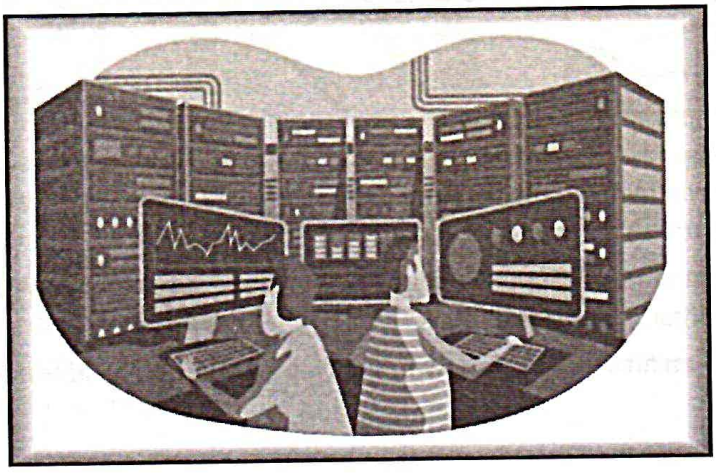
In this phase, we deploy machine learning models to the current system. For example: introducing automatic warehouse labeling using the shape of the product. We will be deploying a computer vision model into the current system, which will use the images from the camera to print the labels.

Generally, the models can be deployed on the cloud and local server, web browser, package as software, and edge device. After that, you can use API, web app, plugins, or dashboard to access the predictions.

In the deployment process, we define the inference hardware. We need to make sure we have enough RAM, storage, and computing power to produce fast results. After that, we will evaluate the model performance in production using A/B testing, ensuring user acceptability.

The deployment strategy is important. You need to make sure that the changes are seamless and that they have improved the user experience. Moreover, a project manager should prepare a disaster management plan. It should include a fallback strategy, constant monitoring, anomaly detection, and minimizing losses.

6. **Monitoring and Maintenance**



After deploying the model to production we need to constantly monitor and improve the system. We will be monitoring model metrics, hardware and software performance, and customer satisfaction.

The monitoring is done completely automatically, and the professionals are notified about the anomalies, reduced model and system performance, and bad customer reviews.

After we get a reduced performance alert, we will assess the issues and try to train the model on new data or make changes to model architectures. It is a continuous process.

In rare cases, we have to revamp the complete machine learning life cycle to improve the data processing and model training techniques, update new software and hardware, and introduce a new framework for continuous integration.

10.4 Types of Machine Learning

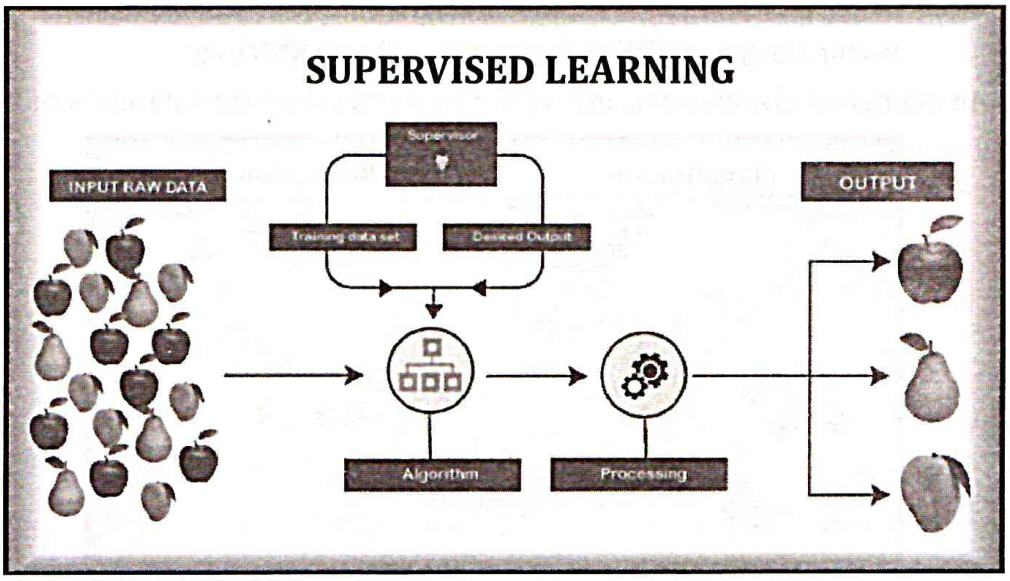
At a broad level, machine learning can be classified into three types:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

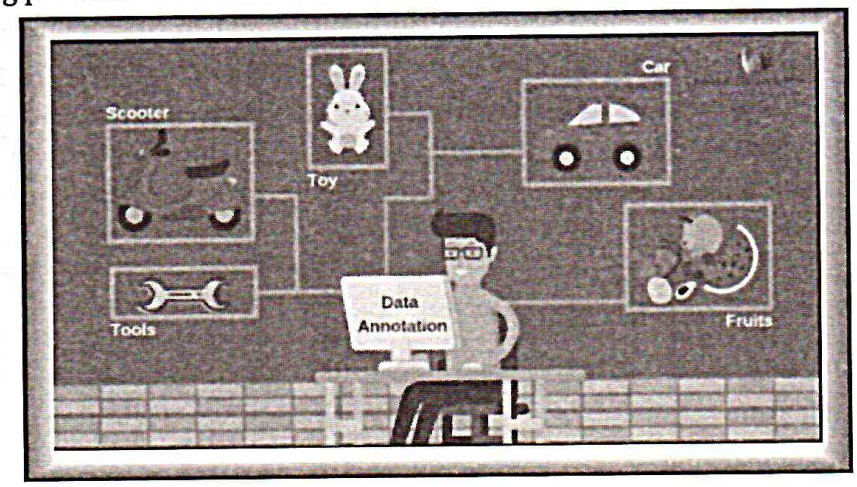
10.4.1. Supervised Machine Learning

Supervised learning is a type of machine learning in which the algorithm is trained on the labelled dataset. It learns to map input features to targets based on labelled training data. In supervised learning, the algorithm is provided with input features and corresponding output labels, and it learns to generalize from this data to make predictions on new, unseen data.

Example 1 :Suppose we have an image of different types of fruits. The task of our supervised learning model is to identify the fruits and classify them accordingly. So to identify the image in supervised learning, we will give the input data as well as output for that, which means we will train the model by the shape, size, color, and taste of each fruit. Once the training is completed, we will test the model by giving the new set of fruit. The model will identify the fruit and predict the output using a suitable algorithm.



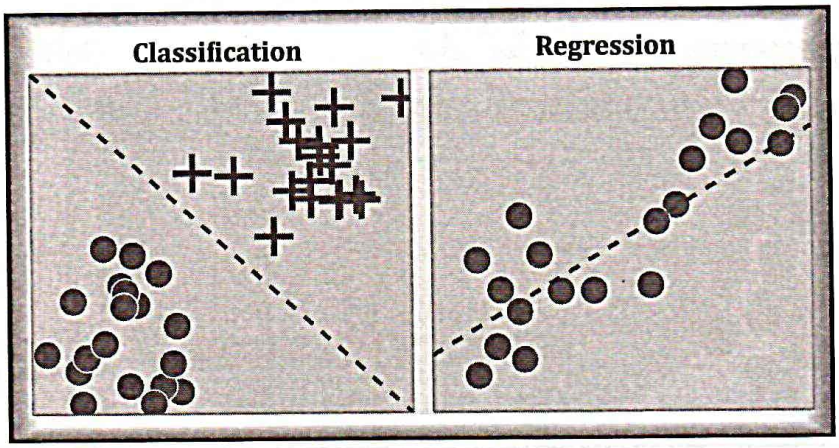
Example 2:Smart Data Labelling with ML- describes the intuition and implementation of a supervised learning model in combination with an active learning algorithm for labelling data. Active learning leverages both manual and automatic labelling to optimise the labelling process.



There are two main types of supervised learning:

- ***Regression:*** Regression is a type of supervised learning where the algorithm learns to predict continuous values based on input features. The output labels in regression are continuous values, such as stock prices, and housing prices. The different regression algorithms in machine learning are: Linear Regression, Polynomial Regression, Ridge Regression, Decision Tree Regression, Random Forest Regression, Support Vector Regression, etc
- ***Classification***: Classification is a type of supervised learning where the algorithm learns to assign input data to a specific category or class based on input features. The output labels in classification are discrete values. Classification algorithms can be binary, where the output is one of two possible classes, or multiclass, where the output can be one of several classes. The different Classification algorithms in machine learning are: Logistic Regression, Naive Bayes, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), etc

An example of classification and regression on two different datasets is shown below:



Advantages & disadvantages of Supervised Learning

Advantages of Supervised Learning

1. Supervised learning permits you to be unmistakable with regards to the meaning of the marks. All in all, you can prepare the calculation to recognize various classes where you can define an ideal choice limit.
2. You can decide the number of classes you need to have.
3. The information in supervised learning is very notable and is marked.
4. The outcomes delivered by the directed strategy are more precise and dependable in contrast with the outcomes created by the solo procedures of AI. This is essentially on the grounds that the information in the regulated calculation is notable and marked. This is a critical distinction between administered and solo learning.
5. The responses in the examination and the result of your calculation are probably going to be known because the relative multitude of classes utilised is known.

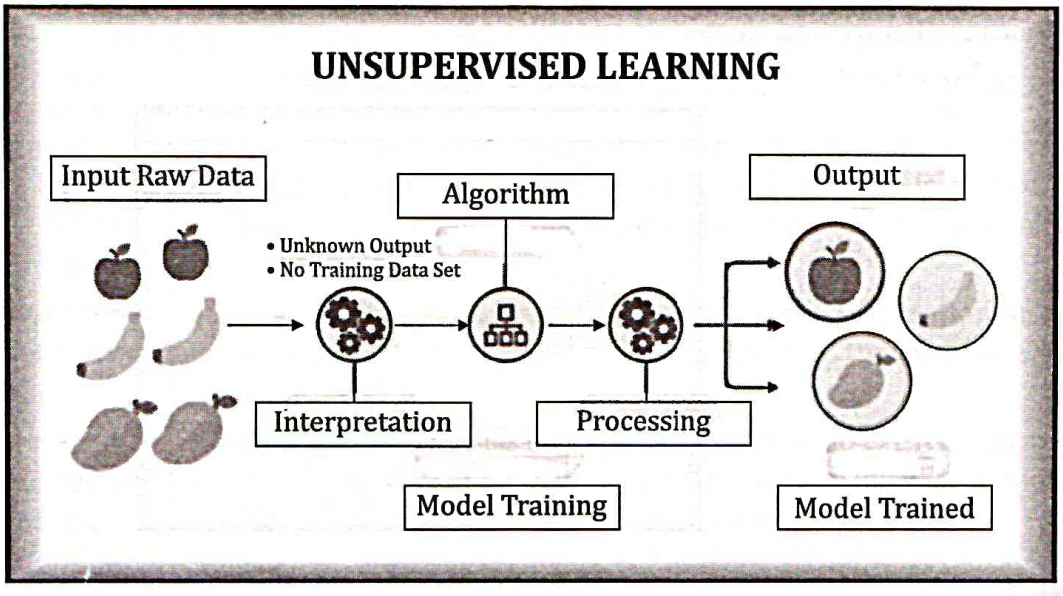
Disadvantages of Supervised Learning

1. Supervised machine learning can be a complicated strategy in examination with the solo technique. The key explanation is that you need to see well overall and mark the contributions to manage learning.
2. It doesn't happen continuously while solo learning is ongoing. This is additionally a significant contrast between supervised and unsupervised learning.
3. It requires a ton of calculation time for preparing.
4. Assuming you have dynamic, large, and developing information, you don't know of the marks to predefine the standards. This can be a genuine test.

