Module 1

CHAPTER 1

Introduction to Artificial Intelligence

Syllabus

Introduction, History of Artificial Intelligence, Intelligent Systems: Categorization of Intelligent System, Components of AI Program, Foundations of AI, Sub-areas of AI, Applications of AI, Current trends in AI.

1.1 Introduction 1-3

1.1.1 What is AI ? 1-3

1.1.1.1 Artificial 1-3

1.1.1.2 Intelligence 1-3

1.1.2 Objectives of AI 1-4

1.2 History of AI 1-5

1.3 Information, knowledge and Intelligence 1-7

1.3.1 Differences between Conventional Computing and Intelligence Computing 1-7

1.4 AI Techniques 1-8

1.5 Intelligent System (IS) 1-8

1.5.1 Categorization of Intelligent Systems 1-9

1.6 Different Definitions of Artificial Intelligence 1-9

H Explain different definitions of artificial intelligence according to different categories.

(MU - Q. 1(a), Dec. 19, 5 Marks) 1-9

1.7 Major Components of Artificial Intelligence 1-10

1.7.1 Functions of Each Components of AI System 1-10

1.8 Foundations of AI 1-11

1.9 SUB-AREAS OF AI 1-13

1.9.1 Machine Learning 1-13

Scanned with CamScanner

1.9.2 Natural Language Processing (NLP) 1-14
1.9.3 Automation and Robotics 1-14
1.9.4 Machine Vision 1-14
1.9.5 Expert System (ES) 1-14
1.9.6 Neural Computing 1-15
1.9.7 Automatic Programming 1-15
1.9.8 Genetic Algorithm 1-15
1.9.9 Fuzzy Logic 1-15
1.9.10 Intelligent Agents 1-15
1.10 AI Problems 1-15
1.10.1 Programming Without and With AI 1-17
UQ. Differentiate Programming Without and With AI (MU - Q. 1(d), Dec. May 18, 4 Marks) 1-17
1.11 Current trends in AI 1-17
1.12 Application Areas of Artificial Intelligence 1-19
UQ. What are applications of AI? (MU - Q. 1(a), May 17, 5 Marks) 1-19
• **Chapter Ends** 1-20

Scanned with CamScanner

1.1 INTRODUCTION

The field of Artificial Intelligence (AI) and Machine Learning (ML) has exploded in last decade. The roots of these words originate from multiple disciplines and not just computer science.

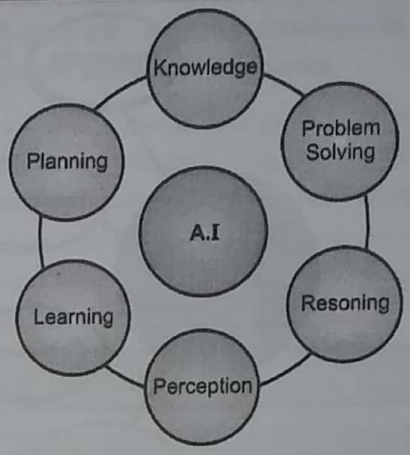
These disciplines include pure mathematics, electrical engineering, statistics, signal processing, and communications along with computer science to name the top few.

Along with the wide variety of origins, the field also finds applications in even greater number of industries ranging from high-tech applications like image processing, natural language processing to online shopping to non-destructive testing of nuclear power plants and so on.

1.1.1 What is AI ?

GQ. What is Artificial Intelligence?

- Alan Turing **defined Artificial Intelligence** as follows: "If there is a machine behind a curtain and a human is interacting with it (by whatever means, e.g. audio or via typing etc.) and if the human feels like he/she is interacting with another human, then the machine is artificially intelligent."
- There are two aspects to AI as viewed from human-like behavior standpoint. One is where the **machine is intelligent** and is **capable of communication with humans**, but does not have any locomotive aspects. The other aspect involves having physical interactions with human-like locomotion capabilities, which refers to the field of robotics.
- Artificial intelligence (AI, also machine intelligence, MI) is intelligence demonstrated by machines, in contrast to the natural intelligence (NI) displayed by humans and other animals. The term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving".
- We can use AI in many special domains and functions like understanding/learning, Planning, Problem solving, Decision making (Fig. 1.1.1). These all are very helpful now days in various industries.



(1A1)**Fig. 1.1.1 : Different phases under AI**

Artificial Intelligence is a term, which consists of two words.

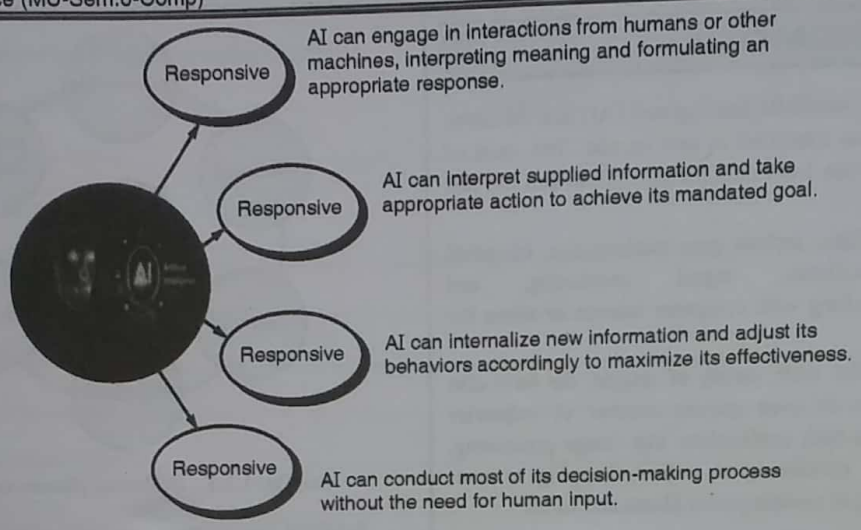
1.1.1.1 Artificial

- Artificial is somewhat that is not real and is simulated. Artificial is not only used in the context of food. Artificial turf is a grass like surface used for sports playing fields.
- Artificial diamonds are those that are manufactured, as opposed to those that have developed through natural processes in the earth. Artificial flowers are often used as decorations.

1.1.1.2 Intelligence

- Defining and classifying intelligence is extremely complicated. Theories of intelligence range from having one general intelligence, to certain primary mental abilities, and to multiple category-specific intelligences.
- Intelligence is defined in many different ways like logic, understanding, self-awareness, learning, emotional knowledge, planning, creativity and problem solving.
- Day by day lot of innovations takes place in today's world. New technologies, new machines with greater efficiency performances are the need of upcoming Techies and this thing is growing gradually exponentially.

Scanned with CamScanner



(1A2)**Fig. 1.1.2 : Artificial Intelligence**

- Tremendous efforts in every domain has to be carried out as a need of future world that may be faster processing speed, reducing memory overhead lesser in size and more intelligent.
- Now the question arises, can we make machine intelligent? Answer is a Yes and this has been possible because of AI (Artificial Intelligence).
- As a result of this, we can make the system, and train the system build the system which can mimic the behavior and multiple properties and function of human being.

1.1.2 Objectives of AI

GQ. What are different objectives of AI ?

Below are the eight aims and objectives of artificial intelligence :

Objective #1 : Artificial intelligence solves problems

- When it comes to artificial intelligence, there is a strong urge to create AI programs that look, act, and feel like real humans.
- However, many scientists now understand that the real goal is not to make a human-like robot. Instead, they would rather create a robot that works to make our lives easier, no matter what it looks or sounds like.
- Moving forward, it is likely that we will see some serious work being put into the ability for AI to learn and understand, and less on forcing them to act like real humans. That will probably just come with time.

Objective #2 : Artificial intelligence completes multiple tasks

- Completing multiple tasks is another aims and objectives of artificial intelligence. One of the largest difficulties to overcome has been making it possible for an AI program or a "robot" to do more than one task. It is very easy to program a system to complete a certain task. For instance, it can bring an item from point A to point B.
- However, if you want the program to understand that it must pick up the item and then either bring it to point A or throw it in the trash based on arbitrary rules that a human would know that's a different story. In simpler terms, it might be a while before your housemaid is a robot.

Objective #3 : Artificial intelligence shapes the future of every company

- AI is quickly becoming a crucial tool for all companies. They are using this technology to streamline their processes.
- It's no secret that the goal is to continue this trend for as many low-level tasks as possible. It ultimately saves the companies money in the long run, and it allows them to up productivity in other areas.

Objective #4 : Artificial intelligence prepares for a boom in big data

- Big data has already taken the world by storm. Big data is the large-scale, and sometimes even random, collection of data about people's lives, habits, conversations and more.

Scanned with CamScanner

- AI will be able to do much more for the analysis of this data than humans ever did, so data-driven research, advertisements, and content are going to explode.

☞ Objective #5 : Artificial intelligence creates synergy between humans and AI

One of the key goals in AI is to develop a strong synergy between AI and humans, so that they can work together to enhance the capabilities of both.

☞ Objective #6 : Artificial intelligence is good at problem-solving

So far, AI is unable to employ advanced problem-solving abilities. That is, it can tell you a factual answer, but cannot analyze a specific situation and make a decision based on the very specific context of that situation.

☞ Objective #7 : Artificial intelligence helps with planning

One of the most human traits in existence is the ability to plan and make goals and subsequently accomplish them. And one of the goals for AI is to have AI be able to do these things.

☞ Objective #8 : Artificial intelligence performs more complex tasks

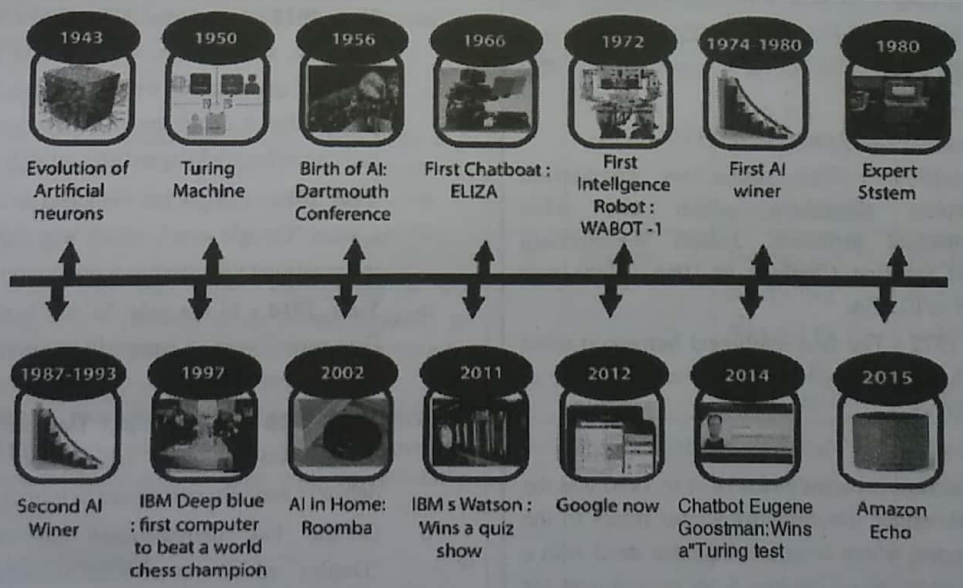
- The key goal is this: to develop AI programs that can complete more and more complex tasks. Already the abilities are shocking, although not yet widespread.
- However, over time these will develop and ultimately, scientists hope, be able to do basically the same things humans can do.

⏭ 1.2 HISTORY OF AI

GQ. Write a short note on : History of Artificial Intelligence ?

Artificial Intelligence is much older than you would imagine. Even there are the myths of Mechanical men in Ancient Greek and Egyptian Myths.

Following are some milestones in the history of AI which defines the journey from the AI generation to till date development.



(1A4)**Fig. 1.2.1 History of AI**

Scanned with CamScanner

- Maturation of Artificial Intelligence (1943-1952)
  - **Year 1943 :** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.
  - **Year 1949 :** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called **Hebbian learning**.
  - **Year 1950 :** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.
- The birth of Artificial Intelligence (1952-1956)
  - **Year 1955 :** An Allen Newell and Herbert A. Simon created the "first artificial intelligence program "Which was named as **"Logic Theorist"**. This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.
  - **Year 1956 :** The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.
  - At that time high-level computer languages such as FORTRAN, LISP, or COBOL were invented. And the enthusiasm for AI was very high at that time.
- The golden years-Early enthusiasm (1956-1974)
  - **Year 1966 :** The researchers emphasized developing algorithms which can solve mathematical problems. Joseph Weizenbaum created the first Chatbots in 1966, which was named as ELIZA.
  - **Year 1972 :** The first intelligent humanoid robot was built in Japan which was named as WABOT-1.
- The first AI winter (1974-1980)
  - The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches.
  - During AI winters, an interest of publicity on artificial intelligence was decreased.
- A boom of AI (1980-1987)
  - **Year 1980 :** After AI winter duration, AI came back with "Expert System". Expert systems were programmed that emulate the decision-making ability of a human expert.
  - In the Year 1980, the first national conference of the American Association of Artificial Intelligence was held at Stanford University.
- The second AI winter (1987-1993)
  - The duration between the years 1987 to 1993 was the second AI Winter duration.
  - Again Investors and government stopped in funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.
- The emergence of intelligent agents (1993-2011)
  - **Year 1997 :** In the year 1997, IBM Deep Blue beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.
  - **Year 2002 :** for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.
  - **Year 2006 :** AI came in the Business world till the year 2006. Companies like Facebook, Twitter, and Netflix also started using AI.
- Deep learning, big data and artificial general intelligence (2011-present)
  - **Year 2011 :** In the year 2011, IBM's Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
  - **Year 2012 :** Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.
  - **Year 2014 :** In the year 2014, Chatbot "Eugene Goostman" won a competition in the infamous "Turing test."
  - **Year 2018 :** The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well.
  - Google has demonstrated an AI program "Duplex" which was a virtual assistant and which had taken hairdresser appointment on call, and lady on other side didn't notice that she was talking with the machine.

Scanned with CamScanner

- Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

1.3 INFORMATION, KNOWLEDGE AND INTELLIGENCE

GQ. Define information, knowledge and intelligence. What is the comparison between artificial and human intelligence.

GQ. Explain Intelligence and Artificial Intelligence. How does conventional computing differ from the intelligence computing?

(1) Information

- All data are information. However, there is some part of information that is not considered as a data. Such **distinguished information** can be considered as a processed data, which makes decision making easier. Processing involves an **aggregation** of data, **calculations** of data, **corrections** on data, etc. in such a way that it generates the flow of messages.
- Information usually has some meaning and purpose that is; data within a context can be considered as information.

(2) Knowledge

Knowledge is a justified true belief. Knowledge is a **store of information** proven useful for a capacity to act.

(3) Intelligence

- Unlike belief and knowledge, intelligence is not information: it is a process, or an **innate capacity to use information** in order to respond to ever-changing requirements.
- It is a capacity to **acquire, adapt, modify, extend** and **use information** in order to solve problems. Therefore, intelligence is the ability to cope with unpredictable circumstances.

(A) Human intelligence

- Human intelligence is the intellectual capacity of humans, which is characterized by perception, consciousness, self-awareness, and volition.
- Intelligence enables humans to remember description of things and use those descriptions in future behaviours. It is a **cognitive process**.
- It gives humans the cognitive abilities to learn from concepts, understand, and reason, including the capacities to recognize patterns, comprehend ideas, plan, solve problems, and use language to communicate. Intelligence enables humans to experience and think.

(B) Artificial intelligence

- Artificial intelligence (or AI) is both the intelligence of machines and the branch of computer science which aims to create it, through "the study and design of intelligent agents" or "rational agents", whereas; an intelligence agent is a system that perceives its environment and takes actions which maximize its chances of success.
- Achievements in artificial intelligence include constrained and well-defined problems such as **games, crossword-solving and optical character recognition** and a few more general problems such as **autonomous cars**. General intelligence or strong AI has not yet been achieved and is a long-term goal of AI research.

1.3.1 Differences between Conventional Computing and Intelligence Computing

Sr. No.	Parameters	Intelligence computing	Conventional computing
1.	Solution	Does not guarantee a solution to a given problem.	Guarantees a solution to a given problem.
2.	Results	Produces results that may not be reliable or consistent.	Produces results that are consistent and reliable.
3.	Instructions	Solves the problem without specific program instructions.	Solves the given problem according to the programmer's exact instructions (algorithm).

Scanned with CamScanner

1.4 AI TECHNIQUES

(1) An artificial intelligence technique is a method that exploits knowledge that is represented so that, the knowledge captures generalizations and situations that share properties which can be grouped together, rather than being allowed in a separate representation. It can be understood by people who must provide the knowledge; although for many programs the bulk of the data may come automatically, such as from readings.

(2) There are various techniques that have evolved and can be applied to a variety of artificial intelligence tasks. These techniques are concerned with how we represent, manipulate and reason with knowledge in order to solve problems.

(3) In short AI technique is a manner to organize and use the knowledge efficiently in such a way that –

- It should be perceivable by the people who provide it.
- It should be easily modifiable to correct errors.
- It should be useful in many situations though it is incomplete or inaccurate.

(4) AI techniques elevate the speed of execution of the complex program it is equipped with.

Following are three important artificial intelligence techniques

- **Search :** Provides a way of solving problems for which no more direct approach is available.
- **Use of knowledge :** Provides a way of solving **complex problems** by exploiting the structures of the objects that are involved.
- **Abstraction :** Provides a way of separating important features and variations from many unimportant ones that would otherwise overwhelm any process.

(1) **Knowledge representation :** Knowledge representation is crucial. One of the clearest results of artificial intelligence research so far is that; solving even apparently simple problems requires lots of knowledge. Really understanding a single sentence requires extensive knowledge of both language and the context. Really understanding a visual scene similarly requires knowledge of the kinds of objects in the scene. Solving problems in a particular domain generally requires knowledge of the objects in the domain and knowledge of how to reason in that domain. Both these types of knowledge must be represented efficiently, and in a meaningful way.

(2) **Efficiency :** It is important, as it would be impossible (or at least impractical) to explicitly represent every fact that we might ever need.

(3) **Search :** Another crucial general technique required while writing an artificial intelligence programs is a search. Often there is no direct way to find a solution to some problem. However, we know how to generate possibilities.

- **For example**, in solving a puzzle you might know all the possible moves, but not the sequence that would lead to a solution. When working out how to get somewhere one might know all the roads/buses/trains, just not the best route to get our destination quickly. Developing good ways to search through these possibilities for a good solution is therefore vital.
- **Brute force techniques**- where we generate and try out every possible solution that may work, but are often very inefficient, as there are just too many possibilities to try.
- **Heuristic techniques** are often better, where we only try the options, which we think (based on our current best guess) are most likely to lead to a good solution.

1.5 INTELLIGENT SYSTEM (IS)

(1) **Intelligent System** (IS) can be defined as the **system** that incorporates intelligence into applications being handled by machines.

(2) **Intelligent systems** perform search and optimization along with learning capabilities.

(3) Different types of machine learning such as supervised, unsupervised and reinforcement learning can be modeled in designing **intelligent systems**.

(4) **Intelligent systems** also perform complex automated tasks which are not possible by traditional computing paradigm.

(5) Various diagnostic, robotics and engineering **systems** are results of **intelligent** procedures implemented in **Intelligent System** Design.

Scanned with CamScanner

1.5.1 Categorization of Intelligent Systems

Sr. No.	Intelligence	Description	Example
1.	Linguistic intelligence	The ability to speak, recognize, and use mechanisms of phonology (speech sounds), syntax (grammar), and semantics (meaning).	Narrators, Orators
2.	Musical intelligence	The ability to create, communicate with, and understand meanings made of sound, understanding of pitch, rhythm.	Musicians, Singers, Composers
3.	Logical-mathematical intelligence	The ability of use and understand relationships in the absence of action or objects. Understanding complex and abstract ideas.	Mathematicians, Scientists
4.	Spatial intelligence	The ability to perceive visual or spatial information, change it, and re-create visual images without reference to the objects, construct 3D images, and to move and rotate them.	Map readers, Astronauts, Physicists
5.	Bodily-Kinesthetic intelligence	The ability to use complete or part of the body to solve problems or fashion products, control over fine and coarse motor skills, and manipulate the objects.	Players, Dancers
6.	Intra-personal intelligence	The ability to distinguish among one's own feelings, intentions, and motivations.	Gautam Buddhha
7.	Interpersonal intelligence	The ability to recognize and make distinctions among other people's feelings, beliefs, and intentions.	Mass Communicators, Interviewers

You can say a machine or a system is **artificially intelligent** when it is equipped with at least one and at most all intelligences in it.

1.6 DIFFERENT DEFINITIONS OF ARTIFICIAL INTELLIGENCE

UQ. Explain different definitions of artificial intelligence according to different categories.

(MU - Q. 1(a), Dec. 19, 5 Marks)

(I) **Strong AI :** This concept was put forward by John Searle in 1980 in his article "Minds, Brains and programs". Strong form AI provides theories for developing some form of computer based AI that can truly reason and solve problems. A strong form of AI is said to be sentient or self-aware.

Strong AI can be categorized as :

1. Human-like AI-In which the computer program thinks and reasons much like a human-mind.
2. Non-human-like AI-IN which the computer program develops a totally nonhuman sentience, and a non-human way of thinking and reasoning.

(II) **Weak AI :** Weak artificial intelligence research deals with the creation of some form of computer-based AI that cannot truly reason and solve problems. They can reason and solve problems only in a limited domain, such a machine would, in some ways, act as if it were intelligent, but it would not possess true intelligence.

There are several fields of weak AI, one of which is natural language. Much of the work in this field has been done with computer simulations of intelligence based on predefined sets of rules. Very little progress has been made in strong AI. Depending on how one defines one's goals, a moderate amount of progress has been made in weak AI.

Scanned with CamScanner

1.7 MAJOR COMPONENTS OF ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers "Developing computer programs to store complex problems by application of process that analogues to human reasoning."

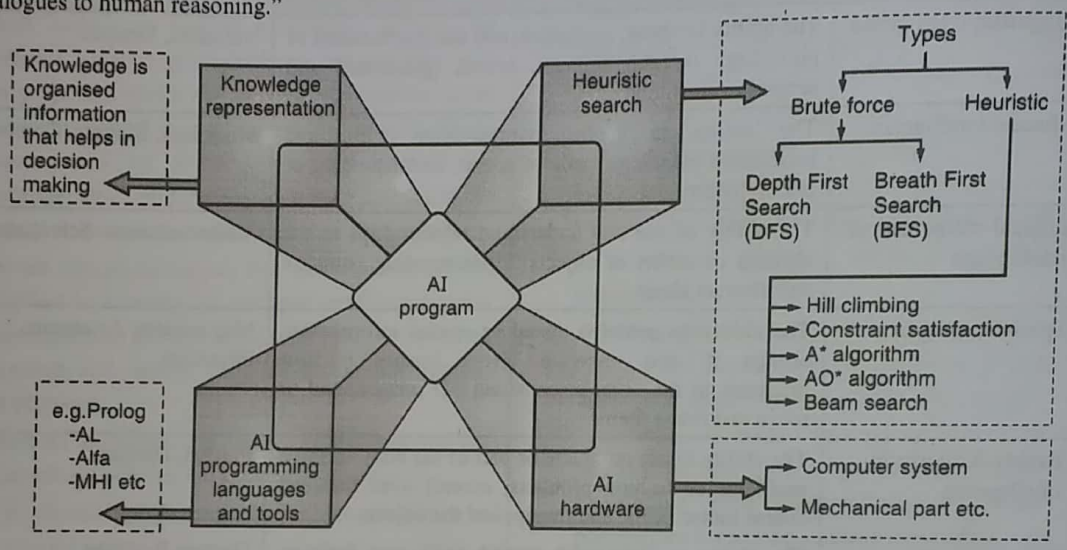


Fig. 1.7.1 : Major components of an AI system

- Any AI system has four major components :
  1. Knowledge representation.
  2. Heuristic search.
  3. AI programming languages and tool.
  4. AI hardware.
- **Definition :** Artificial intelligence (AI) refers to a **"Developing computer programs to store complex problems by application of process that analogues to human reasoning."** The quality of the results depends on how much knowledge the system possesses. The available knowledge must be represented in a very efficient way. Hence **knowledge representation** is a vital component of the system.
- It is not merely enough that knowledge is represented efficiently. The **inference process** should also be equally good for satisfactory results. The inference process is broadly divided into **brute and heuristic search procedures.**
- Today just like we have specialized languages and programs for data-processing and scientific applications, we encounter specialized languages and tools for AI programming. AI languages provide the basic functions for AI programming and tools for the right environment.
- Most of the AI programs are implemented on **Von Neumann machines** only. However, dedicated workstations have emerged for AI programming. AI provides details on the machine architecture for AI machines.

1.7.1 Functions of Each Components of AI System

(1) Knowledge Representation
(2) Heuristic Search
(3) AI programming Language and Tool
(4) AI Hardware

(1) Knowledge Representation

- Knowledge representation in AI describes the representation of knowledge. It expresses how the beliefs, intentions, and judgements of an intelligent agent can be expressed suitably for automated reasoning. One of the primary purposes is modelling intelligent behaviour for an agent.
- It represents information from the real world for a computer to understand and then utilise this knowledge to solve complex real-life problems, like communicating with human-beings in natural language.

Scanned with CamScanner

- It stores data in a database, and allows a machine to learn from that knowledge and behave intelligently like a human being.

(2) Heuristic Search

- In mathematical optimisation and computer science, heuristic is a technique designed for solving a problem more quickly when classic methods fail to find any exact solution.
- This is achieved by trading optionality, completeness, accuracy, or precision for speed. In a way, it can be considered a short-cut.
- A heuristic function ranks alternatives in search algorithms at each balancing step based on available information to decide which branch to follow.

(3) AI programming Language and Tool

(i) Python

- Python language is the forerunner of all other languages. It uses simple syntax.
- The simple syntax allows to spend much more time on planning the core structure, which is why python is an ideal choice for machine learning processes.

(ii) Lisp

- It is the oldest language used for AI processes. Lisp is considered as a tool in AI with its enlarged scope of turning thoughts into reality.
- The language differentiates itself from other AI languages by eying precision.

(iii) R

- This is highly efficient programming language.
- Packages like G models, RODBC, one R and Tm allow huge support for machine learning processes.

(iv) Prolog

- It is completely designed by logic. Prolog requires three important factors : rules, facts and the desired result.
- Once these requirements are provided, the language will figure out the link between the three and design an AI solution.

(v) C++

- One of the major advantages of having C++ as programming language is its processing speed.
- To run complex automated solutions efficiently, C++ is very useful.

(vi) Javascript

- For versatility, Javascript is ahead of Java.
- With continuous developments, multiple domain growth, backend use, ease of use, efficiency etc. Support the above statement.

(vii) Java

- The best benefit of using the JAVA programming language is the presence of virtual machine technology.
- Java virtual machine eases the implementation process.

(viii) Haskell

- This programming language is well-known for resolving errors, during the compilation process and even before that.
- The features like build-in memory and code reusability increase the time allotted for planning the process.

(ix) Julia

- The programming language is well-known for numerical analysis.The best feature of Julia is dynamic type system, which allows you to use the language for literally any process.
- Other features involve in-built package manager, macro programming abilities, multiple dispatch support and suitability with C functions.

(4) AI Hardware

- An AI accelerator is a specialised hardware accelerator or computer system designed to accelerate AI and machine learning applications, including neural networks and machine vision.
- Applications include algorithms for robotics, internet of things and other data-intensive or sensor-driven tasks.

1.8 FOUNDATIONS OF AI

The foundation of AI does not occur in isolation. Fields such as philosophy, linguistics, psychophysics, and theoretical computer science have exercised a historical influence over the field and today there is much discussion over the new field of cognitive science. One consequence of discussion (or dialogue) is that the criticism of positions held in one discipline frequently applies to positions held in other disciplines.

Scanned with CamScanner

Of these, five stand out as particularly fundamental.

(1) Pre-eminence of knowledge and conceptualisation
(2) Disembodiment
(3) Kinematics of cognition are language-like
(4) Learning can be added later
(5) Uniform architecture
(6) Are knowledge and conceptualisation at the heart of AI ?
(7) Linguistic concept
(8) The logicist concept of concept
(9) Inferential abilities
(10) Knowledge and perception
(11) Growth of knowledge

(1) Pre-eminence of knowledge and conceptualisation

Intelligence requires declarative knowledge and some form of reasoning like computation is called 'cognition'. Core AI is the study of the conceptualizations of the world presupposed and used by intelligent systems during cognition.

(2) Disembodiment

Cognition and knowledge it presupposes can be studied largely in abstraction from the details of perception and motor control.

(3) Kinematics of cognition are language-like

It is possible to describe the trajectory of knowledge states or informational states created during cognition using a vocabulary very much like English or some regimented logico-mathematical version of English.

(4) Learning can be added later

The kinematics of cognition and the domain knowledge needed for cognition can be studied separately from the study of concept learning, psychological development, and evolutionary change.

(5) Uniform architecture

There is a single architecture underlying virtually all cognition.

Remark : By 'cognition' is meant to refer to computational processes that resemble both reasoning in a classical sense and computational processes that are more 'peripheral' than reasoning, such as language recognition and object identification, where the representations are not about the entities and relations, but which may usefully be construed as rules operating on representations.

(6) Are knowledge and conceptualisation at the heart of AI ?

A theory in AI is a specification of the knowledge under planning a cognitive skill. A cognitive skill is the information - based control mechanism regarding performance in some domain. It is meant to cover information sensitive activities such as problem solving, language use, decision making, routine activity, perception and some elements of motor control.

(7) Linguistic concept

- In the case of linguistics and higher vision these basic knowledge units tend more generally to be about theoretical entities. Only occasionally will there be pre-existing terms in English for them. Thus, noun phrase, sphere, pyramid and other shapes are commonsense concepts having familiar English names, but governing domain, animate movements, causal launchings and most shape representations are, for most people, novel ideas that are not part of common parlance.
- The basic knowledge units of cognition the conceptualizations under planning cognitive skills-may range, from the familiar to the exotic and theoretical.
- The basic idea that knowledge and conceptualization lie at the heart of AI stems from the seductive view that cognition is inference. Intelligent skills are composed of two parts a declarative knowledge base and an inference engine.
- The inference engine is relatively uncomplicated; it is a domain-independent program that takes as input a set of statements about the current situation plan, a fragment of the declarative knowledge base, it produces as output a stream of knowledge - based programs.
- Logicists are not unmindful of the need to explain for a system to understand a proposition, or to grasp the concepts which constitute propositions. But the party line is that this job can be pursued independently from the designer's main task of inventing conceptualizations.
- The two activities inventing conceptualization and grounding concepts-are modular. Hence the grounding issue has not historically been treated as posing a challenge that might overturn the logicist program.

(8) The logicist concept of concept

- A concept is a modular component of knowledge. If we say that kishor knows 'the pen is on the desk', and we mean this to imply that kishor grasps the fact of their being a particular pen on a particular desk, we assume that he has distinct concepts for 'pen', 'desk' and 'on'.

Scanned with CamScanner

- We assume this because we believe that kishor must know what it is for something to be a pen, a desk, and something to be an something else.
- If concepts and conceptual schemes seem to play enough of an explanatory role at the disembodied level to be seen as robust entities, then we can study their structure without concern for their grounding.

(9) Inferential abilities

- It is the capacity of an agent to draw inferences. For instance, given the premises that the pen on the desk, is matt-black pen on the desk then a knowledgeable agent ought to be able to infer that matt-black pen is on the desk.
- Thus as long as it makes sense to view agents to be 'something' drawing inferences about a domain, or performing reason - like operations, it makes sense to suppose they have a network of concepts which structures their knowledge.

(10) Knowledge and perception

- Immanuel Kant, a german philosopher, once said, 'sensation without conception is blind'. What he meant is that 'I do not know what I am seeing, if I have no concept to categorise my experience'. It is hard to imagine how we could identify entities if we did not have concepts. The reason is because object identification is an active process.
- Perception is not a passive system. It is a method for systematically gathering evidence about the environment. We can think of it as a guide offering answers to questions about the external world. Not direct answers but perceptual answers, that serve as an evidence for an agent certain perceptual 'conjectures'. One job of the perceptual system is to ask the right questions. Our eyes jump about an image looking for clues of identity; then search for confirmation of conjectures.

(11) Growth of knowledge

- A third feature of rational intelligence - learning - can be partially explained if we give to a system a set of disembodied concepts. Domain knowledge is much like a theory, it is a system of axioms relating basic concepts. Some axioms are definitional. Hence learning is movement along a trajectory of theories. It is a conceptual advance.
- This approach brings us closer to understanding the principles of learning, but a theory of intelligence which did not mention concepts would have to explain learning as a chance in capacities behaviourally or functionally classified. Disembodied concepts serve well enough.

1.9 SUB-AREAS OF AI

Various sub-areas of AI are as follows :

(1) Machine Learning.
(2) NLP (Natural Language Processing)
(3) Automation and Robotics.
(4) Machine Vision.
(5) Expert System (ES)
(6) Neural Computing
(7) Automatic Programming
(8) Genetic Algorithm
(9) Fuzzy Logic
(10) Intelligent Agents

1.9.1 Machine Learning

[Machine learning is one of the applications of AI where machines are not explicitly programmed to perform certain tanks, rather they learn and improve from experience automatically] .

It is a subfield of AI, which is defined **as the capability of a machine to imitate intelligent human behaviour**.

A.I. systems are used to perform complex tasks in a way that is similar to how humans solve problems. **Machine learning is one way to use Artificial Intelligence.**

Example of Machine Learning

(1) **Image recognition** is a well-known example of machine learning. It can identify an object as a digital image, based on the intensity of the pixels in black and white images or colour images.

(2) Machine learning is also **used** in 'internet search engines', e-mails filters to sort out spam, websites to make personalized recommendation, banking software to unusual transactions and lots of apps. on our phones such as voice recognition.

(3) Machine learning models learn, identify, pattern and make decisions with minimal intervention from humans. Ideally, machines increase accuracy and efficiency and remove or greatly reduce the possibility of human error.

(4) Machine learning is fascinating because programs learn from examples. From the collected data, a machine learning method can automatically analyse and learn the structure already resident in that data in order to provide a solution to the problem one is trying to solve.

Scanned with CamScanner

1.9.2 Natural Language Processing (NLP)

- NLP is a branch of AI that enables machines to understand the human language. Its goal is to build systems that can make sense of text and automatically perform tasks like **translation, spell check or topic classification**.
- It is an intuitive behavior used to convey information and meaning with **semantic clues** such as words signs or images. The terms AI and NLP might conjure images of futuristic robots; there are already basic examples of NLP at work in our daily lives.
- The NLP techniques are widely used in Log Analysis and Log Mining. The different techniques such as tokenization, stemming parsing etc. are used to convert log messages into structural form.

1.9.3 Automation and Robotics

- Automation is the process of using physical machines, computer software and other technologies to perform tasks that are usually done by humans. Robotics is the process of designing creating and using robots to perform a certain task.
- If we are looking to assemble our products more quickly, a robot may be able to help you speed up your factory operations. If you are looking to make repetitive administrative tasks more efficient, software driven process automation would be more appropriate.

There are 3 types of automation

(i) Fixed automation

(ii) Programmable automation.

(iii) Flexible automation

Uses of Automation

- In order to freeing human workers to more engaging (and better paying) tasks, robots can reduce scrap and waste through improved accuracy, increased efficiency and operate around the clock.
- All of these benefits add up to much higher productivity in manufacturing operations.

1.9.4 Machine Vision

- Machine vision in robotics is the ability of a computer to see, it employs one or more **video cameras, analog-to-digital conversion (ADC)** and **digital signal processing (DSP)**. The resulting data goes to a computer or robot controller. Machine vision is similar in complexity to voice recognition.
- According to **Automated Imaging Association** (AIA) machine vision encompasses all industrial and non-industrial applications in which a combination of hardware and software provide operational guidance to devices in the execution of their functions based on the capture and processing of images.
- Machine value allows a robot to see what it is doing, in a sense. Without machine vision the robot would be blind-only capable of repeating the same exact tasks over and over until it is reprogrammed.
- Machine vision systems are embedded components that use data extorted from online images to automatically guide manufacturing and production operations, such as go/no testing and quality control processes. These systems can also play a role in **inspection operations** and **automated assembly verification**.

Types of Machine Vision

(1) D vision system,

(2) D vision systems,

(3) Line Scan or Area Scan

(4) D vision systems.

- A 'Vision System' is defined as a **rebuilt solution** that is ready to configure and deploy, containing the sensor, processor and software. Imaging systems are built up from many components including optics, illumination, cameras and software.
- Robotics is the intersection of science, engineering and technology that produces machines called **Robots**. It substitutes or replicate for human action. Pop culture has always been fascinated with robots.
- It is important to note that machine vision is in no way associated with image processing which is a process where the output is another image. The information that is captured by machine vision will be translated into a form of data to make sense of it such as the identity, position and orientation of the objects that are being captured by the machine vision system.

1.9.5 Expert System (ES)

- Expert System are a computerized program that attemtps to imitate the reasoning processes and knowledge of experts in solving the specific problems.
- Expert System are of great important and interest to organization because of their ability to enhance productivity and to augment workforces in many special areas where human experts are becoming increasingly difficult to find and retain.

Scanned with CamScanner

1.9.6 Neural Computing

A neural network is a mathematical model of the way a brain functions. Neural networks are starting to have positive impact in many business disciplines.

1.9.7 Automatic Programming

- The goal of automatic programming is to create special program that act as intelligent tools to assist programmers and expedite each phase of the programming process.
- The ultimate aim of automatic programming is a computer system that could develop program itself, in response to and in accordance with the specifiations of a program developer.

1.9.8 Genetic Algorithm

- These are intelligent heuristic search methods that follows a process that simulates evolution in the computer.
- Genetic algorithm have been applied to many large-scale combinational mathematical programming problems such as large scale scheduling problem and even to producing police sketches of criminals.

1.9.9 Fuzzy Logic

- This extends the notions of logic beyond simple True/False to allow for partial or even continuous truths. Inaccurate knowledge and imprecise reasoning and important aspects of expertise in applying common sense to decision making situation.
- In fuzzy logic, degrees of set membership are important. In the traditional Boolean logic frame work, a car may be siad to skidding or not skidding when the break are applied.
- However, an expert driver can recognize the degree to which the car is skidding and can apply control according to variable amount of skidding. One of the commercial application of fuzzy logic was in producing superior antilock breaks.

1.9.10 Intelligent Agents

These are intelligent search methods that follows a process that simulates evolution in the computer. For example, in a specific problem, the solution is representing as chromosome, which generally contains a sequence of 0s and 1s, including the value of vector decisions variables.

For this string of chromosomes, the objective value can be compute.

1.10 AI PROBLEMS

Every new technology has its own share of problems. AI problems

AI problems

As an AI technology consumer and developer, we must know about both the merits and the challenges associated with the adoption of AI. Knowing these problems, helps the user to face the risks linked to the technology as well as take the full advantage of it.

It is very important how a developer addresses the AI problems in the real world. AI technologies must be accepted as a friend and not as a foe. We mention the top 10 potential artificial intelligence problems :

1. Lack of technical knowledge
2. The price factor
3. Data Acquisition and storage
4. Rare and expensive workforce
5. Issue of responsibility
6. Ethical challenges
7. Lack of computation speed
8. Legal challenges
9. AI Myths and expectation
10. Difficulty of assessing vendors'

(1) Lack of technical knowledge

- To implement AI applications in the enterprise, the organisation must have the knowledge of the current AI advancement and technologies and also its shortcomings.
- Enterprise requires a specialist to identify the roadblocks in the functioning process. Skilled human resources would help the teamwork with return on the tracking of adopting AI solutions.

(2) The price factor

- AI technology is a costly affair. Small and mid-sized organizations have to struggle a lot to adopt AI technologies.
- Even big firms like facebook, google, amazon have to allocate a separate budget for adopting and implementing AI technologies.

(3) Data Acquisition and storage

- One of the biggest AI problems is data acquisition and storage. Business AI systems depend on sensor data as its input.
- AI works best when it has a good amount of quality data available to it.

Scanned with CamScanner

- The algorithm becomes strong and performs well as the relevant data grows.
- The AI system fails badly when enough quality data is not fed into it. With small input variations in data quality has profound results on outcomes and predications.
- Hence there is a real need to ensure greater stability and accuracy in Artificial Intelligence.

(4) Rare and expensive workforce

- Adoption and deployment of AI technologies require specialists like data scientists, data engineer and other Subject Matter Experts (SMEs).
- These experts are expensive and rare in the current marketplace. Small and medium -sized enterprises fall short of their tight budget to bring in the man-power according to the requirement of the project.

(5) Issue of responsibility

- The implementation of AI application comes with great responsibility.
- Any specific individual must bear the burden of any sort of hardware malfunctions.
- Earlier, it was relatively easy to determine whether an incident was the result of the actions of a user, developer or manufacturer.

(6) Ethical challenges

- One of the major AI problems are the ethics and morality.
- These problems are yet to be solved. The way how the developers are technically grooming AI applications to perfection, so that it can flawlessly imitate human conversations.
- That way it becomes increasingly difficult to spot a difference between a machine and a real customer service representation.
- Artificial intelligence algorithm predicts based on the training given to it. The algorithm will label things as per the assumption of data it is based on.
- Therefore, we need to make sure that the algorithms are fair, especially when it is used by private and corporate individuals.

(7) Lack of computation speed

- AI, machine-learning and deep learning solutions require a high degree of computation speeds. And it is offered only by high - end processors.
- Larger infrastructure requirements and pricing associated with these processors has become a hindrance in their general adoption of the AI technology.
- In this regard, cloud computing environment and multiple processors running in parallel offer a big alternative to face to these computational requirements.
- As the volume of data available for processing grows exponentially, the computation speed requirements will grow with it. Now it is required to develop next-generation computational infrastructure solutions.

(8) Legal challenges

- An AI data governance application can cause legal challenges for the company.
- This is again one of the biggest artificial intelligence problems that a developer faces in the real world. An incorrect algorithm will always make unfavourable predictions.
- Problems like data branch can be a consequence of weak and poor data governance. To an algorithm, a user's 'personal identifiable information' (PII) acts as a feed stock which may slip into the hands of hackers.
- And the organisation may fall into traps of legal challenges.

(9) AI Myths and expectation

- There is a lot of discrepancy between the actual potential of the AI system and the expectations of this generation.
- Media says, Artificial intelligence, with its cognitive capabilities, will replace human's jobs.
- AI is just a tool that can operate only with the help of human brains. AI can replace human roles like automation of routine or common work optimizations of every industrial work, data-driven predictions etc.
- However, in most of the occasions, AI cannot substitute the caliber of the human brain. Not everything we hear about AI is true. AI is often over-hyped.

(10) Difficulty of assessing vendors

- In any emerging field, a technological procurement is quite challenging as AI is particularly vulnerable. Businesses face a lot of problems to know how exactly they can use AI effectively.
- The key lies in minimising the AI problems and maximising the benefits through the creation of an extensive technology adoption roadmap that understands the main capabilities of artificial intelligence.

Scanned with CamScanner

1.10.1 Programming Without and With AI

UQ. Differentiate Programming Without and With AI **(MU - Q. 1(d), Dec. May 18, 4 Marks)**

Parameters	Programming Without AI	Programming With AI
(1) Computer program	A computer program without AI can answer the Specific questions it is meant to solve.	A computer program with AI can answer the Generic questions it is meant to solve.
(2)Modifications	Modification in the program leads to change in its structure.	AI programs can absorb new modifications by putting highly independent pieces of information together. Hence we can modify even without affecting its structure.
(3) Complexity	Modification is not quick and easy. It may lead to affecting the program adversely.	Quick and Easy program modification.

1.11 CURRENT TRENDS IN AI

While the COVID - 19 pandemic impacted many aspects of how we do business, it did not diminish the impact of Artificial Intelligence (AI) on our everyday lives. AI remains a key trend when it comes to technology and innovations that will fundamentally change how we live, work, and play in the near future.

AI is the force behind many modern technological comforts that are now part of our day -to - day lives. With continuous research, technology has made massive developments in major fields such as health-care, retail, automotive, manufacturing and finance. AI is one essential component that transforms the digital age with high precision and accuracy. So, here there is an overview of what we can expect in the years to come.

1. Robotic Process Automation (RPA)
2. Conversational AI
3. The role of AI in healthcare
4. Increase in demand for ethical AI
5. AI for cyber security and knowledge breach
6. The Intersection of the Internet of Things with AI (AIOT)
7. Natural Language Processing (NLP)
8. Reinforcement Learning
9. Quantum AI
10. AI-Powered Business and Forecasting and Analysis
11. Edge computing
12. Rise of a hybrid work force

(1) Robotic Process Automation (RPA)

- To streamline business processes and reduce costs, they are turning to an evolving technology practice called as robotic process automation (RPA)
- RPA is aimed at the automation of business processes, governed by business logic, and organised inputs.
- RPA solution range from producing an automated email response to deploying thousands of bots. Each is programmed in an ERP system to automate rule-based tasks.

(2) Conversational AI

- Conversational AI increases the customer experience's reach, responsiveness and personalisation.
- To better understand what the human says and needs, AI uses natural language processing (NLD) and machine learning to provide a more natural, near human - level interaction.

(3) The role of AI in healthcare

- Big data has been extensively used to identify COVID patients and critical hot points.
- AI is already helping the health-care sector to a great degree with high accuracy besides, researchers have developed thermal cameras and mobile applications to collect data for healthcare organisations.

Scanned with CamScanner

- By leveraging data analysis and predicting various outcomes, AI can support healthcare facilities in several unique ways.
- AI instruments offer insights into human health and also recommend preventive steps to avoid the spread of diseases.
- AI solutions also help doctors remotely track the health of their patients, thereby advancing teleconsultation and remote care.

(4) Increase in demand for ethical AI

- This demand is at the top of the list of emerging developments in technology.
- Looking at how trends are rapidly changing, values-based customers and workers expect businesses to implement AI responsibly.
- Companies will actively choose to do business with partners committed to data ethics in the next few years.

(5) AI for cyber security and knowledge breach

- In the coming years, knowledge will grow and will be accessible, and digital data will be at greater risk of being compromised and exposed to hacking . AI will help deter cybercrimes in the future with improved cyber security measures.
- Fake digital activity that match criminal trends will be detected by the AI-enabled framework.

(6) The Intersection of the Internet of Things with AI (AIOT)

- There is hardly any boundary between AI and IOT. Although both technologies have individual characteristics, when used together, better and more unique possibilities open up.
- The ability of AI to gain insights from data quickly makes IOT solutions more intelligent.

(7) Natural Language Processing (NLP)

- NLP is one of the widely used applications of AI. N LP is used in Amazon, Alexa and Google Home.
- The need for writing or communicating with a screen has been eliminated by NLP as now humans can communicate with robots that understand their language.
- NLP is used for sentiment analysis, machine translation, process description, auto-video caption generation and chatbots is expected to increase.

(8) Reinforcement Learning

- Reinforced Learning (RL) is a specific application of deep learning. Its work is based on its experience to enhance the efficiency and effectiveness of data.
- Some cases of use of RL are robotics in planning business strategies, optimising advertisement content, automating industries, controlling aircraft, and making motion control robots.

(9) Quantum AI

- To measure the Qubits for use in supercomputers, advanced companies will begin using quantum supremacy. Because of quantum bits, quantum computers solve problems at a quicker pace than classic computers do.
- Also they assist in the interpretation of data and then forecast several unique trends.
- Quantum computers will help multiple organisations identify inaccessible issues and also predict meaningful solutions. Future computers will also be used in fields like healthcare, finance and chemistry.

(10) AI-Powered Business and Forecasting and Analysis

- AI solutions help in redefining business processing with real-time alerts.
- Content-intelligent technologies, along with AI-supportive practices, will assist digital workers to develop outstanding abilities.
- Such skills can help them cope with the automation of natural language, judgment, context formation, reasoning and data-related insights.

(11) Edge computing

- Edge computing provides gadgets with servers and data storage to access their devices and allows them to put data into them. It is defined as data processing in real-time and is more powerful than 'cloud computing services'.

Scanned with CamScanner

- There is another instance of edge computing that uses nodes. It is a mini-server located in the vicinity of a local telecommunications provider.
- Nodes help to build a bridge between the local service provider. It costs less, saves time and provides customers with fast service.

(12) Rise of a hybrid work force

- Post the COVID-19 pandemic, companies will change on to RPA bandwagon, which means that cognitive AI and RPA will be widely applied to cope with high volume, repetitive activities.
- If usages grow, the office will move to a hybrid workforce environment.
- The human workforce will work with various digital assistants. The emergence of a hybrid workforce will imply more collaborative experiences with AI.

1.12 APPLICATION AREAS OF ARTIFICIAL INTELLIGENCE

UQ. What are applications of AI?

(MU - Q. 1(a), May 17, 5 Marks)

(1) Logical artificial intelligence : What a program knows about the world in general the facts of the specific situation in which it must act, and its goals are all represented by sentences of some **mathematical logical language**. The program decides what to do by inferring that certain actions are appropriate for achieving its goals.

(2) Search : Artificial intelligence programs often examine large numbers of possibilities, **e.g.** moves in a chess game or inferences by a theorem proving program. Discoveries are continually made about how to do this more efficiently in various domains.

(3) Pattern recognition : When a program makes observations of some kind, it is often programmed to compare what it sees with a pattern. For example, a vision program may try to match a pattern of eyes and a nose in a scene in order to find a face. More complex patterns, e.g. in a natural language text, in a chess position, or in the history of some event are also studied.

(4) Representation : Facts about the world have to be represented in some way. Usually languages of mathematical logic are used.

(5) Inference : From some facts, others can be inferred. **Mathematical logical deduction** is adequate for some purposes, but new methods of non-monotonic inference have been added to logic since the 1970s. The simplest kind of **non- monotonic reasoning** is default reasoning in which a conclusion is to be inferred by default, but the conclusion can be withdrawn if there is evidence to the contrary. **For example**, when we hear of a bird, we can infer that it can fly, but this conclusion can be reversed when we hear that it is a penguin. It is the possibility that a conclusion may have to be withdrawn that constitutes the **non-monotonic character** of the reasoning. Ordinary logical reasoning is monotonic in that the set of conclusions that can be drawn from a set of premises is a monotonic increasing function of the premises.

(6) Common sense knowledge and reasoning : This is the area in which artificial intelligence is farthest from human-level, in spite of the fact that it has been an active research area since the 1950s. While there has been considerable progress, e.g. in developing systems of non-monotonic reasoning and theories of action, yet more new ideas are needed.

(7) Learning from experience : Programs learn from experience. The approaches to artificial intelligence based on connectionism and neural nets specialize in that. There is also learning of laws expressed in logic. Programs can only learn what facts or behaviours their formalisms can represent, and unfortunately learning systems are almost all based on very limited abilities to represent information.

Scanned with CamScanner

(8) Planning : Planning programs start with general facts about the world (especially facts about the effects of actions), facts about the particular situation and a statement of a goal. From these, they generate a strategy for achieving the goal. In the most common cases, the strategy is just a sequence of actions.

(9) Epistemology : This is a study of the kinds of knowledge that are required for solving problems in the world.

(10) Ontology : Ontology is the study of the kinds of things that exist. In artificial intelligence, the programs and sentences deal with various kinds of objects, and we study what these kinds are and what their basic properties are. Emphasis on ontology begins in the 1990's.

(11) Heuristics : A heuristic is a way of trying to discover something or an idea embedded in a program. Heuristic functions are used in some approaches to search to measure how far a node in a search tree seems to be from a goal. Heuristic predicates that compare two nodes in a **search tree** to see if one is better than the other, i.e. constitutes an advance toward the goal, and may be more useful.

(12) Genetic Programming : Genetic programming is a technique for getting programs to solve a task by making random LISP programs and selecting fittest in millions of generations.

Chapter Ends...

❑❑❑

Scanned with CamScanner

Module 2

CHAPTER 2

Intelligent Agents

Syllabus

Agents and Environments, The concept of rationality, The nature of environment, The structure of Agents, Types of Agents, Learning Agent.

Solving problem by Searching : Problem Solving Agent, Formulating Problems, Example Problems.

2.1 Intelligent agent 2-3

2.1.1 Characteristics of Intelligent Agent 2-3

UQ. Define Intelligent Agent. What are the characteristics of Intelligent Agent ? **(MU – Q. 1(a), May 18, 5 Marks)** 2-3

2.2 Rational Agent in AI 2-4

UQ. Define Rationality and Rational Agent. Give an example of rational action performed by any intelligent agent. **(MU – Q. 1(c), Dec. 2015, 5 Marks)** 2-4

2.2.1 Concept of Rationality 2-4

2.2.2 Omniscience and Rationality 2-4

2.3 Structure of Intelligent agents 2-5

2.3.1 Simple Reflex Agent 2-5

2.3.2 Model-based Reflex Agent 2-6

UQ. Explain Model based Reflex agent with block diagram **(MU – Q. 2(a), Dec. 19, 10 Marks, Q. 2(c), Dec. 16, 5 Marks, Q. 5(b), May 16, 5 Marks)** 2-6

2.3.3 A Goal-based Reflex Agent 2-6

UQ. Explain Goal Based with block diagram. **(MU – Q. 2(b), Dec. 18, Q. 2(B), Dec. 17, 10 Marks, Q. 1(d), May 17, 4 Marks)** 2-6

2.3.4 An Utility-based Reflex Agent 2-7

UQ. Explain Utility based agent with block diagram. **(MU – Q. 2(a), Dec. 19, 10 Marks, Q. 2(c), Dec. 16, 5 Marks, Q. 2(b), Dec. 18, Q. 2(B), Dec. 17, 10 Marks, Q. 1(d), May 17, 4 Marks, Q. 2(a), Dec. 19, 10 Marks, Q. 5(b), May 16, 5 Marks)** 2-7

2.3.5 Comparison of Model Based Agent and Utility Based Agent 2-8

Scanned with CamScanner

UQ. Compare of Model based Agent and Utility based Agent. **(MU - Q. 4(b), May 17, 4 Marks)** 2-8

2.3.6 Comparion Model Based Agent with Goal Based Agent 2-8

UQ. Compare Model Based Agent with Goal Based Agent. **(MU - Q. 1(a), May 19, 5 Marks)** 2-8

2.4 Learning Agents 2-8

UQ. Explain Learning agent with block diagram. **(MU - Q. 4(c), May 19, 5 Marks, Q. 1(b), May 18, Q. 1(D), Dec. 17, Q. 4(b), May 17, 10 Marks, Q. 5(b), May 16, 5 Marks)** 2-8

UQ. What are the basic building blocks of Learning Agent ? Explain each of them with a neat block diagram. **(MU - Q. 6(b), May 16, 10 Marks)** 2-8

UQ. What will be the job of each of the components (performance element, Learning element, Critic and problem generator) of learning agent ? **(MU - Q. 1(c), Dec. 18, 5 Marks)** 2-8

UQ. Draw and illustrate the Architecture of Learning agent. Describe each of its component w.r.t. Medical diagnosis system. **(MU - Q. 2(a), May 19, 10 Marks)** 2-8

2.4.1 Differentiate between S-R Agent Model and Goal-Based Agent 2-9

2.5 Environment 2-9

2.5.1 Definition of Environment 2-9

2.5.2 Three Types of Environment 2-10

2.5.3 The Task Environment 2-10

2.5.4 Properties of Task-Environment 2-11

UQ. Write short notes on: Properties of Agent Task Environment. **(MU - Q. 6(a), May 19, 5 Marks, Dec. 15, 3 Marks, Q. 5(b), Dec. 15, 5 Marks)** 2-11

2.5.5 Examples of Task Environments and their Characteristics 2-12

2.6 Solving problems by searching/State Space Search 2-12

UQ. Explain any AI problem with suitable example. **(MU - Q. 1(d), Dec. 18, 4 Marks)** 2-12

2.7 Problem Solving Agents 2-13

2.7.1 Simple Problem-Solving Agent 2-13

2.7.2 Well - Defined Problems and Solutions 2-14

2.7.2(A) A Problem can be Defined by Five Components 2-14

2.7.3 Formulating Problems 2-14

2.8 Example problems 2-15

2.8.1 Toy-problem Example 2-15

2.8.2 The 8 Puzzle Problem 2-16

2.8.3 8-Queens Problem 2-17

2.9 Examples of Real world problem 2-17

2.9.1 Route-Finding Problem 2-17

2.9.2 The Travelling Salesperson Problem (TSP) 2-18

UQ. Give the initial state, goal test, successor function, and cost function for the Travelling salesman problem **(MU - Q. 1(b), Dec 17, 5 Marks)** 2-18

2.9.3 A VLSI Layout 2-18

2.10 Problem Characteristics of IA 2-18

- **Chapter Ends.** 2-19

Scanned with CamScanner

2.1 INTELLIGENT AGENT

An agent is just something that acts (agent comes from the Latin agree, to do).

In artificial intelligence, an **intelligent agent (IA)** is an autonomous entity which observes through sensors and acts upon an environment using actuators (i.e. it is an agent) and directs its activity towards achieving goals.

- Intelligent agents may also learn or use knowledge to achieve their goals.
- They may be very simple or very complex, **Example :** a reflex machine such as a thermostat is an intelligent agent.
- An agent is anything that can perceive its environment through sensors and acts upon that environment through effectors.

  Agent's structure can be viewed as :

  (1) Agent = Architecture + Agent program.

  (2) Architecture = the machinery that an agent executes on.

  (3) Agent program = an implementation of an agent function.
- IA like Rahul and Gopal are examples of intelligence as they use sensors to perceive a request made by the user and automatically collect the data from the internet without the user's help.
- That can be used to gather information about its perceived environment such as weather and time. Thus, an intelligent agent is an autonomous entity which acts upon an environment using sensors and actuates for achieving goals. An intelligent agent may learn from the environment to achieve their goals.
- The term **'percept'** means the agent's perceptual inputs at any given instant. An agent's percept sequence is the complete history of everything that the agent has perceived. We illustrate this idea in the Fig. 2.1.1.

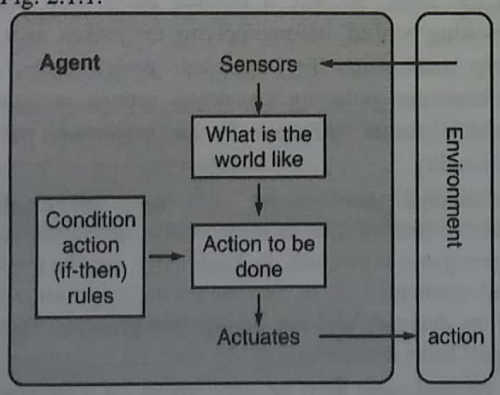


Fig. 2.1.1 : Agent's interact with environments through sensors and actuators

Thus, an agent's behaviour can be described by the **'agent function'** that maps any given percept sequence to an action.

- An intelligent agent is a programme that can make decisions or perform a service based on its environment, user input and experiences.
- These programs can be used autonomously to gather information on a regular, programmed schedule or when prompted by the user in real time.
- IA may be simple or complex - a **thermostat** is considered as an example of an intelligent agent, as is a human being, as is any system that meets the definition, such as firm, a state or total quantity.

Module 2

2.1.1 Characteristics of Intelligent Agent

UQ. Define Intelligent Agent. What are the characteristics of Intelligent Agent ?

(MU - Q. 1(a), May 18, 5 Marks)

1. **Mobility :** Using computer-Network, Intelligent agents engaged in e-commerce gather information until search parameter are complete.
2. **Goal-oriented :** Intelligent agents carry out the particular task provided by **'user statement of goals'**. It moves around from one machine to another and can react in response to their environment and takes initiative to exhibit goal directed behaviour.
3. **Independent :** Intelligent agent is self-dependent, in the sense that it functions an its own without human intervention. It makes decisions on its own and initiate them. It communicates independently with data or information and other agents and achieves the objectives and tasks an behalf of the user.
4. **Intelligent :** Intelligent agent can collect data more intelligently. They can reason out things based on the existing knowledge of its user and environment on past experiences intelligently. To evaluate condition in the external environmental intelligent agents follow present rules.
5. **Reduce net traffic :** Agents communicate and co-operate with other agents quickly. This way, they can perform the tasks, such as information searches quickly and efficiently. And network traffic gets reduce thereby.
6. **Multiple tasks :** Multiple tasks can be performed by an intelligent agent simultaneously. This helps the human from monotonous clerical work.

Scanned with CamScanner

2.2 RATIONAL AGENT IN AI

UQ. Define Rationality and Rational Agent. Give an example of rational action performed by any intelligent agent.

(MU - Q. 1(c), Dec. 2015, 5 Marks)

Rational agent in AI is an agent which has **clear preference, models uncertainty**, and acts in a way to **maximize its performance measure** with all possible actions. AI is about crating rational agents to use for same theory and decision theory for various real-world phenomenons.

Intelligent agents can be grouped into the following classes based on their **degree of perceived intelligence** and capability.

Agent	Example
Simple reflex agents	The vacuum promises to sense dirt and debris on floors and clean those areas accordingly. This is an example of a simple reflex agent that operates on the condition (dirty floors) to initiate an action (vacuuming).
Model-based agent example	Some examples of items with model-based agents aboard include **Roomba vacuum cleaner** and the **autonomous car** known as **Waymo**. Both interact with their environments by using what they know, an internal model of the world and their on-board sensors as well, to make **moment-to-moment decisions** about their actions.
Goal-based agent example	Google's **Waymo driverless cars** are good examples of a goal-based agent when they are programmed with an end destination, or goal, in mind. The car will then 'think' and make the right-decisions in order to deliver the passenger where they intend to go.

2.2.1 Concept of Rationality

The concept of rationality depends upon the following four things :

(i) The performance measure that defines the criterion of success.

(ii) Prior knowledge of the environment

(iii) Agent can perform the action

(iv) The percept sequence of the agent.

Example of Vacuum cleaner Agent

Let us consider the above example of a vacuum-cleaner agent. It cleans a square if it is dirty and moves to the next square if not;

(i) The performance measure is that it cleans a square if it is dirty, at each time.

(ii) The area of space of the floor or **'geography'** of the environment is known as **'a priori'**. Clean squares stay clean and sucking cleans the current square. The **'Left'** and **'Right'** actions move the agent left and right.

(iii) Left, right and suck are the only available actions of the agent.

(iv) The agent correctly perceives its location, if there is dirt. Under these circumstances, we say that the agent is **'rational'**.

2.2.2 Omniscience and Rationality

(1) An omniscient's agent knows the exact outcome of its actions and act accordingly, but in practice it is impossible. In omniscience there is a total perfection. Rationality is not same as perfection. Rationality tries to maximise the expected performance while perfection gives actual total performance.

(2) Suppose I apply urgent brakes to my BMW car, as I want to take a halt. Car will come to halt instantly but the other car behind me may hit my car.

Does the definition of rationality say that it is OK to apply sudden brakes? First, it is not rational to apply sudden brakes, given the **uninformative percent sequence**; the risk of accident without looking behind is too great. Second, a rational agent will mention **'looking behind'** before applying the brakes, as it may help maximize the expected performance. This information gathering i.e. doing actions in order to modify future precepts is an important part of rationality.

(3) A rational agent is not only supposed to gather information but also to learn what it perceives. As the agent gains experience, its knowledge may be modified and augmented. If the agent relies on 'prior' knowledge of its designer and not on its own precepts, then the agent lacks autonomy.

(4) A rational agent must be autonomous - it is supposed to learn what it can, in order to compensate for partial prior knowledge. For example, a vacuum-cleaning

Scanned with CamScanner

agent that can learn to locate when and where additional dirt will appear works better than one does not learn. After having sufficient experience of its environment, the behaviour of a rational agent can become effectively independent of its prior knowledge.

2.3 STRUCTURE OF INTELLIGENT AGENTS

(1) The IA structure consists of three main parts: architecture, agent function and agent programme. Architecture refers to machinery or devices that consist of **actuators and sensors**. The IA executes on this machinery. The tool allows the adjusting of image details and clarifies. Using this tool we can obtain great detail or get a smoother picture with less detail. This is the main tool to increase the contrast of the image and visualize more details of the image.

(2) A software agent has file contents, received network packages which act as sensors and display on the screen, files, sent network packets acting as actuators. A human agent has eyes, ears and other organs which act as sensors, and hands, legs, mouth and other body parts acting as actuators.

(3) The learning element is responsible for improvements. This can make a change on any of the knowledge components in the agent. One way of learning is to observe parts of successive steps in the percept sequence; from this the agent can learn how the world evolves.

(4) Agency is the capacity of individuals to act independently and to make their own free choices. The structure versus agency debate may be understood as an issue of socialization against autonomy in determining whether an individual acts as a free agent or in a manner dictated by social structure.

So far we have discussed the behaviour of the agent, i.e. the **action that is performed after a given sequence of precepts**. Now we begin to study how the inside of the agent works. The job of 'Artificial intelligence' is to design an agent program that designs the agent function-the function from precepts to action. This is 'architecture' or 'structure' of intelligent agent.

We mention four basic kinds of agent programs that cover the principles of almost all intelligent systems. They are :

(1) Simple reflex agents
(2) Model-based reflex agents
(3) Goal-based agents; and
(4) Utility-based agents

Each of these programs combine particular component in particular ways to generate actions. Then we shall see how to convert all these agents into learning agents. This will improve the performance of their components so as to have better actions. Then we shall also describe variety of ways in which the components themselves can be represented within the agent.

2.3.1 Simple Reflex Agent

Module 2

(1) In artificial intelligence, a simple reflex agent is a **'type of intelligent agent'** that performs actions based solely on the current situation, with an intelligent agent generally being one that perceives its environment and then acts. The agent cannot learn or take past percepts into account to modify its behavior.

(2) The 'simple reflex agent' works on **condition-action rule**, which means it **maps the current state to action**. Since 'simple reflex agent' is based on the present condition so it is called as condition-action rule. Problems with simple reflex agents are: very limited intelligence. No knowledge of non-perceptual parts of the state is required.

(3) Usually too big to generate and store. The simple reflex based agent is designed only to respond to the currently occurring problem. If the knowledge of the entire environment is given, then simple reflex agent is perfectly rational.

(4) This agent selects actions based on the agent's current perception, and not based on past perceptions. **For example**, if a mars lander found a rock in a specific place it needed to collect then it would collect it, **if** it was a 'simple reflex agent' **then** if it found the same rock in a different place, it would still pick it up, as it does not take into account that it has already picked it up.

(5) This is useful when a **quick automated response** is needed. Humans have a very similar reaction to fire for example; our brain pulls our hand away without thinking about any possibility that there could be danger in the path of your arm this is called as a **reflex action**.

(6) This kind of connection-action rule, written as **if** hand is in fire, **then** pulls it away. The 'simple reflex agent' has a library of such rules so that if a certain situation should arise, it is in the set of condition-action rules, the agent will know how to react with **minimal reasoning**. These agents are simple to work with but have very limited intelligence, such as picking up 2 rock samples. Refer Fig. 2.3.1.

Scanned with CamScanner

(rectangles) : To represent the current internal state of the agent's decision process.

(ovals) : To represent the background information used in the process.

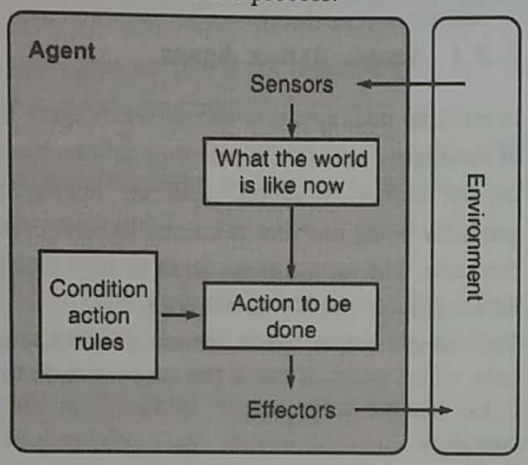


Fig. 2.3.1 : Simple Reflex Agent in AI

Function : SIMPLE–REFLEX–AGENT (percent) returns on action.
Static : Rules, a set of condition – action rules.
State ← INTERPRET – INPUT (percept)
Rule ← RULE – MATCH (State, rules)
Action ← RULE – ACTION (rule)
Return Action

2.3.2 Model-based Reflex Agent

UQ. Explain Model based Reflex agent with block diagram.
(MU - Q. 2(a), Dec. 19, 10 Marks, Q. 2(c), Dec. 16, 5 Marks, Q. 5(b), May 16, 5 Marks)

(1) A model-based reflex agent needs **memory for storing the precept history**; it uses the percept-history to help to reveal the current unobservable aspects of the environment. An example of this IA class is the **self-screening mobile-vision**, where it is necessary to check the percept history to fully understand how the world is evolving.

(2) Model-based reflex agents are made to deal with **partial accessibility**; they do this by keeping track of the world it can see now. It does this by keeping an internal state that depends on what it has seen before, so it holds information on the unobserved aspects of the current state.