10.4.2. Un Supervised Machine Learning

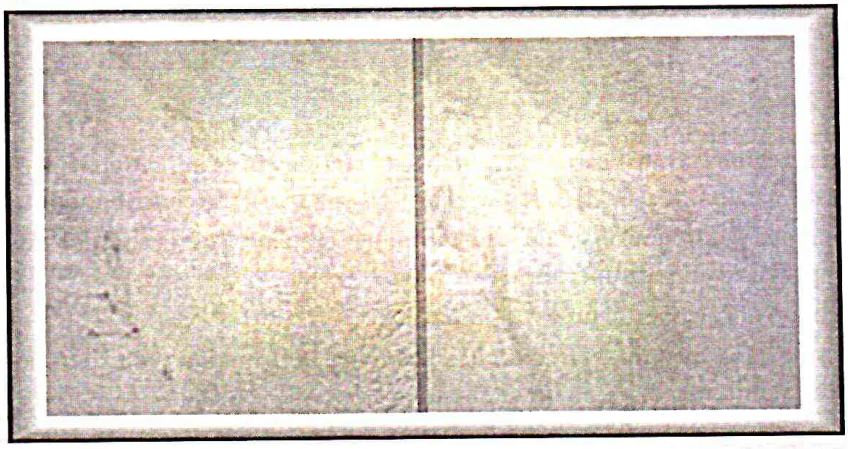
Unsupervised learning is a type of machine learning where the algorithm learns to recognize patterns in data without being explicitly trained using labelled examples. The goal of unsupervised learning is to discover the underlying structure or distribution in the data.

Example 1:



To understand the unsupervised learning, we will use the example given above. So unlike supervised learning, here we will not provide any supervision to the model. We will just provide the input dataset to the model and allow the model to find the patterns from the data. With the help of a suitable algorithm, the model will train itself and divide the fruits into different groups according to the most similar features between them.

Example 2: Using Unsupervised Learning on Satellite Images to Identify Climate Anomalies - Somalia is a small country in the continent of Africa. The country exhibits a lot of natural disasters and terrorism as a result of which people of Somalia go through mass displacements leading towards a situation of lack of food and shelter. This example gives an idea about an anomaly detection system using Machine Learning. The system is capable of capturing sudden vegetation changes, which can be used as an alert mechanism to provide immediate relief to the people and communities in need.

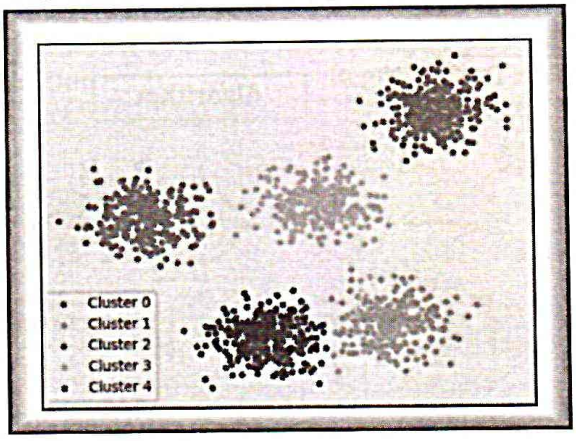


Satellite images to identify Climate Anomalies

There are two main types of unsupervised learning:

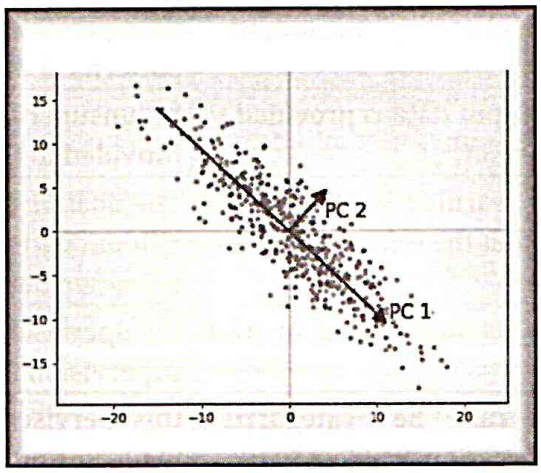
- ***Clustering***: Clustering algorithms group similar data points together based on their characteristics. The goal is to identify groups, or clusters, of data points that are similar to each other, while being distinct from other groups. Some popular clustering algorithms include K-means, Hierarchical clustering, and DBSCAN.

An example for clustering using k-means on spherical data



- ***Dimensionality reduction***: Dimensionality reduction algorithms reduce the number of input variables in a dataset while preserving as much of the original information as possible. This is useful for reducing the complexity of a dataset and making it easier to visualize and analyze. Some popular dimensionality reduction algorithms include Principal Component Analysis (PCA), t-SNE, and Auto encoders.

  Example : PCA guarantees finding the best linear transformation that reduces the number of dimensions with a minimum loss of information



Advantages & disadvantages of Unsupervised Learning

Advantages of Unsupervised Learning

1. Less intricacy in correlation with administered learning. In unsupervised learning, nobody is needed to comprehend and afterward name the information inputs. This makes solo learning less mind boggling and clarifies why many individuals favour unsupervised learning.
2. It is frequently simpler to get unlabelled information – from a PC than marked information, which needs private mediation. This is likewise a critical distinction among managed and solo learning

Disadvantages of Unsupervised Learning

1. You can't get unmistakable with regards to the meaning of the information arranging and the result. This is on the grounds that the information utilized in solo learning is marked and not known. It is the occupation of the machine to name and gather the crude information prior to deciding the secret examples.
2. Less exactness of the outcomes. This is additionally because the info information isn't known and not marked by individuals ahead of time, and that implies that the machine should do this by itself.
3. The consequences of the investigation can't be found out. There is no earlier information in the unaided technique for AI. Moreover, the quantities of classes are likewise not known. It prompts the powerlessness to discover the outcomes created by the examination.

Difference between Supervised Learning and Unsupervised Learning

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labelled data.	Unsupervised learning algorithms are trained using unlabelled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in Classification and Regression problems.	Unsupervised Learning can be classified in Clustering and Associations problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.

10.4.3. Reinforcement Learning

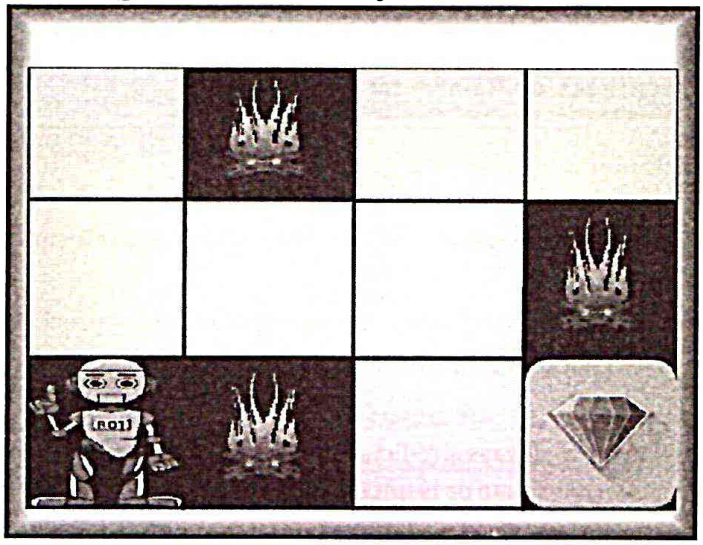
Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

Reinforcement Learning (RL) is the science of decision making. It is about learning the optimal behaviour in an environment to obtain maximum reward. In RL, the data is accumulated from machine learning systems that use a trial-and-error method. Data is not part of the input that we would find in supervised or unsupervised machine learning.

Reinforcement learning uses algorithms that learn from outcomes and decide which action to take next. After each action, the algorithm receives feedback that helps it determine whether the choice it made was correct, neutral or incorrect. It is a good technique to use for automated systems that have to make a lot of small decisions without human guidance.

Reinforcement learning is an autonomous, self-teaching system that essentially learns by trial and error. It performs actions with the aim of maximizing rewards, or in other words, it is learning by doing in order to achieve the best outcomes.

Example: 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.

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.

Types of Reinforcement:

There are two types of Reinforcement:

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

   Advantages of reinforcement learning are:

   - 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 of reinforcement learning:

- Increases Behaviour
- Provide defiance to a minimum standard of performance
- It Only provides enough to meet up the minimum behaviour

Advantages and Disadvantages of Reinforcement Learning

Advantages of Reinforcement Learning

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. Reinforcement learning can handle environments that are non-deterministic, meaning that the outcomes of actions are not always predictable. 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 of Reinforcement Learning

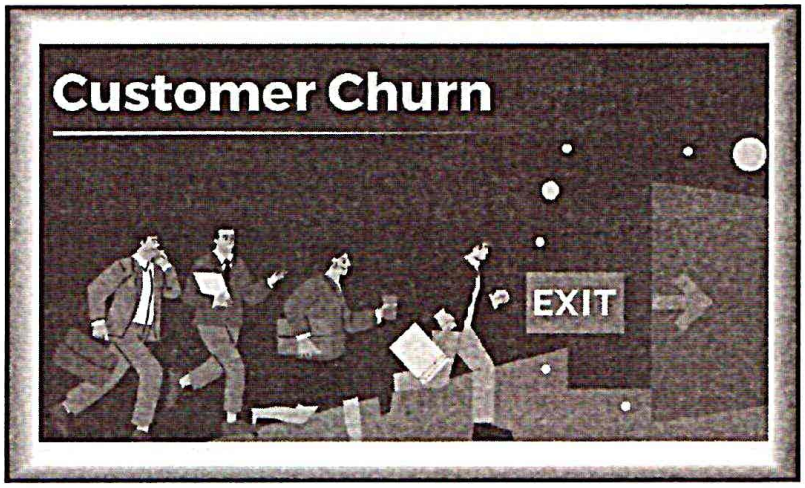
1. Reinforcement learning is not preferable to use for solving simple problems.
2. Reinforcement learning needs a lot of data and a lot of computation
3. Reinforcement learning is highly dependent on the quality of the reward function. If the reward function is poorly designed, the agent may not learn the desired behaviour.
4. Reinforcement learning can be difficult to debug and interpret. It is not always clear why the agent is behaving in a certain way, which can make it difficult to diagnose and fix problems.

Difference between Reinforcement Learning and Supervised Learning

Reinforcement Learning	Supervised Learning
Reinforcement learning is all about making decisions sequentially. In simple words, we can say that the output depends on the state of the current input and the next input depends on the output of the previous input	In Supervised learning, the decision is made on the initial input or the input given at the start
In Reinforcement learning decision is dependent, So we give labels to sequences of dependent decisions	In supervised learning the decisions are independent of each other so labels are given to each decision.
Example: Chess game,text summarization	Example: Object recognition,spam detetction

10.5 Few Case Studies on Machine Learning

- **Customer Churn Prediction**



Predicting customer churn is essential for businesses interested in retaining customers and maximizing their profits. By leveraging historical customer data, machine learning algorithms can identify patterns and factors that are correlated with churn, enabling businesses to take proactive steps to prevent it.

In this case study, you will study how a telecom company uses machine learning for customer churn prediction. The available data contains information about the services each customer signed up for, their contact information, monthly charges, and their demographics. The goal is to first analyse the data at hand with the help of methods used in Exploratory Data Analysis. It will assist in picking a suitable machine-learning algorithm. The five machine learning models used in this case-study are AdaBoost, Gradient Boost, Random Forest, Support Vector Machines, and K-Nearest Neighbours. These models are used to determine which customers are at risk of churn.

By using machine learning for churn prediction, businesses can better understand customer behaviour, identify areas for improvement, and implement targeted retention strategies. It can result in increased customer loyalty, higher revenue, and a better understanding of customer needs and preferences. This case study example will help you understand how machine learning is a valuable tool for any business looking to improve customer retention and stay ahead of the competition.

- **Market Basket Analysis**

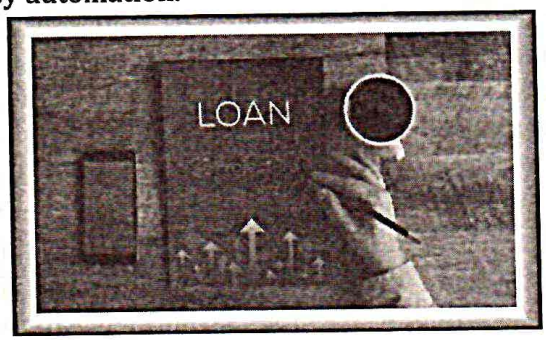
Market basket analysis is a common application of machine learning in retail and e-commerce, where it is used to identify patterns and relationships between products that are frequently purchased together. By leveraging this information, businesses can make informed decisions about product placement, promotions, and pricing strategies.

In this case study, you will utilize the EDA methods to carefully analyse the relationships among different variables in the data. Next, you will study how to use the Apriori algorithm to identify frequent items sets and association rules, which describe the likelihood of a product being purchased given the presence of another product. These rules can generate recommendations, optimize product placement, and increase sales, and they can also be used for customer segmentation.

Using machine learning for market basket analysis allows businesses to understand customer behaviour better, identify cross-selling opportunities, and increase customer satisfaction. It has the potential to result in increased revenue, improved customer loyalty, and a better understanding of customer needs and preferences.

- **Loan Application Classification**

Financial institutions receive tons of requests for lending money by borrowers and making decisions for each request is a crucial task. Manually processing these requests can be a time-consuming and error-prone process, so there is an increasing demand for machine learning to improve this process by automation.



You can work on this Loan Dataset on Kaggle to get started on this one of the most real-world case studies in the financial industry. The dataset contains 614 unique values for 13 columns: Follow the below-mentioned steps to get started on this case study.

1. Analyze the dataset and explore how various factors such as gender, marital status, and employment affect the loan amount and status of the loan application.
2. Select the features to automate the process of classification of loan applications.
3. Apply machine learning models such as logistic regression, decision trees, and random forests to the features and compare their performance using statistical metrics.

This case study falls under the umbrella of supervised learning problems in machine learning and demonstrates how ML models are used to automate tasks in the financial industry.

Summary

- **Machine Learning** is a branch of artificial intelligence that develops algorithms by learning the hidden patterns of the datasets used it to make predictions on new similar type data, without being explicitly programmed for each task.
- 6 series steps in the **life cycle of machine learning** are : (a) Planning (b) Data Preparation (c) Model Engineering (d) Model Evaluation (e) Model Deployment (f) Monitoring and Maintenance

- The **data preparation** section is further divided into four parts: data procurement and labelling, cleaning, management, and processing.
- **Types of Machine Learning**: Supervised learning, Unsupervised learning and Reinforcement learning
- **Supervised Learning** is a type of machine learning in which the algorithm is trained on the labelled dataset. It learns to map input features to targets based on labelled training data.
- The two main types of supervised learning are : Regression and Classification.
- **Regression** is a type of supervised learning where the algorithm learns to predict continuous values based on input features. The output labels in regression are continuous values, such as stock prices, and housing prices.
- The different regression algorithms in machine learning are: Linear Regression, Polynomial Regression, Ridge Regression, Decision Tree Regression, Random Forest Regression, Support Vector Regression, etc
- **Classification** is a type of supervised learning where the algorithm learns to assign input data to a specific category or class based on input features. The output labels in classification are discrete values.
- The different Classification algorithms in machine learning are: Logistic Regression, Naive Bayes, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), etc

- **Advantages of Supervised Learning**
  - Supervised learning allows collecting data and produces data output from previous experiences.
  - Helps to optimize performance criteria with the help of experience.
  - Supervised machine learning helps to solve various types of real-world computation problems.
  - It performs classification and regression tasks.
  - It allows estimating or mapping the result to a new sample.
  - We have complete control over choosing the number of classes we want in the training data.

- **Disadvantages of Supervised Learning**
  - Classifying big data can be challenging.
  - Training for supervised learning needs a lot of computation time. So, it requires a lot of time.
  - Supervised learning cannot handle all complex tasks in Machine Learning.
  - Computation time is vast for supervised learning.
  - It requires a labelled data set.
  - It requires a training process

- **Unsupervised Learning** is a type of machine learning where the algorithm learns to recognize patterns in data without being explicitly trained using labelled examples.

- Two types of unsupervised learning are : Clustering and Dimensionality reduction.
- **Clustering** algorithms group similar data points together based on their characteristics. The goal is to identify groups, or clusters, of data points that are similar to each other, while being distinct from other groups.
- Some popular clustering algorithms include K-means, Hierarchical clustering, and DBSCAN.
- **Dimensionality Reduction** algorithms reduce the number of input variables in a dataset while preserving as much of the original information as possible.
- Some popular dimensionality reduction algorithms include Principal Component Analysis (PCA), t-SNE, and Auto encoders.

- **Advantages of Unsupervised Learning**
  - It does not require training data to be labelled.
  - Dimensionality reduction can be easily accomplished using unsupervised learning.
  - Capable of finding previously unknown patterns in data.
  - Flexibility: Unsupervised learning is flexible in that it can be applied to a wide variety of problems, including clustering, anomaly detection, and association rule mining.
  - Exploration: Unsupervised learning allows for the exploration of data and the discovery of novel and potentially useful patterns that may not be apparent from the outset.
  - Low cost: Unsupervised learning is often less expensive than supervised learning because it doesn't require labelled data, which can be time-consuming and costly to obtain.

- **Disadvantages of Unsupervised Learning**
  - Difficult to measure accuracy or effectiveness due to lack of predefined answers during training.
  - The results often have lesser accuracy.
  - The user needs to spend time interpreting and label the classes which follow that classification.
  - Lack of guidance: Unsupervised learning lacks the guidance and feedback provided by labelled data, which can make it difficult to know whether the discovered patterns are relevant or useful.
  - Sensitivity to data quality: Unsupervised learning can be sensitive to data quality, including missing values, outliers, and noisy data.
  - Scalability: Unsupervised learning can be computationally expensive, particularly for large datasets or complex algorithms, which can limit its scalability.

- **Reinforcement Learning** is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation.

- **Advantages of Reinforcement**
  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. Reinforcement learning can handle environments that are non-deterministic, meaning that the outcomes of actions are not always predictable. 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 of Reinforcement**
  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. Reinforcement learning is highly dependent on the quality of the reward function. If the reward function is poorly designed, the agent may not learn the desired behaviour.
  4. Reinforcement learning can be difficult to debug and interpret. It is not always clear why the agent is behaving in a certain way, which can make it difficult to diagnose and fix problems.

10.6 Review Questions

Short Answer Questions

1. Define Machine learning.
2. State any two difference between machine learning and traditional programming
3. List the stages of machine learning life cycle.
4. List the four parts of data preparation in planning stage of machine life cycle.
5. What do you mean by Model engineering in machine life cycle?
6. List different types of machine learning.
7. Define supervised learning.
8. What do you mean by regression? Name any two regression algorithm.
9. What do you mean by classification? Name any two classification algorithm.
10. List any two advantages of supervised learning.
11. List any two disadvantages of supervised learning.
12. Define unsupervised learning.
13. What do you mean by clustering? Name any two popular clustering algorithm.

14. What do you mean by dimensionality reduction? Name any two popular dimensionality reduction algorithms.
15. List any two advantages of unsupervised learning.
16. List any two disadvantages of unsupervised learning.
17. Define reinforcement learning.
18. List any two advantages of reinforcement learning.
19. List any two disadvantages of reinforcement learning.
20. Differentiate between reinforcement learning and supervised learning.

Long Answer Questions

1. State any four difference between machine learning and traditional programming.
2. Briefly explain machine learning life cycle.
3. Differentiate between supervised and unsupervised learning.
4. With an example, Explain supervised learning.
5. With an example, Explain unsupervised learning.
6. Explain different types of supervised learning.
7. Explain different types of unsupervised learning.
8. List any four advantages and disadvantages of supervised learning.
9. List any four advantages and disadvantages of unsupervised learning.
10. With an example, Explain reinforcement learning.
11. List any four advantages and disadvantages of reinforcement learning.