(3) A simple reflex agent selects actions based on the **agent's current perception of the world and not based on past perception**. A model based reflex agent is designed to deal with partial accessibility. They do this by keeping track of the part of the world it can see now.

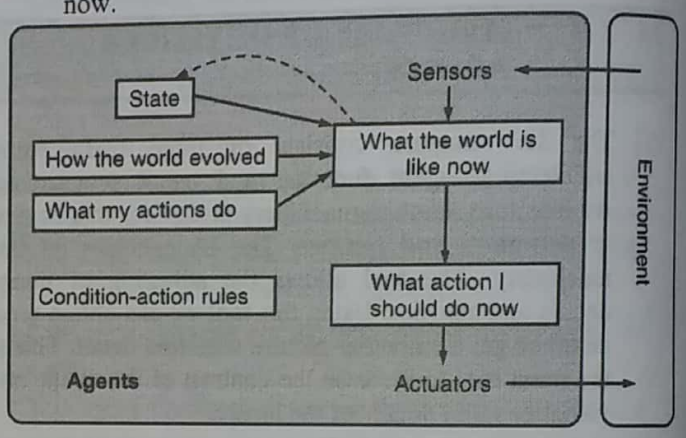


Fig. 2.3.2 : Model-based reflex agent in AI

Function REFLEX-AGENT WITH-STATE (percept)
Returns an action.
Static : State, a description of the current world state
rules, a set condition-action rules.
action, the most recent action, initially none.
State ← UPDATE – STATE (State, action, percept)
Rule ← RULE – MATCH (State, rules)
Action ← RULE – ACTION (rule)
Return Action

2.3.3 A Goal-based Reflex Agent

UQ. Explain Goal Based with block diagram.
(MU - Q. 2(b), Dec. 18, Q. 2(B), Dec. 17, 10 Marks, Q. 1(d), May 17, 4 Marks)

(1) A goal-based reflex agent has a goal and has a strategy to reach that goal. All actions are taken to reach this goal.

(2) More precisely, from a set of possible actions, it selects the one that improves the progress towards the goal (not necessarily the best one).

(3) A goal-based agent has an agenda. Unlike a simple reflex agent that makes decisions based solely on the current environment, a goal-based agent is capable of thinking beyond the present moment to decide the best actions to take in order to achieve its goal.

Scanned with CamScanner

(4) Goal-based agents expand the capabilities of the model-based agent by having the **'goal'** information. They choose an action, so that they can achieve the goal.

(5) These agents may have to consider a long sequence of possible actions, before deciding whether the goal is achieved or not.

(6) A goal-based algorithm uses **searching and planning** to act in the most-efficient solution to achieve the goal.

Remark : Conclusion is a statement to Goal-based agent, but is not considered as goal-based agent.

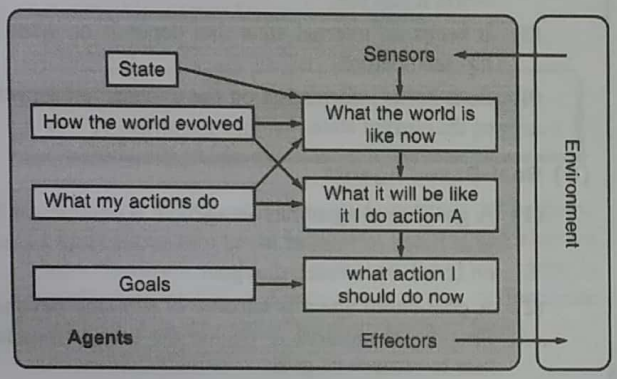


Fig. 2.3.3 : Goal-based agent diagram

2.3.4 An Utility-based Reflex Agent

UQ. Explain Utility based agent with block diagram.
(MU - Q. 2(a), Dec. 19, 10 Marks, Q. 2(c), Dec. 16, 5 Marks, Q. 2(b), Dec. 18, Q. 2(B), Dec. 17, 10 Marks, Q. 1(d), May 17, 4 Marks, Q. 2(a), Dec. 19, 10 Marks, Q. 5(b), May 16, 5 Marks)

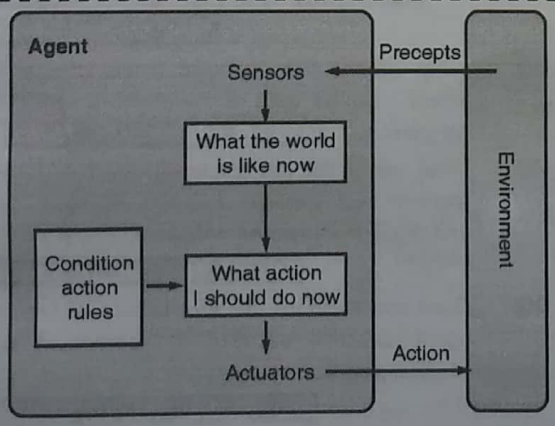


Fig. 2.3.4

(1) An utility-based reflex agent is like the goal-based agent but with measure of **'how much happy'** an action would make it rather than the **goal-based binary feedback [happy, unhappy].**

(2) This kind of agents provides the best solution. An example is the route recommendation system which solves the 'best' route to reach a destination.

(3) The agents which are developed having their end-uses as building blocks are called **utility-based agents.** When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used. They choose actions based on a preference (utility) for each state.

(4) Goal-based agents are important as they are used to expand the capabilities of the model-based agent by having the 'goal' information.

(5) They choose an action, in order that they will achieve the goal. Utility based agent-act based not only goals but also the samples, thanks to achieving the goal.

(6) The utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action. The utility function maps each state to a real number to check how efficiently each action achieves the goals.

(7) In artificial intelligence, utility function **assigns values to certain actions** that the AI can take. An AI agent's preferences over all possible outcomes can be captured by a function that maps the outcomes to a utility value, the higher the number, the more that agent likes that outcome.

(8) In Economics, the utility function **measures the welfare or satisfaction of a consumer as a function of the consumption of real goods**, such as food or clothing. Utility function is widely used in rational choice theory to analyze human behavior.

(9) Utility theory bases its beliefs upon individual's preferences. It is a theory postulated in economics to explain behavior of individuals based on premise; people can consistently rank order their choices depending upon their preferences. We can state that individual's preferences are intrinsic.

Module 2

Scanned with CamScanner

2.3.5 Comparison of Model Based Agent and Utility Based Agent

UQ. Compare of Model based Agent and Utility based Agent. **(MU - Q. 4(b), May 17, 4 Marks)**

Sr. No.	Model based Agent	Utility based Agent.
1.	Goal-based agents are very important as they are used to expand the model-based agent by having the goal information.	Utility based agent-act based not only goals but also the simplest thanks to achieving the goal.
2.	They choose an action in order that they will achieve the goal.	A utility based agent makes decisions based on the maximum utility of its choices.
3.	A model based reflex agent that uses percept history and internal memory to make decisions about the 'model' of the world around it.	It's usefulness (utility) of the agent that makes itself distinct from its counterparts.
4.	Internal memory allows these agents to store some of their navigation history to help understand things about their current environment even when everything that they need to know cannot be directly observed.	A goal-based agent makes decisions based simply on achieving a set goal. Suppose you want to travel from Pune to Mumbai. Mumbai is the goal and the goal based agent will get you there.
5.	Model-based agent uses GPS to understand its location and predict upcoming drivers.	But if you come across a closed road, the **utility-based** agent will analyse other routes to get you there. And it will select the best option for maximum utility. Hence, the utility-based agent is a step above the goal-based agent.
6.	Model-based reflex agents are made to deal with partial accessibility. It does this by keeping an internal state that depends on what it has seen before. So it holds information on the unobserved aspects of the current state.	Utility based agent is more agile and sophisticated since it has some decision making capabilities.

2.3.6 Comparison Model Based Agent with Goal Based Agent

UQ. Compare Model Based Agent with Goal Based Agent. **(MU - Q. 1(a), May 19, 5 Marks)**

(A) Model Based Agents

(1) Model based reflex agents are made to deal with partial accessibility;

(2) They do this by keeping track of the part of the world it can see;

(3) It keeps an internal state that depends on what it has seen before.

(4) So it holds information on the unobserved aspects of the current state.

(B) Goal-Based Agents

(1) A goal-based agent has an agenda. It operates on a goal based in front of it and makes decisions based on how best to reach that goal.

(2) A goal-based agent is capable of thinking beyond the present moment to decide the best actions to take to achieve its goal.

(3) A goal based agent operates as a **search and planning** function.

(4) It targets the goal ahead and finds the right action in order to reach it.

2.4 LEARNING AGENTS

UQ. Explain Learning agent with block diagram. **(MU - Q. 4(c), May 19, 5 Marks, Q. 1(b), May 18, Q. 1(D), Dec. 17, Q. 4(b), May 17, 10 Marks, Q. 5(b), May 16, 5 Marks)**

UQ. What are the basic building blocks of Learning Agent ? Explain each of them with a neat block diagram. **(MU - Q. 6(b), May 16, 10 Marks)**

UQ. What will be the job of each of the components (performance element, Learning element, Critic and problem generator) of learning agent ? **(MU - Q. 1(c), Dec. 18, 5 Marks)**

UQ. Draw and illustrate the Architecture of Learning agent. Describe each of its component w.r.t. Medical diagnosis system. **(MU - Q. 2(a), May 19, 10 Marks)**

Scanned with CamScanner

(1) A learning agent is an agent **capable of learning from experience.** It has the capability of **automatic acquisition and integration** into the system. Any agent designed and expected to be successful in an uncertain environment is considered to be a learning agent.

(2) A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning. The learning algorithm gains **feedback from the critic** on how well the agent is doing and determines how the performance element should be modified it at all to improve the agent.

The Building blocks of learning agent are :

1. Critic	2. Learning Agent
3. Performance element	4. Problem generator

▶ **1. Critic :** The critic element determines the outcome of the action and gives feedback. The learning element uses feedback from the critic on how the agent is performing and then determines how the performance element should be modified to do better in future.

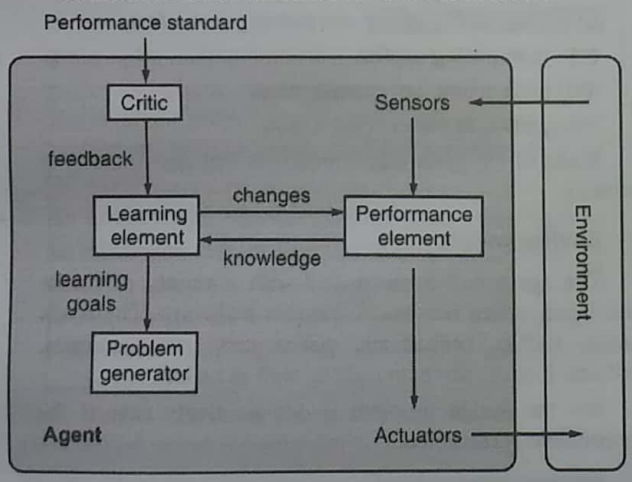


Fig. 2.4.1 : Block Diagram of General Learning Agent

▶ **2. Learning Agent :** The learning agent takes the feedback from the critic element and figures out how to make the action better next time. Thus, it is responsible for making improvements.

▶ **3. Performance element :** The performance element is responsible for selecting external actions. The performance element takes in percepts and decides on actions; it is the entire agent. The performance element chooses the action and later it shifts to a new action based on feedback and suggestions for improvements. Learning is a process of modification of each component of the agent with the available feedback of information, thereby improving the overall performance of the agent.

▶ **4. Problem generator :** The generator is tasked with developing new experiences for the learning agent to try; this is the piece that helps the agent continue to help.

Thus we observe that agents have a variety of components, and hence there is a great variety among learning methods. In short, we can say: Learning in intelligent agent is a process of modification of each component of the agent with the available feedback of information, thereby improving the overall performance of the agent.

Module 2

2.4.1 Differentiate between S-R Agent Model and Goal-Based Agent

Simple Reflex Agents

(1) They choose actions only based on the current percept.

(2) They are rational only if a correct decision is made on the basis of current precept.

(3) Their environment is completely observable.

- **Condition-action rule :** It is a rule that maps a state (condition) to an action.
- **Goal based agents :** They choose their actions in order to achieve goals. Goal-based approach is more flexible than reflex agent since the knowledge supporting a decision is explicitly modelled, thereby allowing for modifications.
- **Goal :** It is the description of desirable situations.

2.5 ENVIRONMENT

The natural environment or natural world encompasses all living and non-living things occurring naturally, meaning that they are not artificial.

2.5.1 Definition of Environment

1. The circumstances, objects or conditions by which one is surrounded.
2. The complex of physical, chemical or biotic factors (**such as climate, soil and living things**) that act upon an organism or an ecological community and ultimately determine its form and survival.

Scanned with CamScanner

3. The sum of the total of the elements, factors and conditions in the surroundings which may have an impact on the development, action or survival of an organism or group of organisms.

2.5.2 Three Types of Environment

The three types of environment are as follows :

(I) The physical environment
(II) Social environment
(III) Cultural environment.

(I) Physical Environment

(1) It consists of all components provided by nature and hence can be called as the natural environment. It is also referred to as physical environment, as it pertains to the physical requirements of life.

(2) These physical conditions are not dependent on the existence of humans. It includes natural resourses, the earth's surface, mountains, plains, land, water, deserts, storms, cyclones, volcanoes, oceans, climatic factors, and so on.

(3) It also refers to biological situations such as complexities associated with plants and animals.

(II) Social Environment

(1) Social environment is the immediate environment or surroundings associated with an individual.

(2) Social environment includes all the social practices, government, occupational structure, family, friends, acquaintances, religioius beliefs. It also refers to trends and developments in changes in attitudes, behaviour, and values in society.

(3) It is closely related to population, life-style, culture, tastes, customs and traditions. Social environments allow us to receive feedback, comments, and suggestions in real time concerning our developmental goals and the observations of our behaviours.

(4) Timely feedback is a critical component for personal development and provides for assessing developmental gaps and progress.

(III) Cultural Environment

(1) Cultural environments are environments shaped by human activities, such as cultural landscapes in the countryside, forests, urban areas and cities, fixed archaeological structures on land or water, constructions and built environments from different ages, along with bridges, roads and power lines.

(2) Cultural environment influences the personality traits of people in the community as well as their ideologies.

(3) The factors of cultural environment are language, social norms, religion, ethics, socio-economics, traditions, social regulations, nationalism aethetics, material culture, attitudes, values and social organisation.

Now we consider the types based on the way the environment appears to the agent.

2.5.3 The Task Environment

In task environment, we consider PEAS (Performance, Environment, Actuators, Sensors) descriptions. In designing an agent, the first step must be to specify the task environment.

Let us consider a problem : an automated taxi-driver. We mention the PEAS-description for the taxi's task environment.

1. Performance Measure

The desirable qualities are :

(i) correct destination
(ii) minimising violations of traffic laws,
(iii) minimising disturbances to other drivers,
(iv) maximising safety and passenger comfort,
(v) maximising profits,
(vi) minimising fuel consumption,
(vii) getting to correct destination.

Some of the goals may conflict, in that case tradeoff is required.

2. Environment

The agent will have to deal with a variety of roads; rural lanes, urban free ways, express highways. The roads contain traffic, pedestrians, police cars, stray animals, potholes. It could drive on right as well as on left.

But the design problem is comparatively easy if the environment is restricted.

3. Actuators

The actuators include control over the engine through steering, accelerator and braking. Also output to a display screen or voice synthesizer to communicate with other vehicles and also passengers and also to communicate in rough language, if so required.

4. Sensors

Sensors include to detect distances to other cars and obstacles, to control the vehicle on curves. It will also need engine, fuel and electrical system sensors.

Scanned with CamScanner

2.5.4 Properties of Task-Environment

UQ. Write short notes on: Properties of Agent Task Environment.
(MU - Q. 6(a), May 19, 5 Marks, Dec. 15, 3 Marks, Q. 5(b), Dec. 15, 5 Marks)

The task-environment in AI is vast. So, we categorise the task-environments in a small number of dimensions. First we enlist the dimensions :

(I) Fully observable versus partially observable
(II) Single agent Vs. Multiagent
(III) Static Vs-Dynamic
(IV) Deterministic Vs. Stochastic
(V) Episodic Vs. Sequential
(VI) Discrete Vs Continuous
(VII) Known Vs Unknown
(VIII) Accessible Vs Inaccessible

(I) Fully observable versus partially observable

(MU - May 19, Dec. 15)

- If the agent sensors describes the environment at each point in detail, then the environment is fully observable. Fully observable environments are convenient because agent need not maintain track of the world.
- An environment may be partially observable because an automated taxi cannot see what other drivers are thinking. A fully observable environment is also termed as accessible environment, while partially observable environment is termed as maccessible environment.

(II) Single agent Vs. Multiagent

(MU - May 19, Dec 15)

- An agent playing on a violine is a single-agent environment while an agent playing a chess is in a two-agent environment.The agent design problems in the multi-agent environment are different from single agent environment.
- Playing tennis against the ball is a single agent environment. The vacuum cleaning environment is single agent environment while chess is a two-agent environment.
- The expert agent where an agent acts like an expert assistant to a user attempting to fulfil some tasks on a computer is multiple-agent environment.

(III) Static Vs-Dynamic

(MU - May 19, Dec 15)

- If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment, else it is called a static environment.
- Static environment are easy to deal because an agent does not need to continue looking at the world while deciding for an action.
- But for dynamic environment, agents need to keep looking at the world at each action.
- Taxi driving is an example of a dynamic environment whereas playing on Piano is an example of a static environment. If the environment does not change with the passage of time, but the agent performance changes by time, then it is semi-dynamic.

Playing card-games is a perfect example of partially observable environment where a player is not aware of the card in the opponent's hand.

(IV) Deterministic Vs. Stochastic

(MU - May 19, Dec. 15)

- If an agent's current state and selected action can completely determine the next state of the environment, then such environment is called a deterministic environment.
- A stochastic environment is random in nature and cannot be determined completely by an agent.
- In a deterministic, fully observable environment, agent does not need to worry about uncertainty.

For a given current state and action executed by agent, the next state or outcome cannot be exactly determined, e.g., if agent kicks the ball in a particular direction, then the ball may or may not be stopped by other players.

Chess : the board is fully observable, so are the opponent's moves.

Driving : the environment is partially observable because what is around the corner is not known.

(V) Episodic Vs. Sequential

(MU - May 19, Dec. 15)

- In an episodic environment, there is a series of one-shot actions, and only the current percept is required for the action. But in sequential environment, an agent requires memory of past actions to determine the next best actions.
- Episodic is an environment where each state is independent of each other. The action on a state has nothing to do with the next state.

Module 2

Scanned with CamScanner

- The sequential environment is an environment where the next state is dependent on the current action. So agent current action can change all of the future states of the environment.
- An AI that looks at radiology images to determine if there is a sickness, is an example of an episodic environment. One image has nothing to do with the next state. Chess with a clock is fully sequential.

(VI) Discrete Vs Continuous

(MU - May 19, Dec. 15)

- If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment.
- A chess game comes under discrete environment as there is a finite number of moves that can be performed.
- A self-driving car is an example of a continuous environment. Input from digital cameras is discrete.

(VII) Known Vs Unknown

(MU - May 19, Dec. 15)

- Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.
- In a known environment, the results for all actions are known to the agent.
- While in unknown environment, agent needs to learn how it works in order to perform an action.
- It is quite possible that a known environment to be partially observable

  For example, in Rummy card game I know the rules of the game but unable to see the card that is not turned over. An unknown environment is fully observable in a video game, the entire game is visible on the screen, but I do not know what the buttons **do till I try them.**

(VIII) Accessible Vs Inaccessible

(MU - May 19, Dec. 15)

- If an agent can obtain complete and accurate information about the state's environment then such an environment is called an Accessible environment else it is called inaccessible.
- An empty room whose state can be defined by its temperature is an example of an accessible environment.

Now ; we enlist the properties of a number of familiar environments :

2.5.5 Examples of Task Environments and their Characteristics

Task environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi-dynamic	Continuous
Chess without clock	Fully	Multi	Stochastic	Sequential	Static	Discrete

2.6 SOLVING PROBLEMS BY SEARCHING/STATE SPACE SEARCH

UQ. Explain any AI problem with suitable example.

(MU - Q. 1(d), Dec. 18, 4 Marks)

1. Defining problem and search

(i) Problem (or problem space) refers to the entire range of components that exist in the process of finding a solution to a problem.

(ii) It is complete set of states including start and goal states, where the answer of the problem is to be searched.

(iii) Problem : It is a question to be solved.

(iv) A problem has four components : initial state, goal test, set of actions, path cost.

(v) A set of rules which describe the actions available.

If we are solving a problem, we are usually looking for some solution which will be the best among others. The space of all feasible solutions (i.e., the set of solutions among which the desired solution exists) is called search (or search space or state space). Each point in the search space represents one possible solution.

Scanned with CamScanner

2. Define State and State Space

(i) A state space is the set of all possible configurations of a system.

(ii) It is a useful abstraction for reasoning about the behaviour of a given system.

(iii) It is widely used in the fields of artificial intelligence and game theory.

(iv) In the theory of dynamical systems, a discrete system defined by a function **f**, the state space of the system can be modelled as a directed graph where each possible state of a dynamical system is represented by a vertex, and there is a directed edge from a to b if and only if **f(a) = b**. This is known as a state diagram.

(v) For a continuous dynamical system defined by a function f, the state space of the system is the image of f.

(vi) State spaces are useful in computer science as a simple model of machines.

(vii) Formally, a state space can be defined as tuple [N, A, S, G] where, N is a set of states

A is a set of arcs connecting the states

S is a nonempty subset of N that contains start stater.

G is a non-empty subset of N that contains the goal states.

State spaces can be either infinite or finite, and discrete or continuous.

3. Defining the problem as state space search

(i) State space search is a process used in the field of computer science, including artificial intelligence (AI).

(ii) In which successive 'configurations' or 'states' of an instance are considered, with the intention of finding a goal state with the desired property.

(iii) The set of states forms a graph where two states are connected it one state can be transformed to another state by the given operation an states.

(iv) State space search differs from traditional computer science search methods because the state space is implicit.

(v) The typical state space graph is much too large to generate.

(vi) Instead nodes are generated as they are explored, and then discarded.

(vii) A solution may consist of the goal state itself, or of a path from some initial state to the goal state.

(viii) A state space problem

P = (S, A, S, T) consists of a set of states S, an initial state $s \in S$, a set of goal states $T \subseteq S$, and a finite set of actions

$A : \{a_1, a_2, ..., a_{n3}\}$ where each

$a_i : S \rightarrow S$ transforms a state into another state.

2.7 PROBLEM SOLVING AGENTS

1. The problem - solving agent performs by defining problems and several solutions. Problem solving agent employees a number of techniques such as a tree, heuristic algorithms to solve a problem.
2. The reflex agent of AI directly maps states into action. When the state of mapping is too large, then the problem gets dissolved and sent to a problem - solving domain. It breaks the large stored problem into smaller storage area and resolves one by one. The final action will be the desired outcome.
3. Intelligent agent organises finite number of steps to formulate a target or goal which require some action to achieve the goal.
4. Problem formulation is one of the most important steps of problem-solving. Agent has to decide what action to be taken to achieve the goal.

   Components to formulate the associated problem :

   (i) Initial state : In this state, new methods also initialise problem solving by a specific class.

   (ii) Action : Here the agent performs all possible actions with a specific class taken from the initial state

   (iii) Transition : Here the agent integrates the actual action by the previous action stage and collets the final stage to forward to the next stage.

   (iv) Goal test : When the goal is achieved, action steps and then forwards into the next stage to determine the cost to achieve the goal.

   (v) Path costing : Here the agent calculates all hardware, software and human working cost.

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

2.7.1 Simple Problem-Solving Agent

```
Function SIMPLE-PROBLEM-SOLVING- AGENT
(percept) returns an action Persistent:
    seq. an action sequence, initially empty,
    state, some description of the current world state,
    goal; a goal initially null,
    problem, a problem formulation,
        state ← UPDATE-STATE(state percept)
    If seq. is empty then
    goal ← FORMULATE-GOAL(State)
    problem ← FORMULATE-PROBLEM (State, goal)
        seq. ← SEARCH( problem}
    If seq. = failure then return a null action action
            ← FIRST(seq.)
        seq. ← REST(seq.) return action
```

Module 2

Scanned with CamScanner

Method

(i) Formulate a problem and a goal.

(ii) Formulate a sequence of actions and execute the sequence of action one - by - one.

(iii) When this is complete, form another goal and proceed.

(iv) Agent uses search-procedure to arrive at a goal.

(v) Agent then uses the solution to guide its actions. Then agent formulates a **new goal**.

Consider an example of a car from Pune to Surat (Fig. 2.7.1)

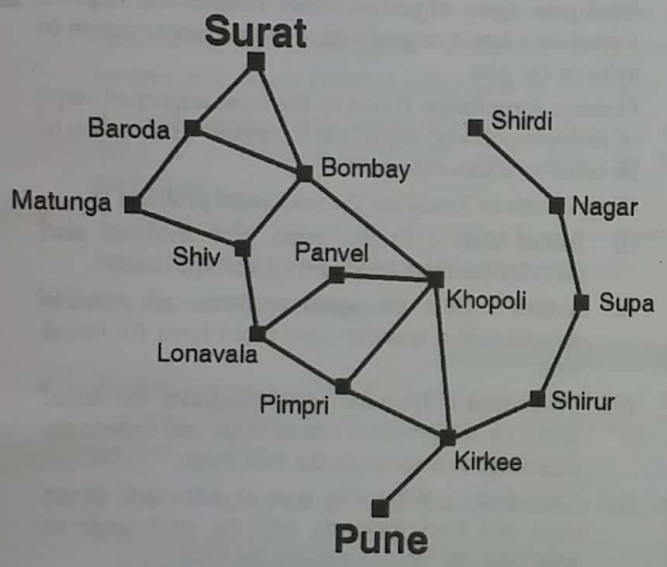


Fig. 2.7.1

2.7.2 Well - Defined Problems and Solutions

Problem - solving consists of using generic or ad-hoc methods in an orderly manner.

Problems can be classified into two different types : ill-defined and well-defined.

1. **Ill defined problems** are those that do not have clear goals, solution paths, or expected Solutions.
2. **Well - defined problems** have specific goals, clearly defined solution paths, and clear expected solutions.

These problems also allow for more initial planning than ill-defined problems.

2.7.2(A) A Problem can be Defined by Five Components

(i) The initial state

It is the state where the agent starts in For example, the initial state for agent is Pune, can be written as In (Pune)

(ii) Action

A description of the possible actions available to the agent.

Given a particular state p, ACTIONS (p) returns the set of actions that can be executed in p. For example, from the state (Pune), the actions could be

{Go (Kirkee), Go (Lonavala), Go (Panvel)}

(iii) Transition model

The formal name for the description of each action is 'transition model'. It is specified by Result (p, a), that results the state from doing action a in p.

We use the term successor to refer to any state reachable from a given state by a single action.

For example, Result {In (Pune), Go (Kirkee)} = In (Kirkee). The initial state, actions and transition model define the **state space** of the problem. It is the set of all states reachable from the initial state by any sequence of actions.

The state space forms a **directed network** or **graph** in which the **nodes** are **states** and **links** between **nodes** are **actions**. A path in the **state space** is a sequence of states connected by a sequence of actions.

(iv) The goal test

It determines whether a given state is a goal-test. There may be an explicit set of possible goal - states, and the test checks whether the given state is one of them.

Sometimes the goal is specified by an abstract property. For example, in chess, the goal is to reach a state called **"checkmate"**, where the opponent's king is under attack and cannot escape.

(v) Path cost

It is a function which assigns numeric value or cost to each path. For the agent going to Surat, time is the essence, so the cost of path may be its length in kilometres. Here the cost of the path is the sum of the costs of individual actions along the path. The **step cost** of taking action a in state p to reach state p′ is denoted by c(p, a, p′). The step costs are non-negative.

2.7.3 Formulating Problems

(i) In the previous article, we made a formulation of the problem of getting to Surat in terms of **initial state, action, transition model, goal test** and **path cost**. This formulation makes sense, but it is still an abstract mathematical description and far away from the reality. For example, consider the simple state description, In (Pune).

Scanned with CamScanner

Where the state of the world includes so many things, the travelling companions, the scenery, condition of the road, the weather and so on.

All these considerations we did not take into our discussion, because they are irrelevant to the problem of reaching to Surat. The process of deleting these details from the representation is called **'abstraction'**

(ii) Now, we take note of abstracting the actions themselves. There are many that we omit altogether, looking out of the window, turning on the radio slowing down for the police-officers, and so on. Also besides changing the location of the vehicle and its occupants it takes up time, consumes fuel, generates pollution and so on.

Our formulation takes into account only the change in location.

(iii) Now we consider a solution of the abstract problem : In the previously mentioned example, the path from Pune to Kirkee to Lonavala to Panvel to Bombay to Surat.

This abstract solution corresponds, to a large number of details. For example, we can drive from Lonavala to Panel, windows open and the shut the windows for the rest of the trip.

Thus the good abstraction is that which involves removing as much details as possible, assuming that abstract actions are easy to carry out.

2.8 EXAMPLE PROBLEMS

We enlist some of the best known problems, distinguishing between **toy** and **real-world** problems.

1. Toy Problem

A toy problem is intended to illustrate various problem-solving methods. Here we mention exact description of the problem and hence is usable to compare the performance of algorithm.

2. Real-world problem

The solution of these problems are useful and people actually care about these problems.

These problems cannot have a unique solution but we can give the general flavour of their formulations.

2.8.1 Toy-problem Example

(a) Vacuum-cleaner World

- This world-problem is very simple so that we can invent many variations in it.
- This world has just two locations : squares A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can move left, move right, suck up the dirt, or do nothing.
- One very simple agent function is : if the current square is dirty, then suck; otherwise, move to the other square.

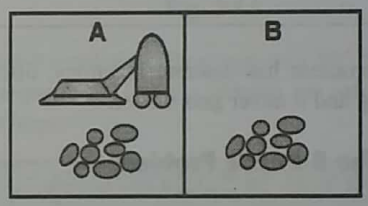


Fig. 2.8.1

We formulate the problem as follows :

1. **States :** The state is determined by the agent location and the dirt locations. The agent is in one of the two locations. And each of them may or may not contain dirt.

   Thus, there are $2 \cdot 2^2 = 8$ possible world states.

   For n locations, it will have $n \cdot 2^n$ states

2. **Initial state :** Any state can be taken as the initial state

3. **Actions :** Each state has three actions : **Left, Right, Suck**

4. **Transition Model :** The actions have their expected effects, except that moving left in the leftmost square, moving right in the rightmost square and sucking in a clean square have no effect.

5. **Goal Test :** It checks whether all squares are clean.

6. **Path cost :** Each step costs 1, so the path cost is the number of steps in the path.

Module 2

Scanned with CamScanner

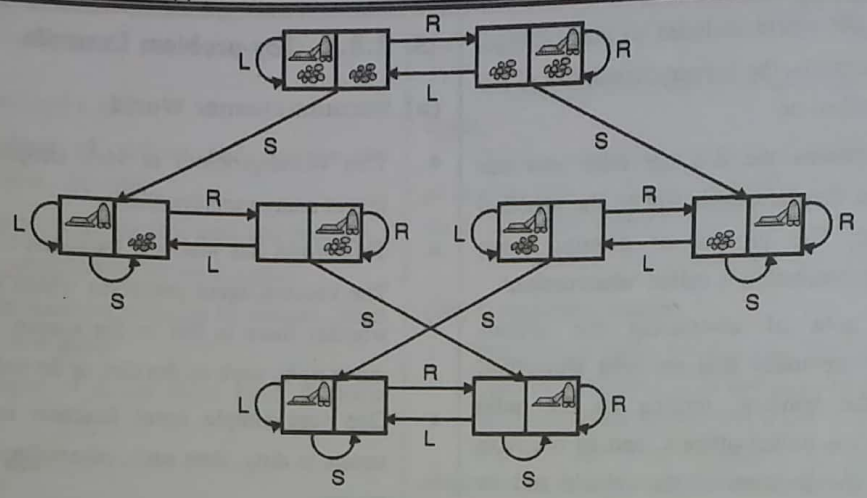


Fig. 2.8.2

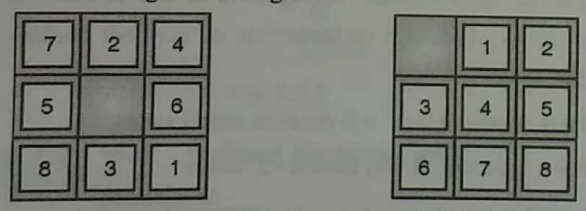
This toy problem has discrete locations, discrete dirt, reliable cleaning and it never gets dirtier.

2.8.2 The 8 Puzzle Problem

It consists of a 3 × 3 board with eight numbered tiles and a blank space.

A tile adjacent to the blank space can slide into the space.

The object is to reach a specified goal state, such as shown on the right of the figure.



(a) Start state (b) Goal state

Fig. 2.8.3

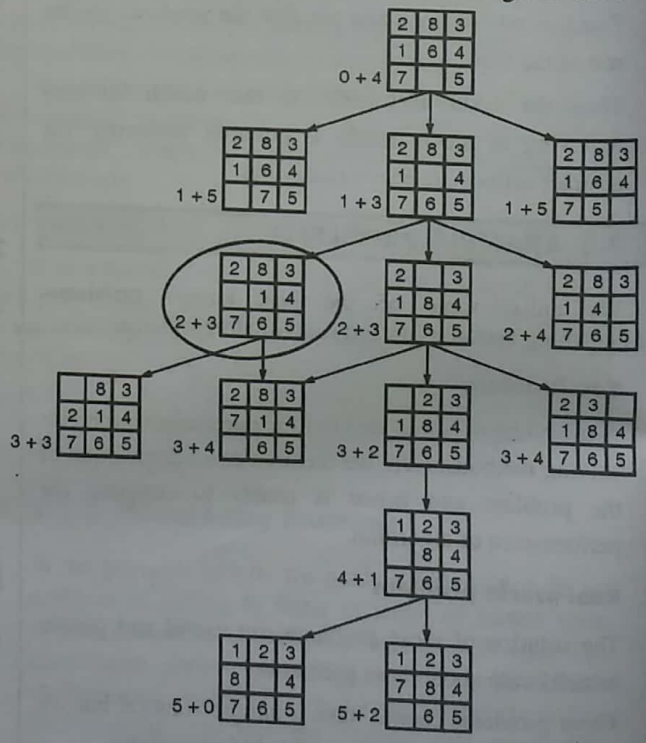
We form the standard **formulations.**

1. **States :** The location of each of the eight tiles and the blank in one of the nine squares is described by a state.
2. **Initial state :** Any state can be taken as the initial state.
3. **Actions :** The actions as movements of the blank space Left, Right, Up or Down are defined by the simplest formulation.
4. **Transition Model :** Given a state and action, this returns the resulting state. For example, if we apply **right** to the start state then the resulting state has 6 and the blank state is switched.
5. **Goal Test :** The test checks whether the state matches the goal configuration.
6. **Path cost :** Each step cost 1, so the path cost is the number of steps in the path.

Here we are interested in the description of the rules of the puzzle, avoiding all the details of physical manipulation.

The 8th puzzle problem

The puzzle can be solved by moving the tiles one by one in the single empty space and achieving the goal state. One can take only one step at a time and no diagonal move.



Scanned with CamScanner

2.8.3 8-Queens Problem

The goal of 8-Queens Problem is to place eight queens on a chessboard such that no queen attacks any other. A queen can attack any piece in the same row, column or diagonal.

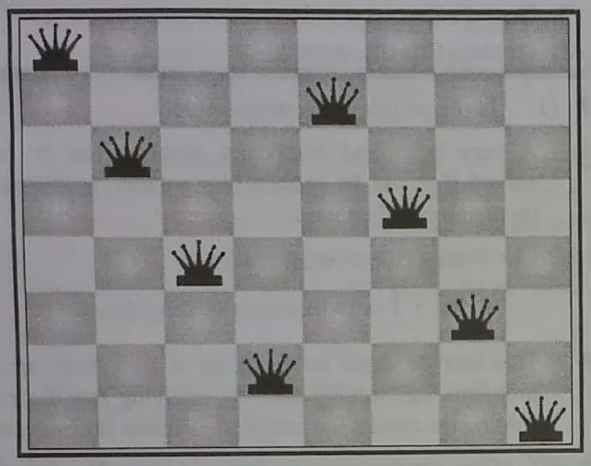


Fig. 2.8.4

In Fig. 2.8.4 the queen in the rightmost column is attacked by the queen at the top left.
(This attempted solution is a failure.)
Here there are two main kinds of formulations.

(1) **An incremental formulations**

This involves operators that **augment** -the state description, beginning with an empty state ; for this problem, it means that each action adds a queen to the state.

(2) **A complete state formulation**

This starts with all 8 queens on the board and moves them around. In either case, the path cost is of no interest because only the final state counts. We form the first incremental formulation as follows :

(i) **States :** Any arrangement of 0 to 8 queens on the board is a state.

(ii) **Initial state :** No queens on the board.

(iii) **Actions :** Add a queen to any empty square.

(iv) **Transition model :** Returns the board with a queen added to the specified square.

(v) **Goal test :** 8 Queens are on the board, no one is attacked.

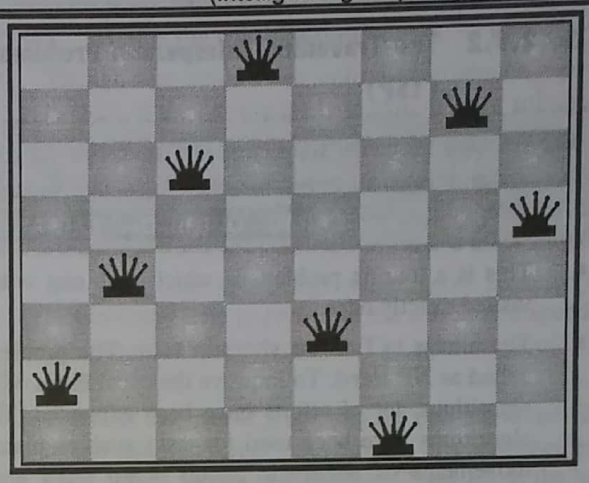


Fig. 2.8.5

In this formulation, there are $64 \cdot 63 \quad 57 = 1.8 \times 10^{14}$ possible sequences. Here the Queens are safe; no queen can attack the other.

2.9 EXAMPLES OF REAL WORLD PROBLEM

2.9.1 Route-Finding Problem

Route-finding problem is defined in terms of specified locations and transitions along links between them. In a variety of applications, route finding algorithms are used.

It is used in Web-sites and in car systems that provide driving directions. Also in routing video-streams in computer networks, military operations planning etc.

We consider the airline travel problem that can be solved by a travel - planning web sites.

(i) **States :** Each state includes a location i.e. an airport and the current time. The state has to record extra information such as domestic or international flight, their fare bases etc.

(ii) **Initial state :** This depends upon users's query.

(iii) **Actions :** Take any flight from the current location, leaving after the current time, leaving enough time for within airport transfer if needed.

(iv) **Transition Model :** Taking a flight will have the flight's destination as the current location and the flights arrival time as the current time.

(v) **Goal Test :** Final destination mentioned by the users.

(vi) **Path cost :** This depends upon monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane and so on.

Module 2

Scanned with CamScanner

2.9.2 The Travelling Salesperson Problem (TSP)

UQ. Give the initial state, goal test, successor function, and cost function for the Travelling salesman problem **(MU - Q. 1(b), Dec. 17, 5 Marks)**

- This is a touring problem in which each city is to be visited exactly once.
- The aim is to find the **shortest tour.** The problem is called as **NP-hard.** To improve the capabilities of TSP algorithms, lot of efforts have been expended. These algorithms have been used for tasks such as planning movements of automatic circuit board drills and of stocking machines on shop floors.

2.9.3 A VLSI Layout

This problem requires vast number of components and connections on a chip to minimise area, minimise stray capacitance, minimise circuit delays and maximising manufacturing yield.

The layout problem is basically logical design phase. It is usually split into two parts. They are (i) cell layout and (ii) channel routing.

(i) **Cell Layout :** In cell layout, the primitive components of the circuit are grouped into cells, each of which performs some recognised function. Each cell has a fixed footprint (size and shape) and a certain number of connections to each of the other cells. Cells are placed on the chip so that they do not overlap and connecting wires can be placed between the cells.

(ii) **Channel Routing :** Channel routing finds a specific route for each wire through the gaps between the cells. These problems even though they are complex but worth solving.

2.10 PROBLEM CHARACTERISTICS OF IA

GQ. Discuss with examples : AI Problem Characteristic.

GQ. Explain how AI techniques improve real-world problem solving.

It includes a variety of techniques. In order to choose an appropriate method, it is necessary to analyze the problem with respect to the following considerations :

(1) Is the problem decomposable ?
(2) Can solution steps be ignored or undone?
(3) Is the Universal Predictable ?
(4) Is good solution absolute or relative ?
(5) The knowledge base consistent ?
(6) What is the role of Knowledge?
(7) Does the task require interaction with the person.
(8) Problem Classification

(1) Is the problem decomposable ?

- A very large and composite problem can be easily solved if it can be broken into smaller problems and recursion could be used.

  Suppose we want to solve.

  Ex. : $\int (x^2 + 3x + \sin 2x \cos 2x)\, dx$

- This can be done by breaking it into three smaller problems and solving each by applying specific rules. Adding the results the complete solution is obtained.

(2) Can solution steps be ignored or undone?

- Problem fall under three classes ignorable, recoverable and irrecoverable. This classification is with reference to the steps of the solution to a problem. Consider thermo proving. We may later find that it is of no help. We can still proceed further, since nothing is lost by this redundant step. This is an example of ignorable solutions steps.
- Now consider the 8 puzzle problem tray and arranged in specified order. While moving from the start state towards goal state, we may make some stupid move and consider theorem proving.
- We may proceed by first proving lemma. But we may backtrack and undo the unwanted move. This only involves additional steps and the solution steps are recoverable.
- Lastly consider the game of chess. If a wrong move is made, it can neither be ignored nor be recovered. The thing to do is to make the best use of current situation and proceed. This is an example of an irrecoverable solution steps.

  1. Ignorable problems Ex:- theorem proving
     In which solution steps can be ignored.
  2. Recoverable problems Ex:- 8 puzzle
     In which solution steps can be undone
  3. Irrecoverable problems Ex:- Chess
     In which solution steps can't be undone

- A knowledge of these will help in determining the control structure.

Scanned with CamScanner

(3) Is the Universal Predictable ?

- Problems can be classified into those with certain outcome (eight puzzle and water jug problems) and those with uncertain outcome (playing cards) in certain – outcome problems, planning could be done to generate a sequence of operators that guarantees to a lead to a solution.
- Planning helps to avoid unwanted solution steps. For uncertain outcome problems, planning can at best generate a sequence of operators that has a good probability of leading to a solution.
- The uncertain outcome problems do not guarantee a solution and it is often very expensive since the number of solution and it is often very expensive since the number of solution paths to be explored increases exponentially with the number of points at which the outcome cannot be predicted.
- Thus one of the hardest types of problems to solve is the irrecoverable, uncertain – outcome problems (Ex:- Playing cards).

(4) Is good solution absolute or relative ? (Is the solution a state or a path ?)

- There are two categories of problems. In one, like the water jug and 8 puzzle problems, we are satisfied with the solution, unmindful of the solution path taken, whereas in the other category not just any solution is acceptable.
- We want the best, like that of traveling salesman problem, where it is the shortest path. In any – path problems, by heuristic methods we obtain a solution and we do not explore alternatives.
- For the best-path problems all possible paths are explored using an exhaustive search until the best path is obtained.

(5) The knowledge base consistent ?

- In some problems the knowledge base is consistent and in some it is not. For example, consider the case when a Boolean expression is evaluated.
- The knowledge base now contains theorems and laws of Boolean Algebra which are always true. On the contrary consider a knowledge base that contains facts about production and cost.
- These keep varying with time. Hence many reasoning schemes that work well in consistent domains are not appropriate in inconsistent domains. **Ex.** Boolean expression evaluation.

(6) What is the role of Knowledge ?

- Though one could have unlimited computing power, the size of the knowledge base available for solving the problem does matter in arriving at a good solution.
- Take for example the game of playing chess, just the rules for determining legal moves and some simple control mechanism is sufficient to arrive at a solution.
- But additional knowledge about good strategy and tactics could help to constrain the search and speed up the execution of the program. The solution would then be realistic.
- Consider the case of predicting the political trend. This would require an enormous amount of knowledge even to be able to recognize a solution , leave alone the best. **Ex :** 1. Playing chess 2. News paper understanding.

Module 2

(7) Does the task require interaction with the person

The problems can again be categorized under two heads.

(1) **Solitary** in which the computer will be given a problem description and will produce an answer, with no intermediate communication and with the demand for an explanation of the reasoning process. Simple theorem proving falls under this category. given the basic rules and laws, the theorem could be proved, if one exists.

Ex. : theorem proving (give basic rules and laws to computer)

(2) **Conversational,** in which there will be intermediate communication between a person and the computer, wither to provide additional assistance to the computer or to provide additional informed information to the user, or both problems such as medical diagnosis fall under this category, where people will be unwilling to accept the verdict of the program, if they cannot follow its reasoning. **Ex :** Problems such as medical diagnosis.

(8) Problem Classification

Actual problems are examined from the point of view, the task here is examine an input and decide which of a set of known classes. **Ex. :** Problems such as medical diagnosis , engineering design.

In Short AI has following Characteristics

(i) They have some level of autonomy that allows them to perform certain tasks on their own.

(ii) They have a learning ability that enables them to learn even as tasks are carried out.

(iii) They can interact with other entities such as agents, humans and system.

Chapter Ends...

□□□

Scanned with CamScanner

Module 3

CHAPTER 3

Problem Solving

Syllabus

Uninformed Search Methods : Breadth First Search (BFS), Depth First Search (DFS), Depth Limited Search, Depth First Iterative Deepening (DFID), Informed Search Methods: Greedy best first Search, A* Search, Memory bounded heuristic Search.

Local Search Algorithms and Optimization Problems: Hill climbing search Simulated annealing, Genetic algorithms.

Adversarial Search: Game Playing, Min-Max Search, Alpha Beta Pruning

3.1 Problem Solving 3-4
- 3.1.1 Problem Solving Agent 3-4
- 3.1.2 Simple Problem-Solving Agent 3-4

3.2 Performance Measures 3-4
- 3.2.1 Types of Performance Measures 3-5

3.3 Searching 3-5
- 3.3.1 Node Representation in Search Tree 3-5

3.4 Uninformed Search (Blind Search) 3-6
- UQ. Explain with example various uninformed search techniques. (MU - Q. 3(a), May 17, 10 Marks) 3-6

3.5 Depth First Search (DFS) 3-6
- 3.5.1 DFS Algorithm 3-7
- 3.5.2 Performance Measures of DFS 3-9
- 3.5.3 Advantages and Disadvantages of Depth First Search 3-9
- 3.5.4 Applications of DFS 3-9
- 3.5.5 Solved Example on DFS 3-10
- UEx. 3.5.1 (MU - Q. 2(b), May 18, 10 Marks, Q. 2(a), Dec. 15, 10 Marks) 3-10
- UEx. 3.5.2 (MU-Q. 5(b), May 16, 10 Marks) 3-10

3.6 Breadth-First Search (BFS) 3-10
- 3.6.1 BFS Traversal Algorithm 3-11
- 3.6.2 Performance Measures for BFS 3-11
- 3.6.3 BFS Algorithm for Extra Memory 3-11
- 3.6.4 Execution of BFS Algorithm 3-12
- 3.6.5 Advantages and Disadvantages of BFS 3-12
- 3.6.6 Applications of BFS Algorithm 3-13
- 3.6.7 Performance Measures of BFS 3-13
- 3.6.8 Limitations of Breadth First Search 3-13
- 3.6.9 Example of Breadth First Search 3-13
- 3.6.10 Solved Example on BFS 3-14
- UEx. 3.6.2 (MU - Q. 5(A), Dec. 17, 10 Marks) 3-14
- 3.6.11 Application of BFS 3-15

3.7 Uniform Cost Search 3-15
- 3.7.1 Algorithm of U.C.S. 3-15
- 3.7.2 Execution of Algorithm of Uniform Cost Search 3-15
- 3.7.3 Example of Uniform Cost Search (Linear Displacement) 3-16
- 3.7.4 Advantages and Disadvantages of U.C.S. 3-18
- 3.7.5 Performance Measures 3-18
- 3.7.6 Solved Example on Curved Displacement 3-18
- UEx. 3.7.1 (MU - Q. 1(b), Dec. 16, 5 Marks) 3-18

3.8 Depth Limited Search 3-20

3.9 Iterative Deepening Search Technique (IDS or IDDFS) 3-21
- UQ. Explain Iterative Deepening search algorithms based on performance measure with justification; complete, optimal, Time and Space complexity. (MU - Q. 4(a), Dec. 18, 10 Marks) 3-21
- 3.9.1 IDDFS Algorithm 3-21
- 3.9.2 Advantages of IDDFS 3-23

3.10 Bidirectional Search 3-23

3.11 Informed search 3-24
- 3.11.1 Example for Informed Search 3-24
- 3.11.2 Algorithm for Beam Search 3-24

3.12 Heuristic Function 3-25

Scanned with CamScanner

UQ. What is heuristic function ? (MU - Q. 3(b), Dec. 18, 10 Marks, Q. 1(c), May 18, 4 Marks, Q. 1(c), May 17, 4 Marks,Q. 3(b), Dec. 16, 5 Marks, Q. 1(c), Dec. 15, 3 Marks) 3-24

3.12.1 Simple Heuristic Functions 3-24

3.12.2 Problem Characteristics for Heuristic Search 3-25

3.12.3 Is the Problem Decomposable ?....3-25

3.12.4 Can Solution Steps be Ignored or Undone ? 3-25

3.12.5 Is the Universal Predictable ?....3-26

3.12.6 Is Good Solution Absolute or Relative ? (Is the Solution a State or a Path?)....3-26

3.12.7 The Knowledge Base Consistent ?....3-26

3.12.8 What is the Role of Knowledge ?....3-26

3.12.9 Does the Task requires Interaction with the Person 3-26

3.12.10 Problem Classification 3-27

3.12.11 Heuristic Function for : Travelling Salesman Problem 3-27

UQ Give the initial state, goal test, successor function, and cost function for the travelling salesperson problem (TSP). There is a map involving N cities some of which are connected by roads. The aim is to find the shortest tour that starts from a city, visits all the cities exactly once and comes back to the starting city. (MU - Q. 5(A), Dec. 16, 6 Marks)....3-27

UQ. Define heuristic function. Give an example heuristics functions for 8-puzzle problem. Find the heuristics value for a particular state of the Blocks World Problem. (MU - Q. 1(b), May 17, 5 Marks)....3-27

3.12.12 Branch And Bound Technique 3-27

3.12.13 Block World Problem....3-28

UQ. Define heuristic function. Give an example heuristics function for Blocks world problem. (MU - Q. 1(a), Dec. 2015, 5 Marks)....3-28

UQ. Find the heuristics value for a particular state of the Blocks World Problem. (MU - Q. 1(b), May 17, 5 Marks)....3-28

3.12.14 Actions....3-28

3.12.15 Motivation....3-28

3.12.16 Symbolic Representation 3-29

3.12.17 Sussman Anomaly 3-29

3.12.18 Heuristic Function : H_1....3-30

3.12.19 Example of 'Block World Problem....3-30

3.12.20 Properties of Heuristic Functions 3-31

3.12.21 Memory Bounded Heurestic Search....3-31

3.13 Local Search Algorithms....3-32

UQ. Write short note on : Local search Algorithms .(MU - Q. 6(b), May 18, Q. 6(c), May 19, 6 Marks)....3-32

3.14 Hill Climbing Algorithm....3-32

UQ. Explain Hill Climbing and its Drawback in details. (MU - Q. 2(b), May 19, 5 Marks, Q. 6(b), May 18, 5 Marks, Q. 2(A), Dec. 17, 10 Marks, Q. 2(b), May 17, 10 Marks)....3-32

UQ. Explain Hill-Climbing Algorithm with an example. (MU - Q. 1(d), Dec. 17, 5 Marks)....3-32

3.14.1 Algorithm for Hill-Climbing Procedure 3-32

3.14.2 Explanation of Hill Climbing Algorithm ...3-32

3.14.3 Simple Hill Climbing Algorithm 3-33

3.15 Steepest-Ascent Hill Climbing Algorithm....3-33

3.15.1 Algorithm : Steepest-ascent Hill Climbing 3-33

3.15.2 Limitations of Steepest-ascent Hill Climbing 3-33

UQ. State limitations of steepest-ascent hill climbing (MU - Q. 2(b), May 19, 5 Marks, Q. 6(b), May 18, 5 Marks, Q. 2(A), Dec. 17, 10 Marks, Q. 2(b), May 17, 10 Marks)....3-33

UQ. What are the problems/frustrations that occur in hill climbing technique? Illustrate with an example. (MU - Q. 4(b), Dec. 15, 6 Marks, Q. 1(d), May 17, 5 Marks)....3-33

3.15.3 Ways of dealing with Local Maxima Plateau and Ridge Problems....3-34

3.16 Simulated Annealing (sa) 3-34

UQ. Define the term simulated annealing. Explain simulated Annealing with suitable example.(MU - Q. 2(A), Dec. 17, 10 Marks, Q. 2(b), May 16, 10 Marks). 3-34

3.16.1 Types and Use of Simulated Annealing 3-35

3.16.2 Simulated Annealing in Machine Learning 3-35

3.17 Parameter for S.A....3-35

3.18 Genetic Algorithm....3-36

UQ. Write a short note on genetic algorithm. (MU - Q. 5(a), Dec. 15, Q. 6, May 17, 10 Marks)....3-36

3.18.1 Comparison between Traditional and Genetic Algorithm....3-36

3.18.2 Basic Terminology....3-36

UQ. Define the terms chromosome, fitness function, crossover and mutation as used in Genetic algorithms. (MU - Q. 5(a), May 18, 10 Marks)....3-36

UQ. Explain how genetic algorithms work. Define the terms chromosome, fitness function, crossover and mutation as used in Genetic algorithms. (MU - Q. 6(c), Dec. 18, 5 Marks)....3-36

3.18.3 Optimisation Problems 3-37

Scanned with CamScanner

3.18.4 Initialisation 3-39
3.18.5 Selection 3-39
3.18.6 Genetic Operators 3-39
UQ. Explain how genetic algorithms work. (MU - Q. 6(c), Dec. 18, 5 Marks) 3-39
3.18.7 Advantages of Genetic Algorithm 3-40
3.18.8 Limitations of Genetic Algorithm 3-40
3.18.9 Applications of Genetic Algorithm 3-41
3.19 Best First Search 3-41
3.19.1 Steps of the Search Process in BFS 3-42
3.19.2 Algorithm for best-first Search 3-42
3.20 Greedy Best First Search and A* Best First Search 3-42
3.20.1 Greedy Best First Search Algorithm 3-43
3.20.2 Solved Examples 3-43
3.21 A* and AO* Search 3-44
UQ. Explain A* Algorithm with example. (MU - Q. 2(b), May 18, 10 Marks, (MU - Q. 2(b), May 19, 5 Marks, Q. 6(c), May 18, 5 Marks, Q. 6(b), May 17, 5 Marks) 3-44
3.22 Admisibility of A* 3-45
UQ. Prove the admissibility of A*. (MU - Q. 5(B), Dec. 16, 6 Marks) 3-45
3.23 Monotonicity 3-46
3.23.1 Monotonicity in AI 3-47
3.23.2 Concept 3-47
UEx. 3.23.1 : (MU - Q. 2(a), Dec. 2015, 10 Marks) 3-47
UQ. Compare following informed search algorithms (a) Greedy first search (b) A* (c) Recursive Best-First (RBFS) (MU- Q. 4(a), May 16, 10 Marks) 3-49
3.24 Adversarial sEARCH 3-50
3.24.1 Types of Games in AI 3-50
3.24.2 Characteristics of Adversarial Search (A.S.) 3-51
3.24.3 Comparison of Search and Games 3-51
3.25 Techniques required to get the best Optimal Solution 3-52
3.26 Game Playing 3-52
3.26.1 Zero Sum Game 3-52
UQ. Write short note on : Game Playing. (MU - Q. 6(a), May 17, 5 Marks, Q. 6(e), May 17, 5 Marks) 3-52
3.26.2 Elements of Game Playing Search 3-52
UQ. Draw a game tree for a Tic-Tac Toe problem. (MU - Q. 4(c), Dec. 2015, 4 Marks) 3-52
3.26.3 Some More Examples of Game Playing/Adversarial Search 3-53
3.26.4 Types of Algorithms in Adversarial Search 3-54
3.27 Game Tree 3-54
3.27.1 tic-tac-toe Problem 3-56
3.27.2 Limitations of Game Trees 3-56
3.28 Minmax Procedure 3-56
UQ. Explain Min max and Alpha beta pruning algorithms for adversarial search with example. (MU - Q. 5(b), May 17, 10 Marks) 3-56
3.28.1 Properties of Min Max Algorithm 3-57
UEx. 3.28.1 : (MU-Dec. 15, Dec. 19 10 Marks) 3-57
UEx. 3.28.2 : (MU - May 18, 10 Marks) 3-58
3.29 Game of Chance 3-59
3.30 Various Categories of Game of Chance 3-59
3.31 Alpha-beta Pruning 3-60
UQ. Explain Min max and Alpha beta pruning algorithms for adversarial search with example. (MU - Q. 5(b), May 17, 10 Marks) 3-60
UQ. Define Alpha and Beta value in game tree? (MU - Q. 1(c), May 16, 4 Marks) 3-60
3.31.1 Calculating Alpha-Beta Values 3-61
3.31.2 Calculating Alpha Values at a Max Node 3-61
3.31.3 Calculating Beta Values at Min 3-61
3.31.4 Alpha-beta Pruning Example 3-61
3.32 Comparisons in Tabular Form 3-63
3.32.1 Difference Between Informed Search & Uninformed Search 3-63
UQ. Difference between Informed and Uninformed Search in AI. (MU - Q. 3(a), Dec. 19, 10 Marks, Q. 4(b), Dec. 18, 10 Marks, Q. 4(b), May 16, 10 Marks) 3-63
3.32.2 Differentiate Hill Climbing vs A* Algorithm 3-64
UQ. How do you compare Hill climbing technique with A* algorithm. MU - Q. 2(A), May. 17, 5 Marks, Q. 2(b), May 16, 5 Marks 3-64
3.32.3 Comparison of Hill-Climbing and Simulated Annealing 3-64
UQ. Explain Hill Climbing and Simulated Annealing with suitable example. (MU - Q. 2(a), Dec. 18, 10 Marks, Q. 2(a), May 18, 10 Marks) 3-64
3.32.4 Difference between Undirectional and Bidirectional Search 3-65
3.32.5 Difference Between BFS and DFS 3-65
UQ. Compare Breadth First Search (BFS), Depth first search (DFS). (MU - Q. 4(a), Dec. 18, 10 Marks) 3-65
• Chapter End 3-66

Module 3

Scanned with CamScanner

3.1 PROBLEM SOLVING

GQ. Briefly explain problem solving.

Problem solving consists of using generic or ad hoc methods, in an orderly manner, for finding solutions to problems. Some of the problem-solving techniques use artificial intelligence, computer science, engineering, mathematics, or psychoanalysis.

Problems can also be classified into two different types: **ill-defined and well-defined,** from which appropriate solutions are to be made.

1. **Ill-defined** problems are those that **do not have clear goals**, solution paths, or expected solution.
2. **Well-defined** problems **have specific goals**, clearly defined solution paths, and clear expected solutions. These problems also allow for more initial planning than ill-defined problems.

   Being able to solve problems sometimes involves dealing with **pragmatics (logic)** and **semantics** (interpretation of the problem). It requires the ability to understand what the goal of the problem is and what rules could be applied to represent the key to solving the problem. Sometimes the problem requires some abstract thinking and coming up with a creative solution.

Problem types

1. **Deterministic, fully observable** ⇒ single-state problem

   Agent knows exactly which state the problem will be in; it's solution is a sequence.
2. **Non-observable** ⇒ conformant problem

   Agent may have no idea where it is; solution (if any) is a sequence.
3. Nondeterministic and/or partially observable ⇒ contingency problem percepts provide new information about current state.
4. **Unknown state space** ⇒ exploration problem ("online") Solution is a tree or policy often interrelated search, and execution.

3.1.1 Problem Solving Agent

GQ. Describe problem solving agent.

Problem-solving agents : A goal formulation, based on the current situation and the performance measure is required for problem solving.

- **Problem formulation** is the process of deciding what actions and states to consider, given a goal. In general, an agent with several options for action of unknown value can decide what to do by first examining different possible sequences of actions (that lead to states of known value) and then choosing the best sequence.
- A search algorithm takes a problem as input and returns a solution in the form of an action sequence.

3.1.2 Simple Problem-Solving Agent

```
Function SIMPLE-PROBLEM-SOLVING- AGENT (percept)
returns an action Persistent:
        seq. an action sequence, initially empty,
        state, some description of the current world state,
        goal; a goal initially null,
        problem, a problem formulation,
        state ← UPDATE-STATE(state percept)
    If seq. is empty then
        goal ← FORMULATE-GOAL(State)
        problem ← FORMULATE-PROBLEM
                (State, goal)
        seq. ← SEARCH( problem}
    If seq. = failure then return a null action action
← FIRST(seq.)
    seq. ← REST(seq.) return action
```

3.2 PERFORMANCE MEASURES

- Performance measurement is the process of collecting, analysing and / or reporting information regarding the performance of an individual, group, organization, system or component.
- Moulin defines the term with a forward looking organizational focus 'the process of evaluating how well organizations are managed and the value they deliver for the customers and other stakeholders'.
- The most common such frameworks include :
  - (i) **Balanced scorecard :** It is used by organisations to manage the implementation of corporate strategies.

Scanned with CamScanner

(ii) **Key performance indicator :** It is a method for choosing important / critical performance measures, usually in an organisational content.

- Performance measure algorithms are as follows :
  1. **Time complexity :** It is a measure of amount of time for an algorithm to execute.
  2. **Space efficiency :** It is a measure of the amount of memory needed for an algorithm to execute.
  3. **Complexity theory :** It is a study of algorithm performance.
  4. **Function dominance :** It is a comparison of cost functions.
  5. Performance of an algorithm means predicting the resources which are required to an algorithm to perform its task.
- It means that when we have multiple algorithms to solve a problem, we need to select a suitable algorithm to solve that problem.
- To compare algorithms, we use a set of parameters or set of elements like memory required by that algorithm, the execution speed of that algorithm, easy to understand, easy to implement, etc.
- To analyse an algorithm, we consider only the space and time required by that particular algorithm and we ignore all the remaining elements.
- Based on this information, performance analysis of an algorithm can be defined as 'Performance analysis of an algorithm is the process of calculating space and time required by that algorithm'.
- Performance analysis of an algorithm is performed by using the following measures :
  - (i) **Space complexity :** Space required to complete the task of that algorithm. It includes program space and data space.
  - (ii) **Time complexity :** Time required to complete the task of that algorithm.
- This involves the following steps :
  - (a) Implement the algorithm completely.
  - (b) Determine the time required for each basic operation.
  - (c) Identify unknown quantities that can be used to describe the frequency of execution of the basic operations.

3.2.1 Types of Performance Measures

(a) Workload or output measures. These measures indicate the amount of work performed or number of services received.

(b) Efficiency measures.

(c) Effectiveness or outcome measures.

(d) Productivity measures.

3.3 SEARCHING

GQ. What is searching?

Search plays a major role in solving many artificial intelligence (AI) problems. Search is a **universal problem-solving mechanism** in AI. In many problems, sequence of steps required to solve a problem is not known in advance but must be determined by systematic trial-and-error exploration of alternatives.

Search techniques try to "pre-play" the game by evaluating the future states (game tree search) and may use also heuristic to prone bad choices or speed things up. They theoretically can make an **exact and perfect** choice, but are **slow**.

The problems that are addressed by AI search algorithms fall into three general- classes:

1. Single-agent path-finding problems
2. Two players games
3. Constraint- satisfaction problems

3.3.1 Node Representation in Search Tree

A binary search tree (BST) is a tree in which all the nodes follow the below mentioned properties

(i) The value of the key of the left sub-tree is less than the value of its parent (root) node's key.

(ii) The value of the key of the right sub-tree is greater than or equal to the value of its parent (root) node's key.

Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and it can be defined as left-sub-tree (keys) < node (key) ≤ right_sub-tree (keys)

Representation

BST is a collection of nodes arranged in a way where they maintain BST properties. Each node has a key and has an associated value.

Module 3

Scanned with CamScanner

While searching, the desired key is compared to the keys in BST and if found, the associated value is retrieved.

We mention below a pictorial representation of BST

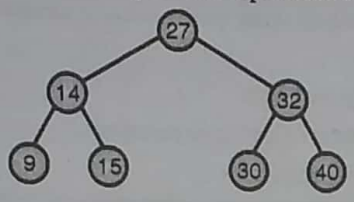


Fig. 3.3.1

Note that the root node key (27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

Basic Operation

The basic operations of a tree

(i) Search : Searches an element in a tree

(ii) Insert : Inserts an element in a tree

(iii) Pre-order Traversal : Traverses a tree in a pre-order manner

(iv) In-order Traversal : Traverses a tree in an in-order manner

(v) Post-order Traversal : Traverse a tree in a post-order manner.

(vi) Remark : There must be no duplicate nodes.

Nodes representing in search tree

Remark

Let us suppose we want to search for the number :

(i) We start at the root

(ii) We compare the value to be searched with the value of the root.

(iii) It is equal, we complete the search.

(iv) If it is lesser, we need to go to the left sub-tree, since in a binary subtree all the elements in the left subtree are lesser and all the elements in the right subtree are greater.

(v) In this traversal, at each step we discard one of the sub-trees.

(vi) We go on reducing like this till we find the element or till our search is reduced to only one node.

(vii) The search here is a binary search and hence is called as Binary search tree.

Note : After reaching the end, just insert that node at left (it less than current), else right.

3.4 UNINFORMED SEARCH (BLIND SEARCH)

GQ. What is blind or uninformed search ?

UQ. Explain with example various uninformed search techniques. **(MU - Q. 3(a), May 17, 10 Marks)**

(1) They have no additional information about states other than those provided in the problem definition. They can only **generate successors and distinguish between goal state and non- goal state**.

(2) These are commonly used search procedures which explore all the alternating options during the search process. They do not have any dome in specific knowledge. All they need are the initial state, the final state and a set of legal operators.

Most important search techniques are as follows :

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

3.5 DEPTH FIRST SEARCH (DFS)

(1) DFS search is the **distributive file system**. The distributive file function provides the ability to logically group shares on multiple servers and to transparently link shares into a single namespace. DFS organizes shared resources on a network in a tree-like structure.

(2) DFS is a file-system with data stored on a server. The data is accessed and processed on if it was stored on the local client machine.

(3) The DFS makes it convenient to share information and files among users on a network in a controlled and authorised way.

(4) DFS is used in topological **sorting, scheduling problems, cycle detection in graphs and solving puzzles**, other applications involve analysing networks **e.g.** testing, if a graph is bipartite

(5) Depth First Search (DFS) is an algorithm for traversing or searching tree or graph data structures.

Scanned with CamScanner

(6) The algorithm starts at the root node (in case of a graph, selecting some arbitrary node as the root node) and explores as far as possible along each branch before backtracking. Refer Fig. 3.5.1.

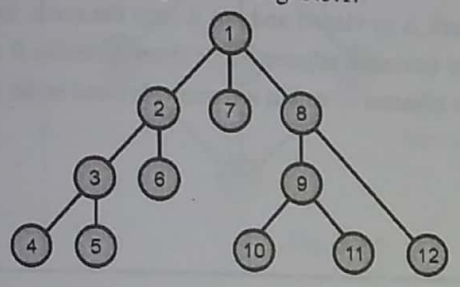


Fig. 3.5.1 : Depth First Search (DFS) order in which the nodes are visited

3.5.1 DFS Algorithm

DFS algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Refer Fig. 3.5.2. In this example, DFS **algorithm** traverses from S to A to D to G to E to B first, then to F and lastly to C. It follows the following rules :

- **Rule 1 :** Visit the adjacent unvisited vertex. Mark it as visited and display it.

  ↓

- **Rule 2 :** If no adjacent vertex is found, choose the vertex from the stack. (There will appear all the vertices from the stack, which do not have adjacent vertices).