UNIT - IV

CHAPTER 11

NEURAL NETWORKS

- Introduction
  - What is Artificial Neural Network?
  - Architecture of Artificial Neural Network
  - Types of Artificial Neural Networks
  - Advantages of Artificial Neural Network (ANN)
  - Disadvantages of Artificial Neural Network
  - Applications of Artificial Neural Networks
  - Example of Neural Network in Tensor Flow
- Deep Learning
  - Difference between Deep learning and Machine learning
  - Important Components of a Deep Neural Network
- Deep Learning Algorithms
  - Convolutional Neural Networks (CNN)
  - Recurrent Neural Networks (RNN)
  - Long Short-Term Memory (LSTM)
- Review Questions

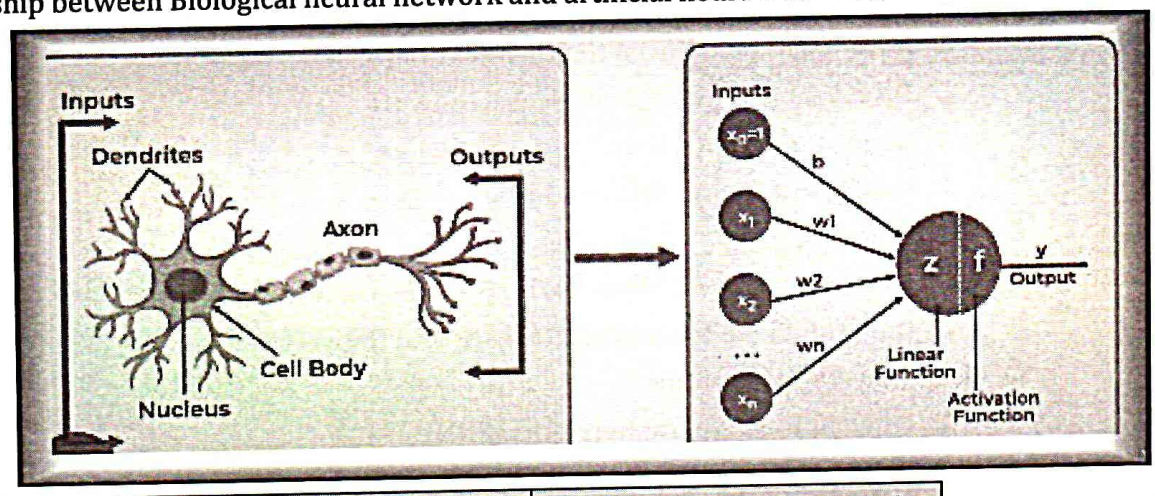
11.1 Introduction

- Neural networks are parallel computing devices, which is basically an attempt to make a computer model of the brain. The main objective is to develop a system to perform various computational tasks faster than the traditional systems. These tasks include pattern recognition and classification, approximation, optimization, and data clustering.
- Neural network is the fusion of artificial intelligence and brain-inspired design that reshapes modern computing. With intricate layers of interconnected artificial neurons, these networks emulate the intricate workings of the human brain, enabling remarkable feats in machine learning.

11.1.1 What is Artificial Neural Network?

An Artificial Neural Network (ANN) is a computer system inspired by biological neural networks for creating artificial brains based on the collection of connected units called artificial neurons. It is designed to analyse and process information as humans. Artificial Neural Network has self-learning capabilities to produce better results as more data is available.

Relationship between Biological neural network and artificial neural network:



Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Cell body	Weights
Axon	Output

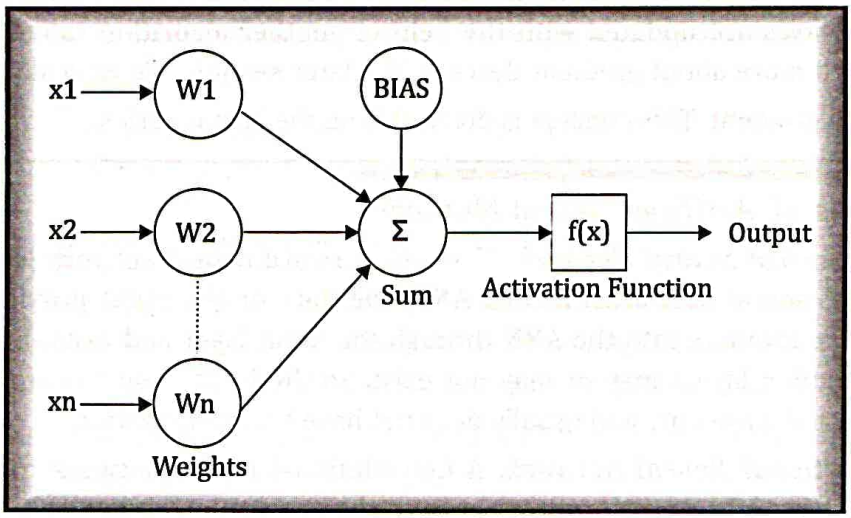
11.1.2 Architecture of Artificial Neural Network :

An Artificial Neural Network (ANN) is composed of four principal objects:

- ***Layers*** - all the learning occurs in the layers. There are 3 layers 1) Input 2) Hidden and 3) Output

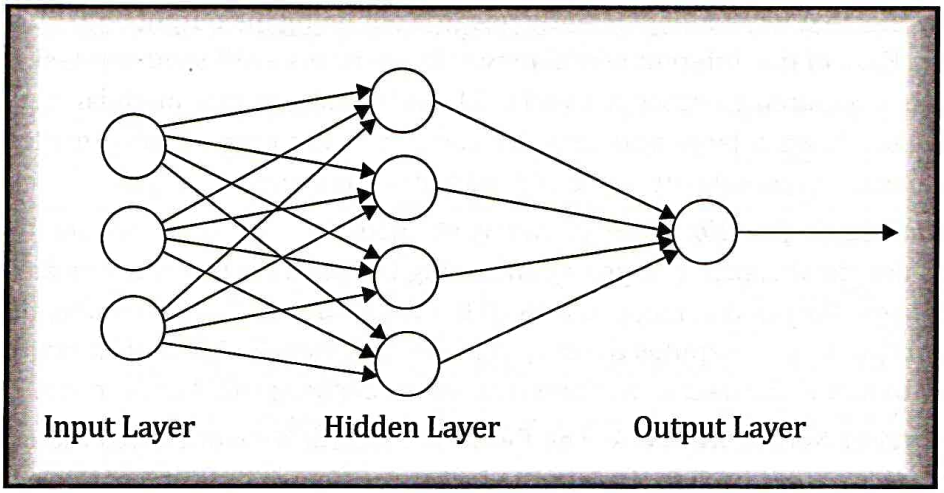
- ***Activation function*** - The activation function of a node defines the output given a set of inputs. You need an activation function to allow the network to learn non-linear pattern.
- ***Loss function*** - The loss function is an important metric to estimate the performance of the optimizer
- ***Optimizer*** - The optimizer will help improve the weights of the network in order to decrease the loss

Suppose we arrange for some automatic means of testing the effectiveness of any current weight assignment in terms of actual performance and provide a mechanism for altering the weight assignment so as to maximize the performance. We need not go into the details of such a procedure to see that it could be made entirely automatic and to see that a machine so programmed would "learn" from its experience.



Explanation:

An artificial neuron can be thought of as a simple or multiple linear regression model with an activation function at the end. A neuron from layer will take the output of all the neurons from the later i-1 as inputs calculate the weighted sum and add bias to it. After this is sent to an activation function as we saw in the previous diagram.



The first neuron from the first layer is connected to all the inputs from the previous layer, Similarly, the second neuron from the first hidden layer will also be connected to all the inputs from the previous layer and so on for all the neurons in the first hidden layer.

For neurons in the second hidden layer (outputs of the previously hidden layer) are considered as inputs and each of these neurons are connected to previous neurons, likewise. This whole process is called Forward propagation.

After this, there is an interesting thing that happens. Once we have predicted the output it is then compared to the actual output. We then calculate the loss and try to minimize it. But how can we minimize this loss? For this, there comes another concept which is known as Back Propagation. We will understand more about this in another article. I will tell you how it works. First, the loss is calculated then weights and biases are adjusted in such a way that they try to minimize the loss. Weights and biases are updated with the help of another algorithm called gradient descent. We will understand more about gradient descent in a later section. We basically move in the direction opposite to the gradient. This concept is derived from the Taylor series.

11.1.3 Types of Artificial Neural Networks

- ***Feed forward Neural Network***: The feed forward neural network is one of the most basic artificial neural networks. In this ANN, the data or the input provided travels in a single direction. It enters into the ANN through the input layer and exits through the output layer while hidden layers may or may not exist. So the feedforward neural network has a front-propagated wave only and usually does not have backpropagation.
- ***Convolutional Neural Network***: A Convolutional neural network has some similarities to the feed-forward neural network, where the connections between units have weights that determine the influence of one unit on another unit. But a CNN has one or more than one convolutional layer that uses a convolution operation on the input and then passes the result obtained in the form of output to the next layer. CNN has applications in speech and image processing which is particularly useful in computer vision.
- ***Modular Neural Network***: A Modular Neural Network contains a collection of different neural networks that work independently towards obtaining the output with no interaction between them. Each of the different neural networks performs a different sub-task by obtaining unique inputs compared to other networks. The advantage of this modular neural network is that it breaks down a large and complex computational process into smaller components, thus decreasing its complexity while still obtaining the required output.
- ***Radial basis function Neural Network***: Radial basis functions are those functions that consider the distance of a point concerning the centre. RBF functions have two layers. In the first layer, the input is mapped into all the Radial basis functions in the hidden layer and then the output layer computes the output in the next step. Radial basis function nets are normally used to model the data that represents any underlying trend or function.
- ***Recurrent Neural Network***: The Recurrent Neural Network saves the output of a layer and feeds this output back to the input to better predict the outcome of the layer. The first layer in

the RNN is quite similar to the feed-forward neural network and the recurrent neural network starts once the output of the first layer is computed. After this layer, each unit will remember some information from the previous step so that it can act as a memory cell in performing computations.

11.1.4 Advantages of Artificial Neural Network (ANN)

Advantages of Artificial Neural Network (ANN)

1. **Parallel Processing Capability:** Artificial neural networks have a numerical value that can perform more than one task simultaneously.
2. **Storing Data on the Entire Network:** Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.
3. **Capability to Work with Incomplete Knowledge:** After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.
4. **Having a Memory Distribution :** For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.
5. **Having Fault Tolerance:** Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance

11.1.5 Disadvantages of Artificial Neural Network

Disadvantages of Artificial Neural Network

1. **Assurance of Proper Network Structure:** There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.
2. **Unrecognized Behaviour of the Network:** It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.
3. **Hardware Dependence:** Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.
4. **Difficulty of Showing the Issue to the Network:** ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.
5. **The Duration of the Network is Unknown:** The network is reduced to a specific value of the error, and this value does not give us optimum results.

11.1.6 Applications of Artificial Neural Networks

- **Social Media:** Artificial Neural Networks are used heavily in Social Media. For example, let's take the 'People you may know' feature on Facebook that suggests people that you might know in real life so that you can send them friend requests. Well, this magical effect is achieved by using Artificial Neural Networks that analyze your profile, your interests, your current friends, and also their friends and various other factors to calculate the people you might potentially know. Another common application of Machine Learning in social media is facial recognition. This is done by finding around 100 reference points on the person's face and then matching them with those already available in the database using convolutional neural networks.
- **Marketing and Sales:** When you log onto E-commerce sites like Amazon and Flipkart, they will recommend your products to buy based on your previous browsing history. Similarly, suppose you love Pasta, then Zomato, Swiggy, etc. will show you restaurant recommendations based on your tastes and previous order history. This is true across all new-age marketing segments like Book sites, Movie services, Hospitality sites, etc. and it is done by implementing personalized marketing. This uses Artificial Neural Networks to identify the customer likes, dislikes, previous shopping history, etc., and then tailor the marketing campaigns accordingly.
- **Healthcare:** Artificial Neural Networks are used in Oncology to train algorithms that can identify cancerous tissue at the microscopic level at the same accuracy as trained physicians. Various rare diseases may manifest in physical characteristics and can be identified in their premature stages by using Facial Analysis on the patient photos. So the full-scale implementation of Artificial Neural Networks in the healthcare environment can only enhance the diagnostic abilities of medical experts and ultimately lead to the overall improvement in the quality of medical care all over the world.
- **Personal Assistants:** I am sure you all have heard of Siri, Alexa, Cortana, etc., and also heard them based on the phones you have!!! These are personal assistants and an example of speech recognition that uses Natural Language Processing to interact with the users and formulate a response accordingly. Natural Language Processing uses artificial neural networks that are made to handle many tasks of these personal assistants such as managing the language syntax, semantics, correct speech, the conversation that is going on, etc.

11.1.7 Example of Neural Network in Tensor Flow

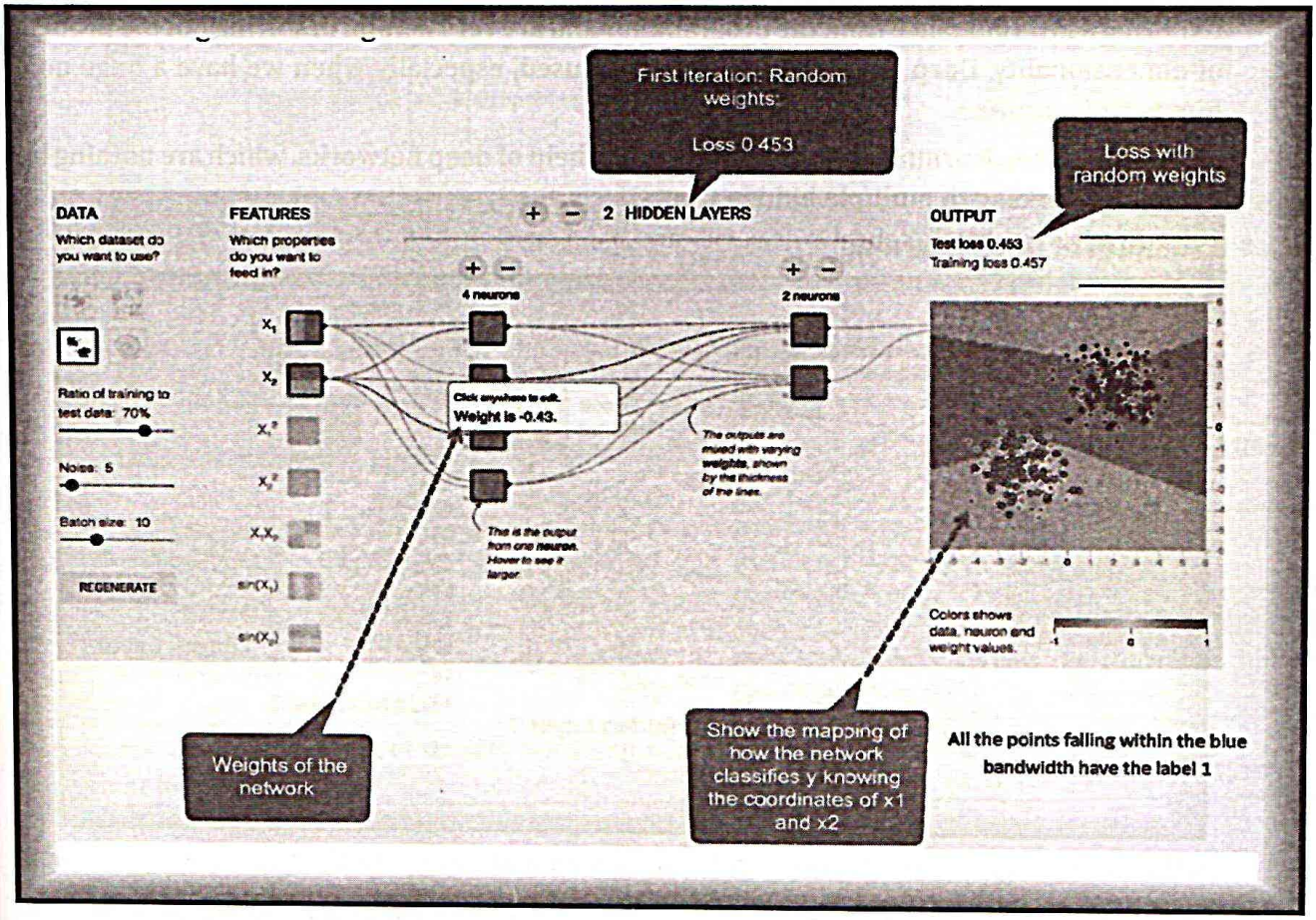
Let's see an Artificial Neural Network example in action on how a neural network works for a typical classification problem. There are two inputs, x1 and x2 with a random value. The output is a binary class. The objective is to classify the label based on the two features. To carry out this task, the neural network architecture is defined as following:

- Two hidden layers
  - ➢ First layer has four fully connected neurons
  - ➢ Second layer has two fully connected neurons
- The activation function is a Relu
- Add an L2 Regularization with a learning rate of 0.003

The network will optimize the weight during 180 epochs with a batch size of 10. In the ANN example video below, you can see how the weights evolve over and how the network improves the classification mapping.

First of all, the network assigns random values to all the weights.

- With the random weights, i.e., without optimization, the output loss is 0.453. The picture below represents the network with different colours.
- In general, the orange colour represents negative values while the blue colours show the positive values.
- The data points have the same representation; the blue ones are the positive labels and the orange one the negative labels.



Inside the second hidden layer, the lines are coloured following the sign of the weights. The orange lines assign negative weights and the blue one a positive weights

As you can see, in the output mapping, the network is making quite a lot of mistake. Let's see how the network behaves after optimization.

The picture of ANN example depicts the results of the optimized network. First of all, you notice the network has successfully learned how to classify the data point. You can see from the picture before; the initial weight was -0.43 while after optimization it results in a weight of -0.95.

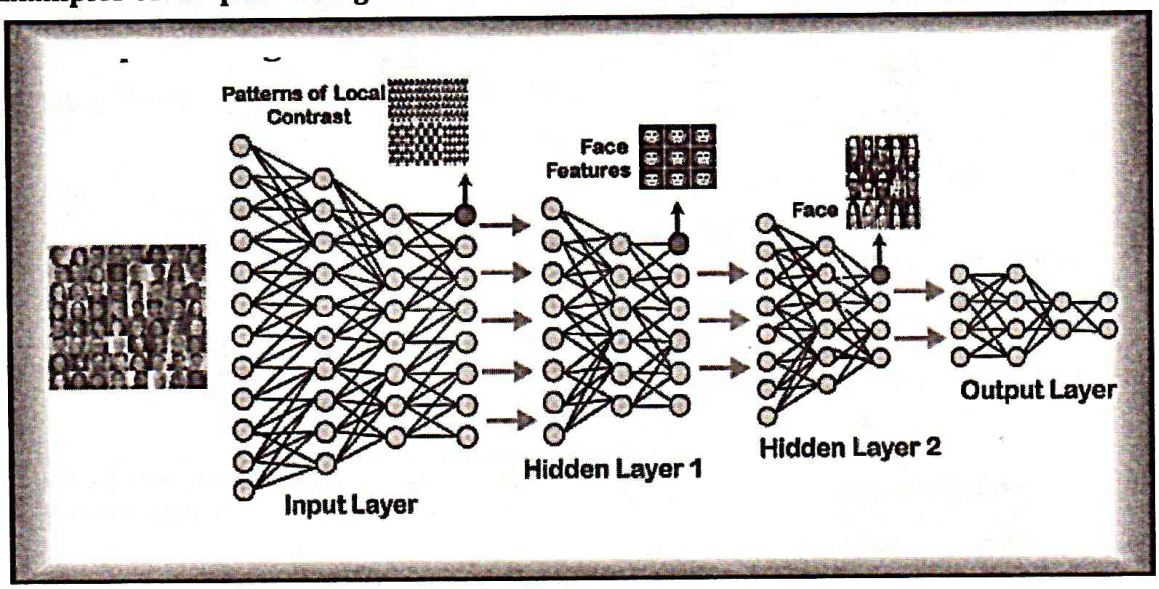
The idea can be generalized for networks with more hidden layers and neurons. You can play around *live by clicking the following link* : A Neural Network Playground (tensorflow.org)

11.2 Deep Learning

- Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers.
- Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge no of inputs and outputs.