  ↓

- **Rule 3 :** Repeat Rule 1 and Rule 2 till the stack becomes empty.

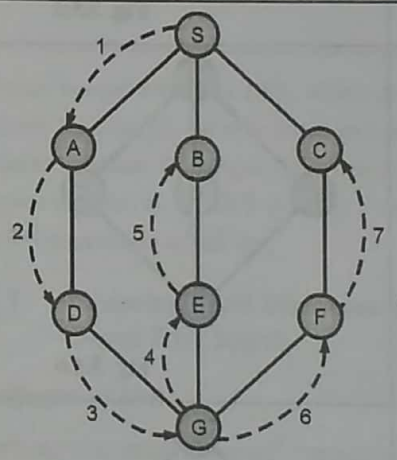
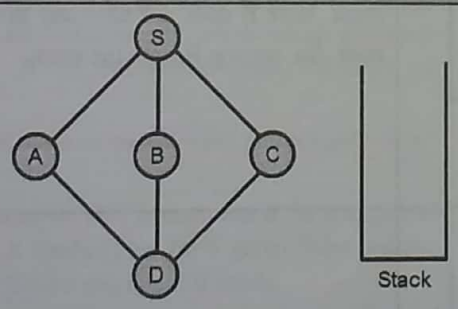
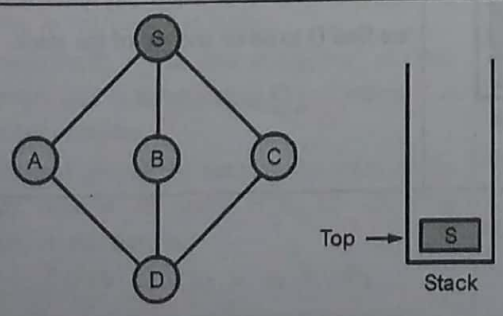
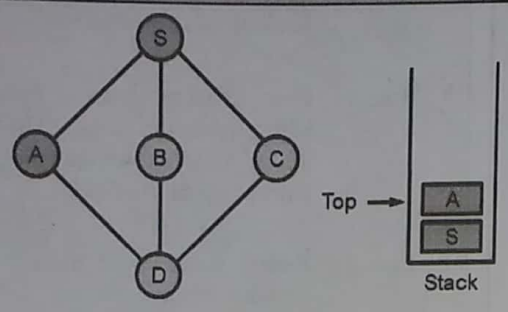
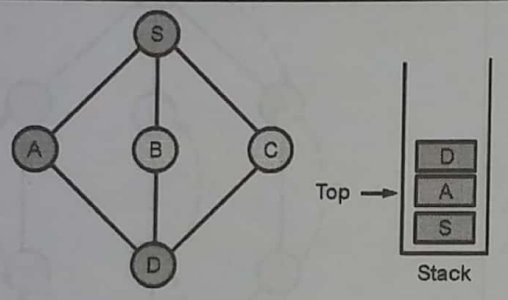
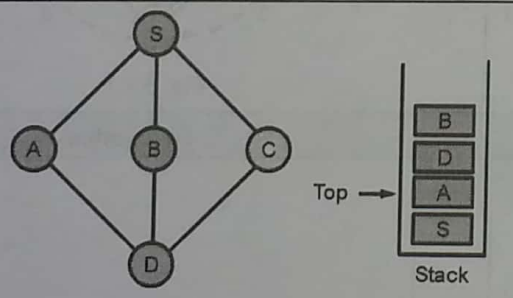
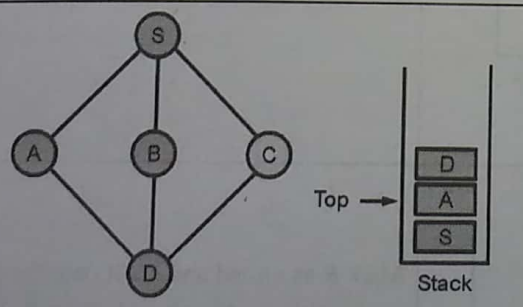


Fig. 3.5.2

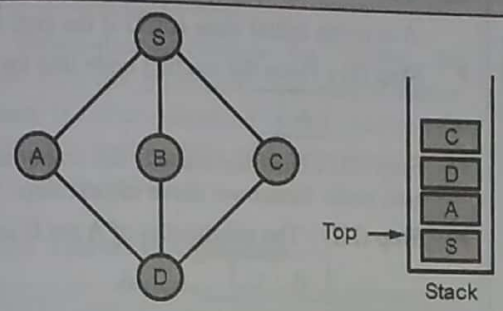
Step	Traversal	Description
1.	 **Fig. 3.5.3**	Initialise the stack.
2.	 **Fig. 3.5.4**	Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. Here we take the node in alphabetical order.

Module 3

Scanned with CamScanner

Step	Traversal	Description
3.	 Fig. 3.5.5	Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Bothe S and D are adjacent to A, but we want unvisited nodes only.
4.	 Fig. 3.5.6	Visit D and mark it as visited and put onto the stack. Here we have B and C nodes, which are adjacent to D and both are unvisited. But again we choose in an alphabetical order.
5.	 Fig. 3.5.7	We choose B, mark it as visited and put onto the stack. Here B does not have any unvisited adjacent node. So, we pop B from the stack.
6.	 Fig. 3.5.8	We check the stack top for return to the previous node and check if it has any unvisited nodes. Here we find D to be on the top of the stack.

Scanned with CamScanner

Step	Traversal	Description
7.	 Fig. 3.5.9	Only unvisited adjacent node is from D is C. So we visit C, mark it and put it into the stack.

- As C does not have any unvisited adjacent node, so we keep popping the stack till we find a node that has an unvisited adjacent node. Here, there is none and the stack is empty and the program is over.

3.5.2 Performance Measures of DFS

The performance measuring factors of an algorithm are as follows :

- The two most common measures are : speed and memory usage; other measures could include transmission speed, temporary disk usage, long-term disk usage, power consumption, total cost of ownership, response time to external stimuli, etc.
- Performance measure is generally defined as regular measurement of outcomes and results which generates reliable data on the effectiveness and efficiency of programs.
- There are four ways to measure the performance of an algorithm :
  - (i) **Completeness :** DFS is complete if the search tree is finite, it implies that for a given finite search, DFS will have a solution if it exists.
  - (ii) **Optimality :** DFS is not optimal, it means that the number of steps in reaching the solution, or the cost spent in reaching it is high.
  - (iii) **Time complexity :** The time complexity of DFS, if the entire tree is traversed, is O (V) where V is the number of nodes.

    For a directed graph, the sum of the sizes of the adjacency lists of all nodes is E. So, the time complexity in this case is

    $$O(V) + O(E) = O(V + E)$$

    For an undirected graph, each edge appears twice.
  - (iv) **Space complexity :** For DFS, which goes along a single 'branch' all the way down and uses a stack implementation, the height of the tree matters. The space complexity for DFS is O (h) where h is the maximum height of the tree.

Module 3

3.5.3 Advantages and Disadvantages of Depth First Search

Advantages of DFS

1. Memory requirement is linear with respect to nodes.
2. Less time and space complexity rather than BFS.
3. Solution can be found out without much more search.

Disadvantages of DFS

1. Not guarantee that it will give you solution.
2. Cut-off depth is smaller so time complexity is more.
3. Determination of depth until the search proceeds.
4. The major drawback of depth-first search is the determination of the depth till which the search is reached. This depth is called cut-off depth. The value of **cut-off depth** is essential because otherwise the search will go on and on.

   If the cut-off depth is smaller, solution may not be found and if cut-off depth is large, time-complexity will be more.

3.5.4 Applications of DFS

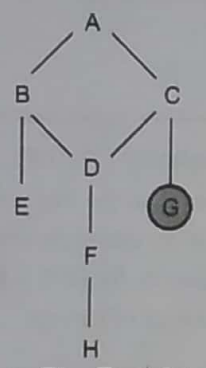
1. Finding connected components.
2. Topological sorting.
3. Finding bridges of graph.

Scanned with CamScanner

3.5.5 Solved Example on DFS

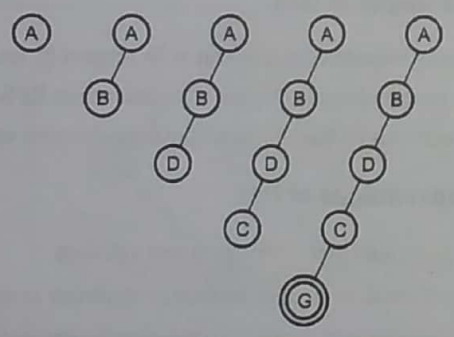
UEx. 3.5.1 (MU - Q. 2(b), May 18, 10 Marks. Q. 2(a), Dec. 15, 10 Marks)

Consider the following graph shown in Fig. 3.5.1 starting from A execute DFS the goal node is G. Show the order in which the nodes are expanded. Assume that the alphabetically smaller node is expanded first to break ties.



(1B13)**Fig. Ex. 3.5.1**

Soln. :



(1B14)**Fig. Ex. 3.5.1(a)**

UEx. 3.5.2 : (MU-Q. 5(b), May 16, 10 Marks)

Consider the graph given in the figure. Assume that the initial state is A and the goal state is G. Find a path from the initial state to the goal state using DFS. Also report the solution cost.

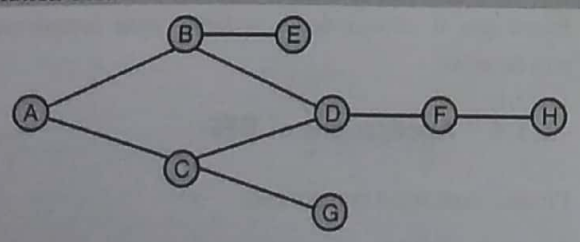


Fig. Ex. 3.5.2

Soln. :

A is given initial state and G is the goal node.

- **Step (I) :** Place the starting node into the stack

A	

- **Step (II) :** Now the stack is not empty and A is not our goal node. Hence we move to next step.
- **Step (III) :** The neighbours of A are B and C.

B	C	

A

- **Step (IV) :** Now, B is top node of the stack. Its neighbours are E and D.

E	D	

B

- **Step (V) :** E is top node of the stack. We find its neighbour, but there is no neighbour of E in the graph, so

B	D	

E

- **Step (VI) :** Now, D is top node; and its neighbours are F and C; we push D into the stack.

F	C	

D

- **Step (VII) :** Now, G is our top node of the stack, which is our goal node.
- **Step (VIII) :** solution cost is

$A \rightarrow B \rightarrow E \rightarrow D \rightarrow G$

3.6 BREADTH-FIRST SEARCH (BFS)

GQ. Comment upon statement that breadth-first search is a special case of uniform cost search.

GQ. Explain breadth first search with its algorithm.

(1) BFS stands for Breadth-First Search is a vertex based technique for finding a shortest path in graph. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.

(2) BFS is the core of many graph analysis algorithms and it is used in many problems, such as social network, computer network analysis and data organization.

(3) BFS involves search through a tree one level at a time. We traverse through one entire level of children nodes first, before moving onto traverse through the grand children nodes.

(4) Breadth first search is an algorithm for node that satisfies a given property. It starts at the **tree root** and

Scanned with CamScanner

explores all nodes at the present depth prior to moving on to nodes at the next depth level.

(5) BFS uses **Queue-data structure** for finding the shortest path. BFS can be used to find **single source shortest path** in an un-weighted graph, because in BFS, we reach a vertex with minimum number of edges from a source vertex.

3.6.1 BFS Traversal Algorithm

- **Step 1 :** Add a node / vertex from the graph to a queue of nodes to be 'visited'.,
  ↓
- **Step 2 :** Visit the topmost node in the queue, and mark it as such.
  ↓
- **Step 3 :** If that node has any neighbours, check to see if they have been 'visited' or not.
  ↓
- **Step 4 :** Add any neighbouring nodes that still need to be 'visited' to the queue.

Illustrative Example

Ex. 3.6.1 : Which solution would BFS find to move from node S to node G if run on the graph below ?

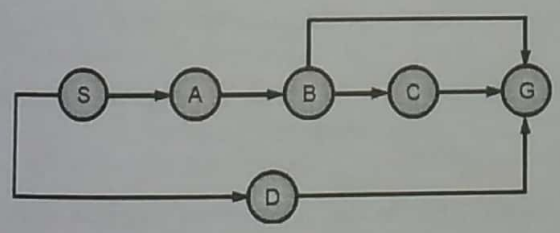


Fig. Ex. 3.6.1

Soln. :

The equivalent search tree for the above graph is as follows :

As BFS traverses the tree "shallowest node first", it would always pick the shallower branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown by dotted line.

Path : $S \rightarrow D \rightarrow G$

= the depth of the shallowest solution.

= number of nodes in level.

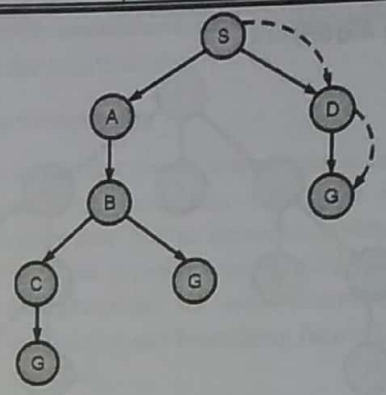


Fig. Ex. 3.6.1(a)

3.6.2 Performance Measures for BFS

Time complexity

Equivalent to the number of nodes traversed in BFS until the shallowest solution.

Note : $T(n) = 1 + n^2 + n^3 + ... + n^s = O(n^s)$.

Space complexity

Equivalent to how large can the fringe get :

$$S(n) = O(n^s)$$

Completeness

BFS is complete, meaning for a given search tree, BFS will come up with a solution if it exists.

Optimality

BFS is optimal as long as the costs of all edges are equal.

3.6.3 BFS Algorithm for Extra Memory

- BFS is an algorithm for searching a tree data structure for a node that satisfies a given property.
- It starts at the tree root and explores all nodes at the present depth prior to moving on to the nodes at the next depth level.
- Extra memory, usually a queue, is needed to keep track of the child nodes that were encountered but not yet explored.

Module 3

Scanned with CamScanner

Sorting algorithm

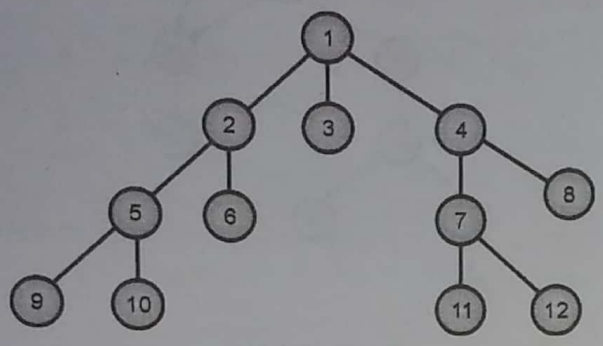


Fig. 3.6.1 : Order in which nodes are expanded

Concept diagram

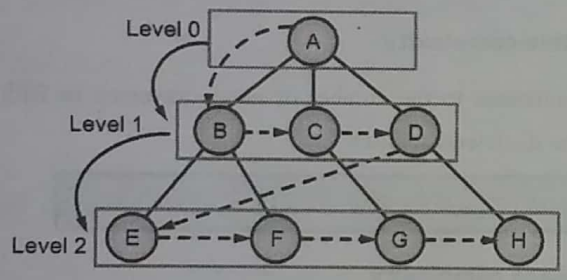


Fig. 3.6.2

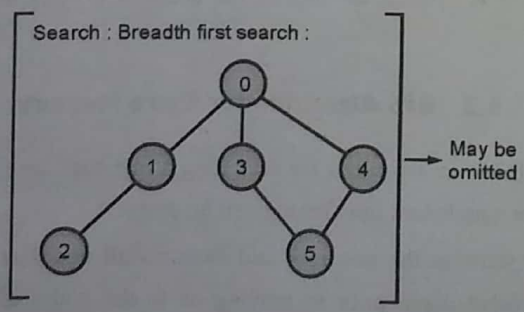
Algorithm

(1) Mark any node as starter or initial

(2) Explore and traverse un-visited nodes adjacent to starting node.

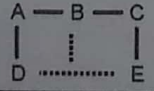
(3) Mark node as completed and move to next adjacent and un-visited nodes.

Search : Breadth first search



BFS will always find the shortest path in an unweighted graph.

Consider an unweighted graph like this :



And my goal is to get from A to E.

I begin at A as it my origin. I queue A, followed by immediately dequeueing A and exploring it. This yields B and D, because A is connected to B and D. I thus queue both B and D.

I cheque B and explore it, and find that it leads to A (already explored), and C, so I queue C. I then dequeue D, and find that it leads to E, and that is my goal.

I then deque C, and find that it also leads to E, my goal.

- BFS can only be used to find shortest path in a graph if
  1. There are no loops.
  2. All edges have same weight or no weight.
- To find the shortest path, all you have to do is start from the source and perform a breadth first source and stop when you find your destination node.
- 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 advantage of both algorithms.

3.6.4 Execution of BFS Algorithm

- **Step 1 :** Start by putting any one of the graph's vertices at the back of the queue.

  ↓

- **Step 2 :** Take the front item of the queue and add it to the visited list.

  ↓

- **Step 3 :** Create a list of that vertex's adjacent nodes.

  ↓

- **Step 4 :** Keep continuing steps two and three till the querry is empty.

3.6.5 Advantages and Disadvantages of BFS

Advantages

(1) Solution will be definitely found out by BFS if there is some solution.

(2) BFS will never get trapped in blind valley, means unwanted nodes.

(3) If there are more than one solution then it will find solution with minimal steps.

Scanned with CamScanner

Disadvantages

(1) Memory constraints as it stores all the nodes of present level to go for next level.

(2) If solution is far away then it consumes time.

3.6.6 Applications of BFS Algorithm

We mention some of the applications where a BFS algorithm implementation can be highly effective.

(1) **Unweighted graphs :** BFS algorithm can easily create the shortest path and a minimum spanning tree to visit all the vertices of the graph in the shortest time possible with high accuracy.

(2) **P2P Networks :** BFS can be implemented to locate all the nearest or neighbouring nodes in a peer to peer network. This will find the required data faster.

(3) **Web Crawlers :** Search engines or web crawlers can easily build multiple levels of indexes by employing BFS. BFS implementation starts from the source, which is the web page, and then it visits all the links from that source.

(4) **Navigation Systems :** BFS can help find all the neighbouring locations from the main or source location.

(5) **Network Broadcasting :** A broadcasted packet is guided by the BFS algorithm to find and reach all the nodes it has the address for.

- It is possible to run BFS recursively without any data structures, but with higher complexity.
- DFS, as opposite to BFS, uses a stack instead of a queue, so it can be implemented recursively. Note that the code used is iterative but it is trivial to make it recursive.
- In BFS, a queue data structure is used. One can mark any node in the graph as root and start traversing the data from it.
- BFS traverses all the nodes in the graph and keeps dropping them as completed.
- BFS visits an adjacent unvisited node, marks it as done, and inserts it into a queue.

3.6.7 Performance Measures of BFS

Time Complexity

Breadth-first search, being a brute search generates all the nodes for identifying the goal. The amount of time taken for generating these nodes is proportional to the depth d and branching factor b and is given by,

$$1+b+b^2+b^3+.....+b^d \simeq b^d$$

Hence the time-complexity = $O(b^d)$

Space Complexity

Unlike depth-first search wherein the search procedure has to remember only the paths it has generated, breadth-first search procedure has to remember every node it has generated. Since the procedure has to keep track of all the children it has generated, the space-complexity is also a function of the depth d and **branching factor b**. Thus space complexity becomes,

$$1+b+b^2+b^3+...+b^d \approx b^d$$

Hence the space-complexity = O (bd)

3.6.8 Limitations of Breadth First Search

1. Amount of time needed to generate all the nodes is considerable because of the time-complexity.
2. Memory constraint is also a major hurdle because of the space-complexity.
3. The searching process remembers all unwanted nodes which is of no practical use for the search.

Module 3

GQ. Which storage structure is preferably chosen for node representation in open list, while performing best-first search over a state space and why?

OPEN is a priority queue in which the elements with the highest priority are those with the most promising value of the heuristic function.

3.6.9 Example of Breadth First Search

GQ. Give an example of a problem for which breadth-first would work better than depth first search and vice-vesa.

In general, BFS is better for problems related to finding the **shortest paths** or somewhat related problems. Because here we can go from one node to all node that are adjacent to it and hence we effectively move from path length one to path length two and so on.

- While DFS on the other hand, helps more in connectivity problems and also in finding cycles in graph (cycles can be found in BFS with a bit of modification). Determining connectivity with DFS is trivial, if we call the explore procedure twice from the DFS procedure, then the graph is disconnected (this is for an undirected graph). We can see the strongly connected component algorithm for a directed graph here, which is a modification of DFS. Another application of the DFS is topological sorting.

Scanned with CamScanner

3.6.10 Solved Example on BFS

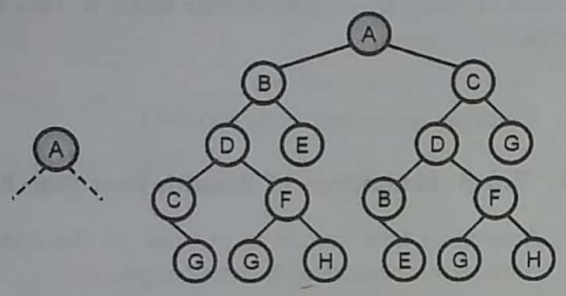
UEx. 3.6.2 (MU - Q. 5(A), Dec. 17, 10 Marks)

Consider the given tree, apply breadth first search algorithm and also write the order in which 10 nodes are expanded.

Soln. :

Step 1 :

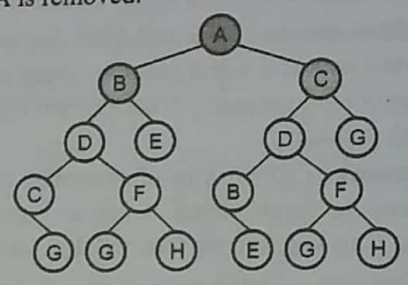
Node, A can be regarded as source node.



(1B1)Fig. Ex. 3.6.2

FRINGE : A

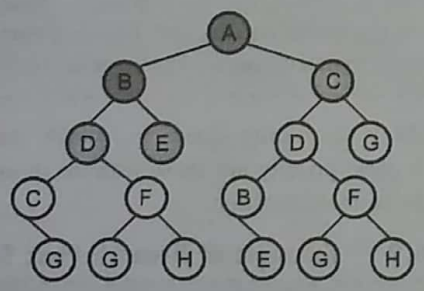
Step 2 : We consider sub-nodes B and C of A. Source node A is removed.



(1B2)Fig. Ex. 3.6.2(a)

FRINGE : B C

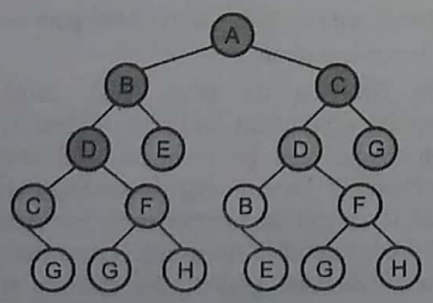
Step 3 : We begin with left nodes. Consider left node B and its subnodes D and E. We remove node B..



(1B3)Fig. Ex. 3.6.2(b)

FRINGE : D E D G

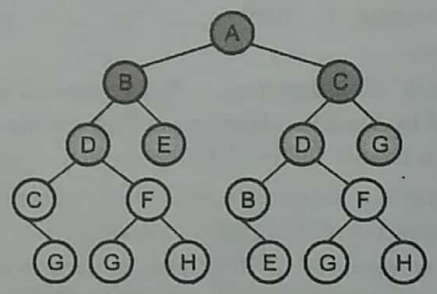
Step 5 : We consider left subnode C and its subnodes are D and G. We remove subnode C from the fringe and add subnodes D and C to the fringe.



(1B5)Fig. Ex. 3.6.2(d)

FRINGE : C D E

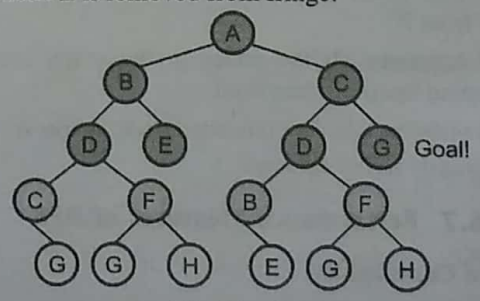
Step 4 : We consider left subnode D. Its subnodes are C and F. We remove node D.



(1B4)Fig. Ex. 3.6.2(c)

FRINGE : E D G C F

Step 6 : Now, consider right subnode E of B and subnode E is removed from fringe.



(1B6)Fig. Ex. 3.6.2(e)

Step 7 : Now, subnode D is expanded.

∴ **Fringe : G C F B F**

Step 8 : Now, we select G for expansion. It is found to be a goal node. And the algorithm terminates.

Scanned with CamScanner

Performance Evaluation

- **Time complexity :** $O(b^d)$
- **Space complexity :** O(bd)

  Where b - branching factor, and d - Depth of the shallowest goal node
- **Optimality** : can be reached

Advantage

Path of minimal length to the goal can be found out.

Disadvantages

1. The search-space should be small in order to use Breadth First Search (BFS) algorithm.
2. Generation and storage of a tree is required whose size is exponential (b^d), the depth of the shallowest goal node.

3.6.11 Application of BFS

1. Finding shortest path.
2. Checking graph with bipertiteness.
3. Copying Cheney's algorithm.

3.7 UNIFORM COST SEARCH

- Uniform-cost-search is an uninformed search algorithm that uses the lowest cumulative cost to find a path from the source to the destination.
- Nodes are expanded, starting from the root, according to the minimum cumulative cost. The uniform-cost search is then implemented using a Priority Queue.
- Here, instead of inserting all vertices into a priority queue, we insert only source, then one by one we insert nodes when needed.

3.7.1 Algorithm of U.C.S.

- Uniform Cost Search is an algorithm used to move around a directed weighted search space to go from a start node to one of the ending nodes with a minimum cumulative cost.
- This search is an uninformed search algorithm, i.e. it does not take the state of the node or search space into consideration.
- It is used to find the path with the lowest cumulative cost in a weighted graph where nodes are expanded according to their cost of traversal from the root node. This is implemented using a priority queue where lower the cost higher is its priority.

Algorithm of Uniform Cost Search : (in AI)

- **Step 1 :** Insert Root Node into the queue.

  ↓
- **Step 3 :** Repeat till queue is not empty.

  ↓
- **Step 3 :** Remove the next element with the highest priority, from the queue.

  ↓
- **Step 4 :** If the node is a destination node, then print the cost and the path and exit, else insert all the children of removed elements into the queue with their cumulative cost as their priorities.

Here root Node is the starting node for the path, and a priority queue is being maintained to maintain the path with the least cost to be chosen for the next traversal.

3.7.2 Execution of Algorithm of Uniform Cost Search

- Uniform-cost search is similar to Dijkstra's algorithm. In this algorithm,

- **Step 1 :** from the starting state we will visit the adjacent states and will choose the least costly state.

  ↓
- **Step 2 :** then we choose the next costly state from the all un-visited and adjacent states of the visited states,

  ↓
- **Step 3 :** in this way we try to reach the goal state.

Module 3

Scanned with CamScanner

Remark

Even if we reach the goal state we continue searching for other possible paths (if there are multiple goals).

- The elements in the priority queue have almost the same costs at a given time, and thus the name Uniform Cost Search.
- It may appear that elements do not have almost the same costs, but when applied on a much larger graph it is certainly so.
- Uniform costing refers to acceptance of identical costing principles and procedures by all or many units in the same industry by mutual agreement.
- 'Uniform Cost Search (UCS)' algorithm is mainly used when the step costs are not the same but we need the optimal solution to the goal state. In such cases, we use Uniform Cost Search, to find the goal and the path including the cumulative cost to expand each node from the root node to the goal node.
- Uniform cost-search is optimal. This is because, at every step the path with the least cost is chosen, and paths never get shorter as nodes are added, ensuring that the search expands nodes in the order of their optimal path cost. To measure the time complexity, we need the help of path cost instead of depth d.

3.7.3 Example of Uniform Cost Search (Linear Displacement)

Consider the example if Fig. 3.7.1; where we need to reach any one of the destination node $\{G_1, G_2, G_3\}$ starting from node S.

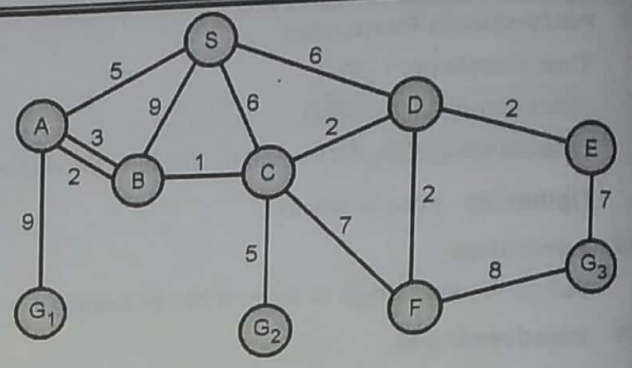


Fig. 3.7.1

Node { A, B, C, D, E and F } are the intermediate nodes. Our motive is to find the path from S to any of the destination state with the least cumulative cost. Each directed edge represents the direction of movement allowed through that path, and its labelling represents the cost is one travels through that path.

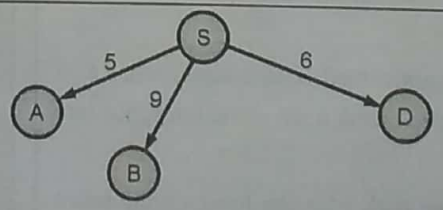
Thus overall cost of the path is a sum of all the paths.

For e.g. : A path from S to G_1 –

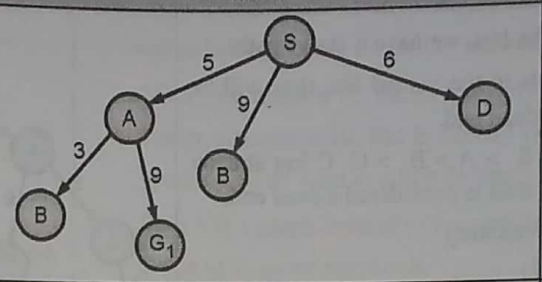
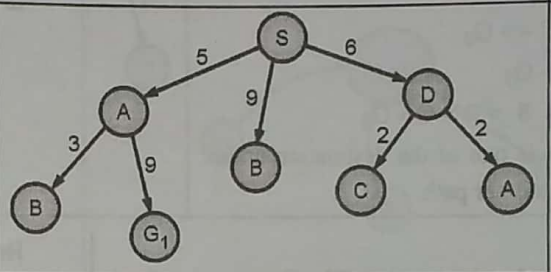
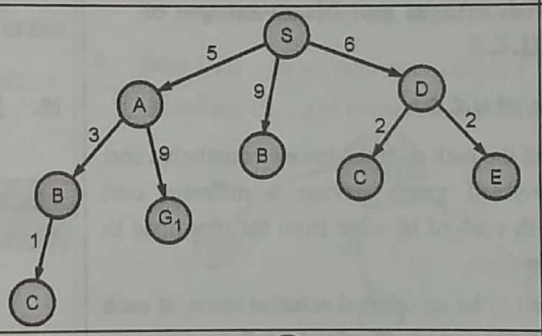
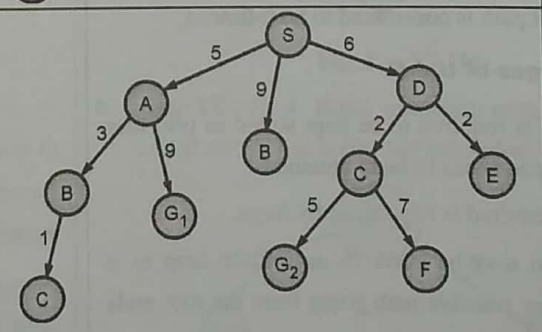
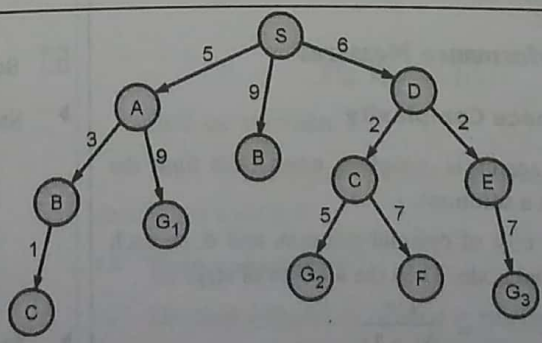
$\{ S > A > G_1 \}$ whose cost is $SA + AG_1 = 5 + 9 = 14$.

Here we maintain a priority queue the same as BFS with the cost of the path as its priority, lower the cost higher is the priority.

We use a tree to show all the paths possible and also maintain a visited list to keep track of all the visited nodes as we need not visit any node twice.

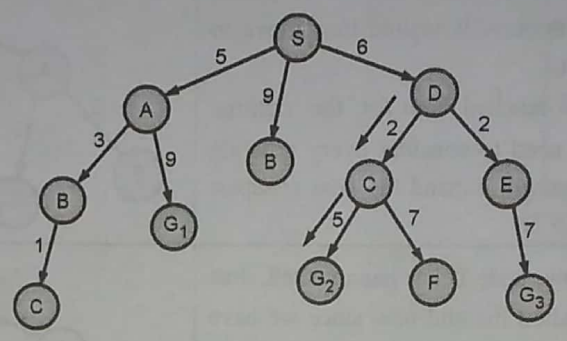
Explanation	Flow	Visited List
Step I : We start with start node and check if we have reached any of the destination, nodes, i.e. No thus continue.		
Step 2 : We reach all the nodes that can be reached from S i.e. A, B, D. And since node S has been visited, thus added to the visited list. Now we select the cheapest path first for further expansion i.e. A.		S

Scanned with CamScanner

Explanation	Flow	Visited List
Step 3 : Node B and G_1 can be reached from A and since node A is visited thus move to the visited list. Since G_1 is reached but for the optimal solution, we need to consider every possible case, thus, we will expand the next cheapest path i.e. S –> D		S, A
Step 4 : Now, node D has been visited, thus it goes to visited list and now since we have three paths with the same cost, we choose alphabetically thus will expand node B.		S, A, D
Step 5 : From B, we can only reach node C. Now the path with minimum weight is S –> D –> C i.e. 8 Thus expand C, and B has now visited D node.		S, A, D, B
Step 6 : From C we can reach G_2 and F node with 5 and 7 weights respectively. Since S is present in the visited list, thus we are not considering the C –> S path. Now, C will enter the visited list. Now the next node with the minimum total path is S > D > E , i.e. 8 Thus we will expand E.		S, A, D, B, C
Step 7 : From E we can reach only G_3. E will move to the visited first.		S, A, D, B, C, E

Module 3

Scanned with CamScanner

Explanation	Flow	Visited List
Step 8 : In the last, we have 6 active paths. S – > B B is in the visited list, thus will be marked as a dead end. Same for S > A > B > C C has already been visited thus is considered a dead end. Out of the remaining S > A – > G_1 S > D – > C – > G_2 S > D > E > G_3 Minimum is S > D > C > G_2 and also G_2 is one of the destination nodes. Thus we found our path.		S, A, D, B, C, E

3.7.4 Advantages and Disadvantages of U.C.S

Advantages of U.C.S

1. It helps to find the path with the lowest cumulative cost inside a weighted graph having a different cost associated with each of its edge from the root node to the destination node.
2. It is considered to be an optimal solution since, at each state, the least path is considered to be followed.

Disadvantages of *U.C.S*

1. The open list is required to be kept sorted as priorities in priority queue needs to be maintained.
2. The storage required is exponentially large.
3. The algorithm may be stuck in an infinite loop as it considers every possible path going from the root node to the destination node.

3.7.5 Performance Measures

Time and Space Complexity

Uniform cost search is complete, when UCS finds the solution, (if there is a solution).

Let C* be the cost of optimal solution, and $\in$ be each step closer to the goal node. Then the number of steps is,

$$= \frac{C^*}{(\in + 1)}$$

Here, we have taken + 1, as we start from state 0 and end to $\frac{C^*}{\in}$.

3.7.6 Solved Example on Curved Displacement

UEx. 3.7.1 **MU - Q. 1(b), Dec. 16, 5 Marks**

Apply uniform cost search algorithm on given graph.

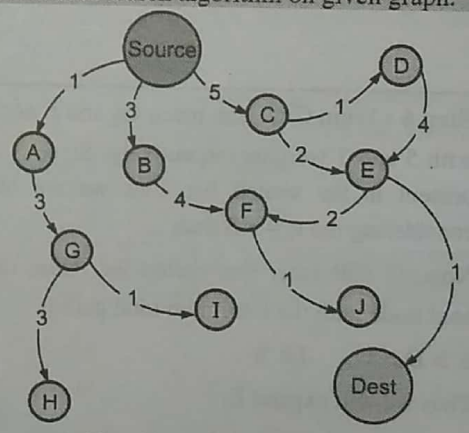


Fig. Ex. 3.7.1 : Graph

Soln. :

Step I : We mention source-node :

Fig. Ex. 3.7.1(a) : Node

Step II : We add the nodes A, B, C to the source node, :

Scanned with CamScanner

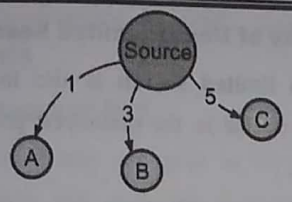


Fig. Ex. 3.7.1(b)

- **Step III :** The node A has minimum distance 1, so we keep it aside and add the node G.

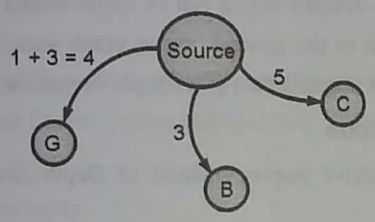


Fig. Ex. 3.7.1(c)

- **Step IV :** For nodes, B and C, B has minimum distance, so we add node F to B.

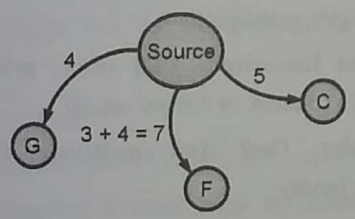


Fig. Ex. 3.7.1(d)

- **Step V :** Now, G has minimum distance, so we keep G aside and add H, note that C and I have same distance ; we remove C or I alphabetically ; but I has no further subnodes.

  Now, we remove D, and bring subnode E, with total distance of 10. But E has already lesser distance 7, so we add E with distance 7.

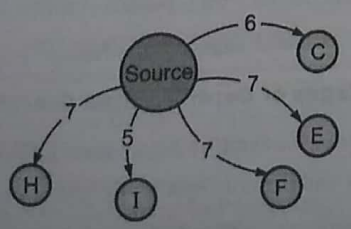


Fig. Ex. 3.7.1(e)

- **Step VI :** I has minimum distance, but I has no subnode, i.e. there is no further updating.

  Node, we remove D, but D has only one sub-node E, but its distance is 10. But E already exists with a lesser distance, so no need to add it further.

- **Step VII :** (Alphabetically) the next minimum distance is that of E, so we remove E.

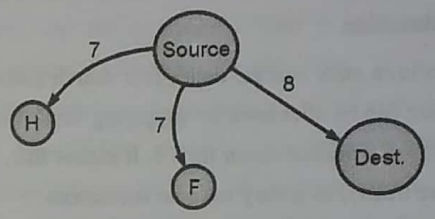


Fig. Ex. 3.7.1(f)

- **Step VIII :** Now, minimum cost is F, so it is removed (alphabetically) and subnode J is added.

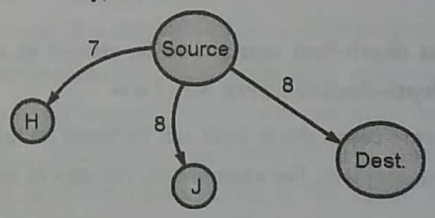


Fig. Ex. 3.7.1(g)

- **Step IX :** Now, H has minimum cost, so it is removed and H also H has no further subnodes.

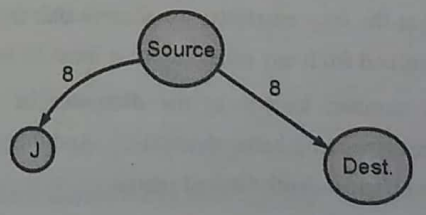


Fig. Ex. 3.7.1(h)

Here the algorithm ends.

Thus, the minimum distance between the source and destination node is **8**.

☞ Time complexity

The time complexity needed to run uniform cost search is : $O(b(1 + C / \varepsilon))$

Module 3

Scanned with CamScanner

Where : b - branching factor, C - optimal cost, ε - cost of each step

Optimal :

Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

3.8 DEPTH LIMITED SEARCH

(I) Explanation

In infinite state spaces, depth-first search method fails. This failure can be alleviated by supplying depth-first search with a pre -determined depth limit *l*. It means that, nodes at depth *l* are treated as if they have no successors.

This approach is called **depth-limited search**.

The depth limit can solve infinite path problem.

Its time-complexity is O (b^l) and its space-complexity is O (b*l*)

Thus depth-first search can be viewed as a special case of depth-limited search with $l = \infty$

In some cases, depth limit can be based on knowledge of the problem. For example, on the map of surat there are

25 cities.

So, if there is a solution, it must be of length 24 at the longest, hence $l = 24$ is a possible choice. But if we look at the map carefully, we observe that any city can be reached from any other city in at most 15 steps.

This number, known as the **diameter** of the **state space**, gives us a better depth limit. And this leads to a more efficient depth-limited search.

Observe that **depth-limited search** can terminate with two kinds of failure :

(i) the standard **failure value** which indicates no solution;

(ii) the **cut-off value** which indicates no solution within the **depth-limit**

☞ Drawbacks of Depth-Limited Search

(i) The **depth limited search** is also incomplete if we choose $l < d$; that is, the shallowest goal is beyond the depth limit.

(ii) Depth-limit search will also be non-optimal if we choose $l > d$.

Implementation

Depth –limited search can be implemented as a simple modification to the general tree-or graph search algorithm. Or, it can be implemented as a **simple recursive algorithm.**

(II) Algorithm

A recursive implementation of Depth-limited Search Algorithm

Function : Depth-Limited-search (Problem, limit) **returns** a solution, or failure/cut off.

Return : Recursive - DLS (Make-Node (Problem. Initial-state), problem, limit)

Function Recursive - DLS (node, problem, limit) **returns** a solution, or failure/ cut-off

If problem. Goal -Test (node-state) **then return** solution (node)

else if limit = 0 then **return cut off**

else

cut off-occurred ? ← false

for each action **in** problem. Action (node, state) **do**

child ← child – Node (problem, node, action)

result ← Recursive – DLS (child, problem, limit- 1)

If result = cutoff **then** cutoff-occurred? ← true

else if result ≠ failure **then return** result

If cutoff-occurred ? **then return** cutoff

else return failure

☞ Advantages of Depth Limited Search

1. Depth limited search is better than DFS as it requires less time and memory space.
2. DFS assures that the solution will be found if it exists in infinite time.

Scanned with CamScanner

3. DLS has applications in graph theory particularly similar to DFS.

Disadvantages of DLS

(i) The goal node may not exist in the depth limit set earlier, which will push the user to iterate further adding execution time.

(ii) The goal node cannot be found out if it does not exist in the desired limit.

Optimality

The DLS is a non-optimal algorithm since the depth that is chosen can be greater than d ($l > d$). Thus DLS is not optimal if $l > d$.

Time complexity

It is similar to DFS, i.e. O (b^l), where 1 is the specified depth limit.

Space complexity

It is similar to DFS, it is O (b^l), where 1 is the specified depth limit.

Conclusion – DLS

(i) DLS is not the case for uninformed search strategy.

(ii) DLS algorithm is used when we know the search domain, and there exists a prior knowledge of the problem and its domain.

(iii) There is little idea of the goal nodes depth.

(iv) The problem with depth-limited search is to set the value of *l* optimally, so as not to leave out any solution.

Also keep the time and space complexity to a minimum.

3.9 ITERATIVE DEEPENING SEARCH TECHNIQUE (IDS OR IDDFS)

UQ. Explain Iterative Deepening search algorithms based on performance measure with justification; complete, optimal, Time and Space complexity.

(MU - Q. 4(a), Dec. 18, 10 Marks)

1. Iterative deepening search or more specifically iterative deepening depth-first search (IDS or IDDFS) is a **state space/graph search strategy** in which a **depth-limited version** of depth-first search is run repeatedly with increasing depth limits until the goal is found.
2. IDDFS is equivalent to breadth-first search, but uses much less memory; on each iteration, it visits the nodes in the search tree in the same order as depth-first search, but the cumulative order in which nodes are first visited is effectively breadth- first.
3. DDFS combines depth-first search's space-efficiency and breadth-first search's completeness (When the branching factor is finite). It is optimal when the **path cost is a non-decreasing function of the depth of the node.**
4. The **time complexity of IDDFS is O (b^d)** and its **space complexity is O(b^d),** where **b** is the branching factor and **d** is the depth of the shallowest goal.
5. Since iterative deepening, visits states multiple times, it may seem wasteful, but it turns out to be not costly, since in a tree most of the nodes are in the bottom level, so it does not matter much if the upper levels are visited multiple times.

Module 3

3.9.1 IDDFS Algorithm

```
Function iterative-Deepening-search (problem) returns a
solution or failure
inputs : problem, a problem
for depth ← O to ∞ do
result ← Depth-Limited-Search (problem, depth)
if result ≠ cutoff then return result
```

The iterative deepening search algorithm, which repeatedly applies depth-limited search with increasing limits.

It terminates when a solution is found or if the depth-limit search returns **failure** meaning that no solution exists.

Scanned with CamScanner

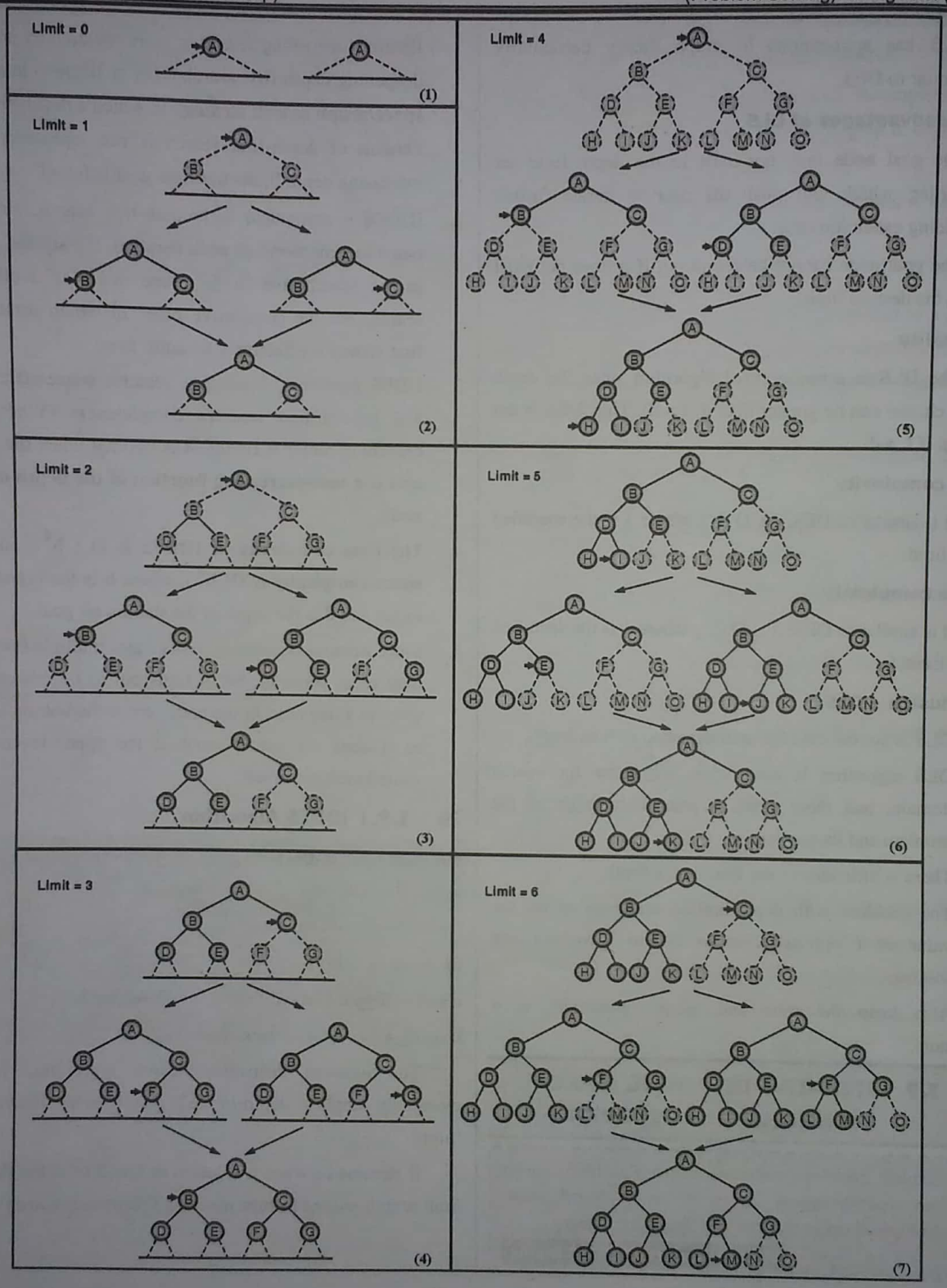


Fig. 3.9.1

Scanned with CamScanner

3.9.2 Advantages of IDDFS

1. The **main advantage** of IDDFS in game tree searching is that the earlier searches tend to improve the commonly used heuristics, such as the killer heuristic and alpha- beta pruning, so that a more accurate estimate of the score of various nodes at the final depth search can occur and the search completes more quickly since it is done in a better order.

   For example, alpha-beta pruning is most efficient if it searches the best moves first.

2. A **second advantage** is the responsiveness of the algorithm. Because early iterations use small values for **d**, they execute extremely quickly. This allows the algorithm to supply early indications of the result almost immediately, followed by refinements as d increases, when used in an interactive setting. Such as in a chess-playing program, this facility allows the program to play at any time with the current best move found in the search it has completed so far. This can be phrased at each depth of the search core, producing a better approximation of the solution, though the work done at each step is recursive. This is not possible with a traditional depth-first search, which does not produce intermediate results.

3. The time complexity of IDDFS in well-balanced trees works out to be the same as **Depth-first search: O (b^d)**

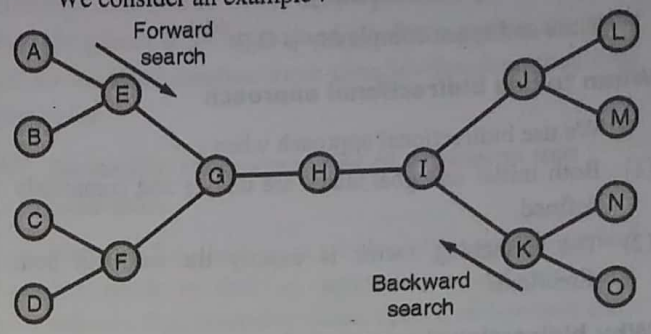
3.10 BIDIRECTIONAL SEARCH

The principle used in a bidirectional heuristic search algorithm is to find the shortest path from the current node to the goal node. The only difference being the two simultaneous searches from the initial point and from goal vertex. The main idea behind bidirectional searches is to reduce the time taken for search drastically.

This takes place when both searches happen simultaneously from the 'initial node depth' or 'breadth-first' and 'backwards from goal nodes'. They intersect somewhere in between of the graph.

The path traverses from the initial node through the intersecting point to goal vertex and that is the shortest path found because of this search.

We consider an example :



Bidirectional search algorithm

- **Step I :** Let A be initial node and O the goal node and H is the intersection node.
- **Step 2 :** We start searching simultaneously from start to goal node and backwards from goal node to start node.
- **Step 3 :** When the forward search and backward search intersect at one node, then searching stops.

Also observe that bidirectional searches are complete if a breadth-first search is used for both traversals, i.e., for both paths from start node till intersection and from goal node till intersection.

Two main types of bidirectional searches are as follows :

(1) Front to Back or B F E A

(2) Front to Front or B F F A

(1) Front to Back or B F E A

In bidirectional front, to front search, two heuristic functions are needed.

First is the estimated distance from a node to goal state using forward search and second, node to start state using reverse action. Here, h is calculated in the algorithm, and it is the heuristic value of the distance between the node n to the root of the opposite tree s or t. This is the most widely used bidirectional search algorithm.

(2) Front to front BFFA

Here the distance of all nodes is calculated, and h is calculated as the minimum of all heuristic distances from the current node to nodes on opposing fronts.

Performance measure

(1) **Completeness :** Bidirectional search is complete if BFS (Breadth-first search) is used in both searches.

(2) **Optimality :** It is optimal if BFS is used for search and paths have uniform cost.

Module 3

Scanned with CamScanner

(3) Time and space complexity

Time and space complexity is O ($b^{d/2}$).

When to use bidirectional approach

We use bidirectional approach when :

(1) Both initial and goal states are unique and completely defined.

(2) The branching factor is exactly the same in both directions.

Why bidirectional approach ?

(i) In many cases it is faster, and it reduces the amount of required exploration.

(ii) Suppose if branching factor of tree is b and distance of goal vertex from source is d, then the normal BFS/DFS searching complexity is O (b^d). But for two search complexity is O ($b^{d/2}$) which is far less than O (b^d).

3.11 INFORMED SEARCH

GQ. What is informed search ?

Informed search (heuristic search) : This can decide whether one non-goal state is more promising than another non-goal state. The advantages of the informed search comes from the fact that :

1. It adds **domain-specific information** to select the best path along which to continue searching.
2. Define a heuristic function **h(n)** that estimates the "goodness" of a node n. Specifically, **h(n) = estimated cost** (or distance) of minimal cost path from n **to a goal state.**
3. The heuristic function is an estimate of how close we are to a goal, based on domain-specific information that is computable from the current state description. Some of the examples of informed search are best first search, beam search, **A*** and **AO*** algorithms etc.

Informed search algorithms

Informed search algorithm contains an array of knowledge that tells us how far we are from the goal, path cost, how to reach to goal node etc. This knowledge helps agents to explore less to the search space and find the goal node more efficiently.

The informed search algorithm is more useful for large search space. Informed search uses the idea of heuristic, hence it is also called as Heuristic search.

3.11.1 Example for Informed Search

GQ. Give an example for informed search.

Beam search is an example for informed search

Beam search: This is an attractive heuristic search technique because it permits searching to be done on a **multi-processor machine**, thereby reducing computations. Reduction in computations is achieved by pursuing some paths and selecting only selected paths.

The searching process is similar to breadth-first search wherein searching proceeds level by level. At each level, heuristic functions are applied to reduce the number of paths to be explored. In fact, it is done to keep the width of the beam to be minimal. The width of the beam is fixed and whatever be the depth of the tree, the number of alternatives to be scanned is the product of the width and the depth.

3.11.2 Algorithm for Beam Search

- **Step 1 :** Let width_of_beam = w.
- **Step 2 :** Put the initial node on a list START.
- **Step 3 :** If (START is empty) or (START = GOAL), then terminate search.
- **Step 4 :** Remove the first node from START, Call this as node a.
- **Step 5 :** If (a = GOAL), then terminate search with success.
- **Step 6 :** Else if node a has successors, generate all of them and add them at the tail of START.
- **Step 7 :** Use a heuristic function to rank and sort all the elements of START.
- **Step 8 :** Determine the nodes to be expanded. The number of nodes should not be greater than w. Name these as START1.
- **Step 9 :** Replace START with START1.
- **Step 10 :** Goto Step 2.

Fig. 3.11.1 shows how beam search proceeds. This search has been used in an expert system called **ISIS for factory scheduling**.

Scanned with CamScanner

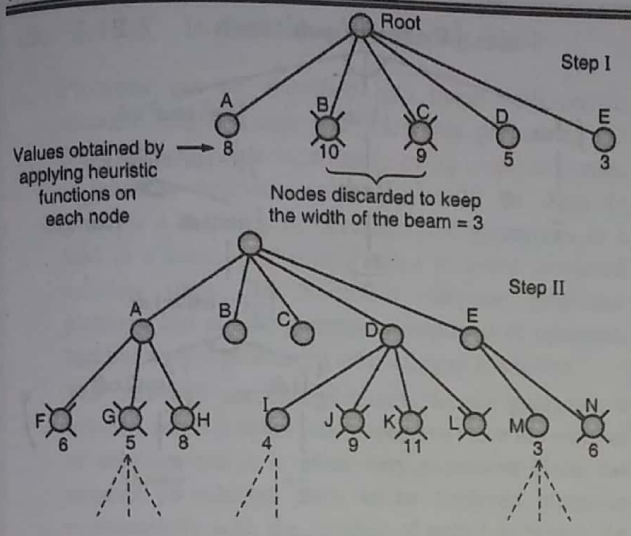


Fig. 3.11.1 : Beam search procedure

3.12 HEURISTIC FUNCTION

UQ. What is heuristic function ?

MU - Q. 3(b), Dec. 18, 10 Marks, Q. 1(c), May 18, 4 Marks, Q. 1(c), May 17, 4 Marks, Q. 3(b), Dec. 16, 5 Marks, Q. 1(c), Dec. 15, 3 Marks

The process of searching can be drastically reduced by the use of heuristics. Heuristics are approximations used to minimize the searching process.

- It is a function that maps from problem state description to measures of desirability, usually represented as number. Which aspect of the problem state are considered, how these aspects are evaluated, and weight given to the individual aspects are chosen. Define a heuristic function h(n) that estimates the "goodness" of a node n.

  Specifically, **h(n) = estimated cost (or distance) of minimal cost path from n to a goal state.**
- The heuristic function is an estimate of how close we are to a goal, based on domain-specific information that is computable from the current state description.

Computed in such a way that the value of the heuristic function at a given node in the search give as good estimate as possible of whether that node is on the desired path to a solution.

Well-designed heuristic function can play an important role in efficiently guiding a search process toward a solution. Sometimes very simple heuristic function can provide a fairly good estimate of whether a path is any good or not. In other situation more complex function should be employed.

Generally, two categories of problems use heuristics

1. Problems for which no exact algorithms are known and one needs to find an approximate and satisfying solution. E.g., computer vision, speech recognition etc.
2. Problems for which exact solutions are known, but computationally infeasible. E.g., Rubik's cube, chess etc. The heuristics which are needed for solving problems are generally represented as a heuristic function which maps the problem states into number. These numbers are then appropriately used to guide search.

3.12.1 Simple Heuristic Functions

Module 3

1. In the famous 8-tile puzzle, the hamming distance is a popular heuristic function. It is an indicator of the number of tiles in the position they are to be.
2. In a game like chess, the material advantage one has over the opponent is an indicator. Normally, the following values are assigned to the pieces. Queen-9, etc.

The following algorithms make use of heuristic evaluation functions

1. Hill climbing	2. Constraint satisfaction
3. Best-first search	4. A* algorithm
5. AO* algorithm	6. Beam search

- The purpose of heuristic function is to guide the search process in the most profitable direction by suggesting which path to follow first when more than one is available.
- The more accurate the heuristic function estimate the true merit of each node in the search tree, the more direct the solution process.
- In the extreme, the heuristic would be so good that essentially no search would be required system would move directly to a solution.
- But for many problems, the cost of computing the value of such a function would outweigh the effort saved in the search process.

Scanned with CamScanner

- In general, there is a trade-off between the cost of evaluating a heuristic function and the saving of search time that the functions provide.

3.12.2 Problem Characteristics for Heuristic Search

GQ. What are the various problem characteristics for heuristic search? **Or** Explain the problem characteristic briefly with appropriate example.

Heuristic search is a very general method applicable to a large class of problems. It encompasses a variety of specific techniques, each of which is particularly effective for a small class of problems.

Several key dimensions for Heuristic Search

1. Is the problem decomposable into a set of (nearly) independent smaller or easier sub problems?
2. Can solution steps be ignored or at least undone if they prove unwise?
3. Is the problem's universe predictable?
4. Is a good solution to the problem obvious without comparison to all other possible solutions?
5. Is the desired solution a state of the world or a path to a state?
6. Is a large amount of knowledge absolutely required to solve the problem, or is knowledge important only to constrain the search?
7. Can a computer that is simply given the problem return the solution, or will the solution of the problem require interaction between the computer and a person?

3.12.3 Is the Problem Decomposable ?

A very large and composite problem can be easily solved if it can be broken into smaller problems and recursion could be used. Suppose we want to solve.

Ex : $\int (x^2 + 3x + \sin^2 x \cos 2x)\, dx$

This can be done by breaking it into three smaller problems and solving each by applying specific rules. On adding the results, complete solution is obtained.

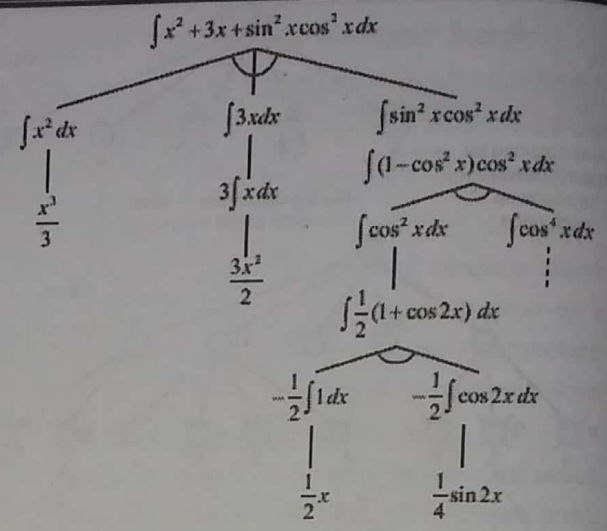


Fig. 3.12.1

3.12.4 Can Solution Steps be Ignored or Undone ?

- Problem fall under three classes ignorable, recoverable and irrecoverable. This classification is with reference to the steps of the solution to a problem. For example, consider proving the theorem. We may later find that it is of no help. We can still proceed further, since nothing is lost by this redundant step. This is an example of ignorable solutions steps.
- Now consider the 8 puzzle problem tray and arranged in specified order. While moving from the start state towards goal state, we may make some stupid move and consider theorem proving. We may proceed by first proving lemma. But we may backtrack and undo the unwanted move. This only involves additional steps and the solution steps are recoverable.
- Lastly consider the game of chess. If a wrong move is made, it can neither be ignored nor be recovered. The thing to do is to make the best use of current situation and proceed. This is an example of an irrecoverable solution steps.

(i) Ignorable problems

Ex : Theorem proving

In which solution steps can be ignored.

(ii) Recoverable problems Ex: 8 puzzle

In which solution steps can be undone.

(iii) Irrecoverable problems Ex : Chess

In which solution steps can't be undone.

A knowledge of these will help in determining the control structure.

Scanned with CamScanner

3.12.5 Is the Universal Predictable ?

- Problems can be classified into those with certain outcome (eight puzzle and water jug problems) and those with uncertain outcome (playing cards) in certain – outcome problems, planning could be done to generate a sequence of operators that guarantees to a lead to a solution. Planning helps to avoid unwanted solution steps. For uncertain outcome problems, planning can at best generate a sequence of operators that has a good probability of leading to a solution.
- The uncertain outcome problems do not guarantee a solution and it is often very expensive since the number of solution and it is often very expensive since the number of solution paths to be explored increases exponentially with the number of points at which the outcome cannot be predicted. Thus one of the hardest types of problems to solve is the irrecoverable, uncertain-outcome problems. Ex : Playing cards.

3.12.6 Is Good Solution Absolute or Relative ? (Is the Solution a State or a Path?)

- There are two categories of problems. In one, like the **water jug and 8 puzzle problems**, we are satisfied with the solution, unmindful of the solution path taken, whereas in the other category not just any solution is acceptable.
- We want the best, like that of travelling salesman problem, where it is the shortest path. In any path problems, by heuristic methods we obtain a solution and we do not explore alternatives. For the best path problems all possible paths are explored using an exhaustive search until the best path is obtained.

3.12.7 The Knowledge Base Consistent ?

- In some problems the knowledge base is consistent and in some it is not. For example, consider the case when a Boolean expression is evaluated.
- The knowledge base now **contains theorems and laws of Boolean algebra which are always true**. On the contrary, consider a knowledge base that contains facts about production and cost.
- These keep varying with time. Hence many reasoning schemes that work well in consistent domains are not appropriate in inconsistent domains. **Ex:** Boolean expression evaluation.

Remark

- A Boolean expression is a logical statement that is either True or False.
- Boolean expressions can compare data of any type as long as both parts of the expression have the same basic data type.
- One can test data to see, if it is equal to, greater than, or less than other data.

Boolean data is as follows :

Boolean values :

(Yes and no, and their synonyms, on and off, and true and false).

3.12.8 What is the Role of Knowledge ?

- Though one could have unlimited computing power, the size of the knowledge base available for solving the problem does matter in arriving at a good solution. Take for example, the game of playing chess, just the rules for determining legal moves and some simple control mechanism is sufficient to arrive at a solution. But, additional knowledge about good strategy and tactics could help to constrain the search and speed up the execution of the program. The solution would then be realistic.
- Consider the case of predicting the political trend. This would require an enormous amount of knowledge even to be able to recognize a solution, leave alone the best. Ex : Playing chess, newspaper understanding.

Module 3

3.12.9 Does the Task requires Interaction with the Person

The problems can again be categorized under two heads as follows :

(i) Solitary Problems
(ii) Conversational Problems

(i) Solitary Problems

In which the computer will be given a problem description and will produce an answer, with **no intermediate communication** and with the demand for an explanation of the reasoning process. Simple theorem proving falls under this category. Given the basic rules and laws, the theorem could be proved, if one exists. **Ex:** Theorem proving (give basic rules and laws to computer)

Scanned with CamScanner

(ii) Conversational Problems

In which there will be **intermediate communication between a person and the computer**, whether to provide additional assistance to the computer or to provide additional informed information to the user, or both problems such as medical diagnosis fall under this category, where people will be unwilling to accept the verdict of the program, if they cannot follow its reasoning. Ex : Problems such as medical diagnosis.

3.12.10 Problem Classification

Actual problems are examined from the point of view; the tasks here is examining an input and decide which of a set of known classes. **Ex :** Problems such as medical diagnosis, engineering design.

3.12.11 Heuristic Function for : Travelling Salesman Problem

GQ. Explain with suitable example, heuristic function for : travelling salesman problem

UQ Give the initial state, goal test, successor function, and cost function for the travelling salesperson problem (TSP). There is a map involving N cities some of which are connected by roads. The aim is to find the shortest tour that starts from a city, visits all the cities exactly once and comes back to the starting city. **(MU - Q. 5(A), Dec. 16, 6 Marks)**

UQ. Define heuristic function. Give an example heuristics functions for 8-puzzle problem. Find the heuristics value for a particular state of the Blocks World Problem. **(MU - Q. 1(b), May 17, 5 Marks)**

Travelling salesman problem : A salesman has a list of cities, each of which he must visit exactly once. There are direct roads between each pair of cities on the list. Find the route the salesman should follow for the shortest possible round trip that both starts and finishes at any one of the cities.

Why informed (Heuristic) search method?

Informed search method are more problem specific and acquires related knowledge and hence finds solution more efficiently than uniformed method. **Exploration :** Exploration is one of the technique used for information gathering.

There are a few possible issues with this algorithm. One of the most important issues is completeness. It can be a typical extension to generate and test using the knowledge that directs the search. This knowledge is based on heuristic function that detects the closeness of current state to the goal state.

Example : Let us assume that we are solving travelling salesman problem with hill climbing.

1. Begin with the initial state (that is a city salesman has visited).
2. Move in the direction so that the city is not repeated, but additional city is visited (better state).
3. Heuristic can be the number of cities visited.
4. Keep moving in the state that improves the number of cities with legal action.
5. When there is no way to improve heuristic function, stop.

Hill climbing algorithm makes incremental changes in the optimal direction based on heuristic.

There are many variants of hill climbing algorithms :

- To apply the steepest ascent hill climbing to the travelling salesman problem, we need to consider all successor states (possible moves) to choose the best one. This algorithm has higher complexity than the basic hill climbing, since it requires selecting move from a number of possible moves.
- Both basic and steepest ascent hill climbings may not able to reach the goal state, and hence, may fail to find the solution. Generally, it terminates in finding a state that does not have a better state in the vicinity or from where it is not possible to get to a better state in a single move.

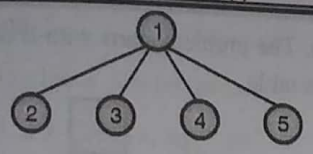
3.12.12 Branch And Bound Technique

GQ. What do you mean by branch and bound technique? Give some examples.