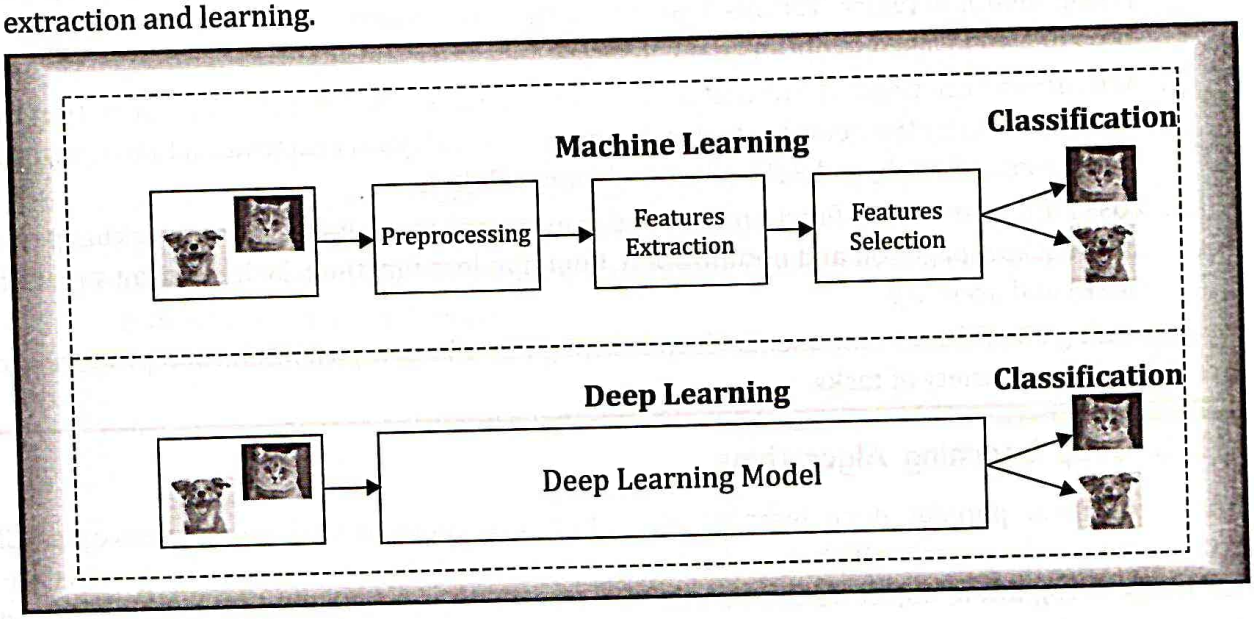
  So basically, deep learning is implemented by the help of deep networks, which are nothing but neural networks with multiple hidden layers.

Examples of Deep Learning



In the example given above, we provide the raw data of images to the first layer of the input layer. After then, these input layer will determine the patterns of local contrast that means it will differentiate on the basis of colors, luminosity, etc. Then the 1st hidden layer will determine the face feature, i.e., it will fixate on eyes, nose, and lips, etc. And then, it will fixate those face features on the correct face template. So, in the 2nd hidden layer, it will actually determine the correct face here as it can be seen in the above image, after which it will be sent to the output layer. Likewise, more hidden layers can be added to solve more complex problems, for example, if you want to find out a particular kind of face having large or light complexions. So, as and when the hidden layers increase, we are able to solve complex problems.

- In traditional machine learning techniques, the classification task typically involves a sequential process that includes pre-processing, feature extraction, meticulous feature selection, learning, and classification. The effectiveness of machine learning methods heavily relies on accurate feature selection, as biased feature selection can lead to incorrect class classification. In contrast, deep learning models enable simultaneous learning and classification, eliminating the need for separate steps. This capability makes deep learning particularly advantageous for automating feature learning across diverse tasks The below figure visually illustrates the distinction between deep learning and traditional machine learning in terms of feature extraction and learning.



11.2.1 Difference between Deep learning and Machine learning

Deep Learning	Machine Learning
Uses artificial neural network architecture to learn the hidden patterns and relationships in the dataset.	Apply statistical algorithms to learn the hidden patterns and relationships in the dataset.
Requires the larger volume of dataset compared to machine learning	Can work on the smaller amount of dataset
Better for complex task like image processing, natural language processing, etc.	Better for the low-label task.
Takes more time to train the model.	Takes less time to train the model.
Relevant features are automatically extracted from images. It is an end-to-end learning process.	A model is created by relevant features which are manually extracted from images to detect an object in the image.
More complex, it works like the black box interpretations of the result are not easy.	Less complex and easy to interpret the result.
It requires a high-performance computer with GPU.	It can work on the CPU or requires less computing power as compared to deep learning.

11.2.2 Important Components of a Deep Neural Network

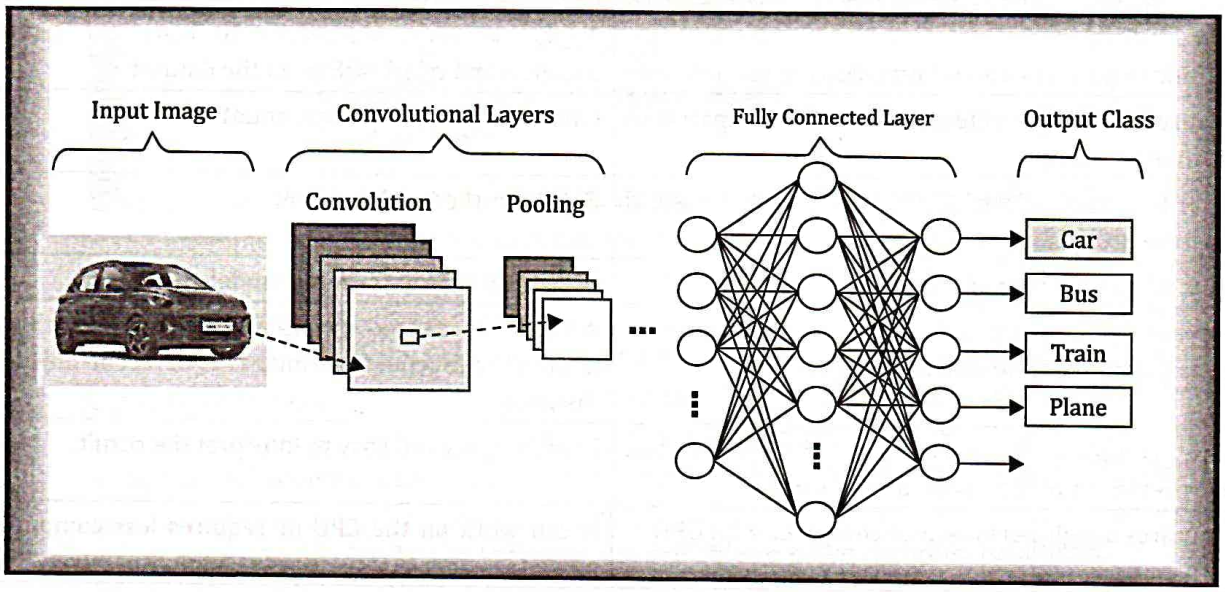
1. **Forward Propagation:** In this process, input is passed forward from one layer of the network to the next until it passes through all layers and reaches the output.
2. **Backpropagation:** This is an iterative process that uses a chain rule to determine the contribution of each neuron to errors in the output. The error values are then propagated back through the network, and the weights of each neuron are adjusted accordingly.
3. **Optimization:** This technique is used to reduce errors generated during backpropagation in a deep neural network. Various algorithms, such as gradient descent and stochastic gradient descent, can be used to optimize the network.
4. **Activation Functions:** Activation functions are used to convert inputs into an output that can be recognized by the neural network. There are several types of activation functions, including linear, sigmoid, tanh, and ReLu (Rectified Linear Units).
5. **Loss Functions:** These functions are used to measure how well a neural network has performed after backpropagation and optimization. Common loss functions include mean squared error (MSE) and accuracy.

By combining all of these components, deep learning can take complex inputs and produce accurate predictions for a variety of tasks.

11.3 Deep Learning Algorithms

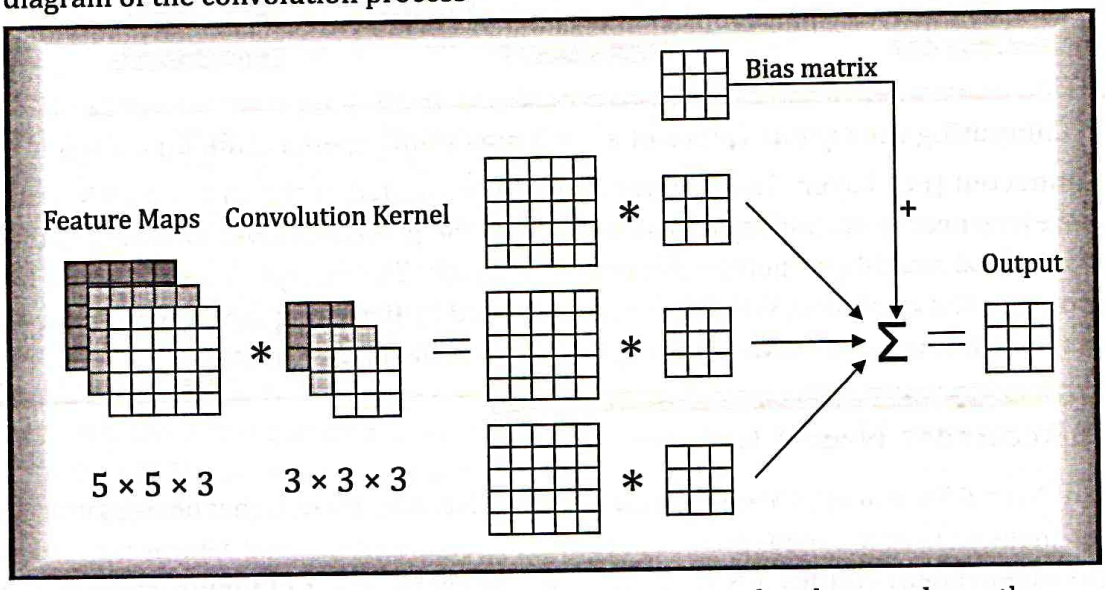
The three most popular deep learning algorithms are convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory networks (LSTMs). CNNs are used for image recognition, object detection, and classification. RNNs are used for sequence modelling, such as language translation and text generation. LSTMs use a special type of memory