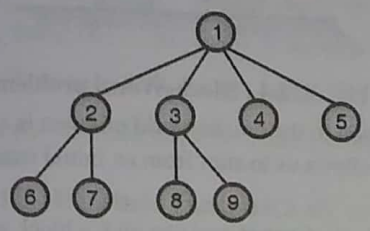
Branch and bound is a state space search method in which all the children of a node are generated before expanding any of its children.

1. Live-node : A node that has not been expanded.
2. It is similar to backtracking technique but uses BFS-like search.
3. Dead-node : A node that has been expanded.

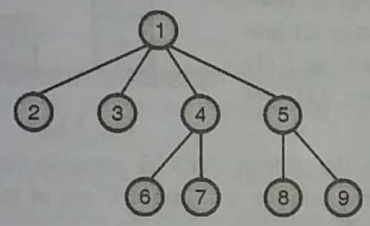
Scanned with CamScanner



Live node : 2, 14.2 and 5



FIFO Branch and Bound Children of E-node are inserted in a queue
Fig. 3.12.2



LIFO Branch and Bound Children of E-node are inserted in a stack
Fig. 3.12.3

3.12.13 Block World Problem

UQ. Define heuristic function. Give an example heuristics function for Blocks world problem.
(MU - Q. 1(a), Dec. 2015, 5 Marks)

UQ. Find the heuristics value for a particular state of the Blocks World Problem.
(MU - Q. 1(b), May 17, 5 Marks)

- The Block World is a 'planning domain' in artificial intelligence. The algorithm is similar to a set of wooden blocks of various shapes and colours but of same dimensions, sitting on table.
- The goal is to build one or more vertical stacks of blocks.
- The blocks in a world are infinite. Though one can see only a chunk of the world at a time. The chunk area is 16 blocks width by 16 block length by 256 blocks height, meaning that one can only see a total volume of 65,536 blocks.
- Artificial intelligence blocks is an intuitive WYSIWYG (What you see is what you get) interface that allows users to create machine learning models.
- The idea of AI-blocks is to have a simple scene with draggable objects that have scripts attached to them.
- The Blocks World is one of the most famous planning domains in Artificial Intelligence.
- A solution to the blocks world problem is a sequence of actions that allows us to start from an initial state and end up to a goal state. Therefore the optimal solution is the solution with the minimum cost (or else minimum actions required to get from the initial state to the goal state).

3.12.14 Actions

- The first member of the tuple represents the block that is about to move and second member of the tuple represents the block (or table) that will have the block (first member of the tuple) stacked on top of it.

  For example : If possible action = [(1, 2), (3, 0)] that means we have two actions :

  (i) The first action will move block 1 on top of block 2, and

  (ii) Second action will move block 3 on the table. (One cannot move a block that has other blocks stacked on top of it).
- After that we have list of all valid actions (list of tuples) and we are ready to move on and apply those actions to a state.
- From an algorithm perspective, blocks-world is an np-hard search and planning problem.
- The task is to bring the system from an initial state into a goal-state.

  [**Remark** : 'NP-complete' is non-deterministic polynomial-time complete. In computational complexity theory, a problem for which the correctness of each solution can be verified quickly and a brute-force search algorithm can actually find a solution by trying all possible solutions].

3.12.15 Motivation

- The simplicity of this toy-world (block world) / ends itself to classical symbolic AI approaches, in which the world is modelled as a set of abstract symbols, which may be reasoned about).
- AI can be researched in theory and with practical applications.
- To develop program on AI system, we invent an easy to solve domain which is called a 'toy-problem'.
- Toy problems were invented with the aim to program

Module 3

Scanned with CamScanner

an AI which can solve it.

- The blocks-world domain is an example of a toy-problem. Its major advantage over more realistic AI applications is that, many algorithms and software programs are available which can handle the situation. This allows to compare different theories against each-other.
- More complicated derivations of the problem consist of cubes in different sizes, shapes and colours. From an algorithm perspective, blocks world is an np-hard search and planning problem.
- The task is to bring the system from an initial state into a goal state.

3.12.16 Symbolic Representation

(A) Four actions

(1) Unstack (A, B) : pick up **clear** block A from block B.

(2) Stack (A, B) : place block A using the arm onto **clear** block B.

(3) pickup (A) : lift clear block A with the empty arm.

(4) putdown (A) : place the held block A onto a free space on the table.

(B) Five predicates

(i) On (A, B) : block A is on block B.

(ii) On table (A) : block A is on table.

(iii) Clear (A) : block A has nothing on it.

(iv) Holding (A) : the arm holds block A.

(v) Arm empty : the arm holds nothing.

Remark : We use the blocks world as an example, because :

(i) it is sufficiently simple and well-defined.

(ii) easily understood.

(iii) yet still provides a good sample environment to study planning.

(iv) problems can be broken into nearly distinct subproblems.

Here, we can show how partial solutions need to be combined to form a realistic complete solution.

3.12.17 Sussman Anomaly

This is a problem in Artificial Intelligence described by Gerald Sussman.

In the problem, three blocks (labelled A, B and C) rest on table. The agent must stack the blocks such that A is a top B, which in turn is a top C. But it may only move one block at a time. The problem starts with B on the table, C a top A and A on table.

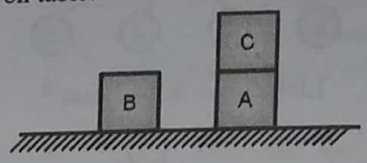


Fig. 3.12.4 : Block-World problem

A solution to the blocks world problem is a sequence of actions that allows us to start from an initial state and end up to a goal.

A mechanical **robot** arm can pick a block and place the cubes either on the table or on top of another block.

A planner typically separate the goal (stack A a top B atop C) into subgoals, such as : (1) get A a top B ; (2) get B a top C

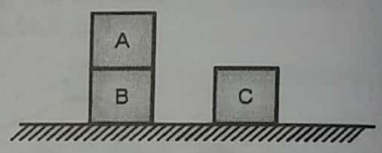


Fig. 3.12.5

Suppose the planner starts by pursuing subgoal 1, the straightforward solution is to move C out of the way, then move A a top B.

While sequence accomplishes subgoal (1), the agent cannot pursue subgoal (2) without undoing subgoal (1), since both A and B must be moved atop C.

Fig. 3.12.6

If instead the planner starts with Goal 2, i.e. to move B. But again the planner cannot pursue Goal 1 without undoing Goal 2.

Thus one of the solutions is

(i) unstack (C, A)

(ii) put down (C)

(iii) stack (B, C)

(iv) stack (A, B) and we get

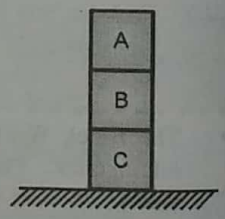


Fig. 3.12.7

Remark

Sussman believed that intelligence requires a list of **exceptions** or **tricks**, and developed a modular planning system for 'debugging' plans.

Most modern planning systems handle this anomaly, but it is still useful for explaining why '**planning** is **non-trivial**'.

Scanned with CamScanner

First order logic

(i) On table (A) – block A is on the table

$\neg \exists x$ On (x, A) or

$\forall x \neg$ On (x, A) [i.e On (x, A) → x is on A].

(ii) Initially : [clear (B) ∧ on (C, A) ∧ on table (A)]

(iii) Solution : [unstack (C, A) ∧ put down (C)]

∧ clear (C) ∧ stack (B, C) ∧ stack (A, B)]

3.12.18 Heuristic Function : H_1

- This function gives better results. It requires less explored nodes and less execution time.
  - (i) First we check how many blocks have different positions.
  - (ii) Compare the node-state and goal-state (basically how many blocks are not where they should be in the current node-state).
  - (iii) The H-function H_1 is admissible because it never overestimates the cost of reaching the goal and is also consistent because for every node N and each successor P of N, the estimated cost of reaching the goal from N is no greater than the step cost of getting to ρ plus the estimated cost of reaching the goal from ρ.
- Thus heuristic function estimates the cost of getting from one place to another (from the current state to the goal state). Also called simply a heuristic.
- A heuristic function algorithm or simply heuristic is a shortcut of solving a problem when there are no exact solutions for it or the time to obtain the solution is too long.

Remark

Automated planning and scheduling problems are usually described in the planning Domain Definition Language (PDDL) notation which is an AI planning language for symbolic manipulation tasks. If something was formulated in PDDL notation, it is called a 'domain'. Therefore the task of stapling blocks is a block world domain, which stays in contrast to other planning problems like the 'dock worker robot' domain and the 'monkey and banana' problem.

3.12.19 Example of 'Block World Problem

(I) Local Heuristic

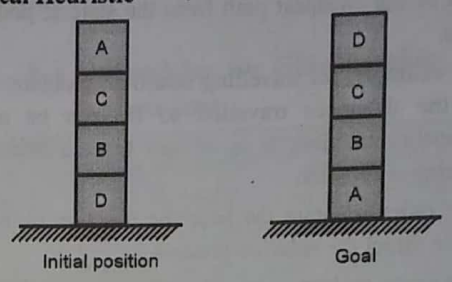


Fig. 3.12.8

(i) pickup (A) ∧ pickup (C) ∧ pickup (B) ∧ pickup (D).

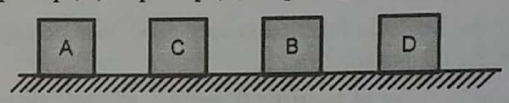


Fig. 3.12.9

(ii) Stack (B, A) ∧ stack (C, B) ∧ stack (D, C)

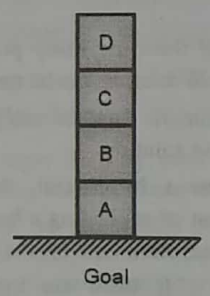


Fig. 3.12.10

(II) Global Heuristic

Here, every block has an incorrect support.

(i) pickup (A) ∧ pickup (C) ∧ pickup (B) ∧ pickup (D).

(ii) stack (B, A)

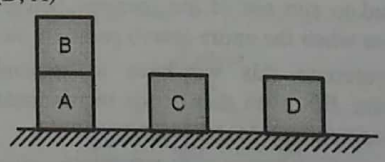


Fig. 3.12.11

(iii) stack (C, B) ∧ stack (D, C)

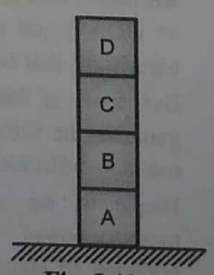


Fig. 3.12.12

Module 3

Scanned with CamScanner

3.12.20 Properties of Heuristic Functions

(1) The heuristic function for a node n is, h (n) = estimated cost of the cheapest path from the state at node n to a goal.

For example, for travelling salesman problem, the sum of the distances travelled so far can be a simple heuristic function. It is of two types : Maximise or minimise function.

(2) The more accurate the heuristic function estimate, the more direct the solution process.

(3) For many problems, the cost of computing the value of such a function outweighs the effort saved in the search process.

(4) If the heuristic function is very good, then no search is required to a solution.

(5) The purpose of heuristic function is to guide the search process in the most profitable direction by suggesting which path to follow first when more than one is available.

(6) **Optimality :** If there are many possible solution to a problem, optimal solution can be found out.

(7) **Accuracy :** Heuristic function can yield approximately good and precise solution.

(8) **Execution time :** In general, there is a trade-off between the cost of evaluating a heuristic function and the saving of execution time for the function.

(9) **Completeness :** If there are many solutions to a problem, we have to find all of them and choose the appropriate one.

3.12.21 Memory Bounded Heurestic Search

- Generally, while carrying out the search out the search, we tend to run out of the memory before time. This happens when the entire search paths are stored.
- To overcome this we have to remember partial solutions. But even then space requirements, memory bounded heuristic searches are used.
- Recursive best first search (RBFS), iterative deepening a* (IDA*) and memory bounded A* (MA*) are the algorithms that fall under this search.
- But there is drawback at RBFS and IDA*. They generate the same states again and again. The cost of memory-utilisation is same for all the above methods.
- Hence, if we are having more memory, we use memory-bound variants MA* or simplified memory bounded A* algorithms, (SMA*).
- SMA* or simplified memory bounded A* is a shortest path algorithm based on the A* algorithm.
- The main advantage of SMA* is that it uses a bounded memory, while the A* algorithm might need exponential memory. All other characteristics of SMA* are inherited from A*

Properties of SMA*

(1) It works with heuristic, just as A*

(2) It is complete if the allowed memory is high enough to store the shallowest optimal solution.

(3) It is optimal if the allowed memory is high enough to store the shallowest optimal solution.

(4) It avoids repeated states as long as the memory bound allows it.

(5) It uses all memory available.

(6) Enlarging the memory bound of the algorithm will only speed up the calculation.

(7) When enough memory is available to contain the entire search tree, then calculation has on optimal speed.

Implementation

- The implantation of SMA* is very similar to that of A*; the only difference is that nodes with the highest f-cost are deleted from the queue when there is not any space left.
- Since these nodes are deleted, SMA* has to remember the f-cost of the best forgotten child of the parent node.
- When it appears that all explored paths are worse than such a forgotten path, the path is regenerated.

SMA* algorithm

- SMA* is a shortest path algorithm that is based on A* algorithm. The difference between SMA*a and A* is that SMA* uses a bounded memory, while the A* algorithm might need exponential memory.
- Like the A*, it expands the most promising branches according to the heuristic.
- SMA* just like A* evaluates nodes by combining g(n)m, the cost to reach the node, and h(n), the cost to get from the node to the goal : f(n) = g(n) + h(n).
- Since g(n) is the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have f(n) = estimated cost of the cheapest solution through n.
- The lower the f value is, the higher priority the node will have. In addition, the node stores the f-value of the best forgotten successor.

Scanned with CamScanner

3.13 LOCAL SEARCH ALGORITHMS

UQ. Write short note on : Local search Algorithms
(MU - Q. 6(b), May 18, Q. 6(c), May 19, 6 Marks)

- **In computer science local search** is a heuristic method for solving computationally hard **optimisation** problems.
- Local search starts from an initial solution and evolves the single solution that is mostly better solution.
- At each solution in this path it evaluates a number of moves on the solution and applies the most suitable move to take the step to the next solution. It continues this process for a high number of iterations until it is terminated.
- Local search uses a single search path and moves facts around to find a good feasible solution. Hence it is natural to implement.
- Local search algorithms are widely applied to numerous hard computational problems, including problems from **artificial intelligence, mathematics, operations research,** engineering and bioinformatics.

Some problems where local search is applied are :

(1) The **vertex cover problem,** in which a solution is a **vertex cover** of a **graph,** and the target is to find a solution with a minimal number of nodes.

(2) The **travelling salesman problem,** in which a solution is a **cycle containing** all nodes of the graph and the target is to minimise the total length of the cycle.

(3) The Hopfield Neural Networks problem for which finding stable configuration in Hop-field network.

3.14 HILL CLIMBING ALGORITHM

GQ. Briefly define hill climbing algorithm.

UQ. Explain Hill Climbing and its Drawback in details.
(MU - Q. 2(b), May 19, 5 Marks, Q. 6(b), May 18, 5 Marks, Q. 2(A), Dec. 17, 10 Marks, Q. 2(b), May 17, 10 Marks)

UQ. Explain Hill-climbing algorithm with an example.
(MU - Q. 1(d), Dec. 17, 5 Marks)

❑ **Definition :** This algorithm, also called **discrete optimization algorithm,** uses a simple heuristic function viz., the amount of distance the node is from the goal. The ordering of choices is a heuristic measure of the remaining distance one has to traverse to reach the goal node.

In fact, there is practically no difference between hill-climbing and depth-first search except that the children of the node that has been expanded are sorted by the remaining distance.

3.14.1 Algorithm for Hill-Climbing Procedure

- **Step 1 :** Put the initial node on a list START.
- **Step 2 :** If (START is empty) or (START = GOAL), then terminate search.
- **Step 3 :** Remove the first node from START. Call this as node a.
- **Step 4 :** If (a = GOAL), then terminate search with success.
- **Step 5 :** Else if node a has successors, generate all of them. Find out how far they are from the goal node. Sort them by the remaining distance from the goal and add them at the beginning of START.
- **Step 6 :** Goto Step 2.

Module 3

3.14.2 Explanation of Hill Climbing Algorithm

Hill-climbing technique is being used in some activity or other in our day-to-day routine. **Some of them are :**

1. While listening to somebody playing flute on the transistor, tone and volume control are adjusted in a way that makes the music melodious.
2. While tuning the carburettor of a scooter, the accelerator is raised to its maximum once and the carburetor is tuned so that the engine keeps on running for a considerably long period of time.
3. An electronics expert, while making the transistor for the first time, tunes the **radio set at mid-afternoon** when the signal is weak for proper reception.

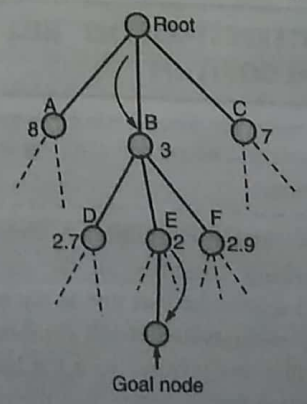


Fig. 3.14.1: Search tree for hill-climbing procedure

Scanned with CamScanner

3.14.3 Simple Hill Climbing Algorithm

GQ. Explain simple hill climbing algorithm with its limitations.

The simplest way to implement hill climbing is as follows :

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until there are no new operators left to be applied in the current state:

(i) Select an operator that has not yet been applied to the current state and apply it to produce a new state.

(ii) Evaluate the new state.

(a) If it is a goal state, then return it and quit.

(b) If it is not a goal state but it is better than the current state, then make it the current state.

(c) If it is not better than the current state, then continue in the loop.

The key difference between this algorithm and generate-and-test is the use of an evaluation function as a way to **inject task-specific knowledge into the control process** makes this method heuristic search methods, and it is that same knowledge that gives these methods their power to solve some otherwise intractable problems.

Notice that in this algorithm, we have asked the relatively vague question, **"Is one state better than another"** For the algorithm to work, a precise definition of better must be provided. In some cases, it means a higher value of the heuristic function. In others, it means a lower value. It does not matter which, as long as a particular hill-climbing program is consistent in its interpretation.

3.15 STEEPEST-ASCENT HILL CLIMBING ALGORITHM

GQ. Explain steepest-ascent hill climbing algorithm with its limitations.

Steepest-ascent hill climbing : A useful variation on simple hill climbing considers all the moves from the current state and selects the best one as the next state. This method is called **steepest-ascent hill climbing** or **gradient search**. Notice that this contrast with the basic method in which the first state that is better than the current state is selected. The algorithm works as follows :

3.15.1 Algorithm : Steepest-ascent Hill Climbing

1. Evaluate the initial state. If it is also a goal state, then return it and quit.

   Otherwise, continue with the initial state as the current state.

   ↓

2. Loop until a solution is found or until a complete iteration produces no change to Current state:

(i) Let SUCC be a state such that any possible successor of the current state will be better than SUCC.

↓

(ii) For each operator that applies to the current state do :

(a) Apply the operator and generate a new state.

(b) Evaluate the new state.

- If it is a goal state, then return it and quit.
- If not, compare it to SUCC. If it is better, then set SUCC to this state.
- If it is not better, leave SUCC alone.

(c) If the SUCC is better than current state, then set current state to SUCC.

To apply steepest-ascent hill climbing to the colored blocks problem, we must consider all **perturbations of the initial state and choose the best**. For this problem, this is difficult since there are so many possible moves. There is a trade-off between the time required to select a move (usually longer for steepest-accent hill climbing) and the number of moves required to get to a solution (usually longer for basic hill climbing) that must be considered when deciding which method will work better for a particular problem.

3.15.2 Limitations of Steepest-ascent Hill Climbing

UQ. State limitations of steepest-ascent hill climbing

(MU - Q. 2(b), May 19, 5 Marks, Q. 6(b), May 18, 5 Marks, Q. 2(A), Dec. 17, 10 Marks, Q. 2(b), May 17, 10 Marks)

UQ. What are the problems/frustrations that occur in hill climbing technique? Illustrate with an example.

(MU - Q. 4(b), Dec. 15, 6 Marks, Q. 1(d), May 17, 5 Marks)

Scanned with CamScanner

Steepest-ascent hill climbing may fail to find a solution. Algorithm may terminate not by finding a goal state but by getting to a state from which no better states can be generated. This will happen if the program has reached either a local maximum, a plateau, or a ridge.

1. **A local maximum** is a state that is better than all its neighbours but is not better than some other states farther away. At a local maximum, all moves appear to make things worse. Local maxima are particularly frustrating because they often occur almost within sight of a solution. In this case, they are called **foothills**.
2. **A plateau** is a flat area of the search space in which a whole set of neighboring states have the same value. On a plateau, it is not possible to determine the best direction in which to move by making local comparisons.
3. **A ridge** is a special kind of local maximum. It is an area of the search space that is higher than surrounding areas and that itself has a slope (which one would like to climb). But the orientation of the high region, compared to the set of available moves and the directions in which they move, makes it impossible to traverse a ridge by single moves.

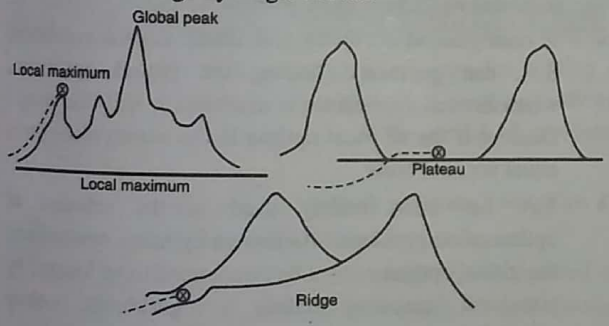


Fig. 3.15.1 : Problems associated with hill climbing : local maxima, plateau and ridge

3.15.3 Ways of dealing with Local Maxima Plateau and Ridge Problems

GQ. What are the ways of dealing with local maxima plateau and ridge problems which arise in hill climbing?

Problems associated with hill climbing are :

(i) Local maxima, (ii) plateaus, (iii) ridges.

There are some ways of dealing with these problems, although these methods are by no means guaranteed :

- Some problem spaces are great for hill climbing and others are terrible.
- **Random restart :** Keep restarting the search from random locations until a goal is found.
- **Problem reformulation :** Reformulate the search space to eliminate these problematic features.
- Backtrack to some earlier node and try going in a different direction. This is particularly reasonable if at that node there was another direction that looked promising or almost as promising as the one that was chosen earlier. To implement this strategy, maintain a list of paths almost taken and go back to one of them if the path that was taken leads to a dead end. This is a fairly good way of dealing with local maxima.
- Make a big jump in some direction to try to get to a new section of the search space. This is a particularly good way of dealing with plateaus. If the only rules available describe single small steps, apply them several times in the same direction.
- Apply two or more rules before doing the test. This corresponds to moving in several directions at once. This is a particularly good strategy for dealing with ridges.
- Even with these first-aid measures, hill climbing is not always very effective. It is particularly unsuited to problems where the value of the heuristic function drops off suddenly as we move away from a solution. This is often the case whenever any sort of threshold effect is present.
- Hill climbing is a local method, by which we mean that it decides what to do next by looking only at the "immediate" consequences of its choice rather than by exhaustively exploring all the consequences. It shares with other local methods, the advantage of being less combinatorically explosive than comparable global methods. But it also shares with other local methods a lack of a guarantee that it will be effective. Although it is true that the hill-climbing procedure itself looks only one move ahead and not any farther, that examination may in fact exploit an arbitrary amount of global information if that information is encoded in the heuristic function.

Module 3

3.16 SIMULATED ANNEALING (SA)

GQ. Define the term simulated annealing. Explain simulated Annealing with suitable example.

MU - Q. 2(A), Dec. 17, 10 Marks, Q. 2(b), May 16, 10 Marks

Scanned with CamScanner

- Simulated annealing is a variation of hill climbing in which, at the beginning of the process, some **downhill moves** may be made. The idea is to do enough exploration of the whole space early on so that the final solution is relatively intensive to the starting state. This should lower the chances of getting caught at a local maximum, a plateau or a ridge.
- Simulated annealing as a computational process is patterned after the **physical process of annealing**, in which physical substances such as metals are melted (i.e., raised to high energy levels) and then gradually cooled until some solid state is reached. The goal of this process is to produce a minimal-energy final state. Thus this process is one of valley descending in which the objective function is the energy level.
- Simulated annealing is an **effective and general form of optimisation**. Annealing refers to an analogy with thermodynamics, specifically with the way that metals cool and anneal. Simulated annealing uses the objective function of an optimisation problem instead of the energy of a material.

Uses of SA

SA is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a met heuristic to approximate global optimisation in a large search space for an optimisation problem.

3.16.1 Types and Use of Simulated Annealing

- Simulated annealing algorithms are essentially random search methods in which the new solutions, generated according to a sequence of **probability distribution** (for example, the Boltzmann distribution) or a random procedure (e.g. a hit-and-run algorithm) may be accepted even if they do not lead to an improvement in the Annealing.
- Simulated Annealing is a process where the temperature is reduced slowly; starting from a random search at high temperature and then eventually it became purely greedy descent as it approaches zero temperature. S.A. maintains a current assignment of values of variables.
- Simulated Annealing is an effective and general form of optimisation. It is useful in finding global optima in the presence of large number of local optima. It is analogous to temperature in an annealing system. At higher value of T, uphill moves are more likely to occur.
- S.A. will accept an increase in the **cost function** with some probability based on annealing algorithm. Simulated Annealing is based on an analogy to a physical system which is first melted and then cooled or annealed into a low energy state.

3.16.2 Simulated Annealing in Machine Learning

- S.A. is a technique that is used to find the best solution for either a **global minimum or maximum** without having a check for every single possible solution, that exists. This is super helpful when addressing massive optimisation problems like the one previously stated.
- S.A. is a **stochastic global search optimisation algorithm**. The algorithm is inspired by annealing in metallurgy where metal is heated to a high temperature quickly, then cooled slowly and that increases its strength and makes it easier to work with S.A. executes the search in the same way.
- Annealing is a heat treatment process that changes the 'physical and sometimes' also the chemical properties of a material to increase ductility and reduce the hardness to make it more workable.
- It configurated correctly and under certain condition, S.A. can guarantee finding the **global optimum**, whereas such a guarantee is available to Hill Climbing / Descent if the all local optima in the search place have equal scores / costs.
- S.A. has been widely used in the solution of optimisation problems. As known by many researchers, the global optima cannot be guaranteed to be located by simulated annealing unless a logarithmic cooling schedule is used.

3.17 PARAMETER FOR S.A.

1. Choice of parameters depends on the **expected variation in the performance** measure over the search space.
2. A good rule of thumb is that the initial temperature should be set to accept roughly **98°** of the moves and that the final temperature should be low enough that the solution does not improve much, if at all. To improve simulated annealing, we have to do the following:
   (i) Improve the accuracy.
   (ii) Alter the parameters of the algorithm.
   (iii) You could run your own meta-optimisation on the parameters of your problem.

Scanned with CamScanner

3. Simulated annealing is gradient based. It is an **extension of gradient descent**, and in the degenerate case (zero temperature) they are the same: It generates **random neighbouring states**, and if the fitness of that state is better than the current one then it jumps there. That is, it seeks a local minimum.
4. In annealing, atoms migrate in the **crystal lattice** and the number of dislocations decreases, leading to a change in ductility and hardness. With the knowledge of composition and phase diagram, heat treatment can be used to adjust from harder and more brittle to softer and more ductile.

Examples of SA

- Travelling salesman problem.
- Task allocation
- Graph colouring and partitioning
- Scheduling problems
- Non-linear function optimization

3.18 GENETIC ALGORITHM

UQ. Write a short note on genetic algorithm. **(MU - Q. 5(a), Dec. 15, Q. 6, May 17, 10 Marks)**

- An algorithm is a progression of steps for solving a problem. A genetic algorithm is a problem-solving technique that uses genetics as its model of problem-solving.
- It is a search method to find estimated solutions to optimization and search issues.
- The genetic algorithm is a method that is applied to solve both constrained and unconstrained optimisation problems.
- It is based on natural selection.
- The genetic algorithm modifies a population of individual solutions.
- Genetic algorithm is meta heuristic in computer science and operations research.
- It is inspired by natural selection. And it belongs to the larger class of evolutionary algorithms.
- Genetic algorithms are used to generate high quality solutions to optimisation and search problems.
- It depends on biologically inspired operators such as crossover, mutation and selection. Some examples of Genetic Algorithm (GA) applications include optimising decision trees for better performance, hyper parameter optimisation etc.
- One can easily distinguish between a traditional and a genetic algorithm.

3.18.1 Comparison between Traditional and Genetic Algorithm

Traditional Algorithm	Genetic Algorithm
It selects the next point in the series by a deterministic computation.	It selects the next population by computation, which utilizes random number generators.
It creates an individual point at each iteration. The sequence of points approaches an optimal solution.	It creates a population of points atevery iteration. The best point in the population approaches an optimal solution.
Advancement in each iteration is problem specific.	Concurrence in each iteration is a problem independent

Module 3

3.18.2 Basic Terminology

UQ. Define the terms chromosome, fitness function, crossover and mutation as used in Genetic algorithms. **(MU - Q. 5(a), May 18, 10 Marks)**

UQ. Explain how genetic algorithms work. Define the terms chromosome, fitness function, crossover and mutation as used in Genetic algorithms. **(MU - Q. 6(c), Dec. 18, 5 Marks)**

We introduce ourselves to basic terminology required to understand GAs.

(i) **Population :** The population of GA is same as the population for human beings except that instead of human beings, we have **candidate solutions** that are similar to human beings.

(ii) **Chromosomes :** Chromosome is regarded as solution to the given problem.

(iii) **Gene :** A gene is one element position (or part) of a chromosome.

(iv) **Allele :** It is the value taken by a gene for a particular chromosome.

Scanned with CamScanner

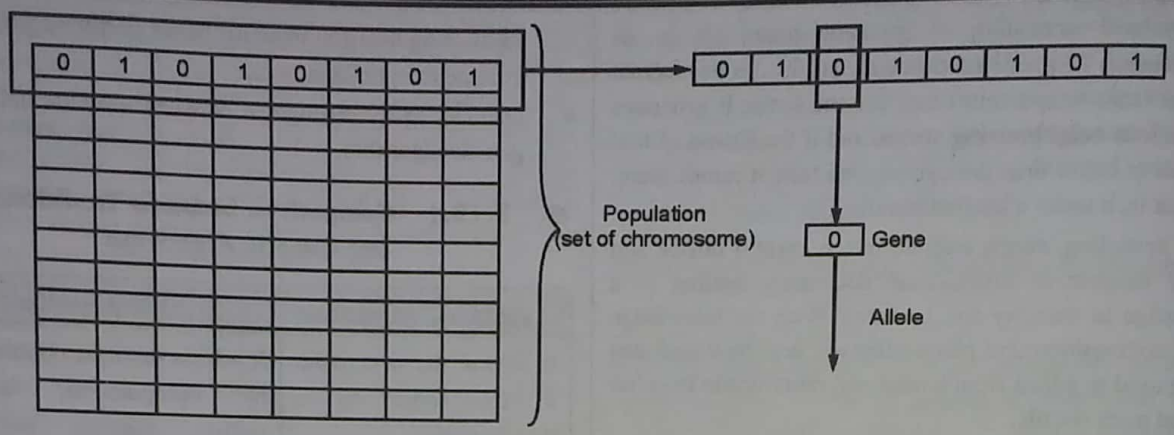


Fig. 3.18.1

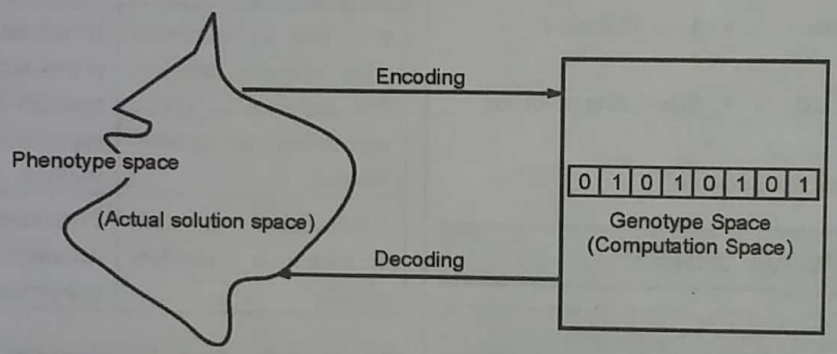


Fig. 3.18.2

(v) **Genotype :** Genotype is the population in the computation space. Using computing system, solutions represented in the computation space can be easily understood and easily manipulated.

(vi) **Phenotype :** Phenotype is the population in which solution are represented in a way they are represented in actual world situations.

(vii) **Decoding and Encoding :** Decoding is a process of transforming a solution from the genotype to the phenotype space.

Encoding is process of transforming from the phenotype to genotype space.

Decoding has to be fast as it is carried out repeatedly in a GA during the fitness value calculation.

Remark : For **simple problems** the phenotype and genotype spaces are **same**.

Fitness Function : A fitness function is a function which takes the solution as input and produces the suitability of the solution as the output.

In some cases the fitness function and objective functions are same. Depending on the problem, they may be different.

Fitness function is a function, we want to optimise.

3.18.3 Optimisation Problems

- In a genetic algorithm, a population of candidate solutions to an optimisation problem is evolved, towards better solutions.
- The evolution generally begins with a population of randomly generated individuals. It is an **iterative process,** and the population in each iteration called a generation.
- The fitness of every individual in the population is evaluated. The fitness of every individual is the valve of the **objective function** in the optimisation problem, to be solved.

Scanned with CamScanner

The more fit individuals are stochastically selected from the current populations to form a new generation. And it is used in the next iteration of the algorithm. The algorithm terminates if (i) a maximum number of generations has been produced or (ii) a satisfactory fitness level is reached for the population.

Requirements of genetic algorithm are :

(i) a genetic representation of the solution domain,

(ii) a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is an array of **bits.**

The main property that makes these genetic representation possible is that their parts are easily aligned due to their fixed size, which causes simple crossover operations.

- Once the genetic representation and the fitness function are defined, a genetic Algorithm initialises a population of solutions and then to improve it through repetitive application of the **mutation crossover, inversion** and **selection operators.**

3.18.4 Initialisation

- The population size depends on the nature of the problem. And it may have several thousands of solutions. Generally, initial population is generated randomly, and it may have all the possible solutions. (This is termed as **search space**).
- The solutions may be 'seeded' in areas where **optimal** solution may be found.

3.18.5 Selection

- During each successive generation, a portion of the existing population is selected to obtain a new generation.
- Individual solutions are selected through a fitness-based process. Certain selection methods rate the fitness of each solution and select the best solution.
- The fitness function is defined over genetic representation. It measures the quality of the represented solution. The fitness function is always problem dependent.
- In some problems, it is not possible to define the fitness expression, in such cases, simulation may be used to determine the fitness function valve of a phenotype (for example, computational fluid dynamics is used to determine the air resistance of a vehicle whose shape is encoded as the **interactive genetic algorithms).**

3.18.6 Genetic Operators

UQ. Explain how genetic algorithms work.
(MU - Q. 6(c), Dec. 18, 5 Marks)

Genetic operators alter the genetic composition of the offspring.

These include crossover, mutation, selection, etc.

Basic structure : The basic structure of GA is

(i) We begin with an initial population, it may be generated at random or seeded by other heuristics.

(ii) Select parents from this population for mafing,

(iii) Apply crossover and mutation operators on the parents to generate new off-springs.

(iv) Finally these off-springs replace the existing individuals in the population and

(v) Process gets repeated.

- In this way genetic algorithms actually copy the human evolution to some extent.

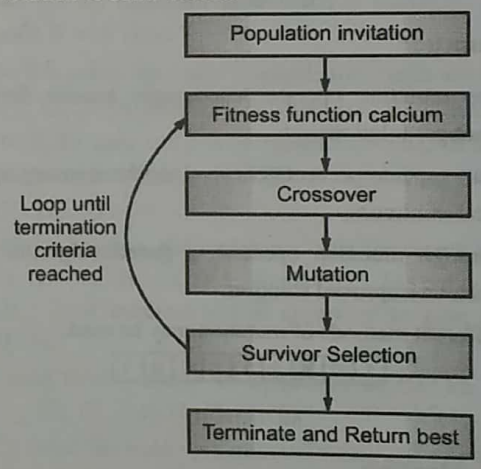


Fig. 3.18.3 : Basic Structure of Genetic Operator

- Genetic variation is a necessity for the process of evolution.
- Genetic operators used in genetic algorithms are analogous to those in the natural world: selection, crossover (also called recombination) and mutation.

(1) Selection

(i) Selection operators give preference to better solutions (chromosomes).

(ii) Allowing them to pass on their 'genes' to the next generation of the algorithm.

Module 3

Scanned with CamScanner

(iii) The best solutions are determined using some form of **objective function** (also known as a 'fitness function' in genetic algorithms), before being passed to the crossover operator.

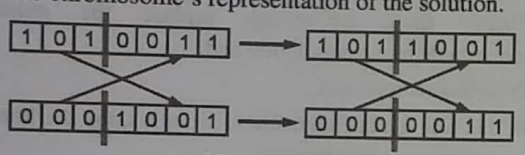
(2) Crossover

(i) Crossover is the process of taking more than one **parent solutions** (chromosomes) and

(ii) Producing a child solution from them. By recombining portions of good solutions.

(iii) The genetic algorithm may create a better solution.

(iv) The crossover method is often chosen to closely match the chromosome's representation of the solution.



CROSSOVER

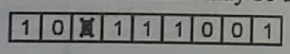
(3) Mutation

(i) The mutation operator encourages genetic diversity amongst solutions,

(ii) It attempts to prevent genetic algorithm converging to a local minimum,

(iii) Through mutation operator, a genetic algorithm can reach an improved solution.

(iv) Different methods of mutation may be used.



MUTATION

(4) Termination

- The above process is repeated until a termination condition is reached.
- In general, terminating conditions are :
  1. A solution is found that satisfies minimum criteria.
  2. Fixed number of generation reached.
  3. computation time/allocated money reached.
  4. The solution's fitness is totally reached or any further successive iteration no longer produce better results.
  5. Manual inspection and combination of the above conditions.

3.18.7 Advantages of Genetic Algorithm

1. The genetic algorithm concept is easy to understand.
2. The genetic algorithm supports multi-objective optimization.
3. A genetic algorithm is suitable for noisy environments.
4. The genetic algorithm is robust with respect to local minima/maxima.
5. The genetic algorithm utilizes probabilistic transition rules.
6. The genetic algorithm utilizes payoff (objective function) information, not derivatives.
7. The genetic algorithm works well on mixed discrete functions.
8. The genetic algorithm concept is modular, separate from application.
9. In the genetic algorithm concept, answer gets better with time.
10. The genetic algorithm concept is inherently parallel; easily distributed.
11. The genetic algorithms work on the Chromosome, which is an encoded version of potential solutions' parameters, rather the parameters themselves.
12. The genetic algorithms use fitness score, which is obtained from objective functions, without other derivative or auxiliary information

3.18.8 Limitations of Genetic Algorithm

1. The Genetic Algorithms might be costly in computational terms since the evaluation of each individual requires the training of a model.
2. These algorithms can take a long time to converge since they have a stochastic nature.
3. The language used to determine candidate solutions must be robust. It must be able to endure random changes such that fatal errors don't mistake.
4. A wrong decision of the fitness function may lead to significant consequences
5. Small population size will not give enough solution to the genetic algorithm to produce precise results.
6. A frequency of genetic change or poor selection scheme will result in disrupting the beneficial schema.
7. Though Genetic algorithms can find exact solutions to analytical sorts of problems, traditional analytic techniques can find the same solutions in a short time with few computational data.

Scanned with CamScanner

3.18.9 Applications of Genetic Algorithm

1. Genetic Algorithm in Robotics

- As we know Robotics is one of the most discussed fields in the computer industry today.
- It is used in various industries in order to increase profitability efficiency and accuracy.
- As the environment in which robots work with the time change, it becomes very tough for developers to figure out each possible behaviour of the robot in order to cope with the changes.
- This is the place where the Genetic Algorithm places a vital role.
- Hence a suitable method is required, which will lead the robot to its objective and will make it adaptive to new situations as it encounters them.
- Genetic Algorithms are adaptive search techniques that are used to learn high-performance knowledge structures.

2. Genetic Algorithm in Financial Planning

- Genetic algorithms are extremely efficient for financial modelling applications.
- As they are driven by adjustments that can be used to improve the efficiency of predictions and return over the benchmark set.
- In addition, these methods are robust, permitting a greater range of extensions and constraints, which may not be accommodated in traditional techniques.

3.19 BEST FIRST SEARCH

GQ. Explain best first search with algorithm.
OR When best first search algorithm, will be applicable ? With a suitable algorithm and example explain the best first search.

❑ **Definition :** This search procedure is an evaluation function variant of best the first search. The heuristic function used here called an evaluation function is an indicator how far the node is from the goal node. Goal nodes have an evaluation function value of zero.

Best-first search is explained using a search graph given in figure :

1. First, the start node S is expanded. It has three children A, B and C with values 3, 6 and 5 respectively. These values approximately indicate how far they are from the goal node.
2. The child with minimum value namely A is chosen. The children of A are generated. They are D and E with values 9 and 8.
3. The search process has now four nodes to search for. i.e., node D with value 9, node E with value 8, node B with value 6 and node C with value 5. Of them, node C has got the minimal value which is expanded to give node H with value 7.
4. At this point, the nodes available for search are (D : 9), (E : 8), (B : 6) and (H : 7) where (α : β) indicates that (α) is the node and β is its evaluation value. Of these, B is minimal and hence B is expanded to give (F : 12), (G : 14).
5. At this juncture, the nodes available for search are (D : 9), (E : 8), (H : 7), (F : 12) and (G : 14) out of which (H : 7) is minimal and is expanded to give (I : 5), (J : 6).
6. Nodes now available for expansion are (D : 9), (E : 8), (F : 12), (G : 14), (I : 5), (J : 6).

   Of these, the node with minimal value is (I : 5) which is expanded to give the goal node.

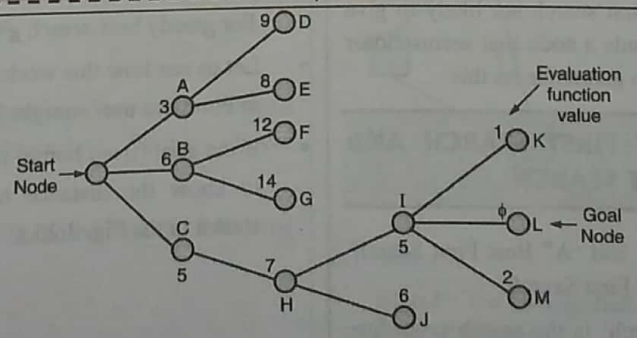


Fig. 3.19.1 : A sample tree for best-first search

Module 3

Scanned with CamScanner

3.19.1 Steps of the Search Process in BFS

The entire steps of the search process are given in Table 3.19.1

Table 3.19.1 : Search process of best-first search

Step	Node being expanded	Children	Available nodes	Node chosen
1.	S	(A : 3), (B : 6), (C : 5)	(A : 3), (B : 6), (C : 5)	(A : 3)
2.	A	(D : 9), (E : 8)	(B : 6), (C : 5), (D : 9), (E : 8)	(C : 5)
3.	C	(H : 7)	(B : 6), (D : 9), (E : 8), (H : 7)	(B : 6)
4.	B	(F : 12), (G : 14)	(D : 9), (E : 8), (H : 7), (F : 12), (G : 14)	(H : 7)
5.	H	(I : 5), (J : 6)	(D : 9), (E : 8), (F : 12), (G : 14), (I : 5), (J : 6)	(I : 5)
6.	I	(K : 1), (L : 0) (M : 2)	(D : 9), (E : 8), (F : 12), (G : 14), (J : 6), (K : 1), (L : 0), (M : 2)	Search stops as goal is reached

As you can see, best-first search **"jumps all around"** in the search graph to identify the node with minimal evaluation function value. There is only a minor variation between hill-climbing and best-first search. In the former, we sorted the children of the first node being generated. Here, we have to sort the entire list to identify the next node to be expanded.

3.19.2 Algorithm for best-first Search

- **Step 1 :** Put the initial node on a list START.
- **Step 2 :** If (START is empty) or (START = GOAL), then terminate search.
- **Step 3 :** Remove the first node from START. Call this as node *a*.
- **Step 4 :** If (*a* = GOAL), then terminate search with success.
- **Step 5 :** Else if node *a* has successors, generate all of them. Find out how far they are from the goal node. Sort them by the remaining distance from the goal.
- **Step 6 :** Name this list as START 1.
- **Step 7 :** Replace START with START 1.
- **Step 8 :** Goto Step 2.

The paths found by best-first search are likely to give solutions faster because it expands a node that seems closer to the goal. However, there is no guarantee on this.

3.20 GREEDY BEST FIRST SEARCH AND A* BEST FIRST SEARCH

- 'Greedy Best First Search' and 'A* Best First Search' are the two variants of Best First Search.
- The 'Greedy best first search' is the search procedure and is evaluation function variant of breadth first search. The heuristic function used here is called an evaluation function and is an indicator how far the node is from the goal node. Goal nodes have an evaluation function of valve zero.
- Speaking about the greedy and best first search, it is the 'best' first search that is named as greedy best first search only.
- But there is a family of best first search algorithms, which have different evaluation functions. The best first search makes use of the function g (n) along with h (n).
- The evaluation function f (n) represents the total cost. Here f (n) = g (n) + h (n), where g (n) is the cost so far to reach n, while h (n) is the estimated cost from n to the goal state.

  For greedy best-search g (n) = 0, and f (n) = h (n)
- Let us see how this works for routine-finding problems in Pune, we use 'straight-line distance' heuristic.
- If the goal (from home) to Roopali restaurant, we need to know the distance to Roopali Hotel, which are shown in the Fig. 3.20.1.

Scanned with CamScanner

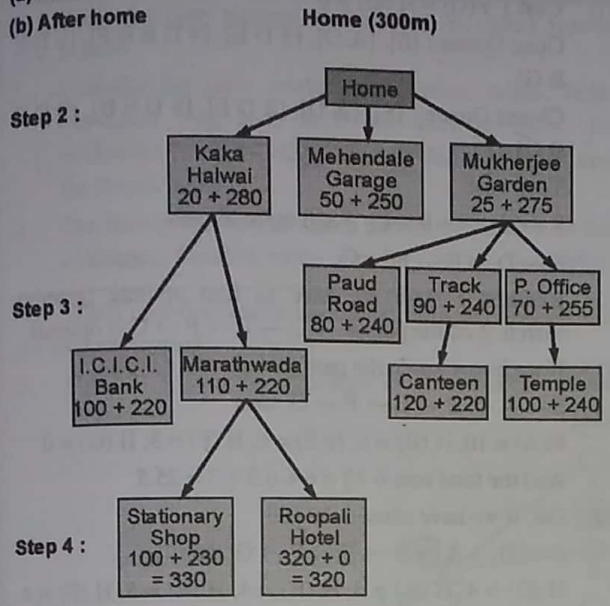


Fig. 3.20.1

- The straight line distance heuristic, we denote by h_{SLD}. Now, h_{SLD} cannot be calculated from the problem description itself. It takes a little experience to know that h_{SLD} is approximately equal to actual road distance, and hence it is a useful heuristic.

  Kaka Halwai – 20 meters

  Mehendale Garage – 40 meters

  Mukherjee Garden – 25 meters

  I.C.I.C.I Bank – 90 meters

  B.M.C.C. College – 120 meters

  Marathwada College – 140 meters

  Stationary shop – 180 meters

  Roopali Hotel – 300 meters

  School – 100 meters

  Walking track – 110 meters

  Post Office – 130 meters

Stages in a greedy best-first tree search for Roopali Hotel with the straight line distance heuristic h_{SLD}. Nodes are labelled with distances.

For the algorithm, a node should contain the following information :

1. Description of the state.
2. Its parent link
3. List of nodes that are generated from it.

A graph containing these information is called 'OR' graph.

3.20.1 Greedy Best First Search Algorithm

Now we discuss the algorithm :

1. Initial state-node.
2. Proceed (till goal is found).
   (i) Select the best node. Add to the list.
   (ii) Mark the successors.
   (iii) For every successor,
      (a) Since priority queue is used, select the best node.
      (b) If there is already successor, change the parent if this new path is better than the earlier one. And update the costs to this node.

   If the successor is not generated then evaluate the cost on the basis of heuristic.
3. Repeat the process.

Properties of Greedy Best-First Search

(i) Greedy Best-First Search is not optimal.

(ii) It is incomplete, even if the given state consists of finite number of nodes.

3.20.2 Solved Examples

Ex. 3.20.1 : Consider the following graph shown in the Fig. Ex. 3.20.1 is the starting state, G is the goal state. Run the greedy search algorithm and write the order of the node in which it is explored. The straight line distance heuristic estimate for the nodes are also given below :

h(s) = 10.5, h (A) = 10, h(B) = 6, h(c) = 4,

h(D) = 8, h(E) = 6.5, h(F) = 3 and h(G) = 0

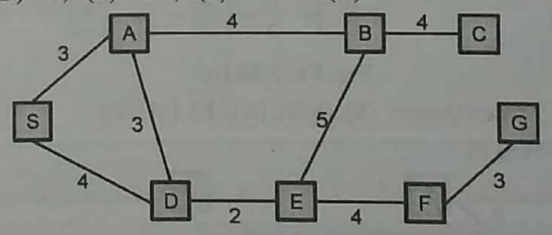


Fig. Ex. 3.20.1

Soln. :

Step I : We have to find greedy search. We choose the order of the node

Consider the tree

Module 3

Scanned with CamScanner

(i) **Cost :** F (A) = H (A) = 10

F(D) = H (D) = 8

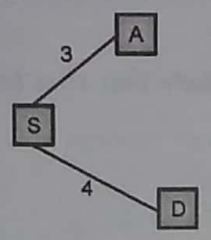


Fig. Ex. 3.20.1(a)

(ii) W closed [S] and [S D] on closed queue and [A D] open Queen.

▶ **Step II :** We consider the tree A S D E

(i) Cost : F (E) = H (E) = 6.5 and closed Queen [S], [S D], [S D E] and open queue : [S], [A D], [S, D, E]

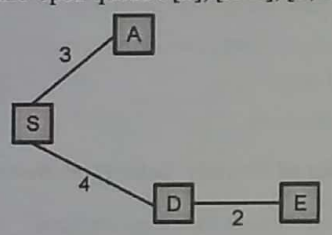


Fig. Ex. 3.20.1(b)

▶ **Step III :**

(i) **Cost :** F (B) = H (B) = 6

F(B) = H (F) = 3

Open Queue : [S], [A D], [S D E], [S D E B F]

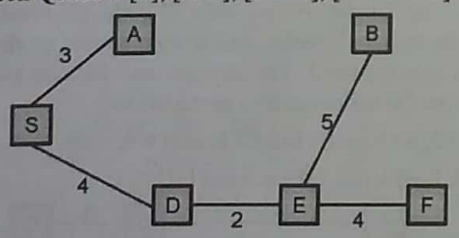


Fig. Ex. 3.20.1(c)

Closed Queue : [S], [S D], [S D E], [SDEB]

▶ **Step IV :**

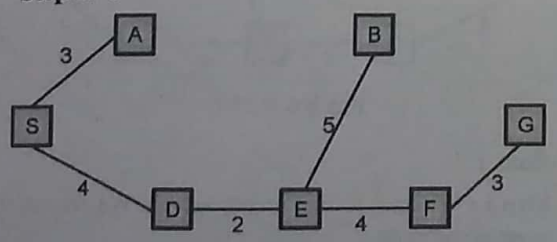


Fig. Ex. 3.20.1(d)

(i) **Cost :** F(G) = H (G) = 0

Open Queue : [S], [A D], [S D E], [S D E B F], [S D E B G]

Closed Queue : [S], [A D], [S D E], [S D E B], [S D E B G]

Total cost

8 + 6.5 + 3 + 0 = 17. 5 and optimal path

$S \to D \to E \to F \to G$

Remark : Here we have to find optimal (greedy) search. And the path $S \to D \to E \to F \to G$ is optimal

(1) If we have chosen the path

$S \to A \to B \to E \to F \to G$, then

H(A) = 10, H (B) = 6, H(E) = 6, H (F) = 3; H (G) = 0

And the total cost = 10 + 6 + 6.5 + 3 = 25.5

(2) OR, if we have chosen the path

$S \to D \to A \to B \to E \to F \to G$; then

H (D) = 4, H (A) = 3, H (B) = 4, H (E) = 5, H (F) = 4 and H (G) = 0

Then the total cost = 4 + 3 + 4 + 5 + 4 = 21

(3) For any other path, the total cost would have been greater than 17. 5

$\therefore S \to D \to E \to F \to G$ is optimal path and optimal cost is 17.5

3.21 A* AND AO* SEARCH

UQ. Explain A* Algorithm with example.
(MU - Q. 2(b), May 18, 10 Marks. (MU - Q. 2(b), May 19, 5 Marks, Q. 6(c), May 18, 5 Marks, Q. 6(b), May 17, 5 Marks)

1. **A* Algorithm :** In best-first search, we brought in a heuristic value called **evaluation function value**. It is a value that estimates how far a particular node is from the goal. Apart from the evaluation function value, one can also bring in cost functions. Cost functions indicate how much resource like time, energy, money etc. have been spent in reaching a particular node from the start. While evaluation function values deal with the future, cost function values deal with the past. Since cost function values are really expended, they are more concrete than evaluation function values.
2. If it is possible for one to obtain the evaluation function values and the cost function values, then A* algorithm can be used. The basic principle is that sum the cost and evaluation function value for a state to get its "**goodness**" worth and use this as a **yardstick** instead of the evaluation function value in best-first search. The sum of the evaluation function value and the cost

Scanned with CamScanner

along the path leading to that state is called fitness number.

3. Consider the same evaluation function values. Now associated with each node are three numbers, the evaluation function value, the cost function value and the fitness number.
4. The fitness number, as stated earlier, is the total of the evaluation function value and the cost-function value.

For example, consider node K, the fitness number is 20, which is obtained as follows :

(Evaluation function of K) + (Cost function involved from start node S to node K)

= 1 + (Cost function from S to C + Cost function from C to H + Cost function from H to I + Cost function from I to K)

= 1 + 6 + 5 + 7 + 1 = 20.

While best-first search uses the evaluation function value only for expanding the best node, A* uses the fitness number for its computation.

5. Fig. 3.21.1 gives the algorithm for A* algorithm method.

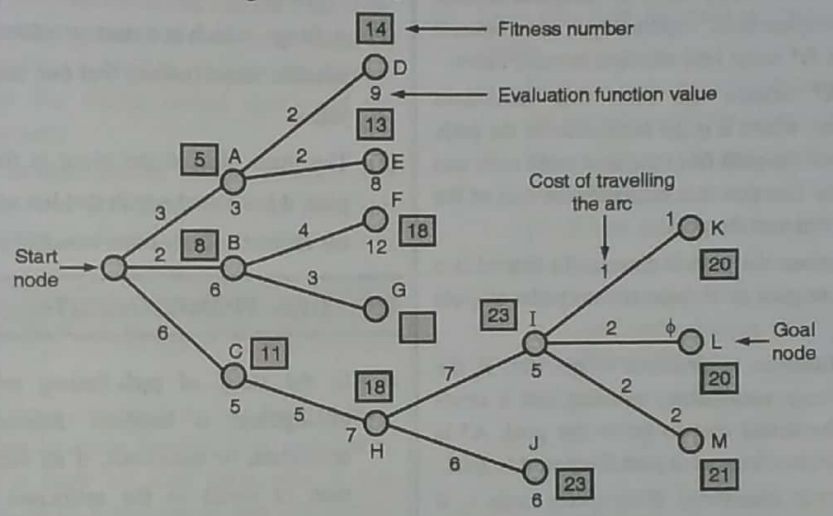


Fig. 3.21.1 : Sample tree with fitness number used for A* search

- **Step 1 :** Put the initial node on a list START.
- **Step 2 :** If (START is empty) or (START = GOAL), then terminate search.
- **Step 3 :** Remove the first node from START. Call this as node a.
- **Step 4 :** If (a = GOAL), then terminate search with success.
- **Step 5 :** Else if node a has successors, generate all of them. Estimate the fitness number of the successors by totaling the evaluation function value and the cost-function value. Sort the list by fitness number.
- **Step 6 :** Name this new list as START 1.
- **Step 7 :** Replace START with START 1.
- **Step 8 :** Go to Step 2.

3.22 ADMISSIBILITY OF A*

UQ. Prove the admissibility of A*. (MU - Q. 5(B), Dec. 16, 6 Marks)

- Admissible functions are functions that should satisfy the essential boundary conditions of the problem.
- An algorithm A is admissible if it is guaranteed to return an optimal solution when one exists.
- A* algorithm is admissible if, it uses an admissible heuristic', and h(goal) = 0 (h(n) is smaller than h*(n)), then A* is guaranteed to find an optimal solution, i.e., f(n) is non-decreasing along any path.

Scanned with CamScanner

[**Theorem :** If h(n) is consistent, f along any path is non-decreasing]

- If is heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least cost path from start to goal.
- Typical implementations of A* use a priority queue to perform the repeated selection of minimum (estimated) cost nodes to expand.
- A* is a graph traversal and path search algorithm, which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency. Thus A* is the best solution in many cases.
- Specifically, A* selects the path that minimizes $f(n) = g(n) + h(n)$, where n is the next node on the path, g(n) is the cost of the path from the start node to n, and h(n) is a heuristic function that estimates the cost of the cheapest path from n to the goal.
- A* terminates when the path it chooses to extend is a path from start to goal or if there are no paths eligible to be extended.
- The heuristic function is problem – specific. If the heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal.
- A heuristic h(n) is admissible if for every node n, if $h(n) < h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.
- **Theorem :** A* is optimal if two conditions are met.
  (1) The heuristic is admissible, as it will never overestimate the cost.
  (2) The heuristic is monotonic, that is, $h(n_i) < h(n_{i+1})$, then real-cost (n_i) < real – cost (n_{i+1}).

Proof : Let G be an optimal goal and let n be an unexpanded node in the fringe such that n is on a shortest path.

Let some suboptimal goal G_1 has been generated and is in the fringe.

Now, we have

$$f(G_1) = g(G_1) + h(G_1)$$

$$\therefore \quad f(G_1) = g(G_1)$$

[$\because h(G_1) = 0$, $\because G_1$ is in the fringe and is an optimal goal]

Again $f(G) = g(G)$

$$\because h(G) = 0$$

Now, $f(G_1) > f(G)$ [$\because$ heuristic is monotonic] ...(i)

$\therefore \quad h(n) \leq h^*(n)$ [$\because$ h is admissible]

$$\therefore g(n) + h(n) \leq g(n) + h^*(n)$$

$\therefore f(n) \leq f(G) < f(G_1)$ ...From equation (ii)

$$\therefore f(G_1) > f(n),$$

$\therefore$ A* will never select G_1 for expansion.

Remarks

(1) A fringe, which is a data structure used to store all the possible states (nodes) that one can go from the current states.

(2) The main idea of the proof is that when A* finds a path, it has found a path that has an estimate lower than the estimate of any other possible path.

3.23 MONOTONICITY

- In the study of path-finding problems in artificial intelligence, a heuristic function is said to be consistent, or monotone, if its estimate is always less than or equal to the estimated distance from any neighboring vertex to the goal, plus the cost of reaching that neighbor.
- Thus, for every node n and each successor p of n, the estimated cost of reaching the goal from n is not greater than the step cost of getting to P plus the estimated cost of reaching the goal from P. That is $h(n) \leq C(n, p) + h(p)$ and $h(G) = 0$.

  Where :

  h is the consistent heuristic function :

  n is any node in the graph,

  p is any descendant of n

  G is any goal node

  C(n, p) is the cost of reaching node p to n.

Hence, every node i will give an estimate that accounting for the cost to reach the next node, is always lesser than the estimate at node (i + 1).

Thus monotonicity is established.

Scanned with CamScanner

3.23.1 Monotonicity in AI

Monotonicity in AI can refer to monotonic classification or monotonic reasoning. Monotonic classification is a mathematical property of an AI model closely related to the concept of a monotonic function. Monotonic reasoning is a form of reasoning that can underlie AI system's logic.

3.23.2 Concept

- A* algorithm is a searching algorithm that searches for the shortest path between the initial and the final state. It is used in various applications, such as maps. In maps the A* algorithm is used to calculate the shortest distance between the source (initial state) and the destination (final state).
- A* search is a combination of lowest cost-first and best-first searches that considers both path cost and heuristic information in its selection of which path to expand.
- It uses cost(p), the cost of the path found, as well as the heuristic function h(p), the estimated path cost from the end p to the goal.
- 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 node g(n). It has combined features of UCS and greedy best-first search, by which it solves the problem efficiently.
- A* is complete and optimal on graphs that are locally finite where the heuristics are admissible and monotonic.
- Because A* is monotonic, the path cost increases as the node gets further from the root.
- Algorithm A* is a best-first search algorithm that relies on an open list and a closed list to find a path that is both optimal and complete towards the goal. It works by combining the benefits of the uniform-cost search and greedy-search algorithms. A* makes use of both elements by including two separate path finding functions in its algorithm that take into account the cost from the root node to the current node and estimates the path cost from the current node to the goal node.
- The first function is g(n), which calculates the path cost between the start node and the current node.
- The second function is h(n), which is a heuristic to calculate the estimated path cost from the current node to the goal node.

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

- It represents the path cost of the most efficient estimated path towards the goal. A* continues to revaluate both g(n) and h(n) throughout the search for all of the nodes that it encounters in order to arrive at the minimal cost path to the goal. This algorithm is extremely popular for path finding in strategy computer games.

The process for A* is basically this :

(1) Create an open list and a closed list that are both empty. Put the start node in the open list.

(2) Loop this until the goal is found or the open list is empty.

  (a) Find the node with the lowest F cost in the open list and place it in the closed list.

  (b) Expand this node and for the adjacent nodes to this node.

    (i) If they are on the closed list, ignore.

    (ii) If not on the open list, add to open list, store the current node as the parent for this adjacent node, and calculate the F, G, H costs of the adjacent node.

    (iii) If on the open list, compare the G costs of this path to the node and the old path to the node. If G cost of using the current node to get the node is the lower cost, change the parent node of the adjacent node to the current node.

(3) If open list is empty, fail.

- The terms locally finite, admissible, and monotonic all aid in the understanding of when A* can be expected to be complete, meaning that it finds a solution, and optimal, meaning that it finds the solution with the lowest path cost.
- A locally finite graph is one where none of the nodes on the graph have an infinite branching factor, thus none of the node paths branch forever. A branching factor of a node refers to the amount of new nodes that can be expanded from that node.

UEx. 3.23.1 : (MU - Q. 2(a), Dec. 2015, 10 Marks)

Consider the graph given in the Fig. Ex. 3.22.1. Assume that the initial state is S and the goal state is 7. Find a path from the initial state to the goal state using A* search. Also report the solution cost. The straight line distance heuristic estimate for the nodes are as follows :

h(1) = 14, h(2) = 10, h(3) = 8, h(4) = 12,

h(5) = 10, h(6) = 10, h(s) = 15

Module 3

Scanned with CamScanner

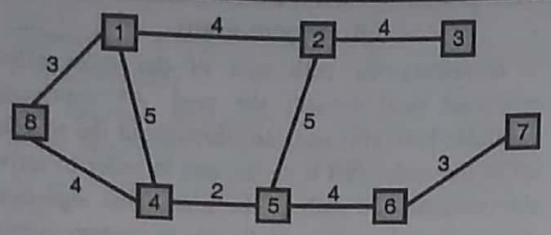


Fig. Ex. 3.23.1

☑ **Soln. :**

▶ **Step I :** We put the initial node S on a list START.

START is not empty and START ≠ GOAL

START : We generate the nodes ('.' It has successors)

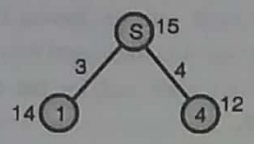


Fig. Ex. 3.23.1(a)

Now, open : 4(12 + 4), 1(14 + 3)

Closed : 5(15)

▶ **Step 2 :** We name the new list as START 1 and go to step (2)

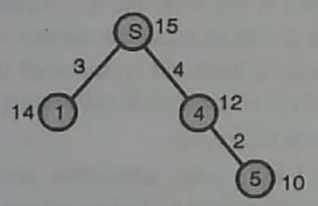


Fig. Ex. 3.23.1(b)

Open : 5 (10 + 6), 1 (14 + 3)

Closed : 5(15), 4 (12 + 4)

Remark :

We maintain two lists, open and closed. In the closed list, we place those nodes which have already expanded and in the open list, it places nodes which have yet not been expanded.

▶ **Step 3 :** We continue the process. We apply successors to node (5);

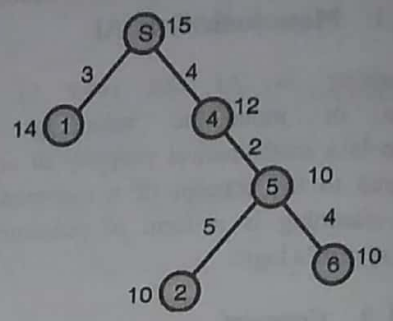


Fig. Ex. 3.23.1(c)

Again : open : 1 (14 + 3), 6(10 + 10), 2(10 + 11)

Closed : 5(15), 4(12 + 4), 5(10 + 6)

▶ **Step 4 :** We apply successors to node (1)

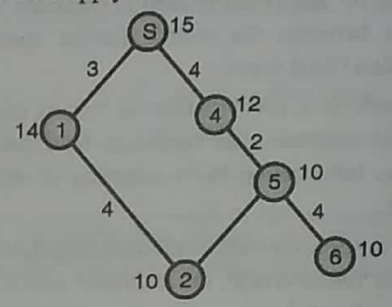


Fig. Ex. 3.23.1(d)

Open : 2 (10 + 7), 6(10 + 10)

Closed : 5(5), 4 (12 + 4), 5(10 + 6), 1(14 + 3)

▶ **Step 5 :** We apply successor to node (2)

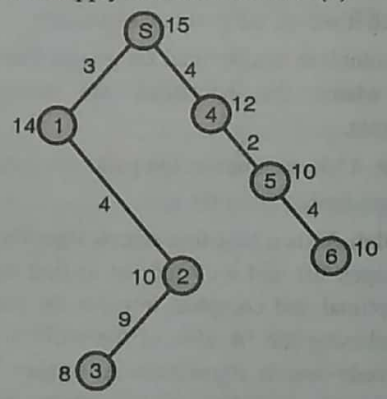


Fig. Ex. 3.23.1(e)

Open : 3(8 + 11), 6(10 + 10)

Closed :

5(15), 4(12 + 4), 5(10 + 6), 1(14 + 3), 2 (10 + 7)

▶ **Step 6 :** On simplification, step (5) becomes

Open : 6(10 + 10)

Closed :

5(15), 4(12 + 4), 5(10 + 6), 1 (14 + 3), 2(10 + 7) 3(8 + 11)

Scanned with CamScanner

Step 7 : We apply goal –node to sub node (6)

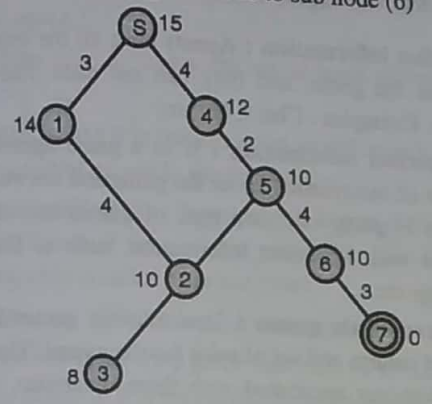


Fig. Ex. 3.23.1(f)

Open : 7 (0 + 13)

Closed : 5(15), 4 (12 + 4), 5(10 + 6), 1(14 + 3), 2(10 + 7), 5 (8 + 4), 6(10 + 10)

UQ. Compare following informed search algorithms
(a) Greedy first search
(b) A*
(c) Recursive Best-First (RBFS)
(MU- Q.4(a), May 16, 10 Marks)

(a) Greedy first search

(i) The Greedy best-first search algorithm tries to explore the node that is closest to the goal.

(ii) The algorithm evaluates nodes by using the heuristic function h(n)

(iii) That is , the evaluation function is equal to heuristic function,

f(n) = h(n). This equivalency is what makes the search algorithm 'greedy'.

(iv) The evaluation function of a target node is a weighted sum of the heuristic function and the distance from the current node to that target.

(v) It is the combination of depth-first search and breadth-first search algorithms.

(vi) It uses the heuristic function and search

(vii) The greedy best first algorithm is implemented by the priority queue.