11.3.1 Convolutional Neural Networks (CNN)

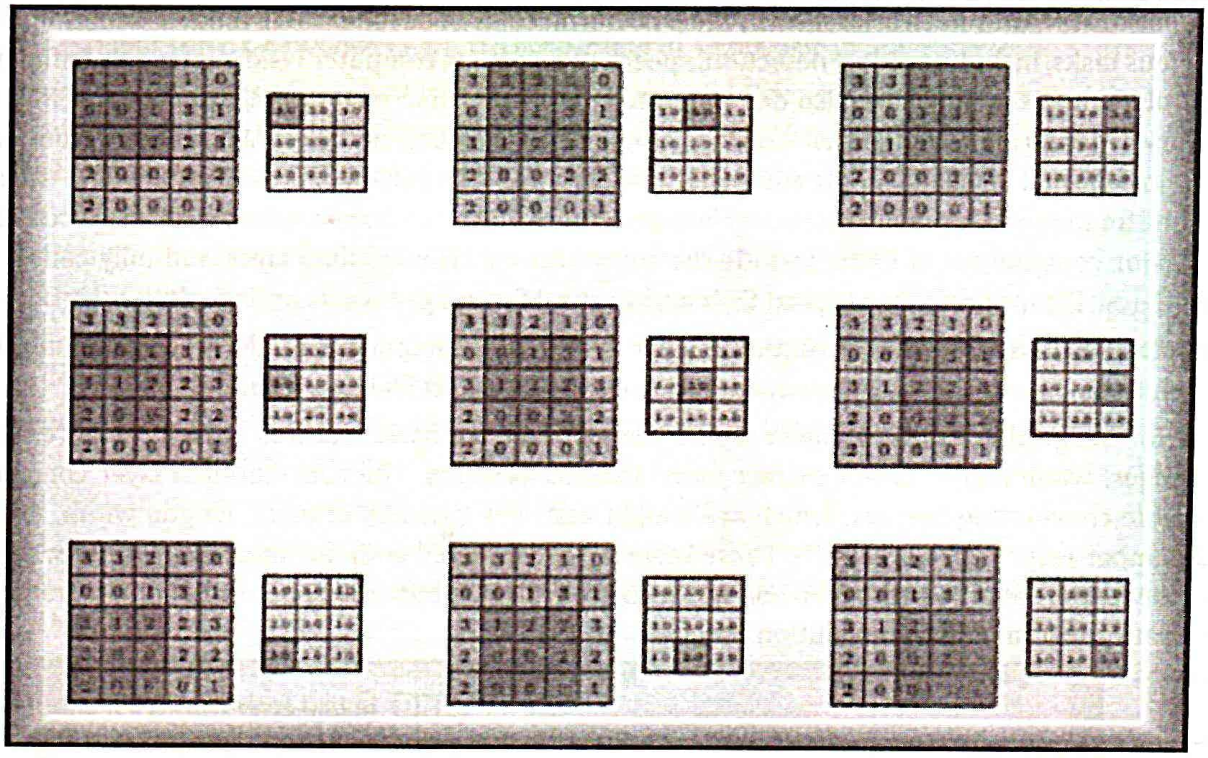


Convolutional Neural Networks (CNNs) are a powerful class of deep learning models widely applied in various tasks, including object detection, speech recognition, computer vision, image classification, and bioinformatics. They have also demonstrated success in time series prediction tasks . CNNs are feedforward neural networks that leverage convolutional structures to extract features from data. Unlike traditional methods, CNNs automatically learn and recognize features from the data without the need for manual feature extraction by humans. The design of CNNs is inspired by visual perception. The major components of CNNs include the convolutional layer, pooling layer, and fully connected layer. Below Figure presents a typical CNN architecture for image classification tasks.

Convolutional Layer: The convolutional layer is a pivotal component of CNNs. Through multiple convolutional layers, the convolution operation extracts distinct features from the input. In image classification, lower layers tend to capture basic features such as texture, lines, and edges, while higher layers extract more abstract features. The convolutional layer comprises learnable convolution kernels, which are weight matrices typically of equal length, width, and an odd number (e.g., 3x3, 5x5, or 7x7). These kernels are convolved with the input feature maps, sliding over the regions of the feature map and executing convolution operations Below figure illustrates the schematic diagram of the convolution process



Pooling Layer: Typically following the convolutional layer, the pooling layer reduces the number of connections in the network by performing down-sampling and dimensionality reduction on the input data. Its primary purpose is to alleviate the computational burden and address overfitting issues . Moreover, the pooling layer enables CNNs to recognize objects even when their shapes are distorted or viewed from different angles, by incorporating various dimensions of an image through pooling . The pooling operation produces output feature maps that are more robust against distortion and errors in individual neurons. There are various pooling methods, including Max Pooling, Average Pooling, Spatial Pyramid Pooling, Mixed Pooling, Multi-Scale Order-Less, and Stochastic Pooling. Below Figure depicts an example of Max Pooling, where a window slides across the input, and the contents of the window are processed by a pooling function.



Computing the output values of a 3 × 3 max pooling operation on a 5 × 5 input

Fully Connected (FC) Layer: The FC layer is typically located at the end of a CNN architecture. In this layer, every neuron is connected to all neurons in the preceding layer, adhering to the principles of a conventional multi-layer perceptron neural network. The FC layer receives input from the last pooling or convolutional layer, which is a vector created by flattening the feature maps. The FC layer serves as the classifier in the CNN, enabling the network to make predictions.

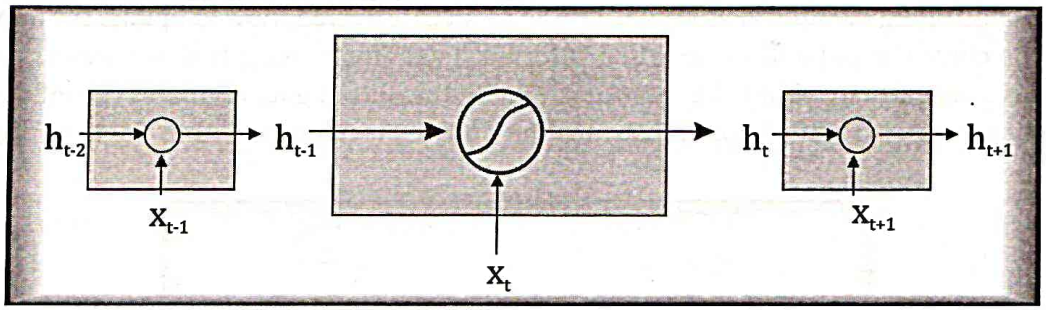
11.3.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are a class of deep learning models that possess internal memory, enabling them to capture sequential dependencies. Unlike traditional neural networks that treat inputs as independent entities, RNNs consider the temporal order of inputs, making them suitable for tasks involving sequential information. By employing a loop, RNNs apply the same operation to each element in a series, with the current computation depending on both the current input and the previous computations.

The ability of RNNs to utilize contextual information is particularly valuable in tasks such as natural language processing, video classification, and speech recognition. For example, in language modelling, understanding the preceding words in a sentence is crucial for predicting the next word. RNNs excel at capturing such dependencies due to their recurrent nature.

However, a limitation of simple RNNs is their short-term memory, which restricts their ability to retain information over long sequences. To overcome this, more advanced RNN variants have been developed, including Long Short Term Memory (LSTM) , bidirectional LSTM , Gated Recurrent Unit (GRU) , bidirectional GRU , Bayesian RNN, and others.

Below figure depicts a simple recurrent neural network,



where the internal memory (ht) is computed using Equation (2) :

$$ht = g(Wxt + Uht + b) \quad (2)$$

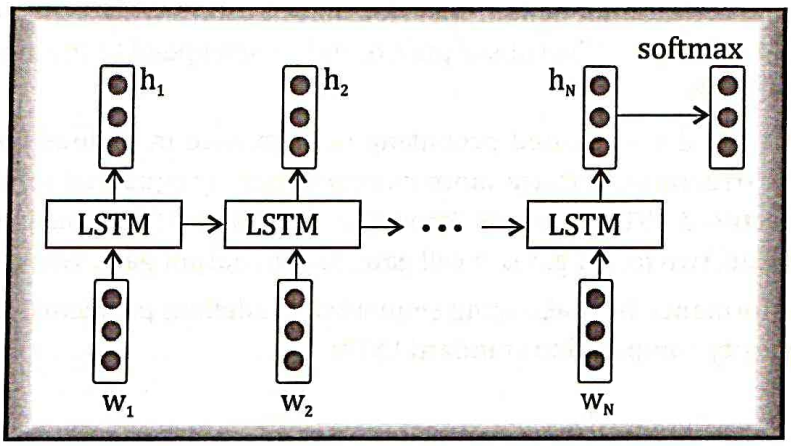
In this equation, $g()$ represents the activation function (typically the hyperbolic tangent), U and W are adjustable weight matrices for the hidden state (h), b is the bias term, and x denotes the input vector.

RNNs have proven to be powerful models for processing sequential data, leveraging their ability to capture dependencies over time. The various types of RNN models, such as LSTM, bidirectional LSTM, GRU, and bidirectional GRU, have been developed to address specific challenges in different applications.

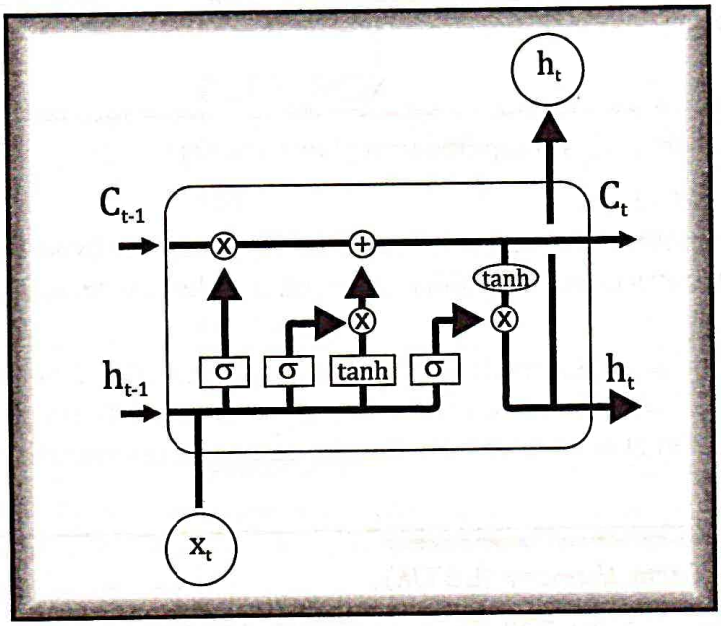
11.3.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is an advanced variant of Recurrent Neural Networks (RNN) that addresses the issue of capturing long-term dependencies. LSTM was initially introduced by in 1997 and further improved by in 2013, gaining significant popularity in the deep learning community. Compared to standard RNNs, LSTM models have proven to be more effective at retaining and utilizing information over longer sequences .

In an LSTM network, the current input at a specific time step and the output from the previous time step are fed into the LSTM unit, which then generates an output that is passed to the next time step. The final hidden layer of the last time step, sometimes along with all hidden layers, is commonly employed for classification purposes. The overall architecture of an LSTM network is depicted in Figure.