(viii) The worst case **time complexity** of Greedy best first search is $O(b^m)$

(ix) **Space complexity :** The worst case space complexity of Greedy best first-search is $O(b^m)$, where m is the maximum depth of the search space.

(x) **Complete :** Greedy best-first search is also incomplete, even if the given space is finite.

(xi) **Optimal :** Greedy best first search is not optimal

(b) A*

(i) A* is a form of best-first search.

(ii) It uses heuristic function h(n).

(iii) Cost is to reach the node n from the start state g(n).

(iv) It has combined features of UCS and greedy best-first search, by which it solves the problem efficiently.

(v) A* search algorithm finds the shortest path through the search space using the heuristic function.

(vi) A* algorithm search finds the shortest path through the search space (using heuristic function).

(vii) This algorithm can solve very complex problems.

(viii) **Time Complexity :** The time complexity of A* search algorithm depends on heuristic function. Time complexity is O(b □ d), where b is the branching factor.

(ix) **Space complexity :** The space complexity of A* search algorithm is O(b □ d).

(x) **Complete :** A* algorithm is complete as long as :
(i) Branching factor is finite; and
(ii) Cost at every action is fixed.

(xi) **Optimal :** A* search algorithm is optimal if it follows two conditions :
(i) h(n) should be an admissible heuristic for A* tree search.
(ii) Consistency for A* graph-search.

(c) Recursive Best-First (RBFS)

(i) It is simple recursive algorithm and resembles the operation of standard best first search.

(ii) It uses only linear space.

(iii) It is similar to recursive DFS.

(iv) It is more efficient than IDA*

(v) It suffers from excessive node regeneration.

(vi) It keeps track of the f value of the best alternative path available from any ancestor of the current node.

(vii) It uses the problem specific information about the environment to determine the preference of one node over the other.

(viii) **Time complexity :** Its time complexity is difficult to characterise because it depends on the accuracy of h(n) and as the nodes are expanded, the best path changes.

Module 3

Scanned with CamScanner

(ix) **Space complexity :**

Space complexity is O(bd)

(x) **Complete :** Yes similar to A*

(xi) It is an optimal algorithm if h(n) is admissible; i.e. it is asymptotically optimal.

3.24 ADVERSARIAL SEARCH

- Adversarial search is a search when there is an **'enemy'** or **'opponent'** changing the state of the problem every step in a direction which we do not want to have.
- Each agent needs to consider the action of other agent and effect of that action on their performance. So, searchers 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.
- Adversarial search is search when there is an "enemy" or "opponent" changing the state of the problem every step in a direction you do not want.
- **Examples :** Chess, business, trading, war. You change state, but then you do not control the next state. Opponent will change the next state in a way

  1. Unpredictable 2. hostile to you

  You only have to change every alternate state.

In adversarial search we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.

(i) We study situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.

(ii) The environment with more than one agent is termed as multi-agent environment in which each agent is an opponent of other agent and playing against each other. Each agent needs to consider the action other agent and effect of that action on their performance.

(iii) 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'.

(iv) 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.

3.24.1 Types of Games in AI

(i) **Perfect information :** Agents have all the information about the game, and they can see each other moves also. Examples : Chess, Go etc.

(ii) **Imperfect information :** If in a game agents do not have all information about the game and not aware with what is going on, such type of games are called the game with imperfect information, such as Battleship, bridge etc.

(iii) **Deterministic games :** Deterministic games follow a strict pattern and set of rules for the games. There is no randomness associated with them. Examples : Chess, Tic-Tac-Toe etc.

(iv) **Non-deterministic games :** These are those games which have various unpredictable events and have a factor of chance or luck. These are random, and each action response is not fixed. Such games are called as **stochastic games**. Example : Poker etc.

(v) **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 tries to maximise one single value, while other player tries to minimise 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.

Zero-sum game : Embedded thinking :

The Zero-sum game involved embedded thinking in which one agent or player is trying to figure out

(i) What to do

(ii) How to decide the move

(iii) Needs to think about his opponent as well

(iv) The opponent also thinks what to do. Each of the players is trying to find out the response of his opponent to their actions. This requires embedded thinking or backward reasoning to solve the game problems in AI.

Formalization of the problem

A game can be defined as a type of search in AI which can be formalised of the following elements :

(i) **Initial state :** It specifies how the game is set up at the start.

Scanned with CamScanner

(ii) **Player (s) :** It specifies which player has moved in the state space.

(iii) **Actions (s) :** It returns the set of legal moves in state space.

(iv) **Result (s, a) :** It is the transition model, which specifies the result of moves in the state-space

(v) **Terminal-Test (s) :** Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.

(vi) **Utility (s, p) :** A utility function gives the final numerical value for a game that ends in terminal states S for player p. It is also called payoff function.

Non-zero sum game : In non-zero sum game, each agent's gain or loss is not balanced by loss or gain.

If player wins the game, then it does not mean that the other player has lost the game.

Positive sum game

Here all players have the same goal and they contribute together to play the game.

Negative sum game

Here nobody wins, everybody loses. Every player has a different goal; e.g. war

Game-Tree

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.

Example : Tic-Tac-Toe game tree

3.24.2 Characteristics of Adversarial Search (A.S.)

- Adversarial search in Artificial Intelligence is a game playing technique, where the agents are surrounded by a competative environment. 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.
- Searchers in which two or more players with conflicting goals are trying to explore the same 'search space' for the solution is called Adversarial Search.
- In A.S. there is enemy or opponent : Example : Chess, business, trading, war.
- Here an agent can change the state but then it cannot control the next state. Opponent will change the next state in an unprecedented way. Each agent needs to consider the action of other agent and effect of that action on its performance.

Module 3

3.24.3 Comparison of Search and Games

Sr. No.	Search	Games
1.	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.	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.
2.	To find optimal solution Heuristics techniques are used. In adversarial search, the result depends on the players which will decide the result of the game.	Games are modelled as a search problem and heuristic evalution function, and these are the two main factors which help to model games in AI.
3.	Pruning is a technique which allows ignoring the unwanted portions of a search tree which make no difference in its final result.	**Perfect information :** A game with the perfect information is that in which agents can look into the complete board. Examples are chess, Go etc.
4.	Heuristic evaluation function allows to approximate the cost value at each level of the search tree, before reaching the goal node. Examples are : Chess business trading war you can change the state but then you cannot	**Imperfect information :** If in a game agents do not have all information about the game and not aware with what is going on, such type of games are called imperfect information. **Example :** Bridge, Battleship blind etc.

Scanned with CamScanner

Sr. No.	Search	Games
	• NOTES •	**Deterministic games :** They follow a strict pattern and set of rules for the games. For example : Chess tic-tac-toe. **Non-deterministic games :** Such games have various unpredictable events and has a factor of change or luck. These are random, and each action response is not fixed. **Example :** Monopoly, Poker etc.

3.25 TECHNIQUES REQUIRED TO GET THE BEST OPTIMAL SOLUTION

There is always a need to choose those algorithms which provide the best optimal solution in a limited time. So, we use the following techniques which could fulfil our requirements :

- **Pruning** : A technique which allows ignoring the unwanted portions of a search tree which make no difference in its final result.
- **Heuristic Evaluation Function** : It allows to approximate the cost value at each level of the search tree, before reaching the goal node.

3.26 GAME PLAYING

3.26.1 Zero Sum Game

UQ. Write short note on : Game Playing.
(MU - Q. 6(a), May 17, 5 Marks, Q. 6(e), May 17, 5 Marks)

- 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.
- The Zero-sum game involved embedded thinking in which one agent or player is trying to figure out:
  - What to do.
  - How to decide the move
  - Needs to think about his opponent as well
  - The opponent also thinks what to do
- Each of the players is trying to find out the response of his opponent to their actions. This requires embedded thinking or backward reasoning to solve the game problems in AI.

3.26.2 Elements of Game Playing Search

UQ. Draw a game tree for a tic-tac-toe problem.
(MU - Q. 4(c), Dec. 2015, 4 Marks)

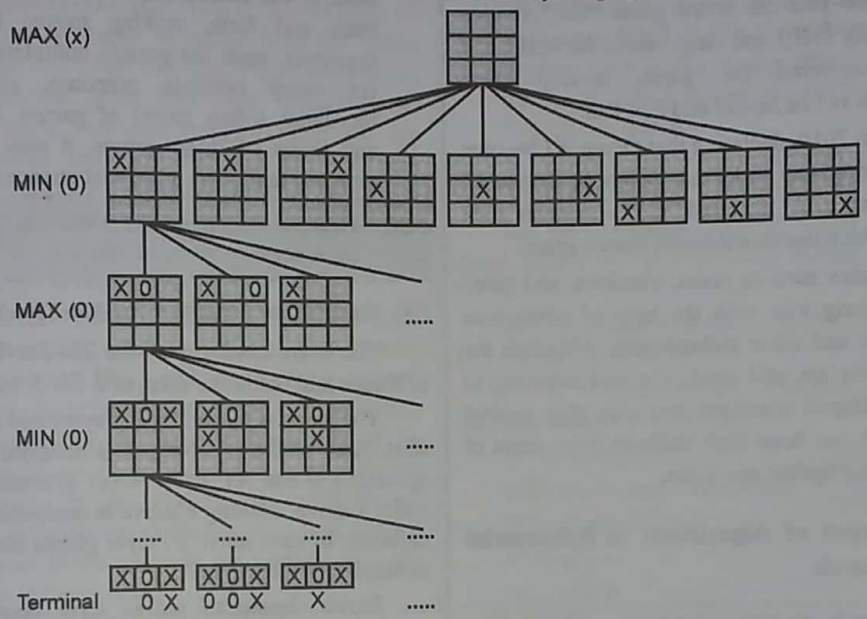
To play a game, we use a game tree to know all the possible choices and to pick the best one out. There are following elements of a game-playing:

- **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.**

Scanned with CamScanner

Let's understand the working of the elements with the help of a game tree designed in Fig. 3.26.1 for tic-tac-toe. Here, the node represents the game state and edges represent the moves taken by the players.



(1B54)**Fig. 3.26.1 : A game-tree for tic-tac-toe**

- **INITIAL STATE (S_0) :** 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.

3.26.3 Some More Examples of Game Playing/Adversarial Search

(1) **Chess :** In 1997, IBM's supercomputer dubbed "Deep Blue" did just that. And it didn't beat just any person, but the world chess champion Garry Kasparov. In an essay, Kasparov wrote about his first game with Deep Blue in 1996. He said that while he played numerous times against a computer, his match against Deep Blue was different. He sensed a "new kind of intelligence across the table." During a rematch in 1997, the chess master lost to the supercomputer. In this instance, we can deduce that Deep Blue was better at adversarial search.

- Indeed, Deep Blue has enormous computational power. It can consider 100–200 billion positions per second and has around 4,000 positions in its opening book.
- In 2006, another world champion, Vladimir Kramnik, was defeated by a machine. This time, it was by Deep Fritz, a German chess computer program.

(2) **Checkers**

- Checkers is another two-player game that can use adversarial search. A computer program called "Chinook" was developed specifically to play in the World Checkers Championship. And in 1990, it reached its goal. Chinook was the first computer program that won the right to play for the World Checkers Championship.
- In contrast to Deep Blue and Deep Fritz, however, Chinook's knowledge of the game was not learned via AI, as its developers programmed everything. Still, we can't discount the fact that it is a powerful application. It has a search space of 5×1020 or 500,000,000,000,000,000,000 sets of possible moves (that's 500 quintillion, in case you're wondering).

Module 3

Scanned with CamScanner

(3) "Go"

- AlphaGo is a computer program designed to learn and master the 3,000-year-old board game "Go." It uses machine learning (ML) and deep neural networks and has indeed mastered the game, beating "Go" champions such as Lee Se-dol and Fan Hui.
- Se-dol retired in 2019, declaring that "Even if I become number one, there is an entity that cannot be defeated." He was referring to AI-powered "Go" opponents such as AlphaGo, which has an enormous search space.
- Two-player games such as chess, checkers, and "Go" have come a long way with the help of adversarial search methods and other technologies. Although the same game rules are still used, it's awe-inspiring to think that intelligent machines can also play against humans. People can hone their skills in these types of games by playing against machines.

3.26.4 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

3.27 GAME TREE

GQ. Explain game tree.

- A **game tree** is a type of **recursive search function** that examines all possible moves of a strategy game, and their results, in an attempt to ascertain the optimal move. They are very useful for artificial intelligence in scenarios that do not require real-time decision making and have a relatively low number of possible choices per play.
- The most commonly-cited example is chess, but they are applicable to many situations. Game trees are generally used in board games to determine the best possible move. For the purpose of this article, **Tic-Tac-Toe** will be used as an example.
- The idea is to start at the current board position, and check all the possible moves the computer can make. Then, from each of those possible moves, to look at what moves the opponent may make. Then to look back at the computers. Ideally, the computer will flip back and forth, making moves for itself and its opponent, until the game's completion. It will do this for every possible outcome, effectively playing thousands (often more) of games. From the winners and losers of these games, it tries to determine the outcome that gives it the best chance of success.

GQ. How AI technique is used to solve tic-tac-toe problem?

Heuristics function for Tic-Tac-Toe problem

The board used to play the Tic-Tac-Toe game consists of 9 cells laid out in the form of a 3×3 matrix.

The game is played by 2 players and either of them can start. Each of the two players is assigned a unique symbol (generally 0 and X). Each player alternately gets a turn to make a move. Making a move is compulsory and cannot be deferred. In each move a player places the symbol assigned to him/her in a blank cell.

Seven classes of moves have been designed using the available heuristics. Each class of moves represents a set of functionally cohesive moves in order to achieve a certain objective during a game. These classes of moves are defined and their roles in playing the game are discussed below :

1. **Prioritized selection (PS) :** The PS class of moves selects the blank cell with the maximum priority. If there exist more than one cell with the maximum priority, then any one of them can be selected. The PS class of moves makes sure that the player has control over the most important cells on the board.
2. **Motion (M) :** The M class of moves finds all tracks with only one cell filled with the symbol assigned to the player and other two cells blank. Then one of these tracks with the highest priority is chosen. After that the blank cell with higher priority in the chosen track is selected. The M class of moves makes sure that the player continues filling a track in which there is still a chance to win.
3. **Definitive offense (DO) :** This class of moves finds a track with exactly two cells filled with the symbol assigned to the player and the third cell blank. This blank cell is selected. This moves is meant to provide an immediate win to the player.
4. **Definitive defense (DD) :** This class of moves finds a track with exactly two cells filled with the symbol assigned to the opponent player and the third cell blank. This blank cell is selected. It is meant to prevent the player from an immediate loss. If the

Scanned with CamScanner

moves are not used then the opponent definitely gets a chance to win in the subsequent move.

5. **Tentative offense (TO) :** This class of moves finds all pairs of intersecting tracks in which both the tracks have exactly one cell filled with the symbol assigned to the player and the other two cells including the common one blank. All such common cells of the intersecting tracks are identified and the one with the maximum priority is selected. This move tries to keep the foundations of victory simultaneously on two tracks. If the TO class of moves can be applied then the player can win in the next move.

6. **Tentative defense (TD) :** This class of moves finds all pairs of intersecting tracks in which both the tracks have exactly one cell filled with the symbol assigned to the opponent player and the other two cells including the common one blank. All such common cells of the intersecting tracks are identified and the one with the maximum priority is selected. It tries to undo the effect of the tentative offense class of moves applied by the opponent. If the move is not applied then the player can lose in the next move.

7. **Diagonal correction (DC) :** The above six classes of moves may be insufficient to prevent a loss for the player moving second if either of the two diagonal tracks are filled in the first three moves. This may happen even if the losing player controls the more important cells. It is used to save the player from losing in such conditions.

In Tic-Tac-Toe, player alternate putting marks in a 3×3 array, one mark (*X*) and other mark (0).

Let the evaluation function, e (P) of a position P be given simply by, If *P* is not a wining position for either player.

e (P) = (number of complete row, column, or diagonal that are still open for X)

– (number of complete row, column, or diagonal that are still open for 0)

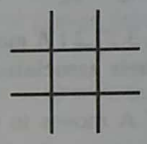
If P is a win for X,

e (P) = ∞ (a very large value)

If P is a win for 0,

e (P) = – ∞

Thus if *P* is



Module 3

We have e(P) = 6.4 = 2

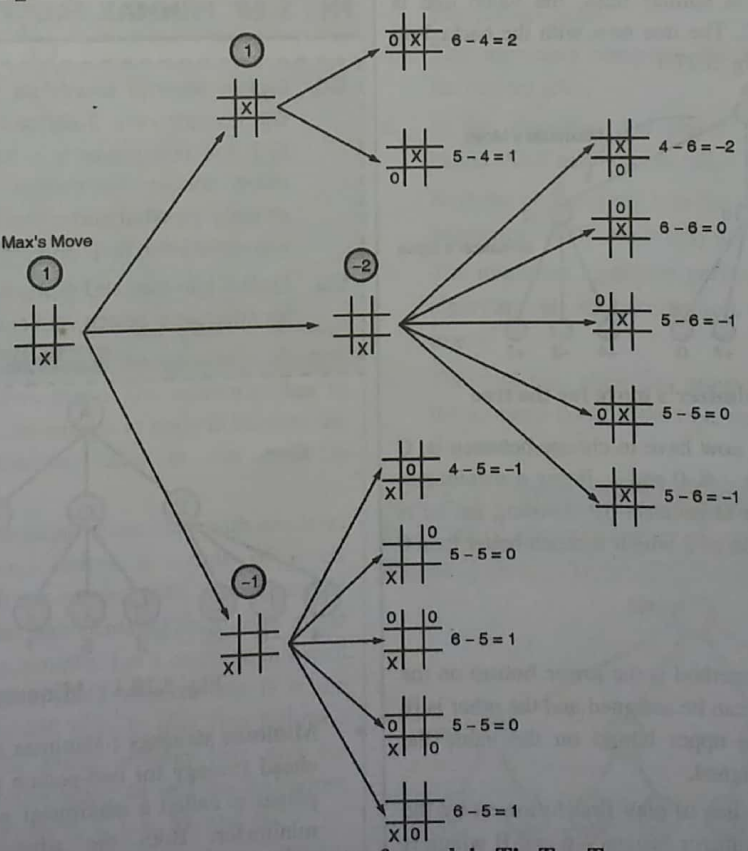


Fig. 3.27.1 : First state of search in Tic-Tac-Toe

Scanned with CamScanner

3.27.1 tic-tac-toe Problem

GQ. How AI technique is used to solve tic-tac-toe problem ?

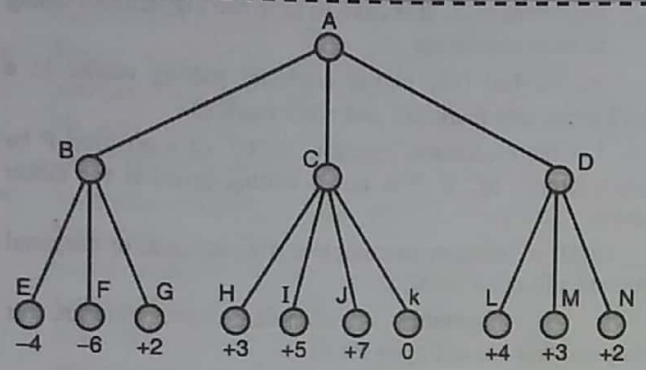


Fig. 3.27.2 : A game tree expanded by two levels and their associated static evaluation function value

8. If A moves to C, then the minimizer will move to K (static evaluation function value = 0) which is the minimum of 3, + 5, 7 and 0. So the value of 0 is backed up at C. On similar lines, the value that is backed up at D is 2. The tree now with the backed up values is given in Fig. 3.27.3.

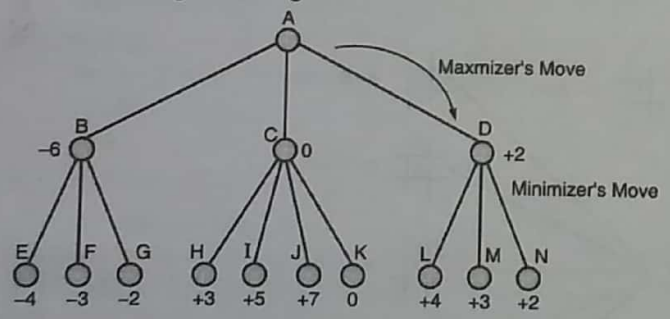


Fig. 3.27.3 : Maximizer's move for the tree

9. The maximizer will now have to choose between B, C or D with the values – 6, 0 and 2. Being a maximizer, he will choose node D because by choosing so, he is sure of getting a value of 2 which is much better than 0 and – 6.

α – β pruning

1. In α – β pruning, α method is the **lower bound** on the value the maximizer can be assigned and the other is β, which represents the upper bound on the value the minimizer can be assigned.
2. Here, the maximizer has to play first followed by the minimizer. Thus maximizer assigns – 6 and B which is passed back to A. This is replaced by 3, value passed by C as A has the maximizer move. Now A will not be examining D and its children. Since value of K is zero and D is having minimizer move making its value 0 or less than 0 only. Thus the tree with the root K will be pruned, which would save a lot of time in searching.

3.27.2 Limitations of Game Trees

1. As mentioned above, game trees are rarely used in real-time scenarios (when the computer isn't given very much time to think).
2. The method requires a lot of processing by the computer, and that takes time.
3. For the above reason (and others) they work best in turn-based games.
4. They require complete knowledge of how to move.
5. Games with uncertainty generally do not mix well with game trees.
6. They are ineffective at accurately ascertaining the best choices in scenarios with many possible choices

3.28 MINMAX PROCEDURE

GQ. Explain Minmax procedure with suitable example.
OR Consider the 2-ply search as shown below : (i) If the first player is a maximizing player, what move should be chosen under the mini-max strategy. (ii) What nodes should not be needed to be examined using α–β pruning technique?

UQ. Explain Min max and Alpha beta pruning algorithms for adversarial search with example.

(MU - Q. 5(b), May 17, 10 Marks)

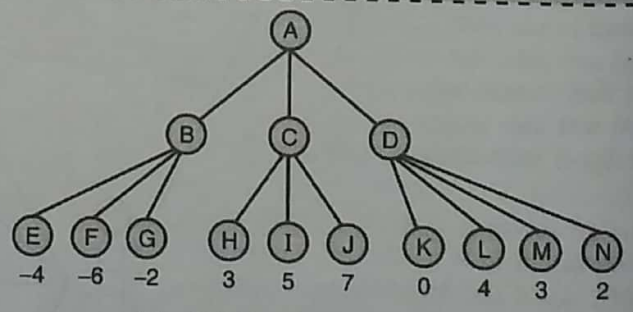


Fig. 3.28.1 : Minmax strategy

- **Minimax strategy :** Minimax strategy is a simple look ahead strategy for two-person game-playing. Here one player is called a maximizer and the other is called a minimizer. Both the adversaries, maximizer and minimizer fight it out to see to that the opponent gets

Scanned with CamScanner

the minimum benefit while they get the maximum benefit. The plausible move generator generates the necessary states for further evaluation and the static evaluation function "ranks" each of the positions.

Mimax Strategy Algorithm

1. The working of the algorithm is described with the aid of Fig. 3.28.2. Let A be the initial state of the game. The plausible move generator generates three children for that move and the static evaluation function generator assigns the values given along with each of the states.
2. It is assumed that the static evaluation function generator returns a value from – 20 to + 20, wherein a value of – 20 indicates a win for the maximizer and a value of – 20 a win for the minimizer. A value of 0 indicates a tie or draw.
3. It is also assumed that the maximizer makes the first move. (It is not essential so. Even a minimizer can make the first move.) The maximizer, always tries to go to a position where the static evaluation function value is the maximum positive value.

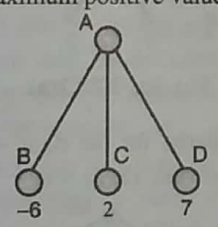


Fig. 3.28.2 : Initial state of the game

4. The maximize being the player to make the first move; will move to node D because the static evaluation function value for that node is the maximum (figure). The same figure shows that if the minimizer has to make the first move, he will go to node B because the static evaluation function value at that node is advantageous to him.
5. But a game-playing strategy never stops with one level but looks ahead, i.e., moves a couple of levels downwards to choose the optimal path. Sometimes, by expanding these nodes and scanning them, one might be forced to retract or rethinks. Let's examine this with the help of Fig. 3.28.2. Let's assume that it is the maximizer again who will have to play first followed by the minimizer. The search strategy here tries for only two moves, the root being M and the leaf nodes being **A, B, C, D, E, F, G, H, I, J, L, M, and N.**
6. Before the maximizer moves to B, C or D it will have to think which move would be highly beneficial to him. In order to evaluate, the children of the intermediate nodes B, C and D are generated and the static evaluation function value generator has assigned values for all the leaf nodes.
7. If A moves to B it is the minimizer who will have to play next. The minimizer always tries to give the minimum benefit to the other and hence he will move to G (static evaluation function value – 6). This value is backed up at N.

3.28.1 Properties of Min Max Algorithm

1. Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory.
2. It provides an optimal move the for player assuming that opponent is also playing optimally.
3. Mini-max algorithm uses recursion to search through the game-tree.
4. Mini-max algorithm is mostly used for game playing in AI. Such as chess, checkers, tic-tac-toe, go and various tow-players game.
5. This algorithm computes the mini-max decisions for the current state.
6. In this algorithm two players play the game, one is called MAX and other is called MIN.

   Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
7. The mini-max algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
8. The mini-max algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

UEx. 3.28.1 : (MU-Dec. 15, Dec. 19 10 Marks)
Problems on min-max search on game tree as shown in the Fig. Ex. 3.28.1.

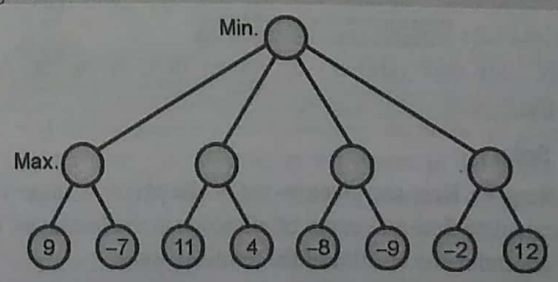


Fig. Ex. 3.28.1

Module 3

Scanned with CamScanner

Soln. :

Step (I) : Since the given move of the player is max; we calculate first maximum of all nodes.

We begin with left node of the layer above the terminal; to calculate the utility of the left node.

Now, max $\{9, -7\} = 9$

$\therefore$ Utility of the left most node $= 9$ ...(i)

The utility of the next node of the same layer is max $\{11, 4\} = 11$ ...(ii)

Again the utility of the node (beginning from left)

$= \max\{-8, -9\} = -8$

and the utility of the last node of the same layer is max $\{-2, 12\} = 12$

Step II :

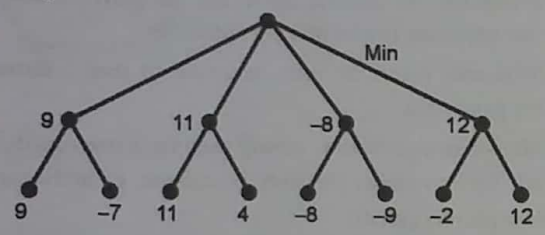


Fig. Ex. 3.28.1(a)

Now, we calculate the utility values, considering one layer at a time, till we reach the root of the tree, i.e. the top-most point.

Here we have 3 layers, so we can directly evaluate min $\{9, 11, -8, 12\} = -8$

Step III : Thus, the best opening move for min is the third node.

This move is called the min max decision. It maximises the utility with the knowledge that the opponent is also playing optimally to minimise it.

Step IV : Thus,

Min-max decision $= \min\{\max(9, -7), \max\{11, 4\}, \max\{-8, -9\}, \max\{-2, 12\}\}$

$= \min\{9, 11, -8, 12\} = -8$

UEx. 3.28.2 : (MU - May 18, 10 Marks)

Apply min-max search on game tree given in the Fig. Ex. 3.28.2.

Soln. :

Step I : Here the given move of the player is max; we calculate first maximum of all nodes in the last layer, to determine the utilities of the terminal nodes.

We begin with the left node of the layer.

Again move of the layer is maximum, we choose the maximum of all the utilities.

(i) $\max\{4, 3, 1\} = 4$

$\therefore$ Utility of the left most-node is 4

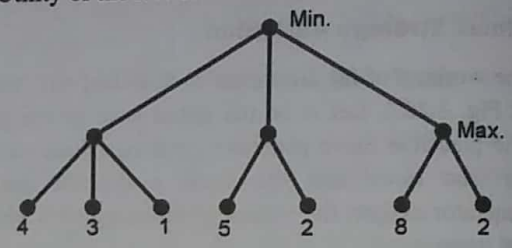


Fig. Ex. 3.28.2

Step II : Again we evaluate maximum of the middle node in the same layer, i.e.

$\max\{5, 2\} = 5$

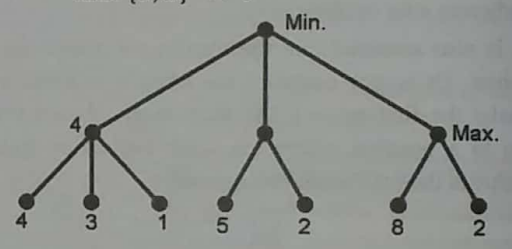


Fig. Ex. 3.28.2(a)

Step III : Similarly, for the right –most of the same layer, we evaluate $\max\{8, 2\} = 8$

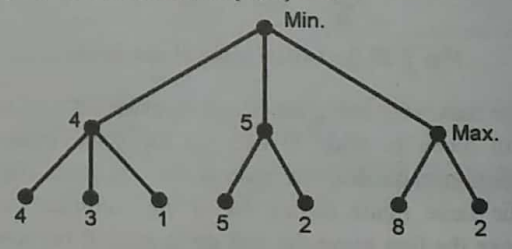


Fig. Ex. 3.28.2(b)

Step IV : Here there are only 3 layers, so we immediately reach to the root. At the root, i.e. the topmost point, min has to choose the minimum value.

So, we evaluate min $\{4, 5, 8\} = 4$

$\therefore$ The best opening move for min is the left most node

Note that this move is called as minmax decision as it maximises the utility under the assumption that the opponent is playing optimally to minimise it.

$\therefore$ Minmax decision $= \min\{\max(4, 3, 1), \max.\{5, 2\}, \max.\{8, 2\}\}$

$= \min\{4, 5, 8\} = 4$

Scanned with CamScanner

3.29 GAME OF CHANCE

Q. Explain game of chance.

- **Games of chance** are among the oldest of human inventions. The use of a certain type of **animal heel bone** (called the astragalus or colloquially the knucklebone) as a crude die dates to about 3600 BC.
- A game of chance is defined as any game where a **randomizing element** plays a large role in the outcome. This could be as simple as choosing heads or tails, or it might involve rolling dice, picking a card, or moving left instead of right. Skill can still play a large part in such games, although a certain unpredictable variable will always exist.
- It is a game whose outcome is strongly influenced by some randomizing device, and upon which contestants may choose to wager money or anything of monetary value. Common devices used include dice, spinning tops, playing cards, roulette wheels, or numbered balls drawn from a container. A game of chance may have some skill element to it, however, chance generally plays a greater role in determining the outcome than skill. A game of skill, on the other hand, also has an element of chance, but with skill playing a greater role in determining the outcome.
- Games of chance may also involve a certain degree of skill. This is especially true where the player or players have decisions to make based upon previous or incomplete knowledge, such as blackjack. In other games like **roulette and punto banco (baccarat)** the player may only choose the amount of bet and the thing he/she wants to bet on, the rest is up to chance, therefore these games are still considered games of chance with small amount of skills required. The distinction between **'chance'** and **'skill'** is relevant as in some countries chance games are illegal or at least regulated, where skill games are not.

3.30 VARIOUS CATEGORIES OF GAME OF CHANCE

Q. What are the various categories of game of chance?

1. **Arcade games :** While console games are predominant in most nations outside of Japan, arcade games were once the primary form of entertainment for players obsessed with video gaming. From the 1980s to the early '90s, arcade games could be found in most malls and restaurants. The all-time most popular titles include **Pac- Man, Space Invaders, Street Fighter II, and Defender.**
2. **Board games :** The primary component for this category is a board, although dice and playing pieces are also common. Notable examples include monopoly, backgammon, risk, and trivial pursuit. Board games have been discovered dating back to 3500 B.C., which is a testament to their enduring popularity.
3. **Card games :** A 52-card deck is the most common, although special decks can also be purchased near the toy section of any major retail chain. Poker and blackjack are common examples of card games where money is placed on the line (and they also fall into the gambling category), while other examples include magic : The gathering, spades, and old maid.
4. **Computer games :** Instead of being generated by a console such as Nintendo or PlayStation, these games are played directly on a Mac or PC. The quality and complexity of these games range from basic to advanced, with superior computing power giving them an edge over console games. Popular games over the years have included The Sims, Diablo III, Half-Life, and Myst.
5. **Console games :** A video game system generates the images, which are then displayed on a television screen. Console games have enjoyed improved graphics and greater popularity over the last several decades, and the leading companies include Nintendo, PlayStation, and Xbox.
6. **Dice games :** One or more dice are a cornerstone of these games. They have been played for thousands of years, and current favorites include Yahtzee, craps, and poker dice. Since roleplaying and board games frequently use dice, you can expect some products to fall into multiple categories.
7. **Drinking games :** Played in pubs and at keg parties, the objective of these games is often to provide participants with a legitimate excuse to drink more alcohol. While some drinking games emphasize speed or skill, others such as Mexico and Ring of Fire have a more random component.
8. **Gambling games :** This category includes all games where money is wagered by the player in the hopes of receiving a larger payout, whether inside or outside of a casino. Examples include sports betting, keno, the lottery, bingo, and slot machines.

Module 3

Scanned with CamScanner

9. **Miniature games :** Often associated with miniature wargaming, this category includes games where tiny figures are used to represent opposing armies. Elaborate gaming surfaces often mimic battlefields and towns, while players spend countless hours painting figures. Popular products include warhammer fantasy battle, flames of war, and space marines.

10. **Mobile games :** If a game is available on a mobile phone or tablet computer, then it falls into this category. Thousands of apps are now designed to entertain consumers, with some of the more popular options being Plants vs. Zombies and Fruit Ninja. This category can also apply to handheld video games such as those created for the Game Boy or PlayStation Portable.

11. **Pencil-and-paper games :** If you're looking for a game that only requires pencil and paper, then you've come to the right category. Games of chance under this heading include Battleships and Hangman.

12. **Roleplaying games :** First popularized by Dungeons and Dragons, roleplaying games allow a player to take on the role of a character and engage in adventures in all manner of settings. Much of the game plays out like theatre, while the success of combat and skills are often determined by rolling dice. Notable examples include D and D, Pathfinder, Traveler, Mutants and Masterminds, Call of Cthulhu, and Rifts.

13. **Tile-based games :** Tiles are used as one of the main elements of play, whether they make up the board or comprise the playing pieces. Examples include dominoes, mahjong, and Scrabble.

14. **Travel games :** Also known as "car games" these games are meant to pass the time on long trips. They are often simple and require no playing pieces, although travel editions of many popular board games are also on the market. Playing a travel game is often as simple as counting animals or predicting the color of the next car to be passed on the highway.

Games of chance support the human need for competition, as well as the desire to flirt with uncertainty.

3.31 ALPHA-BETA PRUNING

GQ. Explain Alpha-beta pruning.

UQ. Explain Min max and Alpha beta pruning algorithms for adversarial search with example.
(MU - Q. 5(b), May 17, 10 Marks)

UQ. Define Alpha and Beta value in game tree?
(MU - Q. 1(c), May 16, 4 Marks)

Alpha-beta search reduces the number of nodes explored in minimax strategy. It reduces the time required for the search.

The exact implementation of alpha-beta keeps track of the best move for each side as it moves throughout the tree.

Alpha values

1. At a Max node we will store an alpha value
2. A lower bound on the exact minimax score
3. The true value might be $\geq \alpha$
4. If we know Min can choose moves with **score** $< \alpha$
5. Then Min will never choose to let Max go to a node where the score will be α or more.

Beta values

- At a min node
  1. The beta value is an upper bound on the exact minimax score.
  2. The true value might be $\leq \beta$.
  3. If we know Max can choose movers with **score** $> \beta$.
- Then Max will never choose to let Min go to a node where the score will be β or less.

Alpha beta in action

Why can we cut off search?

- $\beta = 2 < \alpha = 3$ where the a value is at an ancestor node.
- At the ancestor node max had a choice to a score of at least 3.
- Max is not go going to move right to let min guarantee a score of 2.

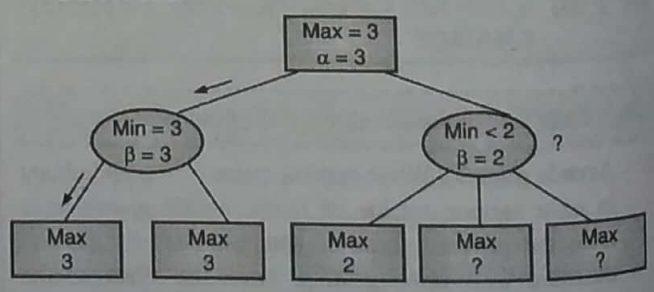


Fig. 3.31.1

Scanned with CamScanner

- **Search can be discontinued at a node if :**
  1. It is a Max node and the alpha value is ≥ the beta of any min ancestor. This is beta cutoff
  2. Or it is a min node and the beta value is ≤ the alpha of any max ancestor. This is alpha cutoff.

3.31.1 Calculating Alpha-Beta Values

- Alpha-Beta calculations similar to Minimax, but the pruning rule cuts down search.
- Final backed up value of node.
  1. Might be the maximum value.
  2. Or might be an approximation where search cut off.
- Less than the minimax value at a Max node.
- More than the minimax value at a Mid node.
- We don't need to know the true value.

3.31.2 Calculating Alpha Values at a Max Node

- After we obtain the final backed up value of the first child.
  1. We set cx of the node to this value.
- When we get the final backed up value of the second child.
  1. We increase ix if the new value is larger.
- When we have the final child, or if β cutoff.
  1. α value becomes the final backed up value.
  2. Only then can we set the β of the parent Min node

3.31.3 Calculating Beta Values at Min

- After we obtain the final backed up value of the first child, we set β of the node to this value
- When we get the final backed up value of the second child
  1. We decrease β if the new value is smaller
- When we have the final child, or if a – cutoff
  1. β value becomes the final backed up value

3.31.4 Alpha-beta Pruning Example

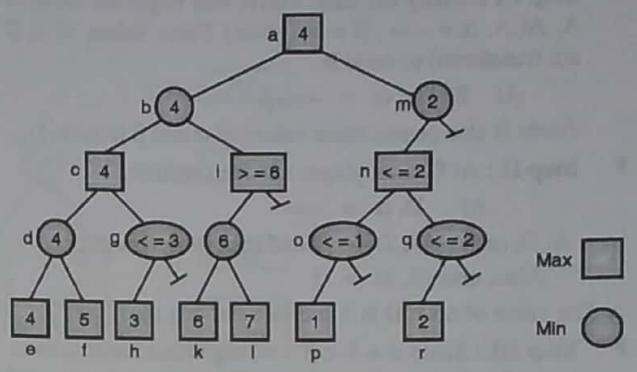


Fig. 3.31.2

- MINIMAX (n, alpha, beta)
- IF n is at the search depth, RETURN V(n)
- FOR each child m of n
- value = MAXIMIN (m, alpha, beta) IF value < beta , beta = value
- IF beta <= alpha, return alpha RETURN beta

Module 3

Performance of Alpha-Beta pruning

1. Efficiency depends on the order in which the nodes are encountered at the search frontier.
2. Optimal - $b^{1/2}$ - if the largest child of a MAX node is generated first, and the smallest child of a MIN node is generated first.
3. Worst - b.
4. Average $b^{3/4}$ - random ordering.

⇒ **Example of alpha-beta pruning**

Ex. 3.31.1 : Apply alpha-beta pruning on two player search tree shown in the Fig. Ex. 3.31.1.

$\alpha = -\infty; \beta = \infty$

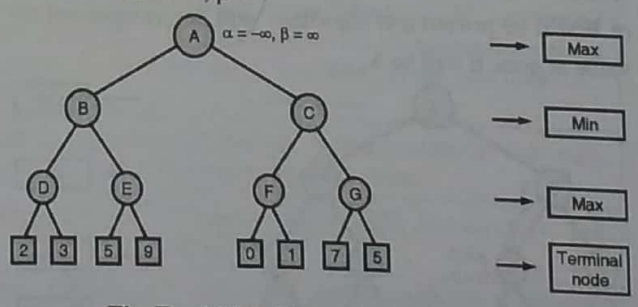


Fig. Ex. 3.31.1: Two player search tree

Scanned with CamScanner

Soln. :

- **Step I :** Initially the max. Player will begin the move at A. At A, $\alpha = -\infty$, $\beta = \infty$ (given) These values of α, β are transferred to node B.

  At B also, $\alpha = -\infty, \beta = \infty$

  Node B also passes these values of α and β to node D.

- **Step II :** At D, max player will start its turn.

  At D, $\alpha = -\infty$

  $\therefore$ At D, max $(-\infty, 2) = 2$ and max $(-\infty, 3) = 3$

  Also, max $(2, 3) = 3$

$\therefore$ The value of node D is 3 and value of α is also (say) 3.

- **Step III :** Since $\alpha = 3 < \beta = \infty$; algorithm backtracks to node B. At B, min player will move and value of β $(= \infty)$ will change.

  Value of β will be compared to the nodal value of D = 3; and min $(\infty, 3) = 3$

  $\therefore$ At B, $\alpha = -\infty$, $\beta = 3$

- **Step IV :** Since $\alpha < \beta$ at B, algorithm traverses the next successor of Node B. Which is node E, and the values of $\alpha = -\infty$, $\beta = 3$ are transferred to node E.

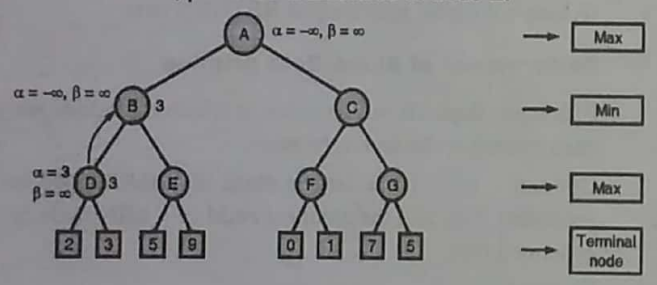


Fig. Ex. 3.31.1(a)

- **Step V :** At E, max player will make a move

  $\therefore$ Value of α will change

  Now max. $(-\infty, 5) = 5$ or max. $(-\infty, 9) = 9$

  $\therefore$ at node E, $\alpha = 5 > \beta = 3$; hence the right success or of E will be pruned and algorithm will not traverse and the value at node E will be 5.

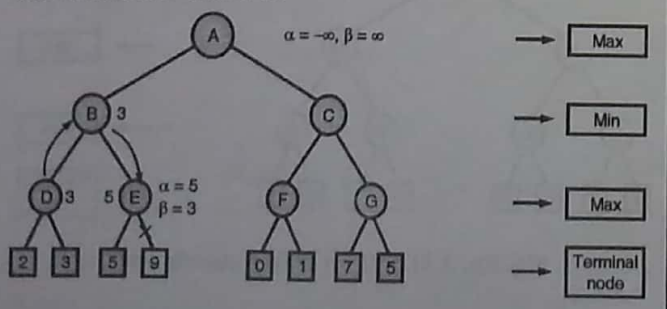


Fig. Ex. 3.31.1(b)

- **Step VI :** Now algorithm again backtrack the tree, from node B to node A.

  At A, there is max player, and α will take the maximum value, that is 3, $\because$ max $(\infty, 3) = 3$ and $\beta = \infty$

  These two values i.e.; $\alpha = 3$, $\beta = \infty$ will transfer to the successor C. And at node C, again $\alpha = 3$, $\beta = \infty$, and these values again will get transferred to node F.

  At F, there is max. Player

- **Step VII :** At node F, we compare the value of α with the left node O, and

  max $(3, 0) = 3$

  Now node value of F = max $(0, 1) = 1$

  $\therefore$ $\alpha = 3 >$ node value of F

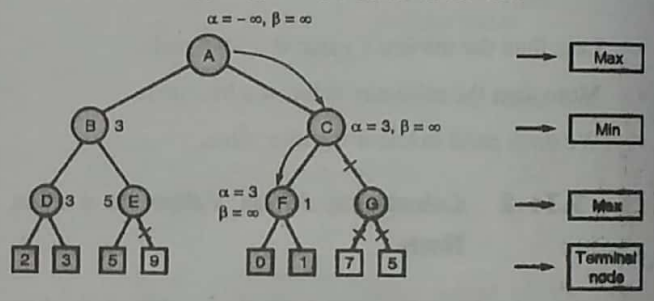


Fig. Ex. 3.31.1(c)

- **Step VIII :** Now the right successor is pruned. ($\because \alpha >$ node value of F)

  $\therefore$ node F gets back to node C and at C,

  $\alpha = 3$, $\beta = \infty$

  Here β will change, since C is min player node. At C node value is 1,

  $\therefore$ $\beta = \min(\infty, 1) = 1$

  Again $\alpha > \beta$ $(\because \alpha = 3, \beta = 1)$

  $\therefore$ Next node G will be pruned and algorithm stops computing at G

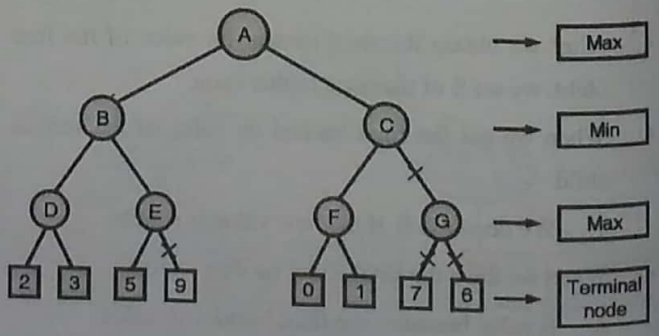


Fig. Ex. 3.31.1(d)

Scanned with CamScanner

▸ **Step IX :** At C, $\alpha = 3$, $\beta = 1$;

($\because \alpha > \beta$); C goes back to A, and for A, max. (3, 1) = 3

$\therefore$ optimiser value for this optimiser is β.

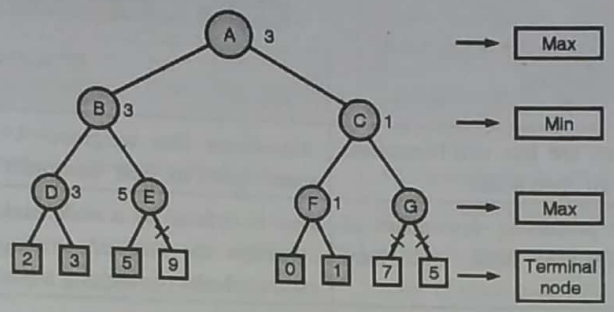


Fig. Ex. 3.31.1(e)

3.32 COMPARISONS IN TABULAR FORM

3.32.1 Difference Between Informed Search & Uninformed Search

UQ. Difference between Informed and Uninformed Search in AI.

(MU - Q. 3(a), Dec. 19, 10 Marks, Q. 4(b), Dec. 18, 10 Marks, Q. 4(b), May 16, 10 Marks)

Module 3

Sr. No.	Parameters	Informed Search	Uninformed Search
1.	Information required in problem solving	Informed strategies use agent's background information about the problem map, costs of actions, approximation of solutions.	Uninformed strategies use only the information available in the problem definition.
2.	Goal state	In informed search, a heuristic is used as a guide that leads to better overall performance in getting to the goal state.	Uninformed search (blind search) has no information about the number of steps or the path costs from the current state to the goal.
3.	Goal state	Instead of exploring the search tree blindly, one node at a time, the nodes that we could go to are ordered according to some evaluation function h(n) that determines which node is probably the "best" to go to next.	They can only distinguish a goal state from a non-goal state. There is no bias to go towards the desired goal.
4.	Search strategies	Also known as "heuristic search", informed search strategies use information about the domain to (try to) (usually) head in the general direction of the goal nodes(s).	Also known as "blind search", uniformed search strategies use no information about the likely "direction" of the goal nodes(s).
5.	Efficiency	Using specific knowledge this method gives solution efficiently.	Uninformed search methods are very inefficient in most cases.
6.	Time and memory consumption	Use comparatively much lesser time and memory.	It leads to higher time and memory time complexity.
7.	Examples	Informed search methods : Hill climbing, best-first, greedy search, beam search, A* algorithm AO* algorithm.	Uninformed search methods : Breadth-first, depth-first, depth-limited, uniform-cost, depth-first iterative deepening, bidirectional.

Scanned with CamScanner

3.32.2 Differentiate Hill Climbing vs A* Algorithm

UQ. How do you compare Hill climbing technique with A* algorithm.
MU - Q. 2(A), May. 17, 5 Marks, Q. 2(b), May 16,5 Marks

Sr. No.	Hill Climbing	A* Algorithms
1.	Hill climbing algorithms are less deli-beratative, rather than considering all open nodes.	Algorithms like weighted A* systematically explore the search space in 'best' first order.
2.	They expand the most promising descendant of the most recently expanded node until they encounter a solution.	Best is defined by a node ranking function which typically considers the cost of arriving at a node, g, as well as estimated cost of reaching a goal from a node h.
3.	Steepest ascent hill climbing is similar to best-first search, which tries all possible extensions of the current path instead of only one.	A* also considers the distance of a node from the goal, d.

3.32.3 Comparison of Hill-Climbing and Simulated Annealing

UQ. Explain Hill Climbing and Simulated Annealing with suitable example.
(MU - Q. 2(a), Dec. 18, 10 Marks, Q. 2(a), May 18, 10 Marks)

	Hill climbing	Simulated Annealing
(1)	Hill climbing gets stuck in a local maxima because downward moves are not allowed.	Simulated Annealing is a variation of hill climbing.
(2)	Hill-climbing can be applied to most combinatorial optimisation problems.	At the beginning of the process, some downhill moves may be made.
(3)	The three algorithms are used to solve the mapping problem, which is the optimal static allocation of communication processes on distributed memory architecture.	The idea is to do enough exploration of the whole space early on so that the final solution is relatively intensive to the starting state.
(4)	A hill-climbing algorithm which never makes a move towards a lower value guaranteed to be incomplete because it can get stuck on a local maximum	The chances of getting caught of a local maximum are lowered.
(5)	One of the widely discussed example of Hill climbing algorithm is Travelling-salesman problem in which we minimise the distance travelled by the salesman.	Simulated annealing as a computational process is patterned after the physical process of annealing.
(6)	It is a greedy local search as it only looks to its good immediate neighbour state and not beyond that.	Physical substances such as metals are melted (i.e. raised to high energy levels) and then gradually cooled until some solid state is reached.

Scanned with CamScanner

3.32.4 Difference between Undirectional and Bidirectional Search

Sr. No.	Unidirectional Search Method	Bidirectional Search Method
1.	It uses search tree, start node, goal node as input for starting search.	It has additional information about search trees nodes, along with the start and goal node.
2.	They use only information from the problem definition.	They incorporate additional measures of a potential of a specific state to reach goal.
3.	Sometimes these methods use past exploration.	All these methods use a potential of a state to reach a goal is measured through heuristic functions.
4.	All these techniques are based on pattern of exploration of the nodes in the search tree.	All these techniques totally depend on the evaluated value of each node generated by heuristic function.
5.	In real time problems these search techniques can be costly with respect to time and space.	In real time problmes these search techniques are cost effective with respect to time and space.
6.	Comparatively more number of nodes will be explored in these methods.	As compared to uniformed techniques less number of nodes are explored in this case.
7.	**Example :** Breadth Frist Search, Depth First search, Uniform Cost search, Depth Limited search, Iterative Depending DFS.	**Example :** Hill Climbing search, Best First search, A* search, IDA search, SMA* search

Module 3

3.32.5 Difference Between BFS and DFS

UQ. Compare Breadth First Search (BFS), Depth first search (DFS). (MU - Q. 4(a), Dec. 18, 10 Marks)

Parameter	BFS	DFS
Definition	It is an abbreviation for Breadth First Search.	It is an abbreviation for Depth First Search.
Structure of Data	It uses the Queue Data Structure for finding the shortest path.	It uses the Stack Data Structure for finding the shortest path.
Speed	It is comparatively slower than the DFS method.	It is comparatively faster than the BFS method.
Distance from Source	BFS acts better when the target stays closer to the source.	DFS acts better when the target stays farther from the source.
Suitability for Decision Tree	Since BFS considers all of its neighbors, it is not very suitable for the decision trees used in puzzle games.	DFS is comparatively much more suitable for the decision trees. Within a decision, it allows a user to traverse further to augment that decision. If you reach a conclusion, you reach the win situation, and you stop.
Type of List	BFS operates using the FIFO list.	DFS operates using the LIFO list.

Scanned with CamScanner

Parameter	BFS	DFS
Tracking Method	BFS uses the queue to keep track of the next location that it should visit.	DFS uses the stack to keep track of the next location that it should visit.
Type of Solution	It ensures a shallow path solution. BFS directly finds the shortest path that leads to a solution.	It does not ensure a shallow path solution. DFS first heads to the bottom of any subtree. Then, it backtracks.
Tree Path	It traverses a path according to the tree level.	It traverses a path according to the tree depth.
Backtracking	You don't need to backtrack in BFS.	You need to follow a backtrack in DFS.
Loop Trapping	A user can happen to get trapped in a finite loop.	Any user can possibly get trapped in an infinite loop.
In Case of No Goal	If a user doesn't find a goal, they may have to expand various nodes before they find the solution.	If a user does not find any goal, it may lead to the leaf node backtracking.
Reaching Destination Vertex	You can use BFS to find a single source of the shortest path in the case of an unweighted graph. It is because, in BFS, one can easily reach the vertex with a minimum number of edges from the source vertex.	You might need to traverse through more edges to find and reach the destination vertex from your source.
Suitable Vertices	BFS works better when a user searches for the vertices that stay closer to any given source.	DFS works better when a user can find the solutions away from any given source.
Memory	The amount of memory required for BFS is more than that of DFS.	The amount of memory required for DFS is less than that of BFS.
Complexity of Time	The time complexity of BFS is O (V+E) when a user deploys the Adjacency List and O(V^2) when the user deploys Adjacency Matrix. Here, E refers to the edges, and V refers to the vertices.	The time complexity of BFS is also equivalent to O (V+E) when a user deploys the Adjacency List and O (V^2) when the user deploys Adjacency Matrix, in which E refers to the edges and V refers to the vertices.

Chapter Ends...

❑❑❑

Scanned with CamScanner

Module 4

CHAPTER 4

Knowledge and Reasoning

Syllabus

Knowledge based Agents, Brief Overview of propositional logic, First Order Logic: Syntax and Semantic, Inference in FOL, Forward chaining, backward Chaining. Knowledge Engineering in First-Order Logic, Unification, Resolution

Uncertain Knowledge and Reasoning: Uncertainty, Representing knowledge in an uncertain domain, The semantics of belief network, Simple Inference in belief network

4.1 Knowledge and Reasoning 4-4
4.1.1 Knowledge Progression 4-4
4.1.2 Types of Knowledge 4-4
4.1.3 Knowledge Agent 4-4
4.1.4 Levels of Knowledge Representation 4-5
4.1.5 Various Levels of Knowledge-based Agent 4-5
4.1.6 Knowledge Level 4-5
4.1.7 Logical Level 4-6
4.1.8 Implementation Level 4-6
4.1.9 Representing Knowledge Methods 4-6
4.1.10 Acquisition of Knowledge 4-7
4.1.11 Significance of Knowledge Representation 4-7
4.1.12 Characteristics Of Knowledge Representation 4-8
4.1.12(A) Knowledge Representation Schemes 4-8
4.1.13 Properties of Knowledge Representation System 4-8
4.1.14 Procedural and Declarative Knowledge 4-8
4.1.15 Difference between Procedural and Declarative Knowledge 4-9
4.2 Propositional Logic (First Order Logic) 4-9
4.2.1 Introduction to Logic 4-9
4.2.2 Logic Language 4-10
4.2.3 Propositions and Logical Operations 4-10
4.2.4 Compound Propositions 4-10
4.3 Basic Operations 4-10
4.4 Propositions and truth-Tables 4-11
4.4.1 Method of constructing Truth-table of the Proposition 4-11
4.4.2 Examples Based on the Proposition 4-12
UEx. 4.4.3 (MU - Q. 5(b), May 19, 10 Marks) 4-12
4.5 Tautologies and Contradictions 4-13
4.5.1 Examples on Tautology 4-13
4.5.2 Theorems 4-13
4.6 Conditional Connectives or Implication 4-13
4.6.1 Examples 4-14
4.6.2 Conditional Statements and Variations 4-14
4.6.3 Advantage and Disadvantages of Propositional Logic 4-14
4.6.4 Theorem of Contra-Positive of the Statements 4-15
4.6.5 Biconditional : $p \leftrightarrow q$ 4-15
4.7 Arguments 4-15
4.7.1 Theorem on Tautology 4-16
4.7.2 Fundamental Principle of Logical Reasoning 4-16

Scanned with CamScanner

4.7.3 Verification of Law of Syllogism 4-16
4.8 Precedence Rule 4-18
4.9 Duality Law 4-18
4.9.1 Illusrative Example based on Duals 4-18
4.9.2 Logical Identities 4-18
4.9.3 Examples on Logical Equivalency 4-18
4.10 Normal Forms 4-19
4.10.1 Disjunctive Normal Form 4-19
4.10.2 Examples of DNF 4-19
4.10.3 Conjunctive Normal Form (cnf) 4-19
4.10.3(A) Conversion from PL to CNF 4-20
UQ. Explain the steps involved in converting the propositional logic statement into CNF with a suitable example
(MU - Q. 6(a), May 2016, 10 Marks) 4-20
UQ. Convert the following propositional logic statement into CNF. $A \rightarrow (B \leftrightarrow C)$
(MU- Q. 1(e), Dec. 17, 5 Marks) 4-20
4.10.4 Examples on Conjunctive Normal Form . 4-20
4.11 Truth Table Method to Find DNF 4-21
4.11.1 Examples on dnf 4-21
4.12 Predicate Logic (First Order Logic) 4-21
4.12.1 Predicates 4-21
4.13 The Universal Quantifier 4-22
4.13.1 Existential Quantifiers 4-22
4.14 Examples of quantification 4-22
4.14.1 Free and Bound Variables 4-23
4.14.2 Logical Equivalences Involving Quantifiers 4-23
4.14.3 Negating Quantified Expressions 4-24
4.14.4 Negating an Existential Qualification 4-24
4.14.5 De Morgan;s Laws for Negations for Quantifiers 4-24
4.14.6 Different inference Rules for FOPL 4-24
UQ. Explain different inference Rules for FOPL.
(MU - Q. 4(b), May 18, 10 Marks) 4-24
4.14.7 Examples 4-26
UEx. 4.14.9 : (MU- Q. 2(b), 2016, 5 Marks) 4-26
4.14.8 FOPL 4-26
UEx. 4.14.10 : (MU- Q. 1(c), May 2018, 6 Marks) 4-26
4.14.9 Comparison between Propositional Logic or First Order Logic (Predicate Logic) 4-27
UQ. Distinguish between Propositional Logic (PL) and first order predicate logic (FOPL) knowledge representation mechanisms. Take suitable example for each point of differentiation.
(MU - Q. 2(b), May 19, 10 Marks) 4-27
4.15 Forward Chaining and Backward Chaining 4-28
UQ. Explain Forward-chaining and Backward-Chaining algorithm with the help of example.
(MU – Q. 1(a), May 17, Q. 6(C), Dec. 17, Q. 6(a), Dec. 18, 6(a), May 19, Q. 5(a), Dec. 19) 4-28
4.15.1 Forward Chaining 4-28
UQ. Illustrate Forward chaining in propositional logic with example.
(MU - Q. 3(b), May 17, 10 Marks) 4-28
4.15.2 Backward Chaining 4-28
UQ. Illustrate Forward chaining and backward chaining in propositional logic with example
(MU - Q. 3(b), May 17, 10 Marks) 4-28
4.15.3 Forward Reasoning 4-29
4.15.4 Modus Ponens 4-29
UQ. Explain modus ponen with suitable example.
(MU - Q. 1(C), Dec. 17, Q. 1(b), May 16, 4 Marks) 4-29
4.15.5 Forward Chaining Proof 4-29
4.15.6 Backward Chaining 4-30
4.15.7 Properties of Backward Reasoning (Chaining) 4-30
4.15.8 Backward Chaining Proof 4-30
4.15.9 Comparison between Forward and Backward Reasoning 4-31
4.16 Example to Compare Forward and Backward Chaining 4-32
4.17 Difference between Forward and Backward Chaining 4-34
4.18 Semantic Networks 4-35
4.18.1 Advantages, Disadvantages of Semantic Nets 4-35
4.18.2 Examples for Semantic Representation . 4-35
4.19 resolution 4-37
4.19.1 Resolution and Unification 4-37
4.19.2 Resolution Algorithm 4-37
4.19.3 University Solved Examples 4-38

Scanned with CamScanner

UEx. 4.19.2 (MU - Q. 1(c), Dec. 19, 5 Marks) 4-38
UEx. 4.19.3 (MU - Q. 3(b), Dec. 15, 10 Marks) ... 4-39
UEx. 4.19.4 (MU - Q. 4(a), Dec. 18, 10 Marks, Q. 4(A), Dec. 17, 10 Marks) ... 4-39
UEx. 4.19.5 (MU - Q. 4(a), May 16, 10 Marks, Q. 4(a), May 19, 10 Marks) 4-40
UEx. 4.19.6 (MU : Dec. 15, 12 Marks) 4-40
4.19.4 Unification 4-41
4.19.5 Conflict Resolution 4-42
4.19.6 Refutation 4-42
UQ. Explain Resolution by Refutation with suitable example. **(MU - Q. 3(a), May 18, 10 Marks)** 4-42
4.19.7 Example based on Propositional Variables 4-43
4.20 Uncertain Knowledge and Reasoning 4-45
4.20.1 Non-monotonic Logics 4-45
4.20.2 Probabilistic Reasoning 4-45
4.20.3 Bayesian Networks 4-46
4.20.4 Challenges to Probabilistic Approaches . 4-46
4.20.5 Fuzzy Logic 4-46
4.20.6 Basic Fuzzy Operators 4-46
4.20.7 Challenges to Fuzzy Approaches 4-46
4.20.8 Causes of Uncertainty in AI 4-46
4.21 Representing Knowledge in an Uncertain Domain 4-46
4.21.1 Probability Theory 4-46
4.21.2 Fuzzy Logic 4-46
4.21.3 Truth Maintenance System : (TMS) 4-46
4.22 Representation of Knowledge 4-47
4.22.1 Categories of Knowledge 4-47
4.22.2 Types of Knowledge Representation 4-47
4.22.3 Causes of Uncertainty 4-48
4.23 The semantics of belief network 4-48
UQ. Define Belief Network. Describe the steps of constructing belief network with an example. What types of inferences can be drawn from that ? **(MU - Q. 4(b), May 19, 10 Marks)** 4-48
4.23.1 Basics of Semantic Networks 4-49
4.23.2 Examples of Semantic Network 4-49
4.23.3 Purpose of Semantics 4-49
4.23.4 Focus on Semantics 4-49
4.24 Inference in belief network 4-49
4.24.1 Purpose of Belief Network 4-49
4.24.2 Method of Belief Network 4-50
4.24.3 PPTC Algorithm 4-50
- **Chapter End** 4-50

Module 4

Scanned with CamScanner

4.1 KNOWLEDGE AND REASONING

- Search-based problem-solving programs require some knowledge to be implemented. Knowledge can be a particular **state or path** toward solution, rules, etc.
- In order to use this knowledge, it must be represented in a particular way with a certain format. Knowledge Representation (KR) is an important issue in computer science, in general and in AI in particular. "The **dominant paradigm** for building intelligent systems since the early 1970s has been based on the premise that intelligence presupposes knowledge".
- Generally, knowledge is represented in the **system's knowledge base**, which consists of **data structures** and **programs**.

4.1.1 Knowledge Progression

GQ. What is knowledge ?

- **Definition :** Knowledge is a progression that starts with data which is of limited utility. By organizing or analyzing the data, we understand what the data means, and this becomes **information.**

The interpretation or evaluation of information yields knowledge. An understanding of the principles embodied within the knowledge is **wisdom**.

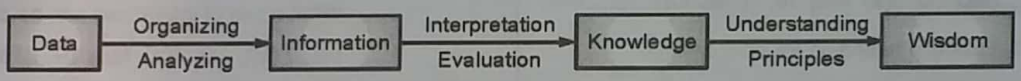


Fig. 4.1.1 : Knowledge Progression

1. **Data:** is viewed as collection of disconnected facts.
   Example: It is raining.
2. **Information** emerges when relationships among facts are established and understood; Provides answers to **"who", "what", "where"**, and **"when"**.
   Example: The temperature dropped 15 degrees and then it started raining.
3. **Knowledge** emerges when relationships among patterns are identified and understood; provides answers as **"how"**.
   Example: If the humidity is very high and the temperature drops substantially, then an atmosphere is unlikely to hold the moisture, so it rains.
4. **Wisdom:** is the pinnacle of understanding, uncovers the principles of relationships that describe patterns. Provides answers as **"why"**.
   Example: Encompasses understanding of all the interactions that happen between raining, evaporation, air currents, temperature gradients and changes.

4.1.2 Types of Knowledge

1. Procedural Knowledge
2. Declarative knowledge

- **1. Procedural knowledge :** is a **compiled knowledge** related to the performance of some task. For example, the steps used to solve an algebraic equation are expressed as procedural knowledge.
- **2. Declarative knowledge :** on the other hand, is passive knowledge expressed as statements of facts about the world. Personnel data in a datable is typical of declarative knowledge.

Heuristic knowledge

There is one more special type of knowledge frequently used by humans to solve complex problems, it is the heuristic knowledge. Heuristics are the knowledge used to make good judgments or strategies, tricks or rules of thumb used to simplify the solution of problems.

For example : In locating a fault in a TV set, an experienced technician will not start by making numerous but instead will immediately reason that the high voltage fly back transformer or related component is the culprit and then it leads to a quick solution.

4.1.3 Knowledge Agent

GQ. What is knowledge agent ?

In artificial intelligence, a knowledge agent is **autonomous entity**, which observes through sensors and acts upon an environment using actuators (i.e. it is an agent) and direct its activities towards achieving their goals. It may also learn or use knowledge to achieve their goals.

Scanned with CamScanner

4.1.4 Levels of Knowledge Representation

GQ. Write a note on "knowledge representation". Or what are the different levels of knowledge representations. **Or** What are the methods of knowledge representation, name them.

- Knowledge consists of facts, concepts, rules, and so on. It can be represented in different forms, as mental images in one's thoughts, as spoken or written words in some language, as graphical or other pictures, and as character strings or collections of magnetic spots stored in a computer.

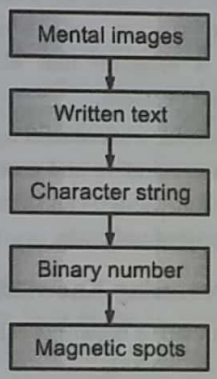


Fig. 4.1.2 : Different levels of knowledge representation

(1) Any choice of representation will depend on the type of problem to be solved and make use of inference methods available. For example, suppose we wish to write a program to play a simple card game using the standard deck of **52** playing cards.

(2) We will need some way to represent the cards dealt to each player and a way to express the rules. We can represent cards in different ways.

(3) The most straightforward way is to record the suit (**clubs, diamonds, hearts, spades**) and face values (ace, 2, 3,....., 10, jack, queen, king) as a symbolic pair. So the queen of hearts might be represented as **<queen, hearts>** Alternatively, we could assign abbreviated codes (c6 for the 6 of clubs), numeric values which ignore suit (1, 2, ..., 13), or some other scheme.

(4) Consider the problem of discovering a pattern in the sequence of numbers **1 1 2 3 4 7**. A change of base in the number from 10 to 2 transforms the number to 01 **101 101 101 101 101 1**.

4.1.5 Various Levels of Knowledge-based Agent

Knowledge-based agent can be described at three different levels : These are :

1. Knowledge level;
2. Logical level;
3. Implementation level.

4.1.6 Knowledge Level

(a) Knowledge-based agents are those agents who have the capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions. These agents can represent the world with some formal representation and act intelligently.

(b) Knowledge-based agents are composed of two main parts :

(I) Knowledge-base and

(II) Inference system.

A knowledge-based agent must be able to do the following :

(i) An agent should be able to do represent states, actions, etc.