LSTM consists of three gates: input gate, forget gate, and output gate. Each gate performs a specific function in controlling the flow of information. The input gate decides how to update the internal state based on the current input and the previous internal state. The forget gate determines how much of the previous internal state should be forgotten. Finally, the output gate regulates the influence of the internal state on the system . Figure illustrates the update mechanism within the inner structure of an LSTM.



The update equations for the LSTM unit are expressed by Equation 3

$$h^{(t)} = g_0^{(t)} fh(s^{(t)})$$
$$s^{(t-1)} = g_f^{(t)} s^{(t-1)} + g_i^{(t)} f_s(wh^{(t-1)}) + uX^{(t)} + b \quad (3)$$
$$g_i^{(t)} = \text{sigmoid } (w_i h^{(t-1)} + u_i X^{(t)} + b_i$$
$$g_f^{(t)} = \text{sigmoid } (w_f h^{(t-1)} + u_f X^{(t)} + b_f$$
$$g_0^{(t)} = \text{sigmoid } (w_0 h^{(t-1)} + u_0 X^{(t)} + b_0$$

where fh and fs represent the activation functions of the system state and internal state, typically utilizing the hyperbolic tangent function. The gating operation, denoted as g, is a feedforward neural network with a sigmoid activation function, ensuring output values within the range of [0, 1], which are interpreted as a set of weights. The subscripts *i*, *o*, and *f* correspond to the input gate, output gate, and forget gate, respectively.

While standard LSTM has demonstrated promising performance in various tasks, it may struggle to comprehend input structures that are more complex than a sequential format. To address this limitation, a tree-structured LSTM network, known as S-LSTM. S-LSTM consists of memory blocks comprising an input gate, two forget gates, a cell gate, and an output gate. While S-LSTM

exhibits superior performance in challenging sequential modelling problems, it comes with higher computational complexity compared to standard LSTM.

Summary

- **Neural Network** is the fusion of artificial intelligence and brain-inspired design that reshapes modern computing. With intricate layers of interconnected artificial neurons, these networks emulate the intricate workings of the human brain, enabling remarkable feats in machine learning.
- An **Artificial Neural Network (ANN)** is a computer system inspired by biological neural networks for creating artificial brains based on the collection of connected units called artificial neurons. It is designed to analyse and process information as humans. Artificial Neural Network has self-learning capabilities to produce better results as more data is available.
- An Artificial Neural Network (ANN) is composed of four principal objects: (a) ***Layers*** (b) ***Activation function*** (c) ***Loss function*** and (d) ***Optimizer***
- Types of Artificial Neural Networks - ***Feedforward Neural Network, Convolutional Neural Network, Modular Neural Network, Radial basis function Neural Network, Recurrent Neural Network***
- **Advantages of ANN** : (a) Parallel processing capability (b) Storing data on the entire network (c) Capability to work with incomplete knowledge (d) Having a memory distribution (e) Having fault tolerance
- **Disadvantages of ANN :** (a) Assurance of proper network structure (b) Unrecognized behaviour of the network (c) Hardware dependence (d) Difficulty of showing the issue to the network (e) The duration of the network is unknown
- **Applications of Artificial Neural Networks :** (a) Social media (b) Marketing and Sales (c) Health Care (d) Personal Assistants.
- Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers.
- **Important Components of a Deep Neural Network:** (a) *Forward Propagation (b) Backpropagation (c) Optimization (d) Activation Functions (e)Loss Functions*
- The three most popular deep learning algorithms are convolutional neural networks **(CNNs),** recurrent neural networks **(RNNs)**, and long short-term memory networks **(LSTMs).**
- **Convolutional Neural Networks** (CNNs) are a powerful class of deep learning models widely applied in various tasks, including object detection, speech recognition, computer vision, image classification, and bioinformatics. They have also demonstrated success in time series prediction tasks . CNNs are feedforward neural networks that leverage convolutional structures to extract features from data. Unlike traditional methods, CNNs automatically learn and recognize features from the data without the need for manual feature extraction by humans.

- **Recurrent Neural Networks (RNNs)** are a class of deep learning models that possess internal memory, enabling them to capture sequential dependencies. Unlike traditional neural networks that treat inputs as independent entities, RNNs consider the temporal order of inputs, making them suitable for tasks involving sequential information. By employing a loop, RNNs apply the same operation to each element in a series, with the current computation depending on both the current input and the previous computations.
- **Long Short-Term Memory (LSTM)** is an advanced variant of Recurrent Neural Networks (RNN) that addresses the issue of capturing long-term dependencies. LSTM was initially introduced by in 1997 and further improved by in 2013, gaining significant popularity in the deep learning community. Compared to standard RNNs, LSTM models have proven to be more effective at retaining and utilizing information over longer sequences .

11.4 Review Questions

Short Answer Questions

1. Define Neural Networks.
2. What is Artificial Neural networks.
3. State the relationship between Biological networks and artificial neural networks.
4. List four principal objects of ANN.
5. List any four types of ANN.
6. State any two advantages of ANN.
7. State any two disadvantages of ANN
8. State any two application areas of ANN.
9. Define Deep learning.
10. List different components of Deep neural network.
11. List popular deep learning algorithms.
12. Define CNN.
13. Define RNN.
14. Define LSTM.

Long Answer Questions

1. With neat diagram, Explain the architecture of ANN.
2. Explain any four types of ANN.
3. With an example, Explain how ANN works for classification problem.
4. Differentiate between Deep learning and Machine learning.
5. Explain CNN in detail.
6. Explain RNN in detail
7. Explain LSTM in detail.

❖❖❖❖

APPENDIX

MODEL QUESTION PAPERS

A

Model Question Paper - 1

Section-A

I. Answer any four questions. Each questions carries two marks **(4 x 2 = 8)**

1. What is intelligence composed of?
2. Define A* search
3. Define Rote learning
4. List the stages of Planning in AI.
5. What is Image classification?
6. List any two characteristics of Expert system

Section – B

II. Answer any four questions. Each questions carries five marks **(4 x 5 = 20)**

7. Define AI. Explain the four categories of AI
8. With an example, Explain BFS
9. Briefly explain Winston's Learning Program
10. Define Green's Approach. Explain Green's Approach for Simple block world problem.
11. With neat diagram explain the architecture of Expert system
12. With an example, Explain reinforcement learning

Section – C

III. Answer any four questions. Each questions carries eight marks **(4 x 8 = 32)**

13. With a neat diagram explain the vacuum-cleaner world example.
14. With an example, Explain Mean End Analysis in AI
15. With an example, Explain FOL in AI.
16. What do you mean by Image classification? Explain the step-by-step process
17. With neat diagram differentiate between Forward chaining and Backward chaining in Inference engine
18. Differentiate between Deep learning and Machine learning.

❖❖❖❖

Model Question Paper - 2

Section-A

I. Answer any four questions. Each questions carries two marks (4 x 2 = 8)

1. List any 4 major disciplines of AI.
2. Define AO* search
3. Define TMS (Truth Maintenance System).
4. What is Uncertainty in AI?
5. List any four application areas of Computer vision
6. List the stages of machine learning life cycle

Section – B

II. Answer any four questions. Each questions carries five marks (4 x 5 = 20)

7. Explain the major disciplines of AI
8. Write a note on Concept of Rationality
9. Write a note on Quantifiers in FOL
10. Define computer vision. Explain any 4 application areas of computer vision.
11. Explain the different types ambiguity in NLP
12. Explain any 5 characteristics of Expert system

Section – C

III. Answer any four questions. Each questions carries eight marks (4 x 8 = 32)

13. Explain the structure of Intelligent Agents .
14. Explain minmax algorithm with an example
15. Briefly explain Samuel's Checker program to illustrate the concept of rote learning.
16. Define STRIPS approach. Explain STRIPS style operators that correspondence to the block world operations.
17. With neat diagram, Explain the architecture of ANN
18. With suitable example, explain how to build a NLP pipeline

❖❖❖❖

Model Question Paper - 3

Section-A

I. Answer any four questions. Each questions carries two marks (4 x 2 = 8)

1. List 4 categories of AI.
2. Define Zero-sum game
3. Define Inference Engine. List the two rules of Inference engine
4. Define Nonmonotonic logics
5. What do you mean by Lemmatization in NLP?
6. Define Deep learning

Section – B

II. Answer any four questions. Each questions carries five marks (4 x 5 = 20)

7. With a neat diagram explain agents and its environment.
8. Explain the Turning Test approach of AI with a neat diagram
9. With a neat diagram, Explain 8 puzzle problem.
10. Differentiate between First order logic and Propositional logic.
11. With a neat diagram, explain the different components of robots.
12. With an example, Explain Lexical or Morphological Analysis

Section – C

III. Answer any four questions. Each questions carries eight marks (4 x 8 = 32)

13. Explain the working of Alpha-Beta pruning tree.
14. Write a note on Decision trees
15. Explain different stages of planning in AI. Differentiate between FSSP & BSSP.
16. With any one of the case study, Explain expert system in detail.
17. With an example, Explain Left most and Right most derivation. Construct parse tree.
18. With a help of game tree, explain tic-tac-toe

❖❖❖❖

Model Question Paper - 4

Section-A

I. Answer any four questions. Each questions carries two marks (4 x 2 = 8)

1. 1.Compare Weak AI with Strong AI
2. What is Goal based agent ?
3. List different types of learning .
4. Justify- "Probabilistic Reasoning in AI - A way to deal with Uncertainty"
5. What do you mean by chunking in NLP?
6. State the relationship between Biological networks and artificial neural networks.

Section – B

II. Answer any four questions. Each questions carries five marks (4 x 5 = 20)

7. With an example, Explain A* search
8. With a neat diagram, Explain 8 -queens problem.
9. How Propositional Logic in Artificial Intelligence Represents Data to Machine ? Explain all logical connectives with truth table.
10. Write a note on Nonmonotonic logics with advantages & disadvantages
11. Explain the challenges of Pragmatic analysis in NLP
12. Briefly explain machine learning life cycle.

Section – C

III. Answer any four questions. Each questions carries eight marks (4 x 8 = 32)

13. Explain any 8 top technologies used in AI.
14. With a neat diagram, Explain the Knowledge-Based System in AI.
15. Briefly explain Winston's Learning Program.
16. What is Robotic Engineering ? With an example, explain forward kinematics & inverse kinematics for 2-DOF. Use Geometrical approach.
17. With an example, Explain Left most and Right most derivation. Construct parse tree.
18. Explain (a) CNN (b) RNN (c) LSTM

❖❖❖❖