(ii) An agent should be able to incorporate new percepts.

(iii) An agent can update the internal representation of the world.

(iv) An agent can deduce the internal representation of the world.

(v) An agent can deduce appropriate actions.

The architecture of knowledge-based agent is shown in Fig. 4.1.3.

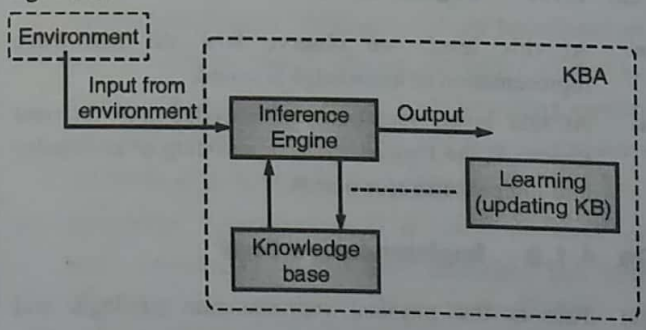


Fig. 4.1.3 : Architecture of knowledge-based agent

The Fig. 4.1.3 is representing a generalised architecture for a knowledge-based agent (KBA).

Module 4

Scanned with CamScanner

KBA takes input from the environment by perceiving the environment.

The input is taken by the inference engine of the agent and it communicates with KB. The learning element of KBA regularly updates the KB by learning new knowledge.

(I) Knowledge-base

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

(II) Inference system

- Inference means deriving new sentences from old.
- Inference system allows us to add a new sentence to the knowledge base.
- Inference system applies logical rules to KB to deduce new information.
- Inference system generates new facts so that an agent can update KB.
- An inference system works mainly in two rules which are given as :
  (i) Forward chaining (ii) Backward chaining

Operations performed by KBA

Following are **three operations** which are performed by KBA in order to show the intelligent behaviour :

1. **Tell :** This operation tells the knowledge base what it perceives from the environment.
2. **Ask :** This operation asks the knowledge base what action it should perform.
3. **Perform :** It performs the selected action.

4.1.7 Logical Level

- At this level, we observe 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.

4.1.8 Implementation Level

(1) This is the physical representation of logic and knowledge. At the implementation level agent performs actions as per logical and knowledge level.

(2) At this level, an automated taxi agent actually implement his knowledge and logic so that he can reach to the destination.

Approaches to designing a knowledge-based agent are as follows

(I) Declarative approach

(1) We can create a knowledge-based agent by initializing with an empty knowledge base and telling the agent all the sentences with which we want to start with.

(2) This approach is called Declarative approach.

(II) Procedural approach

(1) In the procedural approach, we directly encode desired behaviour as a program code which means we just need to write a program that already encodes the desired behaviour or agent.

(2) But in the real world, a successful agent can be built by combining both declarative and procedural approaches, and declarative knowledge can often be compiled into more efficient procedural code.

4.1.9 Representing Knowledge Methods

1. Frames and associative networks (also called semantic and conceptual networks),
2. Fuzzy logic,
3. Model logics and
4. Object- oriented methods.

Clearly a representation in the proper base greatly simplifies finding the pattern solution. A typical statement in this logic might express the family relationship of fatherhood as **FATHER (john, Jim)** where the predicate father is used to express the fact that John is the father of Jim.

Other representation schemes include frames and associative networks (also called **semantic** and **conceptual networks**), **fuzzy logic**, **model logics** and **object- oriented methods**.

(1) **Frames** are flexible structures that permit the grouping closely related knowledge. For example, an object such as a ball and its properties (**size color, function**) and its relationship to other objects, (to the left of on top of, and so on) are grouped together into a single structure for easy access.

(2) **Networks** also permit easy access to groups of related items. They associate objects and linkages TO show their relationship to other objects.

Scanned with CamScanner

(3) **Fuzzy logic** is a generalization of predicate logic developed to permit varying degrees of some property such as tall. In classical two-valued logic, TALL (john) is either true of false, but in **fuzzy logic** this statement may be partially true.

(4) **Modal logic** is an extension of classical logic. It was also developed to better represent common sense reasoning by permitting condition such as **likely or possible.**

(5) **Object oriented representations** package an object together with its attributes and functions, therefore chiding these facts. Operations are performed by sending messages between the objects.

4.1.10 Acquisition of Knowledge

GQ. Write a short note on "knowledge acquisition".

(1) Decisions and actions in knowledge-based systems come from acquisition of the knowledge in specified ways. Some form of input will initiate a search for a goal or decision.

(2) For this known facts in the knowledge base be located, compared, and if necessary altered in some way. This process may set up other subgoals and require further inputs, till a final solution is found. The acquisitions are the computational equivalent of reasoning. This requires a form of inference or deduction, using the knowledge and inferring rules.

(3) All forms of reasoning require a certain amount of searching and matching. These two operations require a lot of computation time in AI systems. In a way, it gives a set-back to the acquisition of knowledge.

(4) One of the greatest bottlenecks in building knowledge-rich systems is the acquisition and validation of the knowledge. Knowledge can come from various sources, such as **experts, textbooks, reports, technical articles**, and the like. Reading research articles, taking college courses, consulting with expert colleagues, and using clinical databases are examples of knowledge acquisition.

(5) Each of these activities provides a way to acquire new knowledge through reading, observation and engaging in life-long learning activities.

(6) To be useful, the knowledge must be accurate, presented at the right level for encoding, in complete sense that all essential facts and rules are included, free of inconsistencies, and so on.

(7) Eliciting facts, heuristics, procedures, and rules from an expert is a slow, and tedious process. Experience in building dozens of expert systems and other knowledge- based systems over the past fifteen years has shown this to be the single most time-consuming and costly part of the building process.

(8) This has led to the development of some sophisticated acquisition tools, including a variety of intelligent editors; editors which provide much assistance to the knowledge engineers and system users.

(9) The acquisition problem has also stimulated much research in machine learning systems, that is; systems which can learn new knowledge autonomously without the aid of humans.

(10) Since knowledge-based systems depend on large quantities of high quality knowledge. for their success, it is essential that better methods of acquisition, refinement, and validation be developed.

(11) The ultimate goal is to develop techniques that permit systems to learn new knowledge autonomously and continually improve the quality of the knowledge they possess.

Module 4

4.1.11 Significance of Knowledge Representation

GQ. What is the significance of knowledge representation ?

(1) The Oxford English Dictionary defines knowledge as intellectual acquaintance with, or perception of, fact or truth. A representation is a way of describing certain fragments or information so that any reasoning system can easily adopt it for inference purposes.

(2) Knowledge representation is a study of ways of how knowledge is actually picturized and how effectively it resembles the representation of knowledge in human brain.

Scanned with CamScanner

4.1.12 Characteristics Of Knowledge Representation

GQ. What are the characteristics of knowledge representation.

A knowledge representation system should provide ways of representing complex knowledge and should possess the following **characteristics** :

1. The representation scheme should have a set of well-defined syntax and semantics. This will help in representing various kinds of knowledge.
2. The knowledge representation scheme should have a good expressive capacity, a good expressive capability will catalyze the inference mechanism in its reasoning process.
3. From computer system point of view, the representation must be efficient. By this we mean that it should use only limited resources without compromising on the expressive power.

4.1.12(A)Knowledge Representation Schemes

Various Knowledge Representation Schemes are as follows :

(i) Semantic networks
(ii) Frames
(iii) Conceptual dependency
(iv) Scripts

4.1.13 Properties of Knowledge Representation System

The following properties should be possessed by a knowledge representation system :

(1) Representational Adequacy : The ability to represent the required knowledge.

(2) Inferential Adequacy : The ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original.

(3) Inferential Efficiency : The ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides.

(4) Acquisitional Efficiency : The ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

4.1.14 Procedural and Declarative Knowledge

(I) Declarative

A declarative representation is one in which knowledge is specified, but the use of that knowledge is not given. A declarative representation, we must augment it with a program that specifies what is to be done to the knowledge and how.

For example : a set of logical assertions can be combined with a resolution theorem prover to give a complete program for solving problems. There is a different way, though, in which logical assertions can be viewed, namely as a program, rather than data to a program. In this view, the implication statements define the legitimate reasoning paths and the atomic assertions provide the starting points (or, if we reason backward, the ending points) of those paths.

(II) Procedural

- A procedural representation is one in which the control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself. To use procedural representation, we need to augment it with an interpreter that follows the instructions given in the knowledge.
- Screening logical assertions as code is not a very essential idea, given that a programs are really data to other programs that interpret (or compile) and execute them. The real difference between the declarative and the procedural views of knowledge lies in where control information resides. For example, consider the knowledge base :
- man (Marcus) ; man (Caesar) ; person(Cleopatra)

  $\forall a$: man $(a) \rightarrow$ person(a)

  Now consider trying to extract from this knowledge base the answer to the question $\exists$: person(z)
- We want to bind y to a particular value for which person is true. Our knowledge base justifies any of the following answers :

  z = Marcus , z = Casear, z = Cleopatra
- For the reason that there is more than one value that satisfies the predicate, but only one value is needed, the answer to the question will depend on the order in which the assertions are examined during the search for response.

Scanned with CamScanner

4.1.15 Difference between Procedural and Declarative Knowledge

Table 4.1.1

Sr. No.	Procedural Knowledge	Declarative Knowledge
1.	Follow black box.	Follow the white box.
2.	Based on process orientation.	Based on data orientation.
3.	Possible to faster usage.	Based on knowledge in the process of system design.
4.	When we have to achieve in a particular result.	According knowledge format that may be manipulated and analyzed.
5.	Knowing how to do something.	It is knowledge about something.
6.	Simple data type can be used.	Large data type can be used.
7.	Followed in C++ and Cobol.	Followed by SQL.
8.	According and programming any simple task.	According SQL any simple task in one line code i.e. one line SQL statement.
9.	Programmer should understand execution.	Programmer should not interact with SQL.
10.	Initially faster but later it can be slow.	Initially slower but possibly faster later .
11.	Work on interpreter of language.	Work on data engine with the DBMS.
12.	Example : If Manmohan or Monu is older.	Example : Manmohan is older than monu.

4.2 PROPOSITIONAL LOGIC (FIRST ORDER LOGIC)

4.2.1 Introduction to Logic

Logic is the discipline that deals with the **methods of reasoning**. On an elementary level, logic provides rules and techniques for determining whether a given argument is valid.

Commonsense logic

It is deriving conclusions from personal experience or knowledge. A conclusion that something makes sense, so it is right or something doesn't make sense, so it is wrong.

Let us consider one classic example: A bear walked one k.m. due south, then turned to the left and walked one k.m. due East. Then it turned to the left again and walked one k.m. due North and arrived back at its starting point.

What was the colour of the bear? Now, actually the bear walked **3 sides** of a **square,** Like this

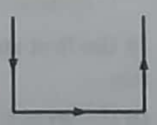


Fig. 4.2.1

But since it ended up where it started from, its actual path must have been a triangle :

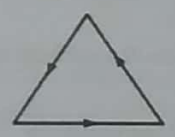


Fig. 4.2.2

The only two places in the world where this could happen is either the **North-pole or South-pole**. The south-pole is not possible, since it is impossible to travel south from south (pole). So the bear was at North pole, i.e. it was a polar bear. So the bear was **white.**

This problem is solved by knowledge, i.e., it requires a logical type of mind to apply that knowledge to a particular problem. It can also be called as **inductive logic,** which does not permit arguments **outside** the facts available.

Logical reasoning

It is used in mathematics to prove theorems; in computer science to verify the correctness of programs and to prove theorems. And in our everyday lives to solve a multitude of problems.

Module 4

Scanned with CamScanner

4.2.2 Logic Language

- One of the basic difficulties in developing an approach to logic is the limitation of ordinary language when it comes to presenting statements and conclusions. [Exactly the same problem arises with computers. You cannot instruct computers in ordinary language-it has to be consistent with the input/output capabilities of the computer].
- Our aim is now very simple- to give each statement an exact meaning and manipulate such statement in a logical manner, determined by the rules and theorems. Here, we discuss a few of the basic ideas; i.e. rules and theorem.

4.2.3 Propositions and Logical Operations

❑ **Definition :** A statement or proposition is a declarative sentence that is either true or false, **but not both.**

Illustrative Ex. 4.2.1 :

(i) The earth is round

(ii) $3 + 4 = 7$

(iii) $4 + x = 9$

(iv) Do you speak Gujarathi ?

(v) Take two aspirins.

(vi) The temperature on the surface of the planet mars is 500°F.

(vii) The sun will come out tomorrow

☑ **Soln. :**

(i) and (ii) are statements which are true.

(iii) it is not a statement, since it is true or false depends on the value of x. But we can say that it is a declarative sentence.

If we put $x = 5$, it becomes a true statement, if we take value of $x \neq 5$, it becomes a false. Such statements are open statements.

Thus if a mathematical statement is **neither true nor false**, it is called **open statement.**

(iv) It is a question, not a statement.

(v) It is a command, but not a statement

(vi) It is a statement, because in principle we can determine if it is true or false.

(vii) It is a statement, since it is true or false but not both.

4.2.4 Compound Propositions

Propositions composed of **sub propositions** are called **compound propositions.** A proposition is said to be **primitive** if it cannot be broken down into simpler propositions; that is, if it is not composite.

Compound propositions or statements are composed of various **logical connectives.**

☞ **Examples of compound propositions**

(i) 'Roses are red and violets are blue' is a compound statement with sub statements: " Roses are red" and "violets are blue".

(ii) "John is intelligent and studies every night" is a compound statement with sub propositions :

"John is intelligent" and" John studies every night.

4.3 BASIC OPERATIONS

(I) Conjunction ($p \wedge q$)

(II) Disjunction ($p \vee q$)

(III) Negation ($\neg$)

(I) $p \wedge q$ conjunction of p and q, read "p and q"

Two propositions p and q can be combined by the word **'and'** to form a compound proposition and called as **conjunction** of the original propositions.

Symbolically, $p \wedge q$ is a compound proposition. And it has a TRUE value.

❑ **Definition :** If p and q are true, then $p \wedge q$ is also true, otherwise $p \wedge q$ is false.

We prepare the table for the truth-value of $p \wedge q$.

➡ **Note :** T stands for true and F stands for false.

Table 4.3.1 : Truth table for Conjunction

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

(i) **In the first row**; if p is true and q is true then $p \wedge q$ is true.

(ii) **In the second row** : if p is true and q is false then $p \wedge q$ is false. And so on.

Scanned with CamScanner

Remark : $p \wedge q$ is true **only when** p and q both are true.

☞ Examples based on conjunction

Form the conjunction of p and q for the following

(i) $p : 3 < 5$ and $q : -2 > -4$

$\therefore$ $p \wedge q : 3 < 5$ and $-2 > -4$

(ii) p : it is hot, q : $2 < 4$

$\therefore$ $p \wedge q$ it is hot and $2 < 4$.

(iii) p : it is snowing, q : I am cold

$\therefore$ $p \wedge q$: it is snowing and I am cold.

☞ Remarks on the above examples

Example : (ii) Shows that we may join two totally unrelated statements by the connective 'and ($\wedge$)'

(II) Disjunction

(i) If p and q are statements, the **disjunction** of p and q is the compound statement "**p or q**" denoted by $p \vee q$.

(ii) The connective **'or'** is denoted by the symbol $\vee$. The compound statement $p \vee q$ is true if **at least one of p or q is true**; it is **false** when both **p and q are false**.

The truth table of $p \vee q$:

Table 4.3.2 : Truth table of $p \vee q$

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

(iii) Negation : ($\neg$)

If p be any statement then negation of p is denoted by $\neg p$ and is read as 'not p'.

If p is true then $\neg p$ is false. If p is false then $\neg p$ is true.

☞ Truth table for Negation

Table 4.3.3 : Truth table for Negation

p	$\neg p$
T	F
F	T

☞ Example based on Negation

If **p** : Gopal is good at sports

then $\neg$ **p** : Gopal is not good at sports.

Remark : Negation is also denoted by $\sim$. Thus if p is true, then $\sim$ p is false

4.4 PROPOSITIONS AND TRUTH-TABLES

Proposition : A proposition is also called a well-formed formula of logical variables **p, q, r,** .... and logical connectives ($\wedge, \vee, \neg$) We denote such a proposition by **P (p, q, r, ...)**.

Truth-table : The truth-value of a proposition depends upon the truth values of its variables. (Thus the truth value of a proposition is known once the truth values of its variables are known). This relationship can be shown through a **truth-table.**

4.4.1 Method of constructing Truth-table of the Proposition

- **Step (i) :** The first columns of the table are for the variables **p, q,**
- **Step (ii) :** Allow the rows for all possible combination of T and F for these variables. (For 2 variables, $2^2 = 4$ rows are necessary; for 3 variables, $2^3 = 8$ rows are necessary, and, in general, for n variables, 2^n rows are required.
- **Step (iii) :** There is a column for each "**elementary**" stage of the constructions for the truth-value of the proposition
- **Step (iv) :** The truth value of each step being determined from the previous stages by the definitions of the connectives $\wedge, \vee, \neg$
- **Step (v) :** Finally, in the last column, we obtain the truth value of the proposition.

Module 4

Scanned with CamScanner

4.4.2 Examples Based on the Proposition

Ex. 4.4.1

(i) Find the truth-table of the proposition $\neg (p \wedge \neg q)$

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

(a)

=

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

(b)

Remark

The truth table of the preposition consists of the columns under the variables and the column under the proposition, as shown in (b).

Ex. 4.4.2 : Find the truth-table of $\neg p \wedge q$.

Soln. : We construct the table :

p	q	$\neg p$	$\neg p \wedge q$
T	T	F	F
T	F	F	F
F	T	T	T
F	F	T	F

(i) For two variables p, q we choose the truth-values in the first two columns as shown.

(ii) We find the truth value of $\neg p$ using the negation $\neg$.

(iii) We find the truth value of $\neg p \wedge q$ using the conjunction $\wedge$.

UEx. 4.4.3 (MU - Q. 5(b), May 19, 10 Marks)

1. John likes all kinds of food.
   $\forall x$: food(x) $\rightarrow$ likes (john, x)
2. Apples are food.
   food(apple)
3. Chicken is food
   food(chicken)
4. Anything anyone eats are isn't kiled by is food
   $\forall x$: ($\exists y$:eats(y, x) $\wedge \neg$ killed by (y, x)) $\rightarrow$ food(x)
5. Bill eats peanuts and is still alive.
   A. eats (Bill, peanuts) B. alive (Bill)
6. Sue eats everything Bill eats
   $\forall x$: eats(Bill, x) $\rightarrow$ eats (Sue, x)
7. $\forall x$: $\forall y$:alive(x) $\rightarrow \neg$ killedby(x, y)

Soln. :

1. John likes all kind of food.
2. Apples and chicken are food.
3. Anything anyone eats is not killed by food.
4. Bill eats peanuts and is still alive.
5. Sue eats everything that Bill eats.

Step I : Converting the given statements into FOPL :

1. $\forall x$: food (x) $\rightarrow$ likes (John, x)
2. food (apple) $\wedge$ food (chicken)
3. $\forall a, \forall b$: eats (a, b) $\wedge \neg$ killed (a) $\rightarrow$ food (b)
4. eats (Bill, peanuts) $\wedge \neg$ killed (Bill)
5. $\forall y$: eats (Bill, y) $\rightarrow$ eats (Sue, y)

Step II : Converting FOPL statements into causal form

1. $\neg$ food (x) $\vee$ likes (John, x)
2. food (apple),
3. food (chicken)
4. $\neg$ eats (a, b) $\vee$ killed (a) $\vee$ food (b)
5. eats (Bill, peanuts)
6. $\neg$ e killed (Bill)
7. eats (Bill, y) $\vee$ eats (Sue, y)

Conclusion : Likes (John, Peanuts)

Step III : Resolution performance

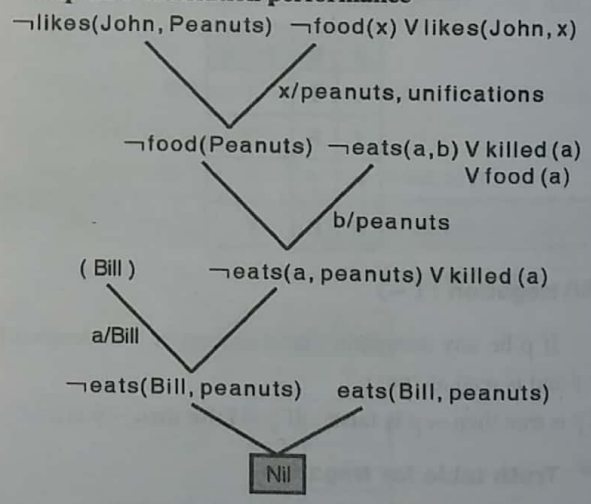


Fig. Ex. 4.4.3

Scanned with CamScanner

4.5 TAUTOLOGIES AND CONTRADICTIONS

- **Definition 1 :** A proposition P (p, q, ...) is a tautology if it contains only T in the last column of its truth table, i.e. if P is true for any truth values of its variables.
- **Definition 2 :** A proposition P (p, q, ...) is a **contradiction** if it contains only 'F' in the last column of its truth table, i.e., it P is false for any truth values of its variables.

For example (i) "p or not p", i.e., $p \vee \sim p$ is tautology

Table 4.5.1 : Truth table of "p or not p is tautology"

P	~ p	p ∨ ~ p
T	F	T
F	T	T

(ii) '$p \wedge \sim p$' is a contradiction By truth table, '$p \wedge \sim p$' is a contradiction.

Table 4.5.2 : Truth table of "p or not p is contradiction"

P	~ p	p ∧ ~ p
T	F	F
F	T	F

4.5.1 Examples on Tautology

Ex. 4.5.1 : Show that $p \vee \neg (p \wedge q)$ is a tautology.

Soln. :

We prepare the truth-table for $p \vee \neg (p \wedge q)$:

p	q	p ∧ q	¬ (p ∧ q)	p ∨ ¬ (p ∧ q)
T	T	T	F	T
T	F	F	T	T
F	T	F	T	T
F	F	F	T	T

(i) We choose the truth-values for p, q in first two columns.

(ii) Then truth value for $p \wedge q$, (conjunction)

(iii) Negation of $(p \wedge q)$

(iv) Truth value of $p \vee \neg (p \wedge q)$; which is a tautology.

4.5.2 Theorems

Theorem (1)

If **P (p, q, ...)** is a tautology, then **¬ P (p, q, ...)** is a contradiction, and conversely.

Proof

Since a tautology is always true, (i.e., it contains only T in the last column); the negation of a tautology is always false, (i.e. it contains only F in the last column)

∴ **¬ P (p, q, ...)** is a contradiction, and conversely.

Theorem (2) : (Principle of Substitution)

Suppose **P (p, q, ...)** is a tautology, then P (p_1, p_2, ...) is a tautology for any propositions $\mathbf{p_1}$, $\mathbf{p_2}$.

Proof :

Since **P (p, q, ...)** is a tautology. And it does not depend on truth values of its variables p, q, ..., we can substitute P_1, for p, P_2 for q, ... in the given tautology P (p, q, ...) and it becomes a tautology.

Ex. 4.5.2 : Verify that $(p \wedge \neg q) \vee \neg (p \wedge \neg q)$ is a tautology.

Soln. :

Let $P = p \wedge \neg q$, then the proposition becomes $p \vee \neg p$. We have seen that proposition $p \vee \neg p$ is a tautology.

∴ $(p \wedge \neg q) \vee \neg (p \wedge \neg q)$ is a tautology.

Module 4

4.6 CONDITIONAL CONNECTIVES OR IMPLICATION

In mathematics, we come across statements like **"If p then q"**. Such statements are called **conditional statements** and are denoted by $\mathbf{p \rightarrow q}$.

Remark

1. The conditional statement $p \rightarrow q$ is sometimes read as **(i)** p implies q, **(ii)** p is **sufficient** for q **(iii)** q is necessary for p (iv) p only if q.
2. The conditional $p \rightarrow q$ is false when the first **p is true** and the second part **q is false** ; i.e., when **p is false**, the conditional $\mathbf{p \rightarrow q}$ is **true** regardless of the truth **value of q.**
3. (i) If p is true, q is true then $p \rightarrow q$ is true.
   (ii) If p is true, q is false then $p \rightarrow q$ is false.
   (iii) If p is false, q is true, then $p \rightarrow q$ is true.
   (iv) If p is false, q is false, the $p \rightarrow q$ is true.

Scanned with CamScanner

Truth table for $p \to q$

Table 4.6.1 : Truth table for $p \to q$

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

4. We observe that the truth tables of $\neg p \vee q$ and $p \to q$ are identical. (check!); i.e., $p \to q$ is logically equivalent to $\neg p \vee q$.
   i.e., $p \to q \equiv \neg p \vee q$.

4.6.1 Examples

Ex. 4.6.1 : Determine the truth value of the following statements :

(i) If Bombay is in India, then $4 + 5 = 9$.

(ii) If Bombay is in India, then $4 + 5 = 3$.

Soln. :

(i) Let p = Bombay is in India, $q = 4 + 5 = 9$

$\because$ p is true, q is true, $\therefore$ $p \to q$ is true.

(ii) Let p = Bombay is in India,

$$q = 4 + 5 = 3$$

$\because$ p is true, but q is false $\therefore$ $p \to q$ is false.

Ex. 4.6.2 : Rewrite the following statements without using the conditional.

(i) If it is hot, he wears a hat

(ii) If F is field, it is integral domain.

Soln. :

(i) Recall that $p \to q = \sim p \vee q$; i.e.

if 'p then q, is equivalent to 'not p or q'.

$\therefore$ It is not hot or he wears a hat.

(ii) F is not field or F is integral domain.

4.6.2 Conditional Statements and Variations

Now we consider other simple **conditional** propositions containing p and q.

1. Let $\mathbf{p \to q}$ be conditional proposition. Then **'$q \to p$'** is called **converse** conditional proposition.
2. **'$\neg p \to q$'** is called **inverse** of the original conditional proposition $\mathbf{p \to q}$.
3. **'$\neg q \to \neg p$'** is called **contrapositive** of the original conditional proposition $\mathbf{p \to q}$.

Example of conditional proposition

Which, if any, of the above propositions are logically equivalent to $p \to q$.

We construct the truth-table for the above conditional proposition.

Table 4.6.2 : Truth table for conditional proposition

p	q	$\neg p$	$\neg q$	Conditional $p \to q$	Converse $q \to p$	Inverse $\neg p \to \neg q$	Contra positive $\neg q \to \neg p$
T	T	F	F	T	T	T	T
T	F	F	T	F	T	T	F
F	T	T	F	T	F	F	T
F	F	T	T	T	T	T	T

Only the contra positive '$\neg q \to \neg p$' is logically equivalent to the original conditional probability $p \to q$.

4.6.3 Advantage and Disadvantages of Propositional Logic

Advantages of Propositional Logic

1. It is used in artificial intelligence for planning, problem-solving, intelligent control and most importantly for decision-making.
2. It is about Boolean functions and the statements where there are more than just true and false values, includes the certainty as well as uncertainty.
3. It led to the foundation of machine learning models.
4. It is a useful tool for reasoning.

Disadvantages of Propositional Logic

1. We cannot represent relations like All, some, or none with propositional logic.
   Example : All the boys are intelligent.
2. Propositional logic has limited expressive power.
3. In propositional logic, we cannot describe statements in terms of their properties or logical relationships.
4. Propositional logic is lacking of syntax for representing objects in the domain of internet.

Scanned with CamScanner

4.6.4 Theorem of Contra-Positive of the Statements

Contra-positive is a proposition or theorem formed by contradicting both the subject and predicate or both the hypothesis and conclusion of a given proposition or theorem and interchanging them "if not-B then not-A" is the contra-positive of "if A then B".

To form the contra-positive of the conditional statement, we interchange the hypothesis and the conclusion of the inverse statement. The contra-positive of "if it rains, then they cancel school" is " If they do not cancel school, then it does not rain". If the converse is true, then the inverse is also logically true.

A conditional statement $p \rightarrow q$ and its contra positive $\neg q \rightarrow \neg p$ are logically equivalent.

If the statement is true, then the contra-positive is also logically true. If the converse is true, then the inverse is also logically true.

If two angles are congruent, then they have the same measure.

Converse, Inverse, Contrapositive

Statement	If p, then q
Inverse	If not p, then not q
Contrapositive	If not q, then not p.

The contrapositive of a conditional statement of the form "if p then q" is "If $\neg$ q then $\neg$ p".

Symbolically the contrapositive of p q is $\neg q \neg p$.

(i) **Conditional :** The conditional of q by p is "If p then q" or "p implies q" and is denoted by '$p \rightarrow q$'.

(ii) **Biconditional : (iff) :** The biconditional of p and q is "p, if and only if, q" and is denoted by $p \leftrightarrow q$.

(iii) **Only if :** p only if q means "If not q then not p" or equivalently, "if p then q".

(iv) **Sufficient condition :** p is a sufficient condition for q means "if p then q".

Ex. 4.6.3 : Determine the contra positive of the statements.

(i) If Bhide is a teacher, then he is poor.

(ii) If Thombare studies, he will pass the test.

☑ **Soln. :**

(i) The contra positive of $p \rightarrow q$ is $\neg q \rightarrow \neg p$.

$\therefore$ The contra positive of the given statement is

If Bhide is not poor, then he is not a teacher.

(ii) The contra positive is :

If Thombare does not study, then he will not pass the test.

4.6.5 Biconditional : $p \leftrightarrow q$

Another common statement is of the form "p if and only if q". Such statements are called biconditional statements and are denoted by $p \leftrightarrow q$.

☞ **Truth-Table for $p \leftrightarrow q$**

Table 4.6.3 : Truth table for $p \leftrightarrow q$

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

The biconditional $p \leftrightarrow q$ is true whenever p and q have the same truth values and false otherwise.

☞ **Theorem**

The propositions **P (p, q, ...)** and **Q (p, q, ...)** are logically equivalent if and only if the proposition.

P(p, q, ...) $\leftrightarrow$ Q (p, q, ...) is a tautology.

Proof

▶ **Step (I) :** Let P (p, q, ...) = Q (p, q, ...). Then they have the same truth table.

$\therefore$ P(p, q, ...) $\leftrightarrow$ Q (p, q, ...) is true for any values of the variables p, q, ..., It means that the proposition is tautology.

▶ **Step (II) :** Since each step is reversible,

$\therefore$ Converse is also true.

Module 4

4.7 ARGUMENTS

❑ **Definition :** An **argument** is an assertion such that for a given set of propositions $P_1, P_2, ..., P_n$ called **premises**, gives rise to another proposition Q (or a consequence Q), called the **conclusion.** And such an argument is denoted by $P_1, P_2, ..., P_n$ Q

❑ **Definition :** Valid Argument or Logical Argument :

1. An argument $P_1, P_2, ..., P_n$ Q is said to be logical or valid if Q is **true whenever** all the premises $P_1, P_2, ..., P_n$ are **true.**
2. An argument which is not valid is called a **fallacy**.

Scanned with CamScanner

Ex. 4.7.1 : Show that the following argument is valid

$p, p \rightarrow q \vdash q$

(This is called as Law of detachment).

Soln. : We prepare the truth-table for $p \rightarrow q$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

(i) p is true in 1 and 2 lines; $p \rightarrow q$ is true in 1, 3, 4 lines.

(ii) p and $p \rightarrow q$ are true in line 1.

Since in this case q is also true.

∴ The argument is valid.

Remark

For the argument to be valid, it is not necessary **that 'p', '$p \rightarrow q$' and 'q' are true in all the rows.**

Ex. 4.7.2 : Show that the following argument is a fallacy :

$p \rightarrow q, q \vdash p$

Soln. :

We observe that $p \rightarrow q$ and q are true in case (3) in the **above** table, but in this case p is false.

Hence the argument is fallacy.

4.7.1 Theorem on Tautology

The argument $P_1, P_2, ..., P_n \vdash Q$ is valid if and only if the proposition $(P_1 \wedge P_2 \wedge .... \wedge P_n) \rightarrow Q$ is a tautology.

Proof

We recall that $P_1, P_2, ...,P_n$ are true if and only if $P_1 \wedge P_2 \wedge .... \wedge P_n$ is true. .... (i)

Thus the argument $P_1, P_2, ...P_n \vdash Q$ is valid if Q is true whenever $P_1, P_2, ... P_n$ is true or equivalently $P_1 \wedge P_2 \wedge ... \wedge P_n$ is true.

That is the proposition $(P_1 \wedge P_2 \wedge ... \wedge P_n) \rightarrow Q$ is a tautology.

4.7.2 Fundamental Principle of Logical Reasoning

The principle states that :

"If p implies q and q implies r, then p implies r."

that is if the following argument is valid :

$p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$. (Law of syllogism).

4.7.3 Verification of Law of Syllogism

Ex. 4.7.3 : Show that the following argument is a fallacy :

$p \rightarrow q, \neg p \vdash \neg q$.

Soln. :

We construct the truth table.

(i) $(p \rightarrow q) \wedge (\neg p)$; and $\neg q$ and then

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

and check whether it is a tautology or not.

Soln. :

p	q	$p \rightarrow q$	$\neg p$	$(p \rightarrow q) \wedge (\neg p)$	$\neg q$	$[(p \rightarrow q) \wedge \neg p] \rightarrow \neg q$
T	T	T	F	F	F	T
T	F	F	F	F	T	T
F	T	T	T	T	F	F
F	F	T	T	T	T	T

Since the proposition $[(p \rightarrow q) \wedge \neg p] \rightarrow \neg q$ is not a tautology (last column), hence it is a fallacy.

Also observe that in line (3) of the truth table $p \rightarrow q$ and $\neg p$ are true but $\neg q$ is false.

Ex. 4.7.4 : Prove that the argument is valid.

$P \leftrightarrow q, q \vdash p$.

Soln. :

We can establish the validity of the argument in two different methods :

Method 1

We construct truth-table for $p \leftrightarrow q$,

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

(i) Now, $p \leftrightarrow q$ is true in lines 1 and 4, and q in lines1 and 3.

∴ $p \leftrightarrow q$ and q are both true in line 1, where p is also true.

∴ The argument $p \leftrightarrow q, q \vdash p$ is valid

Scanned with CamScanner

Method 2

Now, we construct the truth-table of $[(p \leftrightarrow q) \wedge q] \rightarrow p$ as follows :

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

Since $[(p \leftrightarrow q) \wedge q] \rightarrow p$ is a tautology, the argument is valid.

Ex. 4.7.5 : Determine the validity of the argument

$p \rightarrow q, \neg q \vdash \neg p$.

☑ **Soln.:**

We construct the truth table for

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

p	q	$p \rightarrow q$	$\neg q$	$(p \rightarrow q) \wedge (\neg q)]$	$\neg p$	$[(p \rightarrow q) \wedge (\neg q)] \rightarrow \neg p$
T	T	T	F	F	F	T
T	F	F	T	F	F	T
F	T	T	F	F	T	T
F	F	T	T	T	T	T

Since the proposition $[(p \rightarrow q) \wedge \neg q] \rightarrow \neg p$ is a tautology, the given argument is valid.

Ex. 4.7.6 : Show that $(p \wedge q) \rightarrow (p \vee q)$ is a tautology.

☑ **Soln. :**

We prepare truth table :

Since the truth value of $(p \wedge q) \rightarrow (p \vee q)$ is T for all values of p and q, the proposition is a tautology.

p	q	$p \wedge q$	$p \vee q$	$p \wedge q \rightarrow p \vee q$
T	T	T	T	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	T

Ex. 4.7.7 : State the converse of each of the following implications:

(i) If $4 + 4 = 8$, then I am not the PM of India.

(ii) If I am late, then I did not take the train to work.

(iii) If I have enough money, then I will buy a car and I will buy a home.

☑ **Soln. :**

(i) If I am not the PM of India, then $4 + 4 = 8$.

(ii) If I do not take the train to work, then I am late.

(iii) If I buy a car and I buy a house, then I have enough money.

Ex. 4.7.8 : Determine the truth value for each of the following statements :

(i) If 8 is even, then Bombay has a large population.

(ii) If 8 is even, then Bombay has a small population.

(iii) If 8 is odd, then Bombay has a large population.

(iv) if 8 is odd, then Bombay has a small population.

☑ **Soln. :**

Let p = 8 be even, q = Bombay has large population.

We prepare the truth-value table.

Sr. No.	p	q	$p \rightarrow q$
(i)	T	T	T
(ii)	T	F	F
(iii)	F	T	T
(iv)	F	F	T

Ex. 4.7.9 : Construct truth tables to determine whether the given statement is a tautology, a contingency or an absurdity :

(i) $p \rightarrow (q \rightarrow p)$ (ii) $q \rightarrow (q \rightarrow p)$.

☑ **Soln. :**

(i) We prepare the truth-table :

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

Since the truth-value of $p \rightarrow (q \rightarrow p)$ is T for all values of p and q, the proposition is tautology.

(ii) The truth-table is :

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

Since the truth-value of $q \rightarrow (q \rightarrow p)$ is not T for all values of p and q, the proposition is **contingency**.

Module 4

Scanned with CamScanner

4.8 PRECEDENCE RULE

- We can always use a truth table to show that an **argument-form** is valid. We do this by showing that whenever premises are true, the conclusion must also be true.
- However, this can be a tedious approach. For example, when an argument form involves 10 different propositional variables, it requires $2^{10} = 1024$ different rows to show the argument form is valid, or not.
- Fortunately, we do not have to resort to truth tables. Instead we can first establish, Rule of precedence, i.e., the order of preference in which the connectives are applied in a formula of propositions that has no brackets is :

  (i) $\neg$ (ii) $\wedge$ (iii) $\vee$ and $\neg$ (iv) $\rightarrow$ and $\leftrightarrow$

4.9 DUALITY LAW

In this section, we shall consider formulae which contain the connectives $\wedge$, $\vee$, $\neg$. Well shall see later that, any formula containing any other connective can be replaced by an equivalent formula containing only these three connectives.

❑ **Definition :** Two formulae, A and A*, are said to be duals of each other if either one can be obtained from the other by replacing $\wedge$ by $\vee$ and $\vee$ by $\wedge$.

The connectives $\wedge$ and $\vee$ are called duals of each other. If the formula A contains the special variable T or F, then A*, its dual, is obtained by replacing T by F and F by T in addition to the above-mentioned interchanges.

4.9.1 Illusrative Example based on Duals

Ex. 4.9.1 : Write the duals of :

(i) $(p \vee q) \wedge r$; (ii) $(p \wedge q) \vee r)$

(iii) $(p \vee T)$, (iv) $\neg (p \vee q) \wedge (p \vee \neg (q \wedge \neg r)$

☑ **Soln. :** Duals are

(i) $(p \wedge q) \vee r$ (ii) $(p \vee q) \wedge r$

(iii) $(p \wedge F)$, (iv) $\neg (p \wedge q) \vee (p \wedge \neg (q \vee \neg r))$

4.9.2 Logical Identities

1. De Morgan's Laws

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

Dual is : (ii) $\neg (p \wedge q) = (\neg p \vee \neg q)$

2. Associative Laws

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

Dual is : (ii) $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$

3. Commutative Laws

(i) $p \vee q \equiv q \vee p$; **Dual is :** $p \wedge q \equiv q \wedge p$

4. Idempotent Laws

(i) $p \vee p \equiv p$; **Dual is :** $p \wedge p \equiv p$

5. Double Negation : (Involution Laws)

$$\neg (\neg p) = p.$$

6. Distributive Laws

(i) $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$

Dual is : $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$

7. Absorption Laws

(i) $p \vee (p \wedge q) \equiv p$ Dual : (ii) $p \wedge (p \vee q) = p$

8. Complement Laws

$p \vee \neg p \equiv T$; Dual : $p \wedge \neg p \equiv F$

9. $\neg T \equiv F$, $\neg F \equiv T$

4.9.3 Examples on Logical Equivalency

Ex. 4.9.2 : Show that the propositions $\neg (p \wedge q)$ and $\neg p \vee q$ are logically equivalent.

☑ **Soln. :**

> **Note :** There is an error in the problem we have to show that $\neg (p \wedge q)$ and $(\neg p \vee \neg q)]$ are logically equivalent.

We prepare truth-tables for both expressions.

(i) Truth tables for $\neg (p \wedge q)$ Truth table for $(\neg p \vee \neg q)$

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

p	q	$\neg p$	$\neg q$	$\neg p \vee \neg q$
T	T	F	F	F
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

Scanned with CamScanner

Since the truth tables are the same, i.e., both propositions are false in the first case and true in the other three cases, ∴ the propositions, $\neg (p \wedge q)$ and $(\neg p) \vee (\neg q)$ are logically equivalent ;

i.e., $\neg (p \wedge q) \equiv (\neg p) \vee (\neg q)$.

4.10 NORMAL FORMS

(i) Let **P** $(P_1, P_2, ..., P_n)$ be a given statement, where $P_1, P_2, ..., P_n$ are variables. To obtain truth-value of the formula P, we develop truth-table for P. Such a truth-table has 2^n rows.

- If P has **true** value for all possible assignments then P is said to be Tautology.
- If P has **false** value for all possible assignments, then P is said to be **contradiction.**
- If P has **truth** value, T for at least one combination of truth-values of $P_1, P_2, ..., P_n$ then P is said to be **satisfiable.**

(ii) The problem of finding whether a given statement is tautology or contradiction or satisfiable in a finite number of steps is called as **decision problem.**

(iii) For decision problems, the construction of truth-tables may not be a practical solution. Therefore we consider alternate procedure known as reduction to **Normal Forms.**

There are two such forms :

1. Disjunctive Normal Form (DNF).
2. Conjunctive Normal Form (CNF).

These forms are also called as **Canonical forms.**

4.10.1 Disjunctive Normal Form

Disjunctive normal form is a disjunction of fundamental conjunctions (∧). Now fundamental conjunctions (∧) are conjunction of simple statements, i.e. joining two statements by ∧.

i.e. **p, ¬ p, ¬ p ∧ q. p ∧ q, p ∧ ¬ p ∧ q** are fundamental conjunctions.

Some examples of DNF are as follows :

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

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

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

(iv) $(\neg p \wedge q) \vee (p \wedge q) \vee q$.

(v) **Remark :** $q \wedge \neg q$, $p \wedge \neg p$ are always false, (F).

4.10.2 Examples of DNF

Ex. 4.10.1 : Obtain the d.n.f. of the form

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

Soln. :

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

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

Using distributive law, $\because [(p \rightarrow q) = \neg p \vee q]$

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

Using associative and commutative laws ;

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

Using idempotent law ; i.e. $\neg p \wedge \neg p = \neg p$ and $q \wedge q = q$

$= [\neg p \wedge q] \vee [q \wedge \neg p]$ is dnf.

Ex. 4.10.2 : Obtain the d.n.f. of the form: $p \wedge (p \rightarrow q)$

Soln. : We have

$p \wedge (p \rightarrow q)$

$= p \wedge (\neg p \vee q)$ $\quad (\because p \rightarrow q \equiv \neg p \vee q)$

Using distributire law,

$= (p \wedge \neg p) \vee (p \wedge q) = F \vee (p \wedge q)$ $\quad (\because p \wedge \neg p = F)$

$= (p \wedge q)$ is d.n.f.

4.10.3 Conjunctive Normal Form (cnf)

A statement form, which consists of a **conjunction of fundamental disjunction** is called a conjunctive normal form (abbreviated as cnf). Now fundamental disjunctions (v) are disjunction of simple statements, i.e. joining two statements by 'v'.

Remark

1. We know that $p \vee \neg p$ is always true.

   i.e., $p \vee \neg p \vee r$ is logically equivalent to a tautology.

2. cnf is a tautology if and only if every fundamental disjunction contained in it is a tautology.

Examples

(i) $p \wedge r$ (ii) $\neg p \wedge (p \vee r)$

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

Module 4

Scanned with CamScanner

4.10.3(A) Conversion from PL to CNF

UQ. Explain the st25eps involved in converting the propositional logic statement into CNF with a suitable example

(MU - Q. 6(a), May 2016, 10 Marks)

To convert propositional formula to CNF, we perform the following steps.

- **Step 1 :** Push **negations** into the formula, repeatedly applying **De Morgan's Law**, until all negations only apply to atoms. We obtain a formula in **negation normal form** :
  - (i) $\neg (p \vee q)$ to $(\neg p) \wedge (\neg q)$
  - (ii) $\neg (p \vee q)$ to $(\neg p) \vee (\neg q)$
- **Step 2 :** We apply distributive law repeatedly where a disjunction occurs over a conjunction. When it is completed, the formula is in CNF.
  - (i) $p \vee (q \wedge r)$ to $(p \vee q) \wedge (p \vee r)$

Remark

To obtain a formula in disjunctive normal form, simply apply the distribution of $\wedge$ over $\vee$ in step (2)

Example : $A \rightarrow ((B \wedge C))$

$\equiv \neg A \vee (B \wedge C)$

$\equiv (\neg A \vee B) \wedge (\neg A \vee C)$

UQ. Convert the following propositional logic statement into CNF.

$A \rightarrow (B \leftrightarrow C)$ (MU- Q. 1(e), Dec. 17, 5 Marks)

- **Step (I) :** FOL : $A \rightarrow (B \leftrightarrow C)$
- **Step (II) :** Normalise the given statements

  $A \rightarrow (B \rightarrow C \wedge C \rightarrow B)$

  $A \rightarrow (B \rightarrow C) \wedge A \rightarrow (C \rightarrow B)$
- **Step (III) :** Converting to CNF :

  We apply the rule : $P \rightarrow Q \equiv \neg P \vee Q$

  $\therefore \neg A \vee (\neg B \vee C) \wedge \neg A \vee (\neg C \vee B)$

  $\neg A \vee \big((\neg B \vee C) \wedge (\neg C \vee B)\big)$

4.10.4 Examples on Conjunctive Normal Form

Ex. 4.10.3 : Obtain conjunctive normal form of

(i) $(\neg p \rightarrow r) \wedge (p \rightarrow q)$ (ii) $(p \wedge q) \vee (\neg p \wedge q \wedge r)$.

Soln. :

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

$= [\neg (\neg p \vee r)] \wedge [p \rightarrow q]$

$= [(\neg\neg p) \vee r] \wedge [\neg p \vee q]$ $(\because p \rightarrow r = \neg p \vee r)$

$= [p \vee r] \wedge [\neg p \vee q]$ is cnf.

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

Using distributive (left) law ;

$= [p \vee (\neg p \wedge q \wedge r)] \wedge [q \vee (\neg p \wedge q \wedge r)]$

$= [(p \vee \neg p) \wedge (p \vee q) \wedge (p \vee r)] \wedge [(q \vee \neg p) \wedge (q \vee q) \wedge (q \vee r)]$

$= [T \wedge (p \vee q) \wedge (p \vee r)] \wedge [(q \vee \neg p) \wedge (q) \wedge (q \vee r)]$

$= [(p \vee q) \wedge (p \vee r)] \wedge [(q \vee \neg p) \wedge (q) \wedge (q \vee r)]$ is cnf.

Ex. 4.10.4 : Obtain cnf of the following :

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

Soln. :

$(p \rightarrow q) \wedge (q \rightarrow p)$ $(\because (p \rightarrow q) = \neg p \vee q)$

$= (\neg p \vee q) \wedge (\neg q \vee p)$ is cnf.

Ex. 4.10.5 : Obtain the Conjunctive normal form of :

(i) $p \wedge (p \rightarrow q)$ (ii) $\neg (p \vee q) \leftrightarrow (p \wedge q)$.

Soln. :

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

$= p \wedge (\neg p \vee q)$ is cnf. $(\because p \rightarrow q = \neg p \vee q)$

(ii) $\neg (p \vee q) \leftrightarrow (p \wedge q)$.

$= [\neg (\neg (p \vee q) \vee (p \wedge q)] \wedge [\neg (p \wedge q) \vee \neg (p \vee q)]$

$[\because p \leftrightarrow q = (\neg p \vee q) \wedge (\neg q \vee p)]$

$= [(p \vee q) \vee (p \wedge q)] \wedge [(\neg p \vee \neg q) \vee (\neg p \wedge \neg q)]$

$[\because \neg (p \wedge q) = \neg p \vee \neg q; \neg (p \vee q) = \neg p \wedge \neg q]$.

$= [(p \vee q \vee p) \wedge (p \vee q \vee q)] \wedge [(\neg p \vee \neg q \vee \neg p)]$

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

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

$[\because p \vee p = p$ and $\neg p \vee \neg p = \neg p]$

$[\because p \wedge p = p ; \neg p \wedge \neg p = p]$

$= (p \vee q) \wedge (\neg p \vee \neg q)$ is cnf

Scanned with CamScanner

4.11 TRUTH TABLE METHOD TO FIND DNF

Let p be a statement form containing n variables $P_1, P_2, ..., P_n$. Its dnf can be obtained from the truth-table, as follows :

- **Step (I) :** Consider the truth value (T) from P.
- **Step (II) :** Form the conjunction ('$\wedge$') :

  $(P_1 \wedge P_2 \wedge ... \wedge P_i \wedge ... \wedge P_j \wedge ... \wedge P_n)$,

  Where if P_i is true consider P_i and

  if P_j is false consider $\neg P_j$.

  Such a term is called **minterm.**
- **Step (III) :** The disjunction of minterms is the dnf of the given form.

4.11.1 Examples on dnf

Ex. 4.11.1 : Find the dnf of $(\neg p \rightarrow r) \wedge (p \leftrightarrow q)$.

Soln. :

- **Step (I) :** First we prepare the truth-table.

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

- **Step (II) :** (i) Consider only 'T' from last column and choose corresponding values (T) from p, q, r. E.g. for first row, corresponding p, q and r are true, so we consider $(p \wedge q \wedge r)$.

  (ii) Similarly for 2nd row, corresponding p and q are true, but r is false; so we consider $(p \wedge q \wedge \neg r)$.

  (iii) Again for 7th row, corresponding p, q are false and r is true, so we consider, $(\neg p \wedge \neg q \wedge r)$.
- **Step (III) :** Hence the dnf of

  $(\neg p \rightarrow r) \wedge (p \leftrightarrow q)$ is equal to :

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

4.12 PREDICATE LOGIC (FIRST ORDER LOGIC)

Propositional logic, cannot adequately express the meaning of statements in mathematics and in computer programs.

For example

Statements involving variables such as :

(i) "x > 3" (ii) "x = y + 3" (iii) "x = y + z";

(iv) "Computer x is under attack by an intruder"

(v) "Computer y is functioning properly"

These statements are neither true nor false, if the values of the variables are not specified. Here we discuss the ways that propositions can be produced from such statements.

4.12.1 Predicates

- The statement "**x is greater than 3**" has two parts. The first part, the variable x, is the subject of the statement.

  The second part – called as Predicate "**is greater than 3**" – refers to the property that the subject of the statement can have.
- We denote the statement "x is greater than 3" by P(x), where "P" denotes the **predicate** "is greater than 3" and 'x' is the variable. The statement **P(x) is also said to be the value of P at x.**
- Once a value has been assigned to the variable x, the statement P(x) becomes a **proposition** and has a truth value, and P is called as **propositional function.**

Module 4

Ex. 4.12.1 : Let P(x) denote the statement "x > 3". What are the truth values of P(4) and P(2).

Soln. :

To obtain P(4), we put x = 4 in the statement 'x > 3'. ∴ P(4) is "4 > 3", which is **true**. But P(2), which is the statement "2 > 3" is false.

Ex. 4.12.2 : Consider the statement "x = y + 3". We can denote the statement by Q(x, y), where x and y are variables and Q is the predicate. What are the truth values of the propositions Q(1, 2) and Q(3, 0) ?

Soln. :

To obtain Q(1, 2); we put x = 1, y = 2 in the statement Q(x, y). Hence Q (1, 2) : "1 = 2 + 3" which is false.

The statement Q (3, 0) is the proposition "3 = 0 + 3", which is true.

Scanned with CamScanner

Ex. 4.12.3 : Let R(x, y, z) denote the statement "x + y = z". What are the truth values of the propositions R (1, 2, 3) and R (0, 0, 1) ?

Soln. :

The proposition R (1, 2, 3) is obtained by putting x = 1, y = 2, z = 3 in the statement R (x, y, z). We see that R (1, 2, 3) is "1 + 2 = 3" which is true.

Also, note that R(0, 0, 1) which is the statement "0 + 0 = 1", is false.

Remark

1. In general, a statement involving n variables $x_1, x_2,...,x_n$ can be denoted by $P(x_1, x_2,..,x_n)$, and is the value of the propositional function P at the n-tuple $(x_1, x_2, ..., x_n)$

   P is also called an **n-place predicate** or a **n-ary predicate**.
2. A predicate is generally not a proposition but every proposition is a propositional function or a predicate.

4.13 THE UNIVERSAL QUANTIFIER

- Many mathematical statements assert that a property is true for all values of a variable in a particular domain. The 'domain' of the values of a variable is also called as 'universe of discourse', or 'domain of discourse', (often just referred to as the 'domain').
- The universal quantification of P(x) for a particular domain is the proposition that tells that P(x) is true for all values of x in the domain.

Definition : The universal quantification of **P(x)** is the statement.

- "P(x) for all values of x in the domain." The notation ∀ x P(x) as **"for all x P(x)".**
- An element for which P(x) is false is called a counterexample of **∀ x P(x).**

Remark (i)

(1) The symbol '∀' is known as universal quantifier.

(2) The proposition "for all x, P(x)", which is interpreted as "for all values of x, P(x) is true." is a proposition in which the variable x is said to be 'universally quantified' [and '∀' is known as **universal quantifier**]

4.13.1 Existential Quantifiers

- Suppose for the predicate **P(x), ∀ x [P(x)]** is false, but there exists at least one value of x (or some values of x) for which, P(x) is true, then we say that, x is bounded by **existential quantification.**
- The symbol used for 'there exists' is denoted by '∃'. Thus **∃x (P(x))** means 'there exists a value of **x** in the domain for which **P(x)** is true'.

Remark

(1) Negation of ∀ x(P(x)) is "∀ x, P(x) is not true", i.e. 'there exists' at least one x for which P(x) is not true, or in other words, there exists an x for which '¬ P(x)' is true.

(2) Observe that the statement ∃ x P(x) is false if and only if there is no element x in the domain for which P(x) is true.

We **summarise the meaning of universal quantification of P(x) :**

Table 4.13.1 : Universal quantification

Statement	When true ?	When false ?
∀ x P(x)	P(x) is true for every x	There is an x for which P(x) is false
∃ x P(x)	There is an x for which P(x) is true	P(x) is false for every x.

Specifying the domain is **mandatory** when **quantifiers** are used. The truth value of a quantified statement often depends on which elements are in the domain.

4.14 EXAMPLES OF QUANTIFICATION

Ex. 4.14.1 : Let P(x) be the statement "$x^2 > 0$", where the universe of discourse consists of all integers. What is the truth-value of the quantification ∀ x P(x) ?

Soln. :

[**Note :** 'Universe of discourse' is domain of the statement "$x^2 > 0$" consisting of all integers]

We see that x = 0 is a counter-example because $x^2 = 0$ when x = 0, so that x^2 is not greater than 0 when x = 0.

Scanned with CamScanner

Remark

When all the elements in the domain can be listed-say, $x_1, x_2, \ldots x_n$ – it follows that the universal quantification $\forall x\ P(x)$ is same as the conjunction $P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n)$, because this conjunction is true if and only if $P(x_1), P(x_2), \ldots, P(x_n)$ are all true.

Ex. 4.14.2 : What is the truth-value of $\forall x\ P(x)$, where P(x) is the statement "$x^2 < 10$" and the domain consists of positive integers not exceeding 4 ?

Soln. :

The domain consists of integers 1, 2, 3, 4. We observe that (from the above remark) the statement $\forall x\ P(x)$ is the same as the conjunction.

$$P(1) \wedge P(2) \wedge P(3) \wedge P(4).$$

The statement P(4) "$4^2 < 10$" is false

$$\therefore \forall x \quad P(x) \text{ is false.}$$

Ex. 4.14.3 : Let Q(x) denote the statement "x = x + 1". What is the truth value of the quantification $\exists x\ Q(x)$, where the domain consists of all real numbers.

Soln. :

We note that Q(x) is false for every real number x,

$\therefore$ The existential qualification of Q(x), which is $\exists x\ Q(x)$, is **false.**

Remark

- When all elements in the domain can be listed – say, $x_1, x_2, \ldots, x_n$ the existential quantification $\exists x\ P(x)$ is the same as the disjunction.

  $P(x_1) \vee P(x_2) \vee \ldots \vee P(x_n)$
- Because this disjunction is true if and only if at least one of $p(x_1), p(x_2), \ldots, p(x_n)$ is true.

Ex. 4.14.4 : What is the truth value of $\exists x\ P(x)$, where P(x) is the statement "$x^2 > 10$" and the universe of discourse consists of positive integers not exceeding 4 ?

Soln. :

- The given domain is {1, 2, 3, 4}. From the above problem, we say that the proposition $\exists x\ P(x)$ is the same as the disjunction.

  $P(1) \vee P(2) \vee P(3) \vee P(4)$
- Because P(4), which is the statement "$4^2 > 10$", is true, it follows that

  $[P(1) \vee P(2) \vee P(3) \vee P(4)]$ is true

  $\therefore \exists x\ P(x)$ is true

Ex. 4.14.5 : What do the statements $\forall x < 0\ (x^2 > 0)$, $\forall y \neq 0\ (y^3 \neq 0)$ and $\exists z > 0\ (z^2 = 2)$ mean, where the domain in each case consists of the real numbers.

Soln. :

(i) The statement $\forall x < 0\ (x^2 > 0)$ states that for every real number x with $x < 0$, $x^2 > 0$. That is, it states "The square of a real number is positive"

This statement is same as $\forall x\ (x < 0 \rightarrow x^2 > 0)$.

(ii) The statement $\forall y \neq 0\ (y^3 \neq 0)$ states that for every real number y with $y \neq 0$, $y^3 \neq 0$.

That is, it states that "the cube of every non-zero real number is non-zero".

This statement is equivalent to $\forall y\ (y \neq 0 \rightarrow y^3 \neq 0)$

(iii) The statement $\exists z > 0\ (z^2 = 2)$ states that there exists a real number z' with $z > 0$ such that $z^2 = 2$. That is, it states "There is a positive square root of 2". The statement is equivalent to $\exists z\ (z > 0 \wedge z^2 = 2)$.

4.14.1 Free and Bound Variables

- **Bound Variables :** When a quantifier is used on the variable x, we say that this occurrence of the variable is **bound.**
- **Free Variables**: An occurrence of a variable that is not bound by a quantifier is said to be **free.** [A variable is free if it is outside the scope of all quantifiers in the formula that specifies this variable, or set equal to a particular value]
- **For example:** in the statement $\exists x\ (x + y = 1)$, the variable **x** is **bound** by the existential quantification $\exists x$, but the variable **y** is **free** because it is not bound by a quantifier and no value is assigned to this variable.

4.14.2 Logical Equivalences Involving Quantifiers

We have introduced the notion **of logical equivalences of compound propositions**. Now, we extend this notion to expressions involving predicates and quantifiers.

- **Definition :** Statements involving predicates and quantifiers are **logically equivalent** if and only if they have the same truth value no matter which predicates are substituted into these statements and which domain of discourse is used for the variables in these propositional functions.

Module 4

Scanned with CamScanner

We use the notation **S = T** to indicate that two statements S and T involving predicates and quantifiers are logically equivalent.

Ex. 4.14.6 : Show that $\forall x\ [P(x) \wedge Q(x)]$ and $\forall x\ P(x) \wedge \forall x\ Q(x)$ are logically equivalent, where the same domain is used throughout.

Soln. :

Step (I)

- Suppose that $\forall x\ [P(x) \wedge Q(x)]$ is true. This means that for any 'a' in the domain $p(a) \wedge Q(a)$ is true. Hence, p(a) is true and Q(a) is true.
- Because p(a) is true and Q(a) is true **for every element** in the domain, we conclude that $\forall x\ P(x)$ and $\forall x\ Q(x)$ are both true.

$\therefore \forall x\ P(x) \wedge \forall x\ Q(x)$ be true

Step II

- Now let $\forall x\ p(x) \wedge \forall x\ Q(x)$ be true. It follows that
- $\forall x\ P(x)$ is true and $\forall x\ Q(x)$ is true. Hence, if 'a' is in the domain, then p(a) is true and Q(a) is true. [Because P(x) and Q(x) are both true for all elements in the domain].

If follows that **for all** a, $p(a) \wedge Q(a)$ is true. Hence $\forall x\ [P(x) \wedge Q(x)]$ is true.

$\therefore \forall x\ [P(x) \wedge Q(x)] = \forall x\ P(x) \wedge \forall x\ Q(x)$.

4.14.3 Negating Quantified Expressions

Here we consider the **negation** of a quantified expression:

First we consider **'universal quantifier '∀''**,

We consider an example :

"Every student has taken a course in calculus."

- This statement is a universal quantification, namely, $\forall x\ P(x)$, where **P(x)** is the statement : "x has taken a course in calculus" and **domain** consists of the students in your class.
- **The negation of this statement is** : "It is not the case that every student in your class has taken a course in calculus".

  This statement is equivalent to: "There is a student in your class who has not taken a course in calculus".
- This is simply the existential quantification of the negation of the original propositional function, namely, $\exists x \neg P(x)$.
- The above two statements illustrate the following **logical equivalence** : $\neg \forall x\ P(x) = \exists x \neg P(x)$.

4.14.4 Negating an Existential Qualification

We take an example to negate an **existential qualification** : Let us consider an example :

"There is a student in this class who has taken a course in calculus."

This is an existential qualification : $\exists x\ Q(x)$, where Q(x) is the statement "x has taken a course in calculus."

The negation of this statement :

(1) "It is not the case that there is a student in this class, who has taken a course in calculus."

This proposition is equivalent to :

(2) "Every student in this class has not taken calculus."

We phrase the above two statements in the language of quantifiers :

$$\neg \exists x\ Q(x) \equiv \forall x \neg Q(x)$$

This is logical equivalence,

4.14.5 De Morgan;s Laws for Negations for Quantifiers

Table 4.14.1 : De Morgan's laws for quantifiers

Negation	Equivalent statement	When is negative true ?	When false
$\neg \exists x\ P(x)$	$\forall x \neg P(x)$	For every x, P(x) is false.	There is an x for which P(x) is true
$\neg \forall x\ P(x)$	$\exists x \neg P(x)$	There is an x for which P(x) is false	P(x) is true for every x

4.14.6 Different inference Rules for FOPL.

UQ. Explain different inference Rules for FOPL.

(MU - Q. 4(b), May 18, 10 Marks)

- Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences.
- Let us first see some basic terminologies used in FOL.

Scanned with CamScanner

Substitution : Substitution is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL.

If we write F [a/x], so it refers to substitute a constant 'a' in place of variable 'x'.

Equality : First-order logic also uses what is called as Equality in FOL. For this, we can use **equality symbols** which specify that the two terms refer to the same subject.

Example : Brother (Ramesh) = Ashok

Here, the object referred by the Brother (Ramesh) is similar to the object referred by Ashok.

The equality symbol can also be used with negation to represent that two terms are not the same objects.

Example : $\neg$ (x = y) which is equivalent to $x \neq y$.

☞ FOL Inference rules for quantifier

As propositional logic we also have inference rules in first-order logic, the following are some basic inference rules in FOL :

(i) Universal Generalisation
(ii) Universal Instantiation
(iii) Existential Instantiation
(iv) Existential Introduction

▶ (I) Universal Generalisation

Universal generalisation is a valid inference rule which states that if premise P(c) is true for any arbitrary element c, then we can have a conclusion as $\forall x\ P(x)$. It can be represented as :

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

▶ (ii) Universal Instantiation (UI)

Universal Instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.

As per UI, **we can infer any sentence obtained by substituting a ground term for the variable.**

The UI rule states that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from $\forall x\ P(x)$ for any object in the universe of discourse.

It can be represented as $\frac{\forall x\ P(x)}{P(c)}$

Example (i) : If "Every person like ice-cream" $\rightarrow \forall x\ P(x)$ so we can infer that " John likes ice-cream" $\rightarrow$ P(c).

Example (ii) : "All kings who are greedy are Evil".

In FOL form :

$\forall x$ king (x) $\wedge$ greedy (x) $\rightarrow$ Evil (x)

▶ (iii) Existential Instantiation

Existential Instantiation is also called as Existential Elimination, which is a valid inference rule in first-order logic. It can be applied only to replace the existential sentence.

This rule states that one can infer P(c) from the formula given in the form of $\exists$ x P(x) for a new constant symbol c.

It is represented as $\frac{\exists x P(x)}{P(c)}$,

▶ (iv) Existential Introduction

- An existential introduction is also known as an existential generalisation, which is a valid inference rule in first-order logic.
- This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.

It can be expresses as :

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

Example :

'Priyanka got good marks in English'

"Therefore, someone got good marks in English."

☞ Generalised Modus Ponens Rule:

For the inference process in FOL, we have a single inference rule which is called 'Generalised Modus Ponens'.

Generalised modus ponens can be summarised as, " P implies Q and P is asserted to be true, therefore Q must be true".

According to modus ponens, for atomic sentences P_i, P_i, q, where there is a substitution θ such that subst (θ, p_i') = subset (θ, p_i)

It can be written as :

$$\frac{P_1', P_2'}{\ldots, P_n'\ (P_1 \wedge P_2 \wedge \ldots \wedge P_n) \rightarrow q,\ \text{subst} \cdot (\theta, q)}$$

Example : We will use this rule for kings are evil, so we will find some x such that x is king, and x is greedy so we can infer that x is evil.

Here, let p_1' be king (John) P_1 is king (x)
P_2' is greedy (y) P_2 is greedy (x)
θ is {x/ John, y/ John} q is evil (x)
Subst (θ, q)

Module 4

Scanned with CamScanner

4.14.7 Examples

Ex. 4.14.7 : Consider the statements :

"All humming birds are richly coloured"

"No large birds live on honey"

"Birds that do not live on honey are dull in colour."

"Humming birds are small"

Let P(x), Q(x), R(x) and S(x) be the statements "x is a humming bird", "x is large", "x lives on honey," and "x is richly coloured" respectively. Assuming that the domain consists of all birds, express the statements in the argument using qualifiers and P(x), Q(x), R(x) and S(x).

Soln. :

We express the statements using qualifiers

$\forall x\ [P(x) \rightarrow S(x)]$

$\neg \exists x\ [Q(x) \wedge R(x)]$

$\forall x\ [\neg R(x) \rightarrow \neg S(x)]$

$\forall x\ [P(x) \neg Q(x)]$

Remarks

(i) Here we assume "small" as "not large" and

(ii) "dull in colour" as "not richly coloured"

Ex. 4.14.8 : For the universe of integers, let p(x), q(x), r(x) and t(x) be the following open statements :

p(x) : x > 0, q(x) : x is even, r(x) : x is a perfect square,

t(x) : x is divisible by s.

Write the following statements in symbolic form

(i) At least one integer is even

(ii) There exists a positive integer that is even.

(iii) If x is even then x is not divisible by 5,

(iv) No even integer is divisible by 5,

(v) There exists an even integer divisible by 5.

Soln. :

(i) $\exists x\ q(x)$

(ii) $\exists x\ [p(x) \wedge q(x)]$

(iii) $\forall x\ [q(x) \rightarrow \neg t(x)]$

(iv) $\forall x \neg [q(x) \wedge t(x)]$

(v) $\exists x\ [q(x) \wedge t(x)]$

UEx. 4.14.9 : (MU- Q. 2(b), 2016, 5 Marks)

Write first order logic statements for following statements

Soln. :

(1) If a perfect square is divisible by a prime P then it is also divisible by square of P.

$\exists x$: (x divisible by p) $\rightarrow$ x (divisible by p^2)

(2) $x \rightarrow$ likes ($\neg$ chemistry $\wedge$ $\neg$ History)

$\neg x \vee (\neg$ chemistry $\wedge \neg$ History)

(3) If it is Saturday and warm. Then sam is in park

$\forall x,\ y\ (x \wedge y) \rightarrow$ park (sam)

(4) Anything anyone eats and is not killed by is food.

$\forall x\ \forall y$: eat (x, y) $\wedge \neg$ killed (x) $\rightarrow$ food (y)

$\forall x, \forall y$

$\neg$ (eats (x, y) $\wedge \neg$ killed (x)) $\vee$ food (y)

4.14.8 FOPL

UEx. 4.14.10 : (MU- Q. 1(c), May 2018, 6 Marks)

What is FOPL ? Represent the following sentences using FOPL.

(i) John has atleast two friends

(ii) If two people are friends then they are not enemies.

Soln. :

The First Order Predicate Logic (FOPL) is a method of formal representation of Natural Language (NL) text. The prolog language for AI programming has its foundations in FOPL. If demonstrates how to translate NL to FOPL in the form of facts and rules, use of quantifiers and variables, syntax and semantics of FOPL, and conversion of predicate expression to clause forms. This is followed with unification of predicate expressions using instantiations and substitutions, compositions of substitutions, unification algorithm and its analysis.

(i) x : x (John) $\rightarrow$ at least friends (y $\wedge$z)

(ii) x $\wedge$ y (friends) $\rightarrow$ ($\neg$ x) $\wedge$ ($\neg$ y) (enemies)

Scanned with CamScanner

4.14.9 Comparison between Propositional Logic or First Order Logic (Predicate Logic)

UQ. Distinguish between Propositional Logic (PL) and first order predicate logic (FOPL) knowledge representation mechanisms. Take suitable example for each point of differentiation. **(MU - Q. 2(b), May 19, 10 Marks)**

Sr. No.	Parameters	Propositional Logic (PL)	Predicate Logic (FOL)
1.	Definition	Propositional logic deals with simple declarative propositions.	First order logic additionally covers predicates and quantification.
2.	Entities	A proposition is a collection of declarative statements that has either a truth value "true" or a truth value "false".	Predicate logic is an expression of one or more variables defined on some specific domain.
3.	Boolean values	Propositional logic is a simple form of logic which is also known as Boolean logic.	Predicate logic is a collection of formal systems which uses quantified variables over non-logical objects and allows the use of sentences which contain variables.
4.	Truth values	A proposition has truth values (0, 1) which means it can have one of the two values i.e. True or False.	Predicate logic is an expression consisting of variables with a specific domain. It is also known as Boolean logic.
5.	Usefulness	It is the most basic and widely used logic.	Predicate logic is an extension of propositional logic.
		This logic is used for the development of powerful search algorithms including implementation methods.	Predicate logic deals with infinite structures as well. The quantifiers are the linguistic marks that permit one to treat with such infinite structures.
		Propositional logic is used in AI for planning, problem-solving intelligent control and for decision-making.	A predicate with variables can be made a proposition by either authorising a value to the variable or by quantifying the variable.
6.	Nature	It also includes certainty as well as uncertainty.	It consists of objects, functions, relations between the objects.
7.	Representations	It led to the foundation for machine learning models.	Predicate logic helps analyse the scope of the subject over the predicate.
8.	Language	It is a useful tool for reasoning.	It is different from propositional logic which lacks quantifiers.
9.	Level logic	It has limitation because it cannot see inside prepositions and take advantage of relationships among them.	Predicate logic is undecidable, since universal and existential quantifiers treat with infinite structures.

Module 4

Scanned with CamScanner

4.15 FORWARD CHAINING AND BACKWARD CHAINING

UQ. Explain Forward-chaining and Backward-Chaining algorithm with the help of example. (MU – Q. 1(a), May 17 4 Marks, Q. 6(C), Dec. 17, 5 Marks, Q. 6(a), Dec. 18, Q. 6(a), May 19, Q. 5(a), Dec. 19, 10 Marks)

Rule-based system architecture consists of a set of rules, a set of facts, and an **inference engine**. The need is to find what new facts can be derived.

Given a set of rules, there are essentially two ways to generate new knowledge: one, forward chaining and the other, backward chaining.

4.15.1 Forward Chaining

UQ. Illustrate Forward chaining in propositional logic with example. (MU - Q. 3(b), May 17, 10 Marks)

Forward chaining employs the system starts from that a set of facts, and a set of rules, and tries to find a way of using those rules and facts to deduce a conclusion or come up with a suitable course of action.

This is known as **data-driven reasoning** because the reasoning starts from a set of data and ends up at the goal, which is the conclusion.

Steps followed in forward chaining

(i) When applying forward chaining, the first step is to take the facts in the **fact database** and see if any combination of these matches all the antecedents of one of the rules in the rule database.

(ii) When all the **antecedents** of a rule are matched by facts in the database, then this rule is **triggered**.

(iii) Usually, when a rule is triggered, it is then **fired**, which means its conclusion is added to the facts database. If the conclusion of the rule that has been fired is an action or a recommendation, then the system may cause that action to take place or the recommendation to be made.

(iv) **For example**, consider the following set of rules used to control an elevator in a three-story building :

Rule 1: IF on first floor and button is pressed on first floor
THEN open door

Rule 2: IF on first floor
AND buttons is pressed on second floor
THEN go to second floor.

Rule 3: IF on first floor
AND buttons is pressed on third floor THEN go to third floor.

Rule 4 : IF on second floor AND button is pressed on first floor AND already going to third floor THEN remember to go to first floor later.

4.15.2 Backward Chaining

UQ. Illustrate Forward chaining and backward chaining in propositional logic with example. (MU - Q. 3(b), May 17, 10 Marks)

In backward chaining, we start from a conclusion, which is the **hypothesis** we wish to prove, and we aim to show how that conclusion can be reached from the rules and facts in the database.

The conclusion we are aiming to prove is called a **goal**, and so reasoning in this way is known as **goal-driven reasoning**.

Backward chaining used in formulating plans

- A plan is a sequence of actions that a program decides to take to solve a particular problem. Backward chaining can make the process of formulating a plan more efficient than forward chaining.
- Backward chaining in this way starts with the goal state, which is the set of conditions the agent wishes to achieve in carrying out its plan. It now examines this state and sees what actions could lead to it.
- **For example**, if the goal state involves a block being on a table, then one possible action would be to place that block on the table.
- This action might not be possible from the start state, and so further actions need to be added before this action in order to reach it from the start state.
- In this way, a plan can be formulated starting from the goal and working back toward the start state. The benefit in this method is particularly clear in situations where the first state allows a very large number of possible actions.
- In this kind of situation, it can be very inefficient to attempt to formulate a plan using forward chaining

Scanned with CamScanner

because it involves examining every possible action, without paying any attention to which action might be the best one to lead to the goal state.

- Backward chaining ensures that each action that is taken is one that will definitely lead to the goal, and in many cases this will make the planning process far more efficient.

4.15.3 Forward Reasoning

GQ. Explain forward and backward reasoning with examples.

GQ Explain reasoning with example. Compare forward and backward reasoning with example.

GQ Differentiate between forward and backward reasoning.

- **Forward reasoning is also called as forward chaining** in the field of artificial intelligence. It is one of the methods that is used as a reasoning engine with working with inference driven entities.
- Forward reasoning is one of the most population implementation strategies in the concepts of **expert systems and production rule-based systems**.
- With forward chaining, it makes **use of the existing data** alongside the inference rules to extract move data from the user until a certain goal is reached.
- An inference engine **will iterate through the process** of obtaining new data to eventually satisfy the goal.
- The working is based on the **real-world implementation** of the if-then clauses.
- Forward chaining is **a down-up process** in which it works from the bottom to the top in the order of the occurrence of data.

4.15.4 Modus Ponens

UQ. Explain modus ponen with suitable example.

(MU - Q. 1(C), Dec. 17, Q. 1(b), May 16, 4 Marks)

- Forward reasoning (chaining) can be described as repeated application of **'modus ponens'**.
- Forward chaining is a popular implementation strategy for expert systems, business and production rule systems.
- We consider the following famous example which we will use in both forward and backward reasoning.

Remark : 'Modus ponens' is a rule of inference and it states that if P and $P \rightarrow Q$ are true, then Q is also true.

Example : "As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen".

Prove that "Robert is criminal".

☑ **Soln. :**

To solve the problem, we convert all the facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

(I) Facts converting into First Order Logic (FOL) : (a) It is a crime for an American to sell weapons to hostile nations. (Let p, q, r be the variables)

American (p) $\wedge$ weapon (q) $\wedge$ sells (p, q, r) $\wedge$ hostile (r) $\rightarrow$ criminal (p) ...(1)

(b) Country A has some missiles

? P Owns (A, p) $\wedge$ Missile (p)

It can be written in two definite clauses by using **Existential Instantiation,** introducing new constant T_1.

Owns (A, T_1) ...(2)

Missile (T_1) ...(3)

(c) All of the missiles were sold to country A by Robert.

? P Missile (p) $\wedge$ Owns (A, p) $\rightarrow$ Sells (Robert, p, A) ...(4)

(d) Missiles are weapons.

Missile (p) $\rightarrow$ Weapons (p) ...(5)

(e) Enemy of America is known as hostile.

Enemy (p, America) $\rightarrow$ Hostile (p) ...(6)

(f) Country A is an enemy of America

Enemy (A, America) ...(7)

(g) Robert is American

American (Robert) ...(8)

Remark : Existential instantiation is also called as (existential elimination) is a rule of inference which says that, given a formula of the form $(\exists x)\ \phi(x)$, one may infer $\phi(c)$ for a new constant symbol $\subset$.

4.15.5 Forward Chaining Proof

▶ **Step 1 :** We begin with the known facts and will choose the sentences which do not have implications, such as : American (Robert), Enemy (A, America), Owns (A, T_1), and Missile (T_1). All these facts are represented as :

American (Robert)		Missile (T_1)		Owns (A, T_1)		Enemy (A, America)

Module 4

Scanned with CamScanner

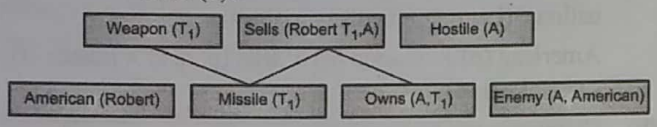
- **Step 2 :** We see the facts which infer from available facts and with satisfied premises.

Rule (1) : does not satisfy premises, so it will not be added in the first iteration.

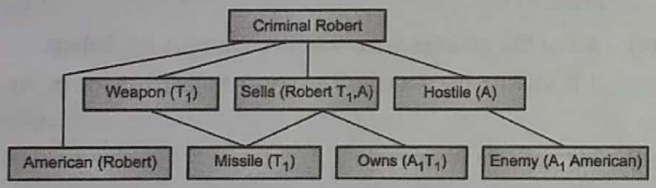
Rule (2) and (3) are already added.

Rule (4) : Satisfy with the substitution $\{p/T_1\}$, so sells (Robert, T_1, A) is added, which infers from the conjunction of Rule (2) and (3)

Rule (5) : is satisfied with the substitution (P/A), so hostile (A) is added and which infers from Rule (6).



- **Step 3 :** As we can check Rule (1) is satisfied with the substitution {p/Robert, q/T_1, r/A}, so we can add criminal (Robert) which infers all the available facts. And hence we reached our goal statement.



Hence it is proved that Robert is criminal using forward reasoning (chaining) approach.

4.15.6 Backward Chaining

Backward chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

4.15.7 Properties of Backward Reasoning (Chaining)

(a) It is known as a top-down approach.

(b) Backward-chaining is based on modus ponens inference rule.

(c) In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.

(d) It is called a goal-driven approach, as a list of goals decides which rules are selected and used.

(e) Backward-chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.

(f) The backward-chaining method mostly used a depth-first search strategy for proof.

Example :

We use the same above problem in backward-chaining.

We rewrite all the rules :

(a) American (p) $\wedge$ weapon (a) $\wedge$ sells (p, q, r) $\wedge$ hostile (r) $\rightarrow$ criminal (D) ...(1)

(b) Owns (A, T_1) ...(2)

(c) ? p Missiles (p) $\wedge$ Owns (A, p) $\rightarrow$ Sells (Robert, p, A) ...(3)

(d) Missile (p) $\rightarrow$ Weapons (p) ...(4)

(e) Enemy (p, America) $\rightarrow$ Hostile (p) ...(5)

(f) Enemy (A, America) ...(6)

(g) American (Robert) ...(7)

4.15.8 Backward Chaining Proof

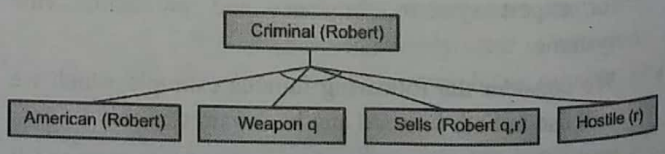
In Backward chaining, we begin with our goal predicate, which is criminal (Robert) and then infer further rules.

- **Step 1 :** We assume the goal fact. And from the goal-fact, we infer other facts, and at last, we prove those facts true.

  So our goal fact is "Rober is Criminal", so following is the predicate of it.

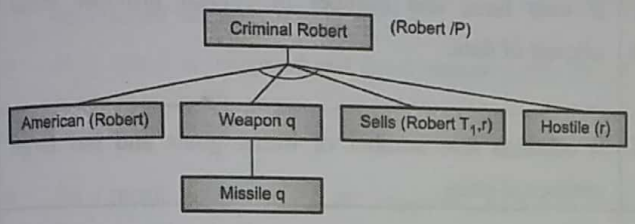
- **Step 2 :** Here we infer other facts from goal fact which satisfies the rules. So as we can see in Rule 1, the goal predicate criminal (Robert) is present with substitution (Robert/p). So we will add all the conjunctive facts below the first level and we replace p with Robert.

Here we can see American (Robert) is a fact, so it is proved here.

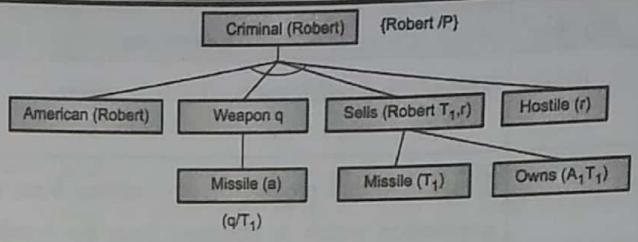


Scanned with CamScanner

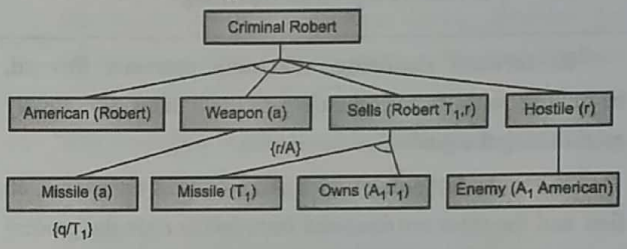
- **Step 3 :** At step 3, we extract further fact Missile (q) which infer from weapon (a) , as it satisfies Rule (5). Weapon (q) is also true with the substitution of a constant T_1 at q.



- **Step 4 :** Now we can infer facts Missile (T_1) and Owns (A, T_1) from sells (Robert, T_1, r) which satisfies the Rule (4) with the substitution of A in place of r.
  So these two statements are proved here.



- **Step 5 :** Now, we can infer the fact Enemy (A, America) from Hostile (A) which satisfies Rule (6). And hence all the statements are proved true using backward chaining.



4.15.9 Comparison between Forward and Backward Reasoning

Sr. No.	Forward Reasoning	Backward reasoning
1	It is a data-driven task.	It is goal driven task.
2	It begins with new data.	It begins with conclusions that are uncertain.
3.	The objective is to find a conclusion	The objective is to find the facts that support the conclusions.
4.	It uses an opportunistic type of approach	It uses a conservative type of approach.
5.	It flows from incipient to the consequence.	It flows from consequence to the incipient.
6.	The precedence of these constraints have to match the current state.	It is based on the decision fetched by the initial state. The system helps choose a goal state, and reasons in a backward direction. It is also known as a decision-driven or goal-driven inference technique.
7.	The inference engine searches the knowledge base with the given information depending on the constraints.	First step is that the goal state and rules are selected.
8.	The first step is that the system is given one or more constraints.	Sub-goals are made from the selected rule, which need to be satisfied for the goal state to be true.
9.	The rules are searched for in the knowledge base for every constraint.	The initial conditions are set such that they satisfy all the sub-goals. The established states are matched to the initial state provided.

Module 4

Scanned with CamScanner

Sr. No.	Forward Reasoning	Backward reasoning
10.	The rule that fulfils the condition is selected.	If the condition is fulfilled, then the goal is the solution. Otherwise the goal is rejected.
11.	Every rule can produce new condition from the conclusion which is obtained from the invoked one.	If tests have less number of rules,it provides small amount of data.
12.	New conditions can be added and are processed again. The step ends if no new conditions exist.	It contains less number of initial goals and has large number of rules
13.	It follows top-down reasoning.	It follows bottom-up reasoning technique.

In forward reasoning, reasoning proceeds forward, beginning with factor, chaining through rules and finally establishing the goal.

When the left side of a sequence of rules is instantiated first and the rules are executed from left to right the process is called forward chaining/reasoning.

This is also known as data-driven search, since, input data are used to guide the direction of the inference process. For example, we can chain forward to show that when a student is encouraged, is healthy, and has goals, the student will succeed.

ENCOURAGED (student) MOTIVATED (students)

MOTIVATED (student) & HEALTHY (student)

WORKHARD (student) WORKHARD (student) & HASGOALS (student)

EXCELL (student) EXCELL (student) → SUCCEED (student)

On the other hand, when the right side of the rules is instantiated first, the left-hand conditions become subgoals. These subgoals may in turn cause sub- subgoals to be established, and so on until facts are found to match the lowest subgoal conditions. When this form of inference takes place, we say that backward chaining is performed. This form of inference is also known as goal-driven inference since an initial goal establishes the backward direction of the inferring.

For example, in MYCIN the initial goal in a consultation is "Does the patient have a certain disease ?" This causes subgoals to be established such as "are certain bacteria present in the patient?" Determining if certain bacteria are present may require such things as tests on cultures taken from the patient. This process of setting up subgoals to confirm a goal continues until all the subgoals are eventually satisfied or fail. If satisfied, the backward chain is established thereby confirming the main goal.

Some systems use both forward and backward chaining/ reasoning, depending on the type of problem and the information available. Likewise rules may be tested exhaustively or selectively, depending on the control structure.

4.16 EXAMPLE TO COMPARE FORWARD AND BACKWARD CHAINING

In this case, we will revert to our use of symbols for logical statements, in order to clarify the explanation, but we could equally well be using rules about elevators or the weather.

Rules :

Rule 1 : $A \wedge B \rightarrow C$ Rule 2 : $A \rightarrow D$

Rule 3 : $C \wedge D \rightarrow E$ Rule 4 : $B \wedge E \wedge F \rightarrow G$

Rule 5 : $A \wedge E \rightarrow H$ Rule 6 : $D \wedge E \wedge H \rightarrow I$

Facts :

Fact 1 : A Fact 2 : B Fact 3 : F

Goal :

Our goal is to prove H :

- First let us use forward chaining. As our conflict resolution strategy, we will fire rules in the order they appear in the database, starting from rule 1.
- In the initial state, rules 1 and 2 are both triggered. We

Scanned with CamScanner

will start by firing rule 1, which means we add C to our fact database. Next, rule 2 is fired, meaning we add D to our fact database.

- We now have the facts A, B, C, D, F, but we have not yet reached our goal, which is G.
- Now rule 3 is triggered and fired, meaning that fact E is added to the database.
- As a result, rules 4 and 5 are triggered. Rule 4 is fired first, resulting in fact G being added to the database, and then rule 5 is fired, and Fact H is added to the database.
- We have now proved our goal and do not need to go on any further. This deduction is presented in the following table :

Table 4.16.1 : Deduction representation

Facts	Rules triggered	Rule fired
A,B,F	1,2	1
A,B,C,F	2	2
A,B,C,D,F,	3	3
A,B,C,D,E,F	4, 5	4
A,B,C,D,E,F,G	5	5
A,B,C,D,E,F,G,H	6	STOP

- Now we will consider the same problem using backward chaining. To do so, we will use a goals database in addition to the rule and fact databases.
- In this case, the goals database starts with just the conclusion, H, which we want to prove. We will now see which rules would need to fire to lead to this conclusion. Rule 5 is the only one that has H as a conclusion, so to prove H, we must prove the antecedents of rule 5, which are A and E.
- Fact A is already in the database, so we only need to prove the other antecedent, E. Therefore, E is added to the goal database. Once we have proved E, we now know that this is sufficient to prove H, so we can remove H from the goals database.
- So now we attempt to prove Fact E. Rule 3 has E as its conclusion, so to prove E, we must prove the antecedents of rule 3, which are C and D.
- Neither of these facts is in the fact database, so we need to prove both of them. They are both therefore added to the goals database. D is the conclusion of rule 2 and rule 2's antecedent, A, is already in the fact database, so we can conclude D and add it to the fact database.
- Similarly, C is the conclusion of rule 1, and rule 1's antecedents, A and B, are both in the fact database. So, we have now proved all the goals in the goal database and have therefore proved H and can stop.
- **This process is represented in the table below :**

Table 4.16.2 : Process representation

Facts	Goals	Matching rules
A,B,F	H	5
A,B,F	E	3
A,B,F	C,D	1
A,B,C,F	D	2
A,B,C,D,F	D	STOP

- In this case, backward chaining needed to use one fewer rule. If the rule database had a large number of other rules that had A, B and F as their antecedents, then forward chaining might well have been even more inefficient.
- In general, backward chaining is appropriate in cases where there are few possible conclusions (or even just one) and many possible facts, not very many of which are necessarily relevant to the conclusion.
- Forward chaining is more appropriate when there are many possible conclusions. The way in which forward or backward chaining is usually chosen is to consider which way an expert would solve the problem. This is particularly appropriate because rule-based reasoning is often used in expert systems.

Module 4

Scanned with CamScanner

4.17 DIFFERENCE BETWEEN FORWARD AND BACKWARD CHAINING

S. No	Forward chaining	Backward chaining
1.	In this a problem solving technique used by expert system when they are faced with a scenario and have to give a solution or conclusion to this scenario.	It is a reasoning technique employed by expert system to take a goal and prove that this goal founded legitimately according to the rule base it possesses.
2.	The system will work its way through the roles, finding which ones fit and which leads to which goals using **deductive reasoning**.	It is a form of reverse engineering, which is very applicable in situations where there are so many roles that could be applied to a single problem. The system could be shifting through roles before it gets anywhere.
3.	Forward chaining is used when a conclusion is not known before hand and it has to reason its way through to one.	Backward chaining is more appropriate when the conclusion is already known.
4.	If matches conditions and then Generates inferences from those conditions. And then generates inferences from those conditions. These conditions can in turn match other roles. Basically, this takes a set of initial conditions and then draws all inferences from those conditions.	Backward chaining is used for **interrogative applications** (finding items that fulfil certain criteria) one commercial example of a backward chaining application might be finding which insurance policies are covered by a particular reinsurance contract.
5.	Starts with initial facts.	Starts with some hypothesis or goal.
6.	Asks many questions.	Asks a few questions.
7.	Tests all the rules.	Tests some rules.
8.	Slow, because it tests all the rules.	Fast, because it tests fewer rules.
9.	Provides a huge amount of information from just a small amount of data.	Provides a small amount of information from just a small amount of data.
10.	Attempts to infer everything possible from the available information.	Searches only that part of the knowledge base that is relevant to the current problem.
11.	Primarily data-driven.	Goal-driven.
12.	Uses input; searches rules for answer.	Begins with a hypothesis; seeks information until the hypothesis is accepted or rejected.
13.	Top-down reasoning.	Bottom-up reasoning.
14.	Works forward to find conclusions from the facts.	Works backward to find facts that support the hypothesis.
15.	Tends to be breadth-first.	Tends to be depth-first.
16.	Suitable for problems that start from data collection, **e.g.** planning, monitoring, control.	Suitable for problems that start from a hypothesis, **e.g.** diagnosis.
17.	Non-focused because it infers all conclusions, and may answer unrelated questions.	Focused; questions all focused to prove the goal and search as only the part of KB that is related to the problem.
18.	Explanation not facilitated.	Explanation is facilitated.
19.	All data is available.	Data must be acquired interactively (i.e. on demand).
20.	A small number of initial states but a high number of conclusions.	A small number of initial goals and a large number of rules that matches the facts.
21.	Forming a goal is difficult.	Easy to form a goal.

Scanned with CamScanner

4.18 SEMANTIC NETWORKS

GQ. What are semantic networks (or semantic nets) and its classification?

- Semantic networks are an **alternative to predicate logic** as a form of knowledge representation.
- The idea is that we can store our knowledge in the form of a **graph**, with **nodes** representing objects in the world, and **arcs** representing relationships between those objects.
- The physical attributes of a person can be represented as in Fig. 4.18.1

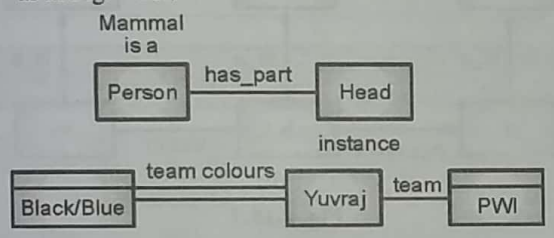


Fig. 4.18.1 : A semantic network

- These values can also be represented in logic as : Is a (person, mammal), instance (Yuvraj, person) team (Yuvraj, PWI). We have already seen how conventional predicates such as lecturer (Poonam) can be written as instance (Poonam, lecturer).
- But, we have a problem: how we can predicate have more than 2 place predicates in semantic nets ? For example score (PWI, India, 20).

  Solution is that, create new nodes to represent new objects either contained or alluded to in the knowledge, game and fixture in the current example. Relate information to nodes and fill up slots.

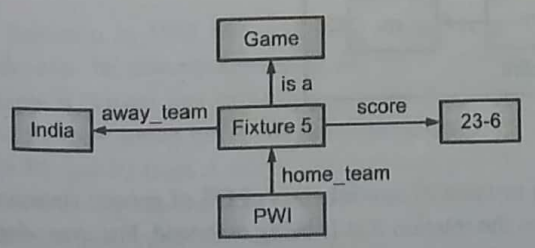


Fig. 4.18.2 : A sematic network for n-place predicate

4.18.1 Advantages, Disadvantages of Semantic Nets

Advantages of semantic nets

1. Semantic network can represent **default values of different categories**.
2. Semantic networks are **simple and easy** to understand.
3. Semantic networks are **easy to translate** in to prologue.
4. Semantic network arcs **represent relationship** between nodes.
5. In semantic networks the **relationships are handled by pointers**.
6. Semantic networks provide good visualization. Being **diagrammatic representation** they are **easy to view**.

Limitations of semantic networks

1. Lack of link names standards; make it difficult to understand the net meaning.
2. Even the nodes naming is not standard. If a node is labelled "**car**" this may mean
   - The class of a car
   - A specific car
   - The concept of a car
3. Answering negative query like "**is XYZ a car**" takes a very long time.
4. Semantic nets are logically inadequate because they cannot define knowledge in a way logic can.
5. Semantic nets can be used better in representing binary relations, but not all types of relations.
6. Logic enhancements have been made and heuristic enhancements have been tried by attaching procedures to the nodes in the semantic nets. The procedures will be executed when the node is activated.

Module 4

4.18.2 Examples for Semantic Representation

GQ. Give semantic representation for following facts : Ram is taller than Hari.

Ram is taller than hari :

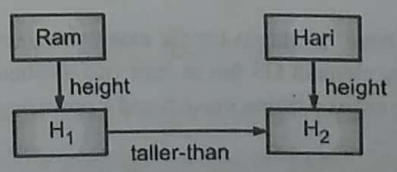


Fig. 4.18.3

Scanned with CamScanner

GQ. Construct semantic net representation for the following :

(i) Pompeian (Marcus), Blacksmith (Marcus)

(ii) Rani gave the green flowered vase to her favorite cousin.

(i) Semantic net representation of "Pompeian (Marcus), Blacksmith (Marcus)" will be,

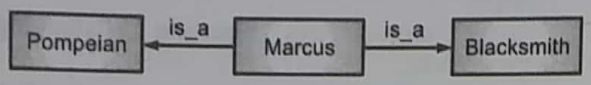


Fig. 4.18.4

(ii) Semantic net representation of "Marry gave the green flowered vase to her favorite cousin" will be,

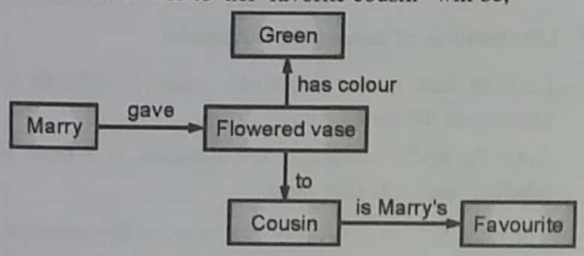


Fig. 4.18.5

GQ.

(i) Represent the following sentence in semantic net :

(a) John gave the book to marry.

(b) Every dog has bitten a postman.

(ii) Bharathiar University Computer Centre, the mini-computer system is a generic node because many mini-computer systems exist and that node has to eater to all of them. On the contrary, Individual of instance nodes explicitly state that they are specific instances of a generic node.

(i) (a) Semantic net representation of sentence "**John gave book to marry**" will be :

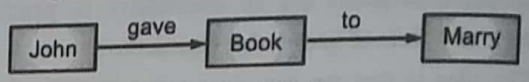


Fig. 4.18.6

(b) Semantic net representation of sentence "**Every dog has bitten a postman**" will be :

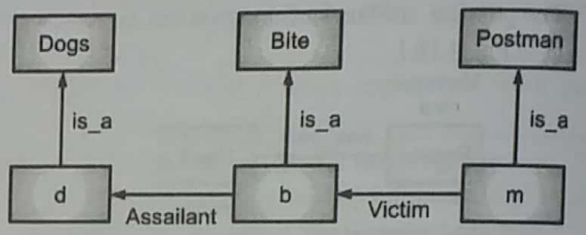


Fig. 4.18.7

The nodes-dogs, bite and hitings and mail earners nodes **d**, b and m represent a particular dog. a particular biting and a particular mail earner This fact can easily be represented by a single net with no partitioning If we represent the fact "Every dog has bitten a postman" or the logic

$\forall(x)$ **Dog(x)** $\rightarrow \exists$ **y Postman-carrier (y)** $\wedge$ **Bite (x, y)**

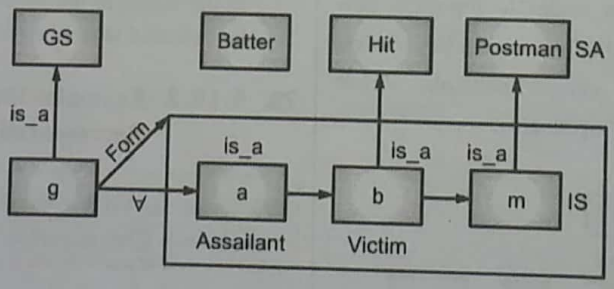


Fig. 4.18.8

(ii) The node '**g**' stands for the **assertion** given above. Node g is an instance of **special class** of **OS** of general statements. Every element GS has at least two attributes, a form which states the relation that is being asserated. For even dog d, there exists a biting event b and a postman m such that d is the assailant of b and m is the victim.

Scanned with CamScanner

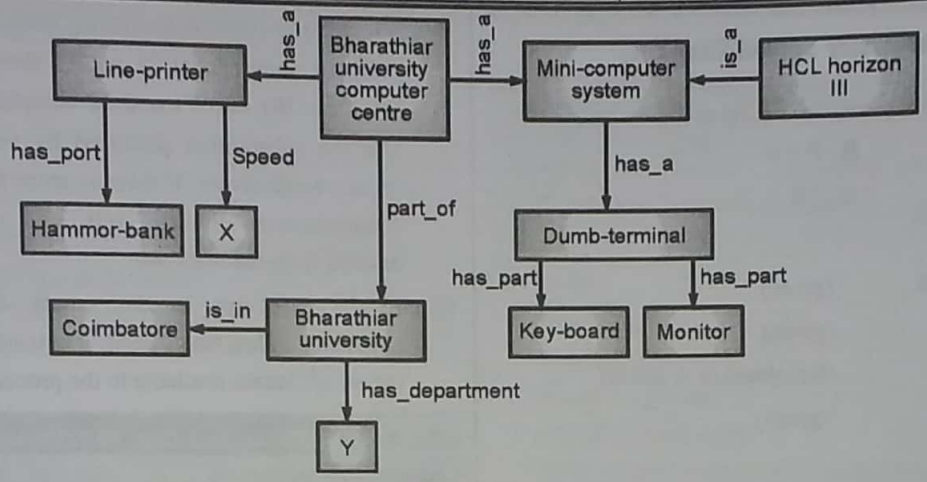


Fig. 4.18.9

4.19 RESOLUTION

GQ. What do you mean by resolution and unification? Explain with example.

Or Write short note on resolution algorithm.

4.19.1 Resolution and Unification

If various statements are given, and we are required to state a conclusion of those statements, then this process is called **Resolution.**

Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form.

Unification is a 'key concept in proofs' by resolutions.

4.19.2 Resolution Algorithm

Robinson in 1965 introduced the resolution principle which can be directly applied to any set of clauses. The principle is "Given any two clauses A and B, if there is a literal P1 in A which has a complementary literal P2 in B, delete P1 and P2 from A and B and construct a disjunction of the remaining clauses. The clause so constructed is called the resolvent of A and B".

For example, consider the following clauses :

A: $P \vee Q \vee R$ B: $\neg P \vee Q \vee R$

C: $\neg Q \vee R$, D = $Q \vee R$

Clause A has the **literal P** which is complementary to **–P** in **B**. Hence, both of them are deleted and a resolvent (disjunction of A and B after the complementary clauses are removed is generated).

That resolvent again has a **literal Q** whose negation is available in C. Hence resolving those two, one has the final resolvent.

A : $P \vee Q \vee R$	(given in the problem)
B : $-P \vee Q \vee R$	(given in the problem)
D: $Q \vee R$	(resolvent of A and B)
C: $-Q \vee R$	(given in the problem)
E: R	(resolvent of C and D)

It is possible to picturize the path of the problem using a **deduction tree**. In fact, it is easier for one to grasp the flow of the problem using the deduction tree. The deduction tree is,

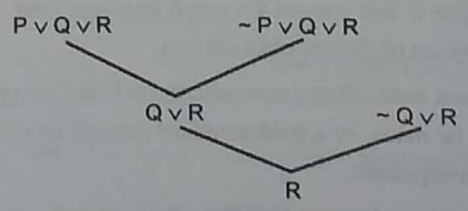


Fig. 4.19.1 : Deduction tree

Sometimes, the resolution might ultimately lead to an empty set or **NIL**. The Following is such an example.

Module 4

Scanned with CamScanner

4.19.3 University Solved Examples

Ex. 4.19.1 : Perform resolution on the set of clauses

A : P ∨ Q ∨ R B : P ∨ R

C : ¬Q D : ¬R

Soln. :

A : P ∨ Q ∨ R	(given)
B¬ P ∨ R	(given)
X : Q ∨ R	(Resolvent of A and B)
C : ¬Q	(given)
Y	
: R	(Resolvent of X and C)
D : ¬R	(given)
Z : NIL	(Resolvent of Y and D)

The deduction tree is,

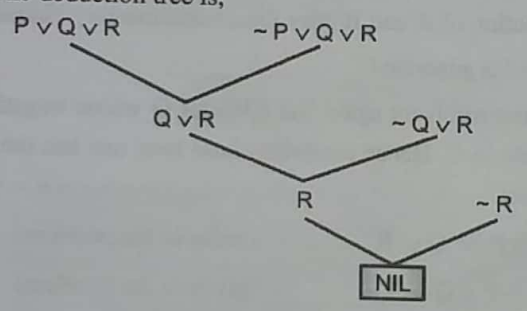


Fig. Ex. 4.19.1

If a A is a formula of predicate calculus, then (x | t). A denotes the formula that results when every occurrence of x in A is substituted by t.

Algorithm Steps

1. Convert all the statements of F to clause form.
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in 1.
3. Repeat until either a contradiction is found, no progress can be made, or a predetermined amount of effort has been expended.

(i) **Select two clauses :** Call these the parent clause.

(ii) **Resolve them together :** The resolvent will be the disjunction of all literal of both parent clauses with appropriate substitution performed and with the following exception. If there is one pair of literals T1 and ¬ T2 such that one of the parent clauses contains T1 and the other contains T2 and if T1and T2 are unifiable, then neither T1 nor T2 should appear in the resolvent. We call T1 and T2 complementary literals. Use the substitution produced by the unification to create the resolvent. If there is more than one, pair of complementary literals, only one pair should be omitted from the resolvent.

(iii) If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure.

UEx. 4.19.2 (MU - Q. 1(c), Dec. 19, 5 Marks)

Consider following facts :

1. It is Humid 2. If it is Humid then it is hot

3. If it is hot and humid then it will rain. Prove that "It will Rain"

Soln. :

Step I : Propositional symbols :

It is humid : H

It is hot : O

It will rain : R

Step II : Propositional logic

(i) H (ii) H → O (iii) H ∧ O → R

Step III : In CNF form

(i) H (ii) ¬ H ∨ O (iii) ¬ H ∨ ¬ O ∨ R

Step IV : We assume negation

It is not raining, i.e. ¬ R

Step V : We form resolution tree.

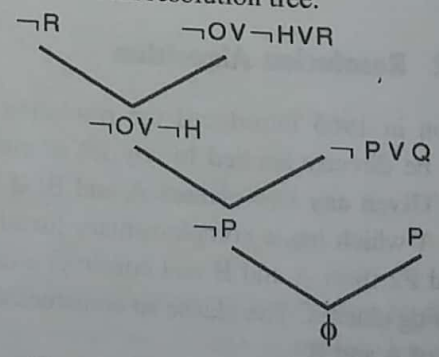


Fig. Ex. 4.19.2 : Resolution Tree

Since the end is empty.

∴ We conclude that it will rain.

Scanned with CamScanner

UEx. 4.19.3 (MU - Q. 3(b), Dec. 15, 10 Marks)

Consider the following axioms:

All people who are graduating are happy.

All happy people smile.

Someone is graduating.

Explain the following :

i) Represent these axioms in first order predicate logic.

ii) Convert each formula to clause form.

iii) Prove that "Is someone smiling?" using resolution technique. Draw the resolution tree.

☑ **Soln. :**

▶ **Step I :** Symbolic logic :

x = people

G = people graduating,

H = happy people,

S = smiling people

▶ **Step II : First order propositional logic**

(i) $\forall x\ G\ \forall x\ H$

(ii) $\forall x\ H,\ \forall x\ S$

(iii) $\exists x\ G \vee \forall x\ G(f(x)) \simeq G(y)$

▶ **Step III : In clause form :**

(i) $\forall x\ G\ \forall x\ H$;

CNF : $\neg G \vee H$

(ii) $\forall x\ H,\ \forall S(x)$;

CNF : $\neg H \vee S(x)$

(iii) Clause form :

$\exists x \neg G \vee H$

(iv) $\forall x \neg H \vee S$

(v) $\exists x\ G$

▶ **Step IV :** We negate the conclusion :

$\neg \exists x\ S(x) \simeq \forall x \neg S$

Resolution tree

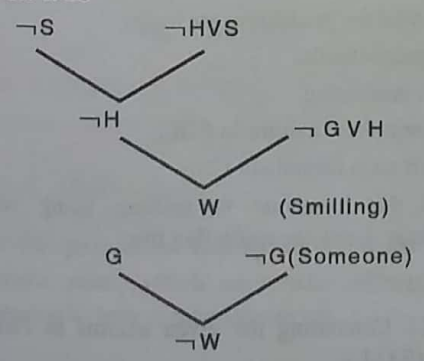


Fig. Ex. 4.19.3

∴ Someone is smiling.

UEx. 4.19.4 (MU - Q. 4(a), Dec. 18, 10 Marks, Q. 4(A), Dec. 17, 10 Marks)

Consider the statements : mammals drink milk, man is mortal, man is a mammal, Tom is a man and we are supposed to prove these.

☑ **Soln. :**

▶ **Step I :** We have

Tom is a man,

Man is a mammal,

Mammals drink milk.

So we have to establish that Tom drinks milk.

First we write down the implication proposition.

M : Mammals drink milk.

Mammals (Tom) → drink (Tom, Milk)

A : Man is mortal.

Man (Tom) → mortal (Tom)

Man is a mammal.

N : Man (Tom) → mammal (Tom)

Tom is a man.

S : Man (Tom)

Goal : Tom drinks (milk)

▶ **Step II :** We note that

(i) Mammal (Tom) → drink (Tom, milk)

(ii) Man (Tom) → mortal (Tom)

(iii) Man (Tom) → mammal (Tom) are propositions and S = man (Tom) is an assertion.

▶ **Step III :** Now in disjunction form,

(i) Not mammal (Tom) OR drink (Tom, Milk)

(ii) Not man (Tom) OR mortal (Tom).

(iii) Not man (Tom) OR mammal (Tom).

▶ **Step III : Resolution Tree :**

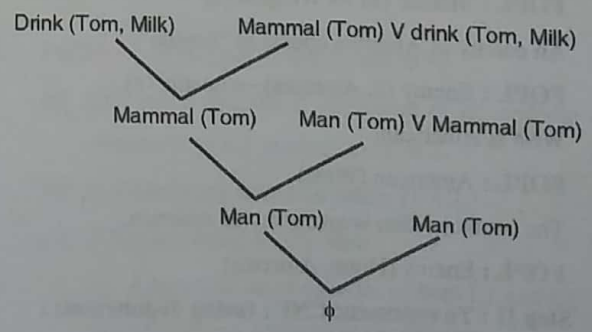


Fig. Ex. 4.19.4

Module 4

Scanned with CamScanner

Thus man (Tom) is not a man but

Tom is a man (∵ We have arrived at empty clause)

∴ Tom does not drink milk is contradiction.

∴ Tom drinks milk.

UEx. 4.19.5 (MU - Q. 4(a), May 16, 10 Marks, Q. 4(a), May 19, 10 Marks)

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles and all of its missiles were sold to it by Colonel West, who is American.

i. Represent the above sentences in first order predicate logic (FOPL).

ii. Convert them to clause form.

iii. Prove that "West is Criminal" using resolution technique.

☑ **Soln. :**

▶ **Step I : To represent sentences in FOPL :**

1. It is a crime for an American to sell weapons to hostile nations.

   Let x : American, y : Weapon

   z : Hostile , M : Missile.

   FOPL : American (x) ∧ weapon (y) ∧ sells (x, y, z) ∧ Hostile (z) → criminal (x).

2. Nono has some missiles :

   ∃x Owns (Nono, x) ∧ Missile (x).

   ∴ FOPL : Owns (Nono, M) and missile (M).

3. All of its missiles were sold by Colonel West.

   FOPL : Missile (x) ∧ Owns (Nono, x) → sells (West, x, Nono).

4. Missiles are weapons.

   FOPL : Missile (x) → Weapon (x)

5. An enemy of America counts as "hostile".

   FOPL : Enemy (x, America) → hostile (x)

6. West is American.

   FOPL : American (West)

7. The country Nono is an enemy of America.

   FOPL : Enemy (Nono, America)

▶ **Step II : To represent CNF : (using disjunction) :**

1. ¬ American (x) ∨ ¬ weapon (y) ∨ ¬ sells (x, y, z) ∨ ¬ Hostile (z) ∨ criminal (x).
2. Owns (Nono, M), Missile (M)
3. ¬ Missile (x) ∨ ¬ owns (None, x) ∨ sells (West, x, Nono)
4. ¬ Missile (x) ∨ weapon (x)
5. (An enemy of America counts as "hostile")

   ¬ Enemy (x, America) ∨ Hostile (x)
6. (West, Who is American)

   American (West).
7. (The country Nono, an enemy of America)

   Enemy (Nono, America).

▶ **Step III : 'Resolution Technique' (using CNF) :**

1. ¬ American (x) ∨ ¬ weapon (y) ∨ ¬ sells (x, y, z) ∨ ¬ Hostile (z) ∨ criminal (x).
2. ¬ Missile (x) ∨ ¬ owns (Nono, x) ∨ sells (West, x, Nono)
3. ¬ enemy (x, America) ∨ Hostile (x)
4. ¬ Missile (x) ∨ weapon (x)
5. Owns (Nono, M)
6. Missile (M)
7. American (West)
8. Enemy (Nano, America)
9. ¬ Criminal (West)

▶ **Step IV : Conclusion**

We discard that West is not criminal. Hence, we conclude that 'West is criminal'.

UEx. 4.19.6 (MU : Dec. 15, 12 Marks)

Consider following axioms :

All people who are graduating are happy.

All happy people smile.

Someone is graduating

(i) Represent these axioms in FOL.

(ii) Convert each formula in CNF

(iii) Prove that someone is smiling using resolution technique. Draw the resolution tree.

☑ **Soln. :**

▶ **Step I :** Converting the given axioms in First Order Logic (F.O.L)

Scanned with CamScanner

(i) Let x stand for people :

(a) $\forall x$: graduating (x) $\rightarrow$ happy (x)

(b) $\forall x$: happy (x) $\rightarrow$ smile (x)

(c) Someone is graduating : $\exists x$: graduating (x)

▶ **Step II :**

Converting First Order Logic (F.O.L.) to conjuctive normal form (C. N. F.)

Note that $(x \rightarrow y)$ is equivalent to $(\neg x \vee y)$

$\therefore$ (a) $\neg$ graduating (x) $\vee$ happy (x)

(b) $\neg$ happy $(x_1) \vee$ smile (x_1)

(c) graduating (x_2)

▶ **Step III :**

We use Resolution Technique

To show that x_3 smile (x_3).

We negate the statement.

i.e. $\neg$ smile (x_3)

Resolution tree

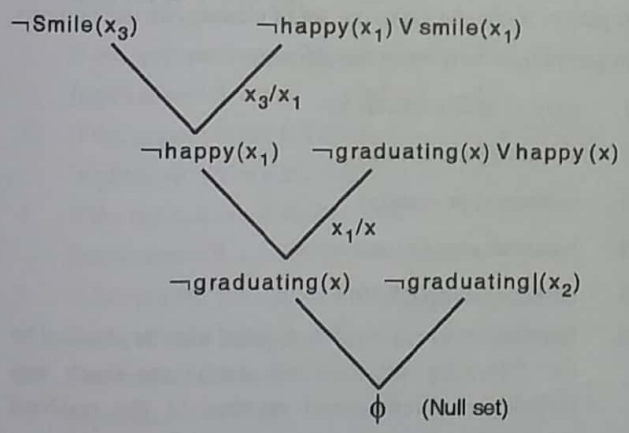


Fig. Ex. 4.19.6

$\therefore$ Our assumption is wrong.

$\therefore$ Someone is smiling.

4.19.4 Unification

1. It is the process of finding substitutions for lifted inference rules, which can make different logical expression to look similar (identical)

2. Unification is a procedure for **determining substitutions** needed to make two first order logic expressions match.

3. Unification is important component of all **first order logic inference algorithms**.

4. The unification algorithm takes two sentences and returns a unifier for them, if one exists.

❑ **Unifier :** A substitution that make two clauses resolvable is called a unifier and the process of identifying such unifiers is carried out by the unification algorithm.

The unification algorithm tries to find out the **most General Unifier (MGU)** between a given set of atomic formulae. Any substitution that makes 2 and more expression equal is called as verifying lineal.

☞ **Algorithm : Unify (L1, L2)**

1. If L1 or L2 are both variables or constants, then:
   (a) If L1 and L2 are identical, then return NIL.
   (b) Else if L1 is a variable, then if L1 occurs in L2 then return {FAIL}, else return (L2/L1).
   (c) Else if L2 is a variable then if L2 occurs in L1 then return {FAIL}, else (L1/L2).
   (d) Else return {FAIL}.
2. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.
3. If L1 and L2 have a different number of arguments, then return {FAIL}.
4. Set SUBST to NIL. (AT the end of this procedure, SUBST will contain all the substitutions unify L1 and L2).
5. For **i ← 1 to** number of arguments in L1:
   (a) Call unify with the ith argument of L1 and the ith argument of L2, putting result in S.
   (b) If S contains FAIL then return {FAIL}.
   (c) If S is not equal to NIL then :
      (i) Apply S to the remainder of both L1 and L2.
      (ii) SUBST : = APPENDS (S, SUBST).
6. Return SUBST.

Module 4

Scanned with CamScanner

4.19.5 Conflict Resolution

GQ. Explain conflict resolution. Explain the term refutation. And let's take an example to explain this :

The following statements are assumed to be true :

1. Steve only likes easy courses.
2. Science courses are hard.
3. All the courses in the basket weaving department are easy.
4. BK30I is a basket weaving course. We ask : What course would steve like?

OR

GQ. What is conflict resolution? Illustrate with an example in any production system.

- **Definition :** Conflict set is the set of rules that have their conditions satisfied by **working memory elements**. Conflict resolution normally selects a single rule to fire.

The popular conflict resolution mechanisms are :

1. Refractory 2. Recency 3. Specificity

1. **Refractory :** A rule should not be allowed to fire more than once on the same data. Discards executed rules from the conflict set. Prevents undesired loops.
2. **Recency :** Rank instantiations in terms of the recency of the elements in the premise of the rule. Rules which use more recent data are preferred. Working memory elements are time-tagged indicating at what cycle each fact was added to working memory.
3. **Specificity :** Rules which have a greater number of conditions and are therefore more difficult to satisfy, are preferred to more general rules with fewer conditions. More specific rules are 'better' because they take more of the data into account.

4.19.6 Refutation

UQ. Explain Resolution by Refutation with suitable example. (MU - Q. 3(a), May 18, 10 Marks)

Refutation is nothing but a technique that a resolution procedure used to prove a statement, i.e., an attempt to show that negation of the statement produces a contradiction with known statements.

We consider an example :

The following statements are assumed to be true :

1. Steve only likes easy courses.
2. Science courses are hard.
3. All the courses in the basket-weaving department are easy.
4. BK 101 is a basket-weaving course.

We ask : What course would Steve like ?

The predicate logic encoding of the premises of the previous problem is as follows:

1. $\forall$(x) easy(x) $\rightarrow$ likes(steve, x)
2. $\forall$ (x) science(x) $\rightarrow$ easy(x)
3. $\forall$ (x) basket weaving(x) $\rightarrow$ easy(x)
4. basket weaving (BK30l)

The conclusion is encoded as, likes(steve, x).

First we put our premises in the clause form and the negation of conclusion to our set of clauses (we use numbers in parentheses to number the clauses):

1. easy(x) or likes(steve, x)
2. science(x) or -easy(x)
3. science(x) or -easy(x)
4. basketweaving(x) or easy(x)
5. basketweaving(BK301)
6. **likes(steve, x) :** A resolution proof may be obtained by the following sequence of resolutions (each step includes a parenthesized number of the resolvent generated in the current step; 1 and 5 means that we resolve clauses (1) and (5).
7. 1 and 6 yields resolvent-easy(x).
8. 4 and 7 yields resolvent ~basketweaving(x).
9. 5 and 8 yields empty clause; the substitution X/BK30I is produced by the unification algorithm which says that the only wff of the form likes (steve, x) which follows from the premises is likes(steve,BK30l).

Thus, resolution gives us a way to find additional assumptions (in this case **x = BK301**) which make our theorem true.

Scanned with CamScanner

4.19.7 Example based on Propositional Variables

GQ.

(i) If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Use these propositional variables :

Y = unicorn is mYthical R = unicorn is mortal

M = unicorn is a maMmal H = unicorn is Horned

G = unicorn is magical

Convert the English into propositional logic implicative form and conjunctive normal form (CNF). The first one is done for you as an example. (Note : "immortal" means "not mortal.")

(ii) Perform Resolute on the set of clauses

A : P ∨ Q ∨ R , B : ~ P ∨ R, C : ~ Q D : ~ R

(i)

1. If the unicorn is mYthical, then it is not moRtal.
   Implicative Y => ¬ R. CNF (¬ Y ∨ ¬ R).
2. If the unicorn is not mYthical, then it is mortal
   Implicative ¬ Y => R. CNF (Y ∨ R).
3. If the unicorn is not mYthical, then it is a maMmal.
   Implicative ¬ Y => M. CNF (Y ∨ M).
4. If the unicorn is not moRtal, then it is Horned.
   Implicative ¬ R => H. CNF (R ∨ H).
5. If the unicorn is a maMmal, then it is Horned.
   Implicative M => H. CNF (¬ M ∨ H).
6. The unicorn is maGical if it is Horned.
   Implicative H => G. CNF (¬ H ∨ G).

(ii)

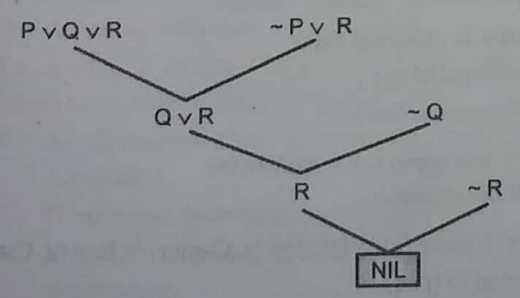


Fig. 4.19.2

Ex. 4.19.6 : Convert the following statement into clause form : (∀x) (∃y) (∀z) [p (x, y) ∧ q (x, z) → r (z, x)]

☑ **Soln. :**

Applying rule (1), we have

(∀x) (∃y) (∀z) [¬ p (x, y) ∧ q (x, z) ∨ r (z, x)]

Applying rule (2), we get

(∀x) (∃y) (∀z) [¬ p (x, y) ∨ ¬ q (x, z) ∨ r (z, x)]

Applying rule (5); we get

(∀x) (∀z) [¬ p (x, A (x)) ∨ ¬ q (x, z) ∨ r (z, x)]

By applying rule (6), we get

[¬ p (x, A (x)) ∨ ¬ q (x, z) ∨ r (z, x)]

By applying rule (7)

¬ p (x, A (x)) ∨ ¬ q (x, z) ∨ r (z, x)

The above expression is a clausal form.

Ex. 4.19.7 : Convert the following sentences into propositional logic and prove by resolution that :

"Schweta will go to college"

1. Schweta likes maths and she likes stories.
2. If she likes maths then she likes algebra.
3. If she likes algebra and likes physics then she will go to college.
4. She does not like stories or she likes physics.
5. She does not like chemistry and history.

Module 4

☑ **Soln. : Propositional logic :**

Let Schweta like maths = M

Schweta like stories = S

Schweta like algebra = A

Schweta like physics = P

Schweta will to go to college = C

Schweta likes history = H

Schweta likes Chemisty = Z

(1) M ∧ S (2) M → A (3) A ∧ P → C

(4) ¬ S ∨ P (5) ¬ H ∧ ¬ Z

Clausal form

1. MA 2. S

3. ¬ M ∨ A 4. ¬ (A ∧ P) ∨ C = ¬ ∨ ¬ P ∨ C

5. ¬ S ∨ P 6. ¬ H 7. ¬ Z

Scanned with CamScanner

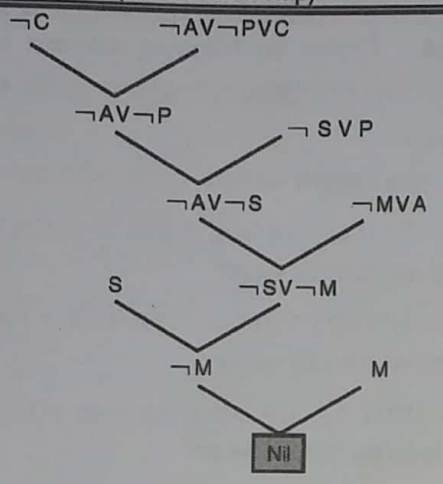


Fig. Ex. 4.19.7

Resolution proof for 'Alice will go to college'

Ex. 4.19.8 : Convert the following sentences into propositional logic and prove by resolution that, "What course would Steeve like?"

1. Steeve only likes easy course
2. Science courses are hard.
3. All the courses in CSE department are easy.
4. DBMS is a CSE course.

Use Resolution to prove that "What course would Steeve like ?"

☑ **Soln. :**

▶ **Step I : First order propositional logic**

1. ∀x course (x) ∧ easy (x) → likes (Steeve, x)
2. Course (science) → ¬ as y (science)
3. ∀a : course (a) ∧ CSE (a) → easy (a)
4. CSE (DBMS)

▶ **Step II : Clausal form**

1. ¬ course (x) ∨ ¬ easy (x) ∨ like (Steeve, x)
2. ¬ course (science) ¬ ∨ ¬ easy (science).
3. ¬ course (a) ∨ ¬ CSE (a) ∨ easy (a)
4. CSE (DBMS).

▶ **Step III :** Conclusion

What course would Steeve like ?

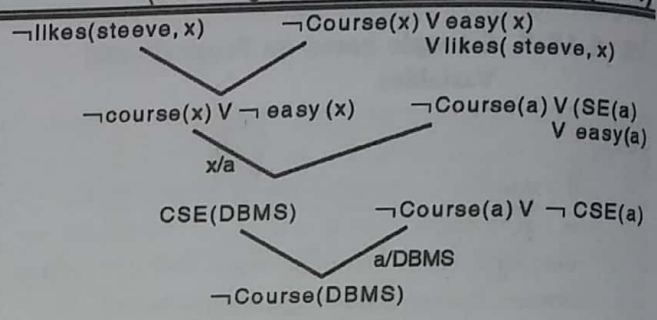


Fig. Ex. 4.19.8

∴ Steeve likes DBMS.

Ex. 4.19.9

1. Markus was a man.
2. Markus was a pompiean.
3. All pompieans were Romans.
4. Caesor ws a ruler.
5. All Romans were either loyal to Caesor or hated him.
6. Everyone is loyal to someone.
7. People only try to assasinate rulers that they are not loyal to.
8. Markus tried to assasinate Caesor.

Proof by resolution that Markus hates Caesor.

☑ **Soln. :**

▶ **Step I : First Order Predicate Logic (FOPL)**

1. Man (Markus)
2. Pompiean (Markus)
3. ∀x : Pompiean (x) → Roman (x)
4. Ruler (Caesor)
5. ∀a : Roman (a) → Loyal to (a, Caesor) ∨ hate (a, Caesor)
6. ∀p, ∃q : Loyal to (p, q)
7. ∀y, ∀z : People (y) ∧ ruler (z) ∧ assassinate (y, z). → ¬ loyal to (y, z).
8. Assassinate (Markus, Caesor).

▶ **Step II : Clausal form**

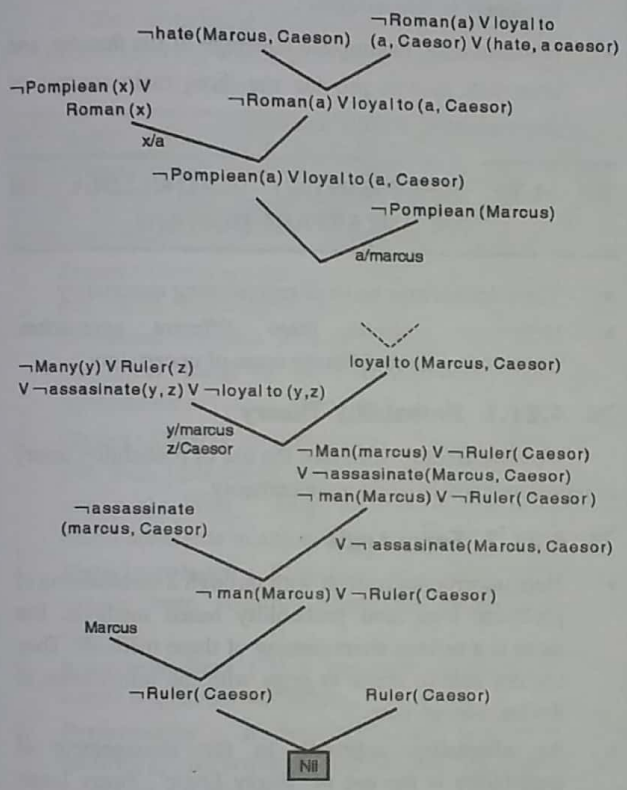
1. Man (Markus)
2. Pompiean (Markus)
3. ¬ Pompiean (x) ∨ Roman (x)
4. Ruler (Caesor)
5. ¬ Roman (a) ∨ loyal to (a, Caesor) ∨ hate (a, Caesor)
6. loyal to (p, q)
7. ¬ (y) ∨ ¬ Ruler (z) ∨ ¬ assasinate (y, z) V → loyal to (y, z)
8. Assasinate (Markus, Caesor)

Scanned with CamScanner

Step III : Conclusion

hate (Markus, Caesor)

Resolution Tree



A resolution proof
Fig. Ex. 4.19.9

4.20 UNCERTAIN KNOWLEDGE AND REASONING

- In a reasoning system, there are several types of uncertainty. Reasoning under uncertainty research in AI is 'focused on uncertainty of truth value' in order to find the values other than True and False.
- To develop a system that reasons with uncertainty means to provide the following :
  1. An explanation about the origin and nature of the uncertainty.
  2. To represent uncertainty in a formal language.
  3. A set of inference rules that derive uncertain conclusions.
  4. An efficient memory-control mechanism for uncertainty management.

4.20.1 Non-monotonic Logics

- A reasoning system is monotonic if the truthfulness of a conclusion does not change when new information is added to the system.
- In contrast, in a system doing non-monotonic reasoning the set of conclusions may either grow or shrink when new information is obtained.
- Simply speaking, the truth value of propositions in a non-monotonic logic can be classified into the following types :
  1. Facts that are definitely true, such as 'Crow is a bird'.
  2. Default rules that are normally true, such as 'Birds fly'.
  3. Tentative conclusions that are true, such as 'Crow flies'.
- **Remark :** When an inconsistency is recognised, only the truth value of the last type is changed.

4.20.2 Probabilistic Reasoning

- The basic idea of probabilistic reasoning is to use probability theory to represent and process uncertainty.
- Though no conclusion from probability theory is absolutely true, the one with the highest probability is preferred. Under certain conditions (assumptions) probability theory gives the optimum solutions.
- "It is remarkable that a science which began with the consideration of games of chance should become the most important object of human knowledge. The most important questions of life are, for the most part, really only problems of probability.The theory of probabilities is at bottom nothing but common sense reduced to Calculus."

  – **Pierre Simon de Laplace(1812).**

We extend the basic Boolean connectives to probability functions :

1. **Negation :** $P(\neg A) = 1 - P(A)$
2. **Conjunction :** $P(A \wedge B) = P(A) \cdot P(B)$, if A and B are independent of each other.
3. **Disjunction :** $P(A \vee B) = P(A) + P(B)$, if A and B never happen at the same time.
4. **Conditional :** Probability of B given A is

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

From which Bayes' theorem is derived

Module 4

Scanned with CamScanner

5. Bayesian reasoning determines the probability of an event with uncertain knowledge.

   It is a way to calculate the value of P (B/A) with the knowledge of P (A/B).

4.20.3 Bayesian Networks

- Bayesian Networks are directed graphs in which the nodes represent a variables of interest and the links represent informational or causal dependencies **among** the variables.
- The strength of dependency is represented by conditional probabilities.

4.20.4 Challenges to Probabilistic Approaches

1. Unknown probability values.
2. Inconsistent probability assignments.
3. Computational expense.

Considering the uncertainty in probability judgements, we go further to study 'imprecise probability'.

4.20.5 Fuzzy Logic

- 'Fuzzy logic' is the reflection of 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 a matter of 'degree'. Fuzzy logic uses a 'degree of membership' which is a real number in [0, 1].

4.20.6 Basic Fuzzy Operators

1. **Negation :** $M(\neg A) = 1 - M(A)$.
2. **Conjunction :** $M(A \wedge B) = \text{Min}\ \{M(A), M(B)\}$.
3. **Disjunction :** $M(A \vee B) = \text{Max}\ \{M(A), M(B)\}$.

In building a Fuzzy system, the designer needs to provide all membership functions included in it, by considering how the concepts are used by average people.

4.20.7 Challenges to Fuzzy Approaches

1. Degree of membership is often dependent on content.
2. General purpose fuzzy rules are hard to obtain.

4.20.8 Causes of Uncertainty in AI

- In machine learning the biggest source of difficulty for beginners is 'Uncertainty'.
- Noise in data, incomplete coverage of the domain, and imperfect models provide the three main sources of uncertainty in machine learning.

4.21 REPRESENTING KNOWLEDGE IN AN UNCERTAIN DOMAIN

- There are various ways of representing uncertainty.
- Here we consider three different approaches, representing three different areas of uncertainty :

4.21.1 Probability Theory

We have already discussed the use of probability theory to represent knowledge in an uncertainty.

4.21.2 Fuzzy Logic

- Here uncertainty is dealt with through a combination of predicate logic and probability based methods. But there is a serious short coming of these methods. They are not able to come to grips with the information in the knowledge-base.
- An alternative approach to the management of uncertainty is the use of 'Fuzzy Logic'. Fuzzy Logic provides a systematic framework for dealing with fuzzy quantifiers, e.g., most, many, about 0.7, etc.
- Fuzzy Logic considers both predicate logic and probability theory, and deals with different types of uncertainty within a single conceptual framework.
- In Fuzzy Logic, the deduction of a conclusion from a set of premises is reduced. And from a non linear program through the application of projection and extension principles, a solution is obtained.

4.21.3 Truth Maintenance System : (TMS)

- To choose the actions, reasoning programs must be able to make assumptions and then revise their beliefs when discoveries contradict these assumptions.
- The Truth Maintenance System (TMS) is a problem solver subsystem for performing these functions by recording and maintaining the reasons for program beliefs. Such recorded reasons are useful in constructing explanations of program actions and in guiding the course of action of a problem solver.

Scanned with CamScanner

- TMS can be best visualised in terms of graphs.
- It stores the latest truth value of any predicate. The system is developed with the idea that truthfulness of a predicate can change with the time, as new knowledge is added or existing knowledge is updated. It keeps a record showing which items of knowledge is currently believed or disbelieved.

4.22 REPRESENTATION OF KNOWLEDGE

In AI systems, knowledge is represented in the following manner.

1. **Events :** Events are the occurrence of things in the real world. Anything which happens in real time are considered as events. It is an important element as it is the initial thing to be considered in knowledge representation.
2. **Object :** Objects are nothing but facts that are actually true. Such facts can be universal truth such as "The sun sets in the West." "Dogs are faithful", or any facts which holds true in any events.
3. **Meta-knowledge :** It is those knowledge which are already been acquired either by human brain or machine.
4. **Knowledge-base :** It is the core component of the agents acquiring knowledge.
5. **Performance :** It describes about how good the knowledge is acquired and it can be applied to machines.

4.22.1 Categories of Knowledge

There are two main categories of knowledge :

1. Tacit knowledge	2. Explicit knowledge

1. Tacit knowledge

- Tacit knowledge is the knowledge which exists within a human being. It does correspond to informal or implicit type of knowledge.
- It is quite difficult to articulate formally and is also difficult to communicate and share.

2. Explicit knowledge

- Explicit knowledge is the knowledge which exist outside a human being.
- It corresponds to formal type of knowledge, it is easier to articulate compared to tacit knowledge and is easier to share, store or even process.

4.22.2 Types of Knowledge Representation

Types of knowledge representations are as following :

A. Declarative knowledge
B. Procedural knowledge
C. Matter knowledge
D. Heuristic knowledge
E. Structural knowledge

A. Declarative Knowledge

- It is a segment of knowledge which stores factual information in a memory and it seems to be static in nature.
- These can be things or events or processes and the domain of such knowledge finds the relation between events or things.

B. Procedural Knowledge

- This knowledge is less general than declarative knowledge and is also known as imperative knowledge.
- It can have the potential to declare the accomplishment of a particular thing. It is generally used by modern mobile robots where they can be planned to attack into a building or perform navigation in a room.
- If we consider declarative knowledge implanted into a modern robot, it will be assigned just a map instead of detailed plan of attack into a building.

C. Meta Knowledge

- In the field of AI, the knowledge of pre-defined knowledge is known as meta-knowledge. A study of planning, tagging and learning are some of the examples of meta knowledge.
- This model tends to change with time and utilise a different specification. A knowledge engineer may utilise different forms of meta-knowledge given below :
- Accuracy, Applicability, Assessment, Consistency, Completeness, Disambiguation, Justification life span, Purpose, Source, Reliability.

D. Heuristic Knowledge

- This knowledge is also known as shallow knowledge and it follows the principle of thumb rule.
- It is very efficient in reasoning process as it solves the problems based on the records of past problems or the problems which are compiled by experts.
- It provides knowledge based on the experiences, it gathered during the past problems

Module 4

Scanned with CamScanner

E. Structural Knowledge

- This is the most basic knowledge used and applied in problem solving.
- It tries to find out a relationship between concepts and objects. It describes relationship between various concepts such as kind of, part of, and grouping of something.
- Let us describe a relationship of the knowledge along with a flowchart.

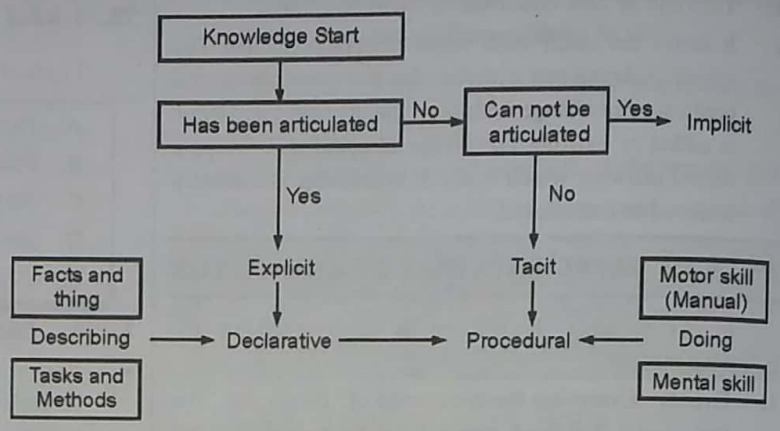


Fig. 4.22.1 : Relationship among types of knowledge

- Here, declarative knowledge is represented as **describing one** and procedural knowledge is represented as **doing one**.
- One more inference is, declarative knowledge is termed as explicit while procedural knowledge is termed as tacit.
- If the knowledge can be articulated, it is a declarative knowledge and if cannot be articulated, it is known as procedural knowledge.

4.22.3 Causes of Uncertainty

- 1. uncertainties relate to situations where it is impossible to exactly describe state of future outcomes.
- 2. In climate change adaptation, uncertainties arise from different sources, e.g. natural climate variability, future emissions, modelling, behavioural, socio-economic and technological responses and ecological dynamics.
- 3. Uncertainty can arise from multiple causes and situations; the lack of information or on the contrary the abundance of information with conflicting different pieces of information, measurement errors, linguistic ambiguity, or the subjectivity of opinions.
- 4. Reducing uncertainties is a major challenge in climate change adaptation. Some uncertainty can be reduced by acquiring knowledge while intrinsic uncertainty cannot be reduced.

4.23 THE SEMANTICS OF BELIEF NETWORK

UQ. Define Belief Network. Describe the steps of constructing belief network with an example. What types of inferences can be drawn from that ?

(MU - Q. 4(b), May 19, 10 Marks)

- A semantic network, or frame network is a knowledge base that represents semantic relations between concepts in a network. This is often used as an form of knowledge representation.
- It is a directed or undirected graph consisting of vertices, which represent concepts, and edges, which represent semantic relations between concepts, mapping or connecting semantic fields.
- A semantic network may be initiated as, for example a graph database or a concept map.

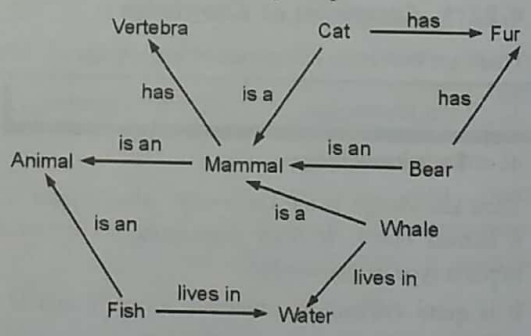


Fig. 4.23.1: Example of semantic network

Semantic networks are used in natural language processing applications such as, semantic parsing and word-sense disambiguation.

Scanned with CamScanner

4.23.1 Basics of Semantic Networks

- A semantic network is used when one has knowledge that is best understood as a set of concepts that are related to one another.
- Most semantic networks are cognitively based. They also consists of arcs and nodes which can be organised into a taxonomic hierarchy.
- Semantic networks contributed ideas of spreading activation, inheritance, nodes as proto-objects.

4.23.2 Examples of Semantic Network

1. WordNet

- An example of semantic network is WordNet. It groups English words into sets of synonyms called synsets, provides short, general definitions and records various semantic relations between these synonym sets.
- Some of the most common semantic relations defined are Meronymy (A is a meronym of B if A is a part of B), holonymy (B is a holonym of A if B contains A), hyponymy(Or troponymy) (A is subordinate of B, A is a kind of B), hypernymy (A is a subordinate of B), synonymy (A denotes the same as B), and antonymy (A denotes the opposite of B).
- In the social sciences people sometimes use the term semantic network to refer to co-occurrence networks.
- The basic idea is that words that occur in a unit of text, e.g. a sentence, are semantically related to one another. Ties based on co-occurrence can then be used to construct semantic networks.

4.23.3 Purpose of Semantics

- The purpose of semantics is to propose exact meanings of words and phrases, and remove confusion, which might lead the readers to believe a word has many possible meanings.
- It makes a relationship between a word and sentence through their meanings.

4.23.4 Focus on Semantics

- The study of semantics looks at how meaning works in language, and because of this it often uses native speaker intuitions about the meaning of words and phrases to base research on.
- We all understand semantics already on a subconscious level, it is how we understand each other when we speak.

4.24 INFERENCE IN BELIEF NETWORK

- An inference is an idea or conclusion that is drawn from evidence and reasoning. An inference is an educated guess.
- We **learn about** some things by experiencing them first hand, but we gain other knowledge by inference-the process of inferring things based on what is already known.
- Belief networks are popular tools for encoding uncertainty in expert systems.
- These networks rely on inference algorithms to compute beliefs in the context of observed evidence, network evidence, expert systems, join tree, probabilistic inference, probability propagation, reasoning under uncertainty.

Module 4

4.24.1 Purpose of Belief Network

- A belief network specifies a joint probability distribution from which arbitrary conditional probabilities can be derived.
- A network can be queried by asking for the conditional probability of any variables conditioned on the values of any other variables.

Scanned with CamScanner

4.24.2 Method of Belief Network

- A belief network is a directed model of conditional dependence among a set of random variables.
  - The precise statement of a conditional independence in a belief network takes into account the directionality.
  - To define a belief network, we begin with a set of random variables that represent all of the features of the model. Let the variables be $\{x_1, x_2, \ldots, x_n\}$ with the total ordering as $x_1, x_2, \ldots, x_n$.
- We define the **parents** of random variable x_i, written parents (x_i), to be a minimal set of predecessors of x_i in the total ordering such that the other predecessors of x_i are conditionally independent of x_i.
- That is, parents $(x_i) \subseteq \{x_1, x_2, \ldots, x_{i-1}\}$ such that

  $P(x_i / x_{i-1}, \ldots, x_1) = P(x_i / \text{parents}(x_i))$.
- The probability over all of the variables, $P(x_1, x_2, \ldots, x_n)$ is called the 'joint probability distribution'.
- A belief network defines a **factorization** of the joint probability distribution, where the conditional probabilities from factors that are multiplied together.

4.24.3 PPTC Algorithm

- Belief networks are popular tools for encoding uncertainty in expert systems.
- These networks rely on inference algorithms to compute beliefs in the context of observed evidence.
- One established method for exact inference on belief networks is the probability propagation in trees of clusters (PPTC) algorithm.
- PPTC converts belief network into a secondary structure, then computer probabilities by manipulating the secondary structure.

Chapter Ends...

❑❑❑

Scanned with CamScanner

Module 5

CHAPTER 5

Planning and Learning

Syllabus

The planning problem, Planning with state space search, Partial order planning, Hierarchical planning, Conditional Planning.

Learning : Forms of Learning, Theory of Learning, PAC learning. Introduction to statistical learning (Introduction only)

Introduction to reinforcement learning: Learning from Rewards, Passive Reinforcement Learning, Active reinforcement Learning

5.1 Components of Planning 5-3
5.1.1 Problem Solving Vs. Planning 5-3
5.2 Problems Associated With Planning Agent 5-4
5.2.1 Planning Agent 5-4
5.2.2 Three Key Ideas behind Planning 5-4
5.2.3 Two Types of Agents 5-5
5.3 Goal Representation 5-6
5.4 Planning with State Space Research 5-7
5.4.1 Forward State-Space Search 5-7
5.4.2 Backward State-Space Search 5-8
5.5 Partial order planning 5-8
UQ. Explain a partial order planner with an example. **(MU - Q. 5(c), Dec. 15, Q. 4(b), May 17, Q. 5(a), May 2019, 10 Marks)** 5-8
5.5.1 Causal Link in Partial Order Planning 5-9
5.5.2 Working of a Partial Order Planner 5-9
5.5.3 Example for Partial Order Planner 5-9
5.6 Hierrarcial Planning 5-10
5.7 Hierarchical Task Network Planning 5-11
5.7.1 Steps to Create Planning Hierarchy 5-12
5.7.2 Steps in Planning 5-12
5.7.3 Key Steps in Hierarchy of Strategies 5-12
5.7.4 Three Levels of Strategy 5-12
5.7.5 Strategy Intent 5-12
5.7.6 Hierarchical Goals in Strategic Management 5-12
5.7.7 Hierarchical Arrangement of Objective 5-12
5.7.8 Advantages of Hierarchy of Objectives 5-12
5.7.9 How Hierarchical Structure Works ? 5-13
5.7.10 Some Examples of Hierarchy 5-13
5.7.11 Advantages and Limitations of Hierarchy 5-13
5.7.12 Characteristics of Hierarchy Data Model 5-13
5.7.13 Importance of Hierarchy 5-13
5.8 Conditional Planning 5-13
5.8.1 Difference in Hierarchical and Conditional Planning 5-13
5.9 Forms of Learning 5-14
5.10 Learning Algorithms 5-14
5.11 Supervised Learning 5-14

Scanned with CamScanner

UQ. What is Supervised Learning ? Give example of each.

(MU - Q. 1(e), May 17, Q. 1(e), May 19, 5 Marks) 5-14

5.11.1 How Supervised Learning Works? 5-15

5.11.2 Advantages of Supervised Learning....5-16

5.11.3 Disadvantages of Supervised Learning 5-16

5.12 Unsupervised Learning 5-16

UQ. What is unsupervised learning ? Give example of each.

(MU - Q. 1(e), May 17, Q. 1(e), May 19, 5 Marks) 5-16

5.12.1 Types of Unsupervised Learning Algorithm 5-17

5.12.2 Advantages of Unsupervised Learning 5-18

5.12.3 Disadvantages of Unsupervised Learning 5-18

5.12.4 Difference between Supervised and Unsupervised Learning 5-18

5.13 Reinforcement Learning 5-19

5.13.1 Approaches to Implement Reinforcement Learning 5-20

5.13.2 Challenges of Reinforcement Learning 5-21

5.13.3 Applications of Reinforcement Learning 5-21

5.13.4 Reinforcement Learning vs. Supervised Learning 5-21

5.14 General Learning Model 5-21

5.15 Techniques used in Learning 5-22

5.15.1 Learning by Memorization (Rote Learning) 5-22

5.15.2 Learning by Direct Instruction 5-22

5.15.3 Learning by Analogy 5-22

5.15.4 Learning by Induction 5-23

5.15.5 Learning by Deduction 5-23

5.15.6 Neural Network 5-23

5.16 Statistical Learning in Artificial Intelligence 5-23

5.17 Theory of Learning 5-23

5.17.1 Machine-Learning Theory 5-23

5.17.2 Artificial Learning and Machine Learning 5-23

5.17.3 A.I and Deep Learning 5-23

5.17.4 Techniques of Deep Learning 5-24

5.18 PAc Learning 5-24

5.18.1 Analysis of PAC 5-24

5.18.2 Epsilon in PAC Learning 5-24

5.18.3 Agnostic PAC Learning 5-25

5.19 Reinforcement Learning (R.I) 5-25

5.19.1 Example : We Consider a Problem 5-25

5.19.2 Method of R.L. 5-25

5.19.3 Types of Reinforcement 5-25

5.19.4 Practical Applications of R.L. 5-26

5.19.5 Challenges with RL 5-26

5.19.6 Challenges in Deep Reinforcement Learning Research and Applications 5-26

5.20 Learning from Rewards and Punishment 5-27

5.20.1 Effectiveness of Rewards and Punishments 5-27

5.21 Passive Reinforcement Learning and Active Reinforcement Learning 5-27

5.22 Passive Reinforcement Learning 5-27

5.22.1 Direct Utility Estimation 5-27

5.22.2 Adaptive Dynamic Programming (ADP) 5-28

5.22.3 Temporal Difference Learning (TDL) 5-28

5.22.4 Active Reinforcement Learning 5-28

5.23 Q-learning 5-28

5.23.1 Goal of the Agent 5-28

5.23.2 Influence of Variables : Learning Rate 5-29

5.23.3 Discount Factor 5-29

5.23.4 Initial Conditions (Q_0) 5-29

5.23.5 Implementation 5-29

5.23.6 Quantization 5-30

5.23.7 Variants (Deep Q-learning) 5-30

5.23.8 Delayed Q-learning 5-30

5.23.9 Limitations 5-30

5.23.10 Temporal Difference Learning (TD) 5-30

5.23.11 TD Algorithm in Neuroscience 5-30

5.24 Introduction to Statistical Learning 5-31

- **Chapter Ends. 5-31**

Scanned with CamScanner

5.1 COMPONENTS OF PLANNING

GQ. Explain the different components of planning system. **OR** Given the components of planning system and briefly explain them. **OR** Explain the various components of planning system. How can you represent a planning action?

Different components of planning system are as follows :

1. Choose best rule to apply
2. Apply the chosen rule
3. Detecting when a solution has been found
4. Detecting dead ends
5. Repairing an almost correct solution

- **1. Choose best rule :** Isolate set of differences between the desired goal state and the current state, identify those rules that are relevant to reducing those differences (means-end analysis). If several rules found then choose the best using heuristic information.
- **2. Apply rules :** In simple systems, applying rules is easy. Each rule simply has specified the problem state that would result from its application. 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.
- **3. Detecting a solution :** A planning system has succeeded in finding a solution to a problem when it has found a sequence of operators that transform the initial problem state into the goal state. In simple problem-solving systems we know the solution by a straightforward match of the state description. But, in complex problem different reasoning mechanisms can be used to describe the problem states, that reasoning mechanisms could be used to discover when a solution had been found.
- **4. Detecting dead ends :** A planning system must be able to detect when it is exploring a path that can never lead to a solution. Above same reasoning mechanism can be used to detect dead ends. In search process reasoning forward from initial state, it can prune any path that lead to a state from which goal state cannot be reached. Similarly backward reasoning, some states can be pruned from the search space.
- **5. Repairing an almost correct solution :** In completely decomposable problems can be solve the sub problems and combine the sub solutions yield a solution to the original problem. But try to solving nearly decomposable problems one way is use means-ends analysis technique to minimize the difference between initial states to goal state. One of the better ways to represent knowledge about what went wrong and then apply a direct patch.

5.1.1 Problem Solving Vs. Planning

GQ. Compare problem solving and planning.

Sr. No.	Planning	Problem Solving
1.	The task coming up with a sequence of actions that will achieve a goal is called planning.	Problem solving is the systematic search through a range of possible actions in order to reach to some predefined goal solution.
2.	Planning is typically viewed as a generic term of problem solving because it deals with search in an abstract level.	Problem solving comprises standard search process.
3.	Planning is involved in plan generation.	Problem solving is more concerned with plan execution.
4.	Planner is viewed as the producer or generator of the solution.	Problem solving just demonstrates one specific solution.
5.	Backward planning is typical for planning.	In problem solving usually forward approach is adopted.
6.	The pre-requirements are almost same for variety of problems in planning.	Problem solving is of two types – special purpose and general purpose. A special purpose is for one problem. General purpose is for variety of problems.
7.	STRIPS (Standard Research Institute problem solver), PDDL (Planning domain definition language) are some examples for planning systems.	A* algorithm is an example.

Module 5

Scanned with CamScanner

- A simple planning agent is very similar to problem-solving agents in that it constructs plans that achieve its goals, and then executes them. The limitations of the problem solving approach motivates the design of planning systems.

☞ **To solve a planning problem using a state-space search approach we would let the**

- Initial state = initial situation
- Goal-test predicate = goal state description
- Successor function computed from the set of operators.
- Once a goal is found, solution plan is the sequence of operators in the path from the start node to the goal node.

In searches, operators are used simply to generate successor states and we cannot look "inside" an operator to see how it's defined. The goal-test predicate also is used as a "black box" to test if a state is a goal or not. The search cannot use properties of how a goal is defined in order to reason about finding path to that goal.

Planning is considered different from problem solving because of the difference in the way they represent states, goals, actions, and the differences in the way they construct action sequences.

☞ **Remember the search-based problem solver had four basic elements**

- **Representations of actions :** Programs that develop successor state descriptions which represent actions.
- **Representation of state :** Every state description is complete. This is because a complete description of the initial state is given, and actions are represented by a program.
- **Representation of goals :** A problem solving agent has only information about its goal which is in terms of a goal test and the heuristic function.
- **Representation of plans :** In problem solving, the solution is a sequence of actions.

☞ **Cordless drill for a problem solving exercise we need to specify**

1. **Initial state :** The agent is at home without any objects that he is wanting.
2. **Operator set :** Everything the agent can do.

5.2 PROBLEMS ASSOCIATED WITH PLANNING AGENT

GQ. Write short notes on : (i) Planning agent (ii) State goal and action representation.

5.2.1 Planning Agent

- The actual branching factor would be in the thousands or millions. The heuristic evaluation function can only choose states to determine which one is closer to the goal. It cannot eliminate actions from consideration.
- The agent makes guesses by considering actions and the evaluation function ranks those guesses. The agent picks the best guess, but then has no idea what to try next and therefore starts guessing again.
- It considers sequences of actions beginning from the initial state. The agent is forced to decide what to do in the initial state first, where possible choices are to go to any of the next places. Until the agent decides how to acquire the objects, it can't decide where to go. Planning emphasizes what is in operator and goal representations.

5.2.2 Three Key Ideas behind Planning

There are Three Key Ideas behind Planning

- To "open up" the representations of state, goals, and operators so that a reasoner can more intelligently select actions when they are needed.
- The planner is free to add actions to the plan wherever they are needed, rather than in an incremental sequence starting at the initial state.
- Most parts of the world are independent of most other parts which makes it feasible to take a conjunctive goal and solve it with a divide-and-conquer strategy.
- An intelligent agent can act independently and has well-defined goals. It can adapt its behavior to its environment "a general-purpose system that, like a human, can perform a variety of different tasks under conditions that may not be known a priori." An agent must be aware of the user's goals and may have to behave in a changing world.

Scanned with CamScanner

5.2.3 Two Types of Agents

1. Physical agents (usually known as robots)
2. Software agents (sometimes known as softbots)

1. Physical agents

These are physical artefacts and act in a physical environment, e.g. send a physical agent into a dangerous building. It must be able to see, it must know where it is, it must be able to move and search, plan its goals, execute its goals (physically), re-plan if necessary, communicate with other (possibly human) agents.

Consist of some or all of the following :

(a) Computers

Top-level controller

Low-level controller, e.g. to manipulate a hand

(b) Sensors

Establish contact / non-contact with objects in the environment

To "see"

(c) Effectors

"arms", "hands", "feet"

(d) Auxiliary equipment

Tools, containers to put things in, tables to put things on

Stages of development of physical agents :

- Slave manipulator operated by human master
- Limited sequence manipulator (hard to adjust)
- Teach-replay robot
- Computer-controlled robot
- Intelligent robot

Incentives for development of physical agents

- Job undesirability
- Lethal (radiation)
- Harmful (paint spraying, chemical handling)
- Risky (fire-fighting, combat)
- Strenuous (heavy loads, visual inspection)
- Noisy (riveting, hammering, forging)
- Boring (sorting, assembling)

2. Software agents

These are software programs and act in computers or computer networks.

Simple examples are programs that sort your incoming mail according to a given priority, that store documents in the correct folder (previously done by a human agent, a secretary).

Three Laws of Robotics

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.
- A robot may not harm humanity, or, by inaction, allow humanity to come to harm.

Agents and goals

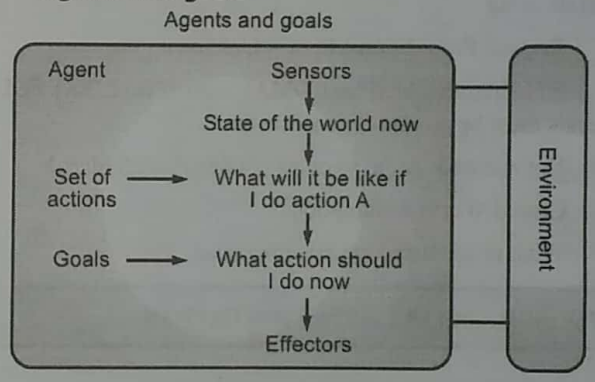


Fig. 5.2.1 : Software agents

(i) **Machine translation (MT)** is the task of automatically converting one natural language into another, preserving the meaning of the input text, and producing fluent text in the output language.

(ii) **State, goal and action representation :** Describe state, goal and action represents to plan, one should be able to represent the problem properly. Representation of the planning problem is **mapping of the states, actions and the goals**. For the representation, the language used should be concrete, understandable and expressive. In a broader perspective, there are different representation methods or ways that are followed like propositional, first order, state variable. Fig. 5.2.2 depicts the high-level diagram for planning.

Module 5

Scanned with CamScanner

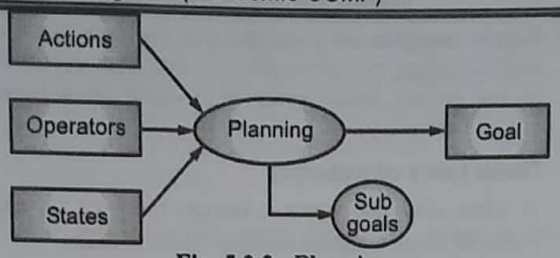


Fig. 5.2.2 : Planning

Planning essentially needs the representation in terms of **time**. This is required so that we are able to reason regarding the actions that are to be taken along with the reactions that we get back.

(iii) State Representation : States are the representation of the facts. States are represented as conjunction. It comprises the positive literals that specify the state.

A state is represented with a conjunction of positive literals using :

Logical Propositions: Poor ∧ Unknown

FOL literals: At (Plan1,OMA) ∧ At (Plan2,JFK) FOL literals must be ground & function-free

Not allowed: At (x, y) or At (Father(Fred),Sydney)

Closed World Assumption :

What is not stated are assumed false.

5.3 GOAL REPRESENTATION

Goal is most often a partially specified State. A state or say proposition is said to achieve or satisfy the given goal if it consists of all the objects required for the goal or may by some other too. As an example, if the goal kind ∧ hardworking, then a state that has kind ∧ hardworking ∧ pretty fulfils the goal.

Example : Rich ∧ Famous ∧ Miserable satisfies the goal Rich ∧ Famous

☞ Action representation

Whenever we decide to do something, we are aware of the state we are in and what possibly the effects are. When it comes to the mapping of the actions for an agent or in case of robots, the pre and the post situations need to be specified. These can also be called as **preconditions** and the after effects are called **post-conditions**.

For example, an action to drive from one place to another can be mapped as follows :

- Action<drive(c,from.to)
- Pre-condition: at(c,from)^car(c)
- Post-condition: ~at<c, from)^at(c, to)]

Action Representation in this case is :

- Action Schema
  → Action name
  → Precondition
  → Effects
- **Example :**
  Action(Fly (p. from to),
  →Pre-condition: At (p from)∧plane (p) ∧Airport(from) ∧Airport (to)
  → Effect: At (p, from) ∧At(p.to))
- Sometimes, Effects are split into ADD list and DELETE list

At (WHI, LNK), Plane(WHI)
Airpost (LNK) Airport (OHA)
Fly(WHI, LNK OHA)
At (WHI, OHA)-At (WHI,LNK)

Here, in the post-condition, where we have written **~at(c, from)** which indicates that; the state is deleted or is to be removed. In some cases, add and delete list can also be used.

In case of a slate variable representation, the state comprises of different state variables. The action here is defined as a partial function on the states.

☞ The actions can be actually applied using following criteria

1. A substitution is to be identified for the variables. That is, for the current state that exists, identify the action with a pre-condition that satisfies the current state (the current state can be subset of the pre-condition).
2. Apply this substitution (for whatever part of the current state it is applicable).
3. Add the post-condition (effects) to the remaining subset of the current state if any.

Scanned with CamScanner

5.4 PLANNING WITH STATE SPACE RESEARCH

We highlight the main planning method to solve AI problems

State space search is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or states of an instance are considered, with **the intention of finding a goal state with the desired property.**

For finding the solution one can make use of explicit search tree that is generated by the initial state and the successor function that together define the state space

Construction of State Space

(i) The root of search tree is a search node corresponding to initial state. In this state we can check whether the goal is reached.

(ii) If goal is not reached we consider another state. This can be done by expanding from the current state by applying successor function which generates new state. And from this, we get multiple states.

(iii) For each one of these, again we need to check goal test or repeat expansion of each state.

(iv) The choice of which state to expand is determined by the search strategy.

(v) It is possible that some state cannot lead to goal state such a state we should not expand.

Example : state-search example :

Making coffee

(1) Take some of the boiled water in a cup and add necessary amount of instant coffee powder to make decoction.

(2) Add milk powder to the remaining boiling water to make milk.

(3) Mix decoction and milk.

(4) Add sufficient quantify of sugar and the coffee is ready.

The most straight forward approach is to use state-space search.

Since the descriptions of actions in a planning problem specify both pre conditions and effects, it is just possible to each along both the directions :

(1) Forward from the initial state or

(2) backward from the goal

5.4.1 Forward State-Space Search

This search is also called as progressive planning, because it moves in the forward direction. It is similar to the problem-solving approach.

We begin with the problem's initial state, considering sequences of actions until we reach a goal state.

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

(a) Forward search is **an algorithm that searches forward from the initial state of the world to try to find a state that satisfies the goal formula.**

(b) The initial state of the search is the initial state from the planning problem here, each state will be set of position ground literals; not appearing literals are taken as false.

(c) The actions which are applicable to a state are all those whose preconditions are satisfied. Adding the positive effect literals and deleting the negative effect literals, the successor sate is generated from the action.

(d) The goal test chocks whether the state satisfies the goal of the planning problem.

(e) The step cost of each action is taken as 1. Different cost for different actions may be allowed.

(f) Forward state-space search is not very practicable. It is because of a big branching factor. Forward search considers all applicable actions, (i.e. all relevant and non-relevant actions are considered).

(g) A forward planner searches the state-space graph from the initial state to the goal-description.

State Representation

- A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes represent the moves from one state to another, an initial state and a goal state.
- Each state space takes the form of a tree or a graph.
- Factors that determine which search algorithm or technique will be used include the type of the problem and how the problem can be represented.
- We have presented state-space search as a forward search method, but it is also possible to search backward from the set of states that satisfy the goal to the initial state.

Module 5

Scanned with CamScanner

5.4.2 Backward State-Space Search

(a) Backward state space search planning is we want to generate possible predecessors of a given goal state, work backwards toward the initial state.

(b) If a solution exists, it should be found by a backward search which **allows only relevant action**

(c) The meaning or restriction to relevant actions only means that backward search often has a much lower branching factor than forward search.

(d) Goal states are often incompletely specified. It expresses only what is desired in the final state, rather than a complete description of the final state.

(e) It is also called as Regression planning because Backward state-space search will find the solution from goal to the action.

(f) If there are many known states possible to be reached from goal state. Reaching to any one of those nodes, we can reach the start node.

(g) In general, backward search works only when we know how to regress form a state description to the predecessor state description. For example, it is hard to search backwards for a solution to the n-queens problem because these is no easy way to describe the states that are one move away from the goal.

(h) To obtain full advantage of backward search, we need to deal with partially uninstantiated actions and states.

For example, suppose the goal is to **deliver a specific piece of cargo to Delhi** :

This suggests the action unload (C, P, Delhi) Thus, Action (unload (C, P. Delhi))

Precondition : In (C, P) $\wedge$ (P, Delhi) $\wedge$ cargo (C) $\wedge$ plane (P) $\wedge$ Airport (Delhi)

Effect : At (C, Delhi) $\wedge \neg$ In (C, P)

5.5 PARTIAL ORDER PLANNING

UQ. Explain a partial order planner with an example.
(MU - Q. 5(c), Dec. 15, Q. 4(b), May 17, Q. 5(a), May 2019, 10 Marks)

- Partial order planning is an approach to automated planning that maintains a partial ordering between actions and only commits ordering between actions when the actions are partial. Also the planning does not specify which action will come out first when two actions are processed.
- A partial order plan or partial plan which specifies all actions that need to be taken, but only specifies the order between actions when necessary, is the result of a partial order planner.
- This is sometimes also called **a non-linear planner**, which is a misnomer because such planners often produce a linear plan.
- A partial ordering is a less-than relation that is transitive and asymmetric.

Partial order planner components

1. **A set of actions** (also known as operators).
2. **A partial order :** A partial-order plan is a set of actions together with a partial ordering, representing a "before" relation on actions, such that any total ordering of the actions, consistent with the partial ordering, will solve the goal from the initial state.
3. **A set of causal links :** It specifies which actions meet which preconditions of other actions. Alternately, a set of bindings between the variables in action. Write $\mathbf{act_0 < act_1}$ if action act_0 is before action act_1 occurs in the partial order.
4. **A set of open preconditions** : For **uniformity**, treat **start** as an action that achieves the relations that are true in the initial state, and treat **finish** as an action whose precondition is the goal to be solved. The **pseudo action start** is before every other action, and finish is after every other action. The use of these as actions means that; the algorithm does not require special cases for the initial situation and for the goals. When the preconditions of finish hold, the goal is solved.
5. In order to keep the possible orders of the actions as open as possible, the set of other conditions and **causal links** must be as small as possible. A plan is itself a solution if the set of open preconditions is empty.
6. An action, other than start or finish, will be in a **partial-order plan** to achieve a precondition of an action in the plan. Each precondition of an action in the plan is either true in the initial state or so achieved by start, or there will be an action in the plan that achieves it.
7. We must ensure that the actions achieve the conditions they were assigned to achieve. Each **precondition P** of an **action** $\mathbf{act_1}$ in a plan will have an **action** $\mathbf{act_0}$ associated with it such that $\mathbf{act_0}$ **achieves precondition P for** $\mathbf{act_1}$.

Scanned with CamScanner

8. The triple (act_0, **P**, act_1) is a **causal link**. The partial order specifies that action act_0 occurs before action act_1, which is written as $act_0 < act_1$. Any other action A that makes P false must either be before act_0 or after act_1.

☞ Linearization of partial order plan

A linearization of partial order plan is a **total order plan** derived from the particular partial order plan, in other words, both order plans consist of the same actions, with the order in the **linearization** being a **linear extension** of the partial order in the original partial order plan.

For example : A plan for baking a cake might start as follows :

- Go to store,
- Obtain flour, get milk, etc.
- Pay for all the goods,
- Go to kitchen.

This is a partial plan because the order for finding flour and milk is not specified; the agent can wander around the store, accumulating all the items as its shopping list is complete.

5.5.1 Causal Link in Partial Order Planning

- Each causal link specifies a **pair of steps and a proposition**, where the proposition is a post condition of the first step and precondition of the second step. The first step is ordered before the second step.
- If a precondition of a step is not supported by a causal link, then it is a flaw in a partial-order plan.
- Any planning algorithm that can place two actions into a plan without specifying which should come first is called partial-order planner.
- **POP (Partial-order-plan) :** is a regression planner; it uses problem decomposition; it searches plan space rather than state space; it builds partially-ordered plans; and it operates by the principle of least commitment.

5.5.2 Working of a Partial Order Planner

GQ. How does partial order planner work ?

1. Begin with the actions start and finish and the partial order **start < finish**.
2. The planner maintains an agenda that is a set of **(P, A)** pairs, where A is an action in the plan and P is an atom that is a precondition of A that must be achieved. Initially the agenda contains pairs (**G, finish**), where G is an atom that must be true in the goal state.
3. At each stage in the planning process, a pair (**G,** act_1) is selected from the agenda, where P is a precondition for action act_1.
4. Then an action, act_0, is chosen to achieve P. That action is either already in the plan it could be the start action, or it is a new action that is added to the plan. Action act_0 must happen before act_1 in the partial order. It adds a causal link that records that act_0 achieves **P for action** act_1. Any action in the plan that deletes P must happen either before act_0 or after act_1.
5. If act_0 is a new action, its preconditions are added to the agenda, and the process continues until the agenda is empty. This is a non-deterministic procedure. The "**choose**" and the "**either ...or ...**" form choices that must be searched over.
6. **POP :** is a **regression planner**; it uses problem decomposition; it searches plan space rather than state space; it builds partially-ordered plans; and it operates by the principle of least-commitment.

☞ A plan in POP (whether it be a finished one or an unfinished one) comprises of following

- **A set of plan steps:** Each of these is a STRIPS operator, but with the variables instantiated.
- **A set of ordering constraints:** $S_i < S_j$ means step Si must occur sometime before S_j (not necessarily immediately before).
- **A set of causal links:** S_i c $- \rightarrow S_j$ means step Si achieves precondition c of step S_j.

So, it comprises actions (steps) with constraints (for ordering and causality) on them. The algorithm needs to start off with an initial plan. This is an unfinished plan, which we will refine until we reach a solution plan.

The initial plan comprises two dummy steps, called **Start and Finish**. Start is a step with no preconditions, only effects: the effects are the initial state of the world. Finish is a step with no effects, only preconditions: the **preconditions** are the goal.

Module 5

5.5.3 Example for Partial Order Planner

GQ. Given an example for partial order planner.

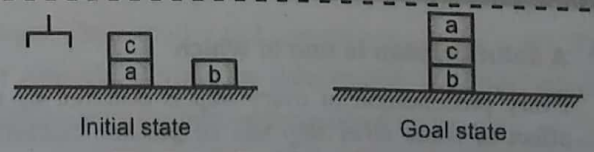


Fig. 5.5.1 : Partial order planner

Scanned with CamScanner

The above initial state can be represented in POP as the following initial plan :

Plan(STEPS: {S1: Op(ACTION: Start, EFFECT: clear(b) ∧ clear(c) ∧ on(c, a) ∧ ONTABLE(a) ∧ ONTABLE(b) ∧ ARMEMPTY), S2: Op(ACTION: Finish, PRECOND: on(c, b) ∧ on(a, c))}, ORDERINGS: {S1 < S2}, LINKS: {})

This initial plan is refined using POP's plan refinement operators. As we apply them, they will take us from an unfinished plan to a less and less unfinished plan, and ultimately to a solution plan.

☞ **There are four operators, falling into two groups**

1. **Goal achievement operators – Step addition** : Add a new **step S_i** which has an effect c that can achieve an as yet unachieved precondition of an existing step Sj . Also add the following constraints: **Si < Sj and Si c– → Sj and Start < Si < Finish.** – Use an effect c of an existing step Si to achieve an as yet unachieved precondition of another existing step Sj . And add just two constraints: **Si < Sj and Si c – → Sj** .
2. **Causal links:** must be protected from threats, i.e. steps that delete (**or negate or clobber**) the protected condition. If S threatens link **Si c – → Sj : – PROMOTE:** add the **constraint S < Si;** or – **DEMOTE:** add the constraint Sj < S the goal achievement operators ought to be obvious enough. They find preconditions of steps in the unfinished plan that is not yet achieved.

The two goal achievement operators remedy this either by adding a new step whose effect achieves the precondition, or by exploiting one of the effects of a step that is already in the plan.

The promotion and demotion operators may be less clear. Why are these needed? POP uses problem-decomposition: faced with a conjunctive precondition, it uses goal achievement on each conjunct separately. But, as we know, this brings the risk that the steps we add when achieving one part of a precondition might interfere with the achievement of another precondition.

And the idea of promotion and demotion is to add ordering constraints so that the step cannot interfere with the achievement of the precondition. Finally, we have to be able to recognize when we have reached a solution plan : a finished plan.

☞ **A solution plan is one in which**

- Every precondition of every step is achieved by the effect of some other step and all possible clobberers have been suitably demoted or promoted.
- There are no contradictions in the ordering constraints, e.g. disallowed is **Si < Sj and Sj < Si** ; also disallowed is **Si < Sj , Sj < Sk** and **Sk < Si** .

The solutions may still be **partially-ordered**. This retains flexibility for as long as possible. Only immediately prior to execution will the plan need linearization, i.e. the imposition of arbitrary ordering constraints on steps that are not yet ordered. (In fact, if there's more than one agent, or if there's a single agent but it is capable of multitasking, then some linearization can be avoided: steps can be carried out in parallel.)

5.6 HIERARCHICAL PLANNING

- Take an unachieved precondition from the plan; achieve it. Resolve any threats using promotion or demotion. But, what the above fails to show is that planning involves search.
- At certain points in the algorithm, the planner will be faced with choices (alternative ways of refining the current unfinished plan). POP must try one of them but have the option of returning to explore the others.

☞ **There are basically two main 'choice points' in the algorithm**

- In **goal achievement**, a condition c might be achievable by any one of a number of new steps and/or existing steps. For each way of achieving c, a new version of the plan must be created and placed on the agenda.
- **Question :** A condition c might be achievable by new steps or existing steps. When placing these alternatives on the agenda, might be we arrange for the latter to come off the agenda before the former?
- When resolving threats, POP must choose between demotion and promotion. All preconditions must eventually be achieved, and so these aren't alternatives. The choice can be made irrevocably.
- Provided implementation of POP uses a complete and optimal search strategy, then POP itself is complete and optimal. However, **POP's branching factor** can still be high and the unfinished plans that we store on the agenda can be quite large data structures, so we typically abandon completeness/optimality to keep time and space more manageable.

Search strategies that are more like depth-first search might be preferable. And we might use heuristics to order alternatives or even to prune the agenda.

Scanned with CamScanner

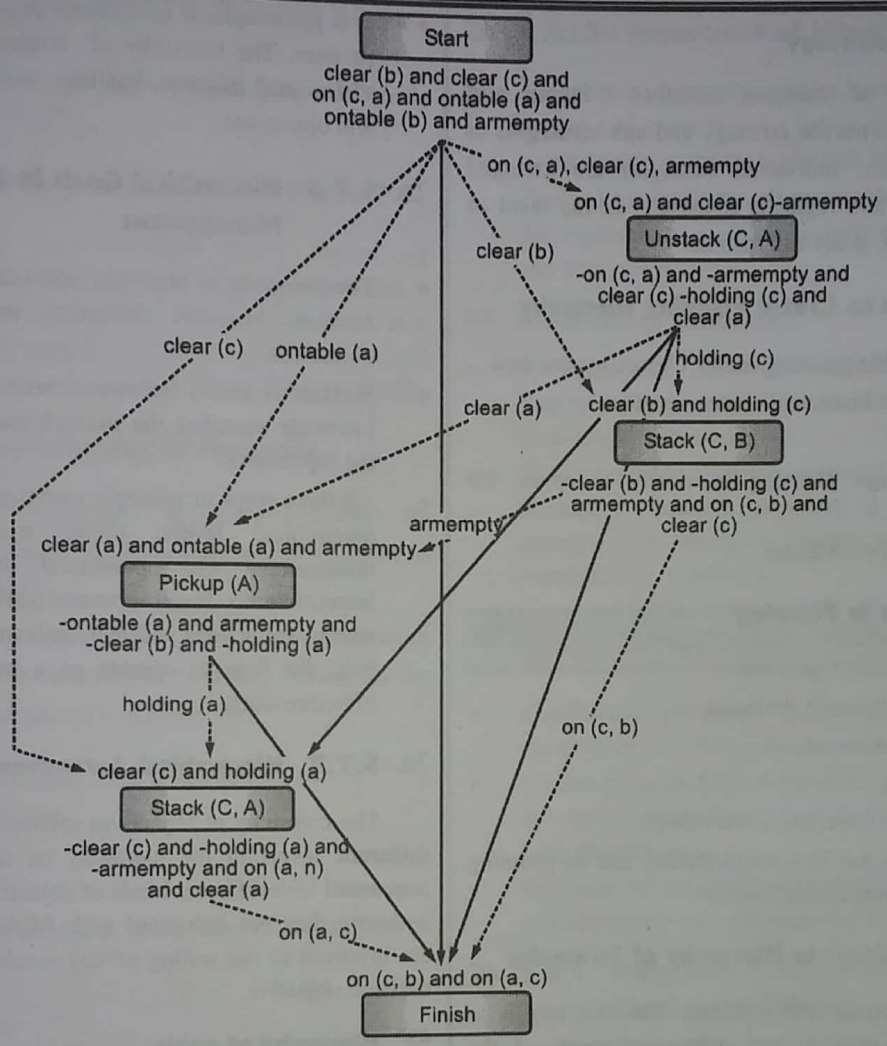


Fig. 5.6.1 : Hierarchical planning

5.7 HIERARCHICAL TASK NETWORK PLANNING

- In an Artificial Intelligence hierarchical task network planning is an approach to **automated planning** in which the dependency among actions can be given in the form of hierarchically structured networks.
- A hierarchical planning represents the **organisational levels** and units in the company for which one wants to plan. A planning hierarchy is a **combination of characteristic values** based on the characteristics of one information structure. You can create only one planning hierarchy for an information structure.
- Planning formed and executed by several subsystems organised according to a layered structure for the system goal.

System

System when applied to production and operations management, the process is called a **hierarchical production planning and control** process, which is the essence of a manufacturing planning and control (**MPC**) system. Also, these decisions are usually made by the upper levels of the management hierarchy.

Hierarchical plan consists of :

There are 3 types of hierarchical plans. They are strategic, administrative and operating (technical core) .The three hierarchical plans are independent, and they support the fulfilment of the three organisational needs.

Module 5

Scanned with CamScanner

Hierarchical strategy

The hierarchy of strategies describes a **layout and relations** of **corporate strategy** and **sub strategies** of the organisation. Individual strategies are arranged hierarchically and logically consistent at the level of vision, mission, goals and metrics.

5.7.1 Steps to Create Planning Hierarchy

1. From the flexible planning menu, choose master data.
2. Leave all fields blank and choose 'confirm'
3. Choose 'save'
4. Choose: Change planning Hierarchy to create the hierarchy.
5. To continue, choose Enter

5.7.2 Steps in Planning

1. Determination of Objectives,
2. Constructing Planning Premises,
3. Evaluation of Alternatives.
4. Selecting plan,
5. Controlling the plan, and a few others.

Every business has its own problems, and so planning details differ from business to business.

5.7.3 Key Steps in Hierarchy of Strategies

Strategic objectives and Analysis. The first step is to define the vision, mission and values-statements of the organisation.

1. Strategic formulation.
2. Strategic implementation
3. Strategic Evaluation and control.

5.7.4 Three Levels of Strategy

1. **Corporate level of strategy**. This level answers the foundational question of what you want to achieve.
2. **Business unit level strategy**. This level focuses on how you are going to compete.
3. **Market level strategy**. This strategy focuses on how you are going to grow.

5.7.5 Strategy Intent

- Strategy intent refers to the **purpose** for which the organisation strives for.
- It is philosophical framework of strategic management process. The hierarchy of strategic intent covers the **vision and mission**, business definition and the goals and objectives.

5.7.6 Hierarchical Goals in Strategic Management

- The hierarchy of strategic intent covers the vision and mission, business definition and the goals and objectives.
- **Stretch** is misfit between resources and aspirations. Leverage stretches the meagre resource base to meet the aspirations.
- All three steps in strategic planning occur within three hierarchical levels; upper management, middle management and operational levels. Thus, it is imperative to foster communication and interaction among employees and managers at all levels. So as to help the firm to operate as a more functional and effective team.

5.7.7 Hierarchical Arrangement of Objective

The hierarchy of objectives indicates that **managers at different levels** in the hierarchy of the organisation are concerned with different kinds of objectives according to the authority they are delegated with **Middle level managers** are involved in the setting of key result area objective and division objective.

Hierarchy of goals

The hierarchy of goals is a **goal-oriented iterative method** that can be used to define the scope of the given project. Thus, by setting goals that can be brought up to date as knowledge expands. The method consists of a hierarchy with a main goal on top, followed by sub-goals, project goals, deliveries and success criterion.

Hierarchy of objectives

The hierarchy of objectives is a **tool** that helps analyse and communicate a **project's objectives**. The hierarchy of objective organises the objectives of a project into different levels of hierarchy or tree.

5.7.8 Advantages of Hierarchy of Objectives

1. Clear lines of authority and reporting within the business.
2. Clearer understanding of employee roles and responsibilities.

Scanned with CamScanner

3. Accountability for actions or decisions at different management levels.
4. Clear career paths and development prospects which can motivate employees.

5.7.9 How Hierarchical Structure Works ?

A hierarchical structure has many layers of management, and business with the structure often use a **'top down'** approach with a long chain of command in a hierarchical structure, managers will have a narrow span of control and a relatively small number of subordinates (staff).

5.7.10 Some Examples of Hierarchy

1. An example of hierarchy is the corporate ladder. An example of hierarchy is the various levels of priests in catholic churches.
2. A structure that has a predetermined ordering from high to low. For example, all files and folders on the hard disk are organised in a hierarchy.

5.7.11 Advantages and Limitations of Hierarchy

Advantage

Clear chain of command, clear paths of Advancement, Specialisation.

Disadvantage

Poor flexibility, communication barriers, organisational disunity.

5.7.12 Characteristics of Hierarchy Data Model

1. **Structure :** The main characteristic of a hierarchical data model is the **tree like** structure. One-to-many and Redundancy, Navigation, local parent pointers.
2. **Hierarchical relation :** Hierarchical relationships are based on degree or levels of superordination and subordination, where the superordinate term represents a class or a whole, and subordinate term refers to its members or parts.

5.7.13 Importance of Hierarchy

- An effective hierarchy makes leaders accountable for results, and provisions for their replacing failures, with someone new-sometimes through internal promotion.
- This is how hierarchy ultimately serves the success of the organisation as whole-including owners, managers and employees.

Hierarchy of concepts

- A conceptual hierarchy defines a sequence of mappings from a set of low-level concepts to higher level managerial concepts.
- These mappings form a concept hierarchy for the dimension location, mapping a set of low-level concepts (i.e. cities) to higher-level, more general concepts (i.e. countries).

5.8 CONDITIONAL PLANNING

- Conditional planning has to work regardless of outcome of an action.
- It takes place in fully **observable environment** where the current state of the agent in known environment is fully observable.
- Outcome of an action cannot be determined so the environment is said to be non-deterministic.

5.8.1 Difference in Hierarchical and Conditional Planning

1. Hierarchical Planning

(a) In hierarchical planning, at each level of hierarchy the objective functions are reduced to a small number of activities at the next lower level.

(b) The computational cost of finding the correct way to arrange these activities for the current problem is small.

(c) Hierarchical methods can result in linear time.

(d) The initial plan of hierarchical planning describes the complete problem which is a very high level description.

(e) The plans are refined by applying action decomposition.

(f) Each action decomposition reduces a high level description to some of the individual lower level descriptions.

Module 5

hierarchic

Scanned with CamScanner

(g) The action decomposers describe how to implement the actions.

2. Conditional Planning

(a) It deals with the planning by some appropriate conditions.

(b) The agents plan first and then execute the plan that was produced.

(c) The agents find out which part of the plan to execute by including sensing actions in the plan to test for the appropriate conditions.

(d) Conditional planning works regardless of the outcome of an agent.

(e) It takes place in fully observable Environment where the current state of the agent in known environment is fully observable.

(f) The outcome of actions cannot be determined so the environment is said to be non-deterministic.

(g) A "state node" is represented with a "square" and "chance node" is represented with a "circle".

(h) Here, we can check what is happening in environment at predetermined points of the plan to deal with ambiguous action.

(i) It needs to take some actions at every state and must be able to handle every outcome for the action it takes.

Hierarchical methods can result in linear time. The initial plan of hierarchical planning describes the complete problem which is a very high level description.

Conditional planning deals with the planning by some appropriate conditions. The agent plans first and then executes the plan that was produced.

As an alternative to outright refusal the planning authority can grant permission subject to one or more condition. Planning condition sometimes limit the use or occupation of land or premises to a named person or company.

5.9 FORMS OF LEARNING

Learning is one of the fundamental building blocks of Artificial Intelligence solutions. From a conceptual stand point, learning is a process that improves the knowledge of an Artificial Intelligence program by making observations about its environment.

Learning process can be effected as :

(1) **Skill refinement :** Repeatedly practicing one can attain **good skill** refined skill; e.g. learning to drive a car.

(2) **Knowledge acquisition :** Knowledge is acquired through learning. One can develop learning based on type of knowledge representation used (e.g. inductive, predictive logic, concepts, predictions etc.)

5.10 LEARNING ALGORITHMS

With the constant advancements in artificial intelligence, the field has become too big to specialize in all together. There are countless problems that can be solved with countless methods. Knowledge of an experienced AI researcher specialized in one field may mostly be useless for another field. Understanding the nature of different machine learning problems is very important. Even though the list of machine learning problems is very long, these problems can be grouped into three different learning approaches:

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

Top machine learning approaches are categorized depending on the nature of their feedback mechanism for learning. Most of the machine learning problems may be addressed by adopting one of these approaches.

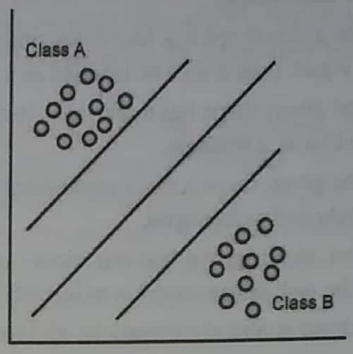
5.11 SUPERVISED LEARNING

UQ. What is supervised learning ? Give example of each.
(MU - Q. 1(e), May 17, Q. 1(e), May 19, 5 Marks)

- Learning that takes place based on a class of examples is referred to as supervised learning. It is learning based on labeled data. In short, while learning, the system has knowledge of a set of labeled data. This is one of the most common and frequently used learning methods
- The supervised learning method is comprised of a series of algorithms that build mathematical models of certain data sets that are capable of containing both inputs and the desired outputs for that particular machine.
- The data being inputted into the supervised learning method is known as training data, and essentially consists of training examples which contain one or more inputs and typically only one desired output. This output is known as a "supervisory signal."
- In the training examples for the supervised learning method, the training example is represented by an array, also known as a vector or a feature vector, and the training data is represented by a matrix.

Scanned with CamScanner

- The algorithm uses the iterative optimization of an objective function to predict the output that will be associated with new inputs.
- Ideally, if the supervised learning algorithm is working properly, the machine will be able to correctly determine the output for the inputs that were not a part of the training data.
- Supervised learning uses classification and regression techniques to develop predictive models. Classification techniques predict categorical responses,
- Regression techniques predict continuous responses, for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.
- Let us begin by considering the simplest machine-learning task : supervised learning for classification. Let us take an example of classification of documents. In this particular case a learner learns based on the available documents and their classes. This is also referred to as labeled data.
- The program that can map the input documents to appropriate classes is called a classifier, because it assigns a class (i.e., document type) to an object (i.e., a document). The task of supervised learning is to construct a classifier given a set of classified training examples. A typical classification is depicted in Fig. 5.11.1.
- Fig. 5.11.1 represents a hyperplane that has been generated after learning, separating two classes - class A and class B in different parts. Each input point presents input–output instance from sample space. In case of document classification, these points are documents.

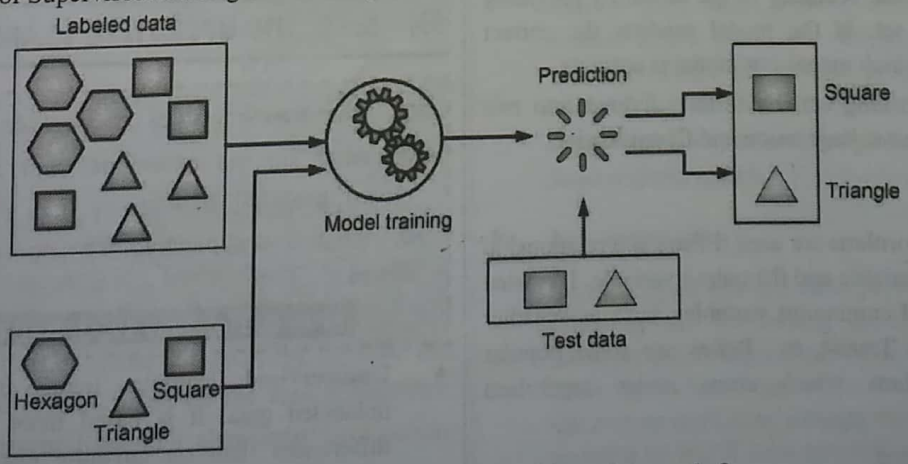


(1D1)**Fig. 5.11.1 : Supervised learning**

- Learning computes a separating line or hyperplane among documents. An unknown document type will be decided by its position with respect to a separator.
- There are a number of challenges in supervised classification such as generalization, selection of right data for learning, and dealing with variations. Labeled examples are used for training in case of supervised learning. The set of labeled examples provided to the learning algorithm is called the *training set*.
- Supervised learning is not just about classification, but it is the overall process that with guidelines maps to the most appropriate decision.

5.11.1 How Supervised Learning Works?

- In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.
- The working of Supervised learning can be easily understood by the below example and diagram (Fig. 5.11.2).



(1D2)**Fig. 5.11.2 : How Supervised learning works?**

Module 5

Scanned with CamScanner

- Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.
  - If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
  - If the given shape has three sides, then it will be labelled as a **triangle**.
  - If the given shape has six equal sides then it will be labelled as **hexagon**.
- Now, after training, we test our model using the test set, and the task of the model is to identify the shape.
- The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.
- Following are the steps involved in Supervised Learning :
  - First Determine the type of training dataset
  - Collect/Gather the labelled training data.
  - Split the training dataset into training dataset, test dataset, and validation dataset.
  - Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
  - Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
  - Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
  - Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.
- Supervised learning can be further divided into two types of problems: Regression and Classification.

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning :

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Random Forest
- Logistic Regression
- Decision Trees
- Support vector Machines

5.11.2 Advantages of Supervised Learning

1. With the help of supervised learning, the model can predict the output on the basis of prior experiences.
2. In supervised learning, we can have an exact idea about the classes of objects.
3. Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

5.11.3 Disadvantages of Supervised Learning

1. Supervised learning models are not suitable for handling the complex tasks.
2. Supervised learning cannot predict the correct output if the test data is different from the training dataset.
3. Training required lots of computation times.
4. In supervised learning, we need enough knowledge about the classes of object.

5.12 UNSUPERVISED LEARNING

GQ. What are the types of unsupervised learning?

GQ. What are the advantages and disadvantages of unsupervised learning?

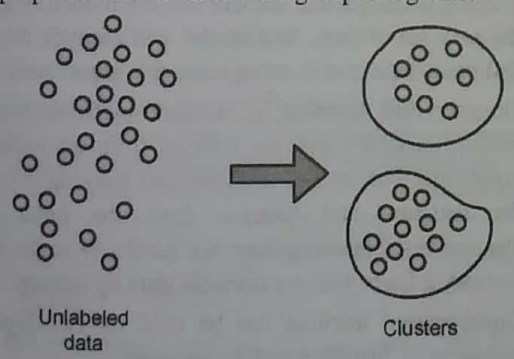
UQ. What is unsupervised learning? Give example of each.

(MU - Q. 1(e), May 17, Q. 1(e), May 19, 5 Marks)

- Unsupervised learning refers to learning from unlabeled data. It is based more on similarity and differences than on anything else. In this type of learning, all similar items are clustered together in a particular class where the label of a class is not known.

Scanned with CamScanner

- It is not possible to learn in a supervised way in the absence of properly labeled data. In these scenarios there is need to learn in an unsupervised way. Here the learning is based more on similarities and differences that are visible. These differences and similarities are mathematically represented in unsupervised learning.
- Given a large collection of objects, we often want to be able to understand these objects and visualize their relationships. For an example based on similarities, a kid can separate birds from other animals. It may use some property or similarity while separating, such as the birds have wings.
- The criterion in initial stages is the most visible aspects of those objects. Linnaeus devoted much of his life to arranging living organisms into a hierarchy of classes, with the goal of arranging similar organisms together at all levels of the hierarchy. Many unsupervised learning algorithms create similar hierarchical arrangements based on similarity-based mappings.
- The task of hierarchical clustering is to arrange a set of objects into a hierarchy such that similar objects are grouped together. Nonhierarchical clustering seeks to partition the data into some number of disjoint clusters. The process of clustering is depicted in Fig. 5.12.1.
- A learner is fed with a set of scattered points, and it generates two clusters with representative centroids after learning. Clusters show that points with similar properties and closeness are grouped together.



(1D3)**Fig. 5.12.1 : Unsupervised learning**

- Unsupervised learning is a set of algorithms where the only information being uploaded is inputs. The device itself, then, is responsible for grouping together and creating ideal outputs based on the data it discovers. Often, unsupervised learning algorithms have certain goals, but they are not controlled in any manner.
- Instead, the developers believe that they have created strong enough inputs to ultimately program the machine to create stronger results than they themselves possibly could. The idea here is that the machine is programmed to run flawlessly to the point where it can be intuitive and inventive in the most effective manner possible.
- The information in the algorithms being run by unsupervised learning methods is not labeled, classified, or categorized by humans. Instead, the unsupervised algorithm rejects responding to feedback in favor of identifying commonalities in the data. It then reacts based on the presence, or absence, of such commonalities in each new piece of data that is being inputted into the machine itself.
- It is used to draw inferences from datasets consisting of input data without labelled responses. Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data. Applications for clustering include gene sequence analysis, market research, and object recognition.

5.12.1 Types of Unsupervised Learning Algorithm

The unsupervised learning algorithm can be further categorized into two types of problems:

1. Clustering
2. Association

1. Clustering

Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

2. Association

An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Module 5

Scanned with CamScanner

Below is the list of some popular unsupervised learning algorithms :

- K-means clustering
- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

5.12.2 Advantages of Unsupervised Learning

1. Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
2. Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

5.12.3 Disadvantages of Unsupervised Learning

1. Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
2. The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

In practical scenarios there is always need to learn from both labeled and unlabeled data. Even while learning in an unsupervised way, there is the need to make the best use of labeled data available. This is referred to as semisupervised learning. Semisupervised learning is making the best use of two paradigms of learning - that is, learning based on similarity and learning based on inputs from a teacher. Semisupervised learning tries to get the best of both the worlds.

5.12.4 Difference between Supervised and Unsupervised Learning

- Supervised and Unsupervised learning are the two techniques of machine learning. But both the techniques are used in different scenarios and with different datasets. Below the explanation of both learning methods along with their difference table is given.
- Supervised learning is a machine learning method in which models are trained using labeled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y).

$$Y = f(X)$$

- Supervised learning needs supervision to train the model, which is similar to as a student learns things in the presence of a teacher. Supervised learning can be used for two types of problems: Classification and Regression.
- **Example :** 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.
- Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data. The goal of unsupervised learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.
- Unsupervised learning can be used for two types of problems : Clustering and Association.
- **Example :** 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.

Scanned with CamScanner

GQ. What is the difference between supervised learning and unsupervised learning?

The main differences between Supervised and Unsupervised learning are given below :

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled 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.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

5.13 REINFORCEMENT LEARNING

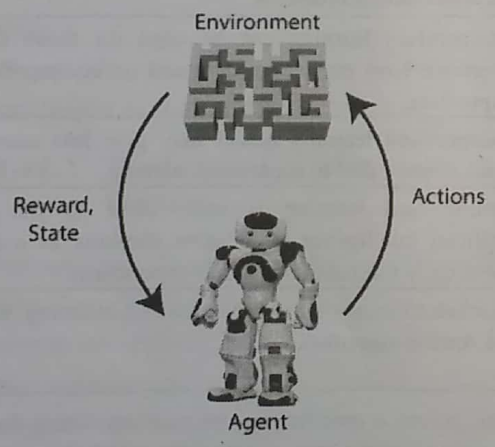
GQ. What is Reinforcement Learning? Explain with an example.

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that "Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that." How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.

Module 5

Scanned with CamScanner

- It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.
- **Example :** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
- The agent continues doing these three things (take action, change state/remain in the same state, and get feedback), and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.



(1D4) **Fig. 5.13.1**

- For machine learning, the environment is typically represented by an "MDP" or Markov Decision Process. These algorithms do not necessarily assume knowledge, but instead are used when exact models are infeasible. In other words, they are not quite as precise or exact, but they will still serve a strong method in various applications throughout different technology systems.
- The key features of Reinforcement Learning are mentioned below.
  - In RL, the agent is not instructed about the environment and what actions need to be taken.
  - It is based on the hit and trial process.
  - The agent takes the next action and changes states according to the feedback of the previous action.
  - The agent may get a delayed reward.
  - The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

5.13.1 Approaches to Implement Reinforcement Learning

GQ. What are the approaches for Reinforcement Learning?

There are mainly three ways to implement reinforcement-learning in ML, which are :

1. **Value-based :** The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.
2. **Policy-based :** Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward. The policy-based approach has mainly two types of policy :

   Deterministic : The same action is produced by the policy (π) at any state.

   Stochastic : In this policy, probability determines the produced action.
3. **Model-based :** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Here are important characteristics of reinforcement learning

- There is no supervisor, only a real number or reward signal
- Sequential decision making
- Time plays a crucial role in Reinforcement problems
- Feedback is always delayed, not instantaneous

Scanned with CamScanner

- Agent's actions determine the subsequent data it receives

RL can be used in almost any application. It is a learning based on experience algorithm, a decision maker algorithm, an algorithm that learns autonomously, an optimization algorithm that over time learns to maximize its reward, the reward can be defined by the engineer to reach the objective of the problem.

5.13.2 Challenges of Reinforcement Learning

Here are the major challenges you will face while doing Reinforcement earning :

1. Feature/reward design which should be very involved
2. Parameters may affect the speed of learning.
3. Realistic environments can have partial observability.
4. Too much Reinforcement may lead to an overload of states which can diminish the results.
5. Realistic environments can be non-stationary.

5.13.3 Applications of Reinforcement Learning

Here are applications of Reinforcement Learning :

1. Robotics for industrial automation.
2. Business strategy planning
3. Machine learning and data processing
4. It helps you to create training systems that provide custom instruction and materials according to the requirement of students.
5. Aircraft control and robot motion control

5.13.4 Reinforcement Learning vs. Supervised Learning

GQ. What is the difference between Reinforcement Learning and Supervised Learning ?

Parameters	Reinforcement Learning	Supervised Learning
Decision style	Reinforcement learning helps you to take your decisions sequentially.	In this method, a decision is made on the input given at the beginning.
Works on	Works on interacting with the environment.	Works on examples or given sample data.
Dependency on decision	In RL method learning decision is dependent. Therefore, you should give labels to all the dependent decisions.	Supervised learning the decisions which are independent of each other, so labels are given for every decision.
Best suited	Supports and work better in AI, where human interaction is prevalent.	It is mostly operated with an interactive software system or applications.
Example	Chess game	Object recognition

5.14 GENERAL LEARNING MODEL

We develop general learning model and note the factors affecting learning performance.

(i) Learning can be done in several ways. But learning requires that new knowledge structure be created from some form of input stimulus.

This type of new knowledge must be assimilated into a knowledge base and be tested in some way for its utility.

(ii) Testing means that the knowledge should be used in the performance of some task from which meaningful feedback can be obtained. The feedback provides some measure of accuracy and usefulness of the newly acquired knowledge. We exhibit the general learning model in the figure.

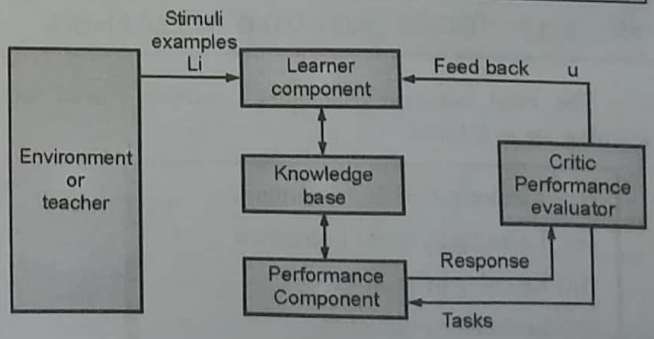


Fig. 5.14.1 : General Learning Model

(iii) In the figure, environment represents overall learner system. The environment may be regarded as one which produces random stimuli or a training source such as a teacher which provides carefully selected training examples to the learner component.

Module 5

Scanned with CamScanner

(iv) Some representation language for communication between environment and the learner component must be used. The language scheme must be same as that used in the knowledge base. When they are same, we say that 'single representation' is used.

(v) Inputs to the learner component may be physical stimuli of some type or descriptive, symbolic training examples.

The information conveyed to the learner component is used to create and modify knowledge structures in the knowledge base.

The performance component uses this knowledge to carry out some tasks, such as solving a problem, playing a game etc.

(vi) When a task is given to performance component, it describes its actions in performing the task and produces a response. Then critic module evaluates this response **relative** to an **optimal** response.

(vii) Critic module sends the response to the learner component to check whether or not the performance is acceptably and if so required modifying the structures in the knowledge base.

If the proper learning is achieved, then the performance of the system will be improved by the knowledge base component.

(viii) The cycle described in the above figure is to be repeated a number of times (i) until the performance of the system reaches some acceptable level, (ii) or a known learning goal is achieved, or (iii) after some repetitions no change **occurs** in the knowledge base.

5.15 TECHNIQUES USED IN LEARNING

The most common techniques (methods) used for learning are as follows :

(i) Memorization (Rote learning)
(ii) Learning by direct instruction
(iii) Learning by analogy,
(iv) Learning by induction
(v) Learning by deduction
(vi) Learning using neural network

5.15.1 Learning by Memorization (Rote Learning)

- It is the simplest form of learning. It requires the least amount of inference. Here, learning is achieved by simply copying the knowledge that is used in the knowledge base. For example, for memorizing multiplication table, we use this type of learning.
- When a computer stores a piece of data, it performs a rudimentary form of learning. It is a simple case of 'data caching'. We store computed values so that we do not have to recompute them later.
- When computation is more expensive then this strategy can save a significant amount of time.
- Caching is used in AI programs to produce some suprising performance improvements. Such caching is known as rote learning.

Remark : Data Caching

- Data Caching is a technique of storing frequently used data in memory, so that when the same data is needed next time, it can be directly obtained from the memory, instead of being generated by the application.
- The Android Phone's cache comprises stores of small bits of information that your apps and web browser use to speed up performance. But cached files can become corrupted and overloaded and cause performance issues. Cache need not be constantly cleared, but a periodic clean-out can be helpful.

5.15.2 Learning by Direct Instruction

- This type of learning is slightly more complex. This requires more inference than rote learning. Here to integrate knowledge into 'knowledge base', it must be transformed into an operational form.
- When a number of facts are presented to us directly in a well-organised manner, we use this type of learning.

5.15.3 Learning by Analogy

- This is a process of learning a new concept or solution by using the similar known concepts or solutions. For example, in an examination previously learned examples help one to solve new problems (in an exam)
- We make frequent use of analogical learning. This form of learning requires more inferring than either of the previous forms.
- It is because 'difficult transformations' must be made between the known and unknown situations.

Scanned with CamScanner

5.15.4 Learning by Induction

- This is a powerful form of teaming which also requires more interring than the first two methods. This form of learning is a form of **invalid** but **useful** inference.
- Here we formulate a general concept after seeing a number of instances or examples of the concept. For example, we learn the concepts of sweet taste or color after experiencing the sensations associated with several examples of sweet foods or coloured objects.

5.15.5 Learning by Deduction

- This form of Learning is achieved through a sequence of deductive inference steps using known facts. From the known facts, new facts or relationships are derived logically. For example, we could learn deductively that sita is the cousin of Ramesh, if we have knowledge of sita and Ramesh's parents and Rules for cousin relationship.
- Deductive learning requires more interence than the other methods

5.15.6 Neural Network

- Neural Networks can be loosely separated into neural methods and learning rules.
- Mculloch and pitts developed the first network models to explain how the signals passed from one neuron to another within **the network.**

5.16 STATISTICAL LEARNING IN ARTIFICIAL INTELLIGENCE

- Statistical learning theory is a branch of Artificial Intelligence. It provides us with the theoretical basis for many of today's machine learning algorithms. The basic premise of learning is to use a set of observations to uncover an underlying process. Thus statistical learning in AI is a set of tools for machine learning that uses statistics and functional analysis.
- The main goal of statistical learning theory is to provide a framework for studying the problem if inference, that is of gaining knowledge, making predictions, making decisions or constructing models from a set of data.

5.17 THEORY OF LEARNING

The theory of AI learning includes the study of

(i) Neural like elements,

(ii) Multi-dimensional neural –like growing networks

(iii) Temporary and long term memory

(iv) The study of functional organization of the brain of AI systems of the sensor systems, modulating system, motor system etc.

5.17.1 Machine-Learning Theory

- Computational learning theory or statistical learning theory refers to mathematical frameworks for qualifying learning tasks and algorithms.
- These are the subfields of machine learning that a machine-learning practitioner does not **needs to know** in great depth in order to achieve good results on wide range of problems.

5.17.2 Artificial Learning and Machine Learning

- Learning in machine learning refers to a machine's ability to learn and is based on data; and machine-learning algorithms ability to train a model, evaluate its performance or accuracy and then make predictions.
- AI is a technology that enables a machine to simulate human behaviour. Machine-learning is a subset of Artificial Intelligence, which allows a machine to learn automatically from past data without programming explicity. The goal of AI is to make a smart compute system like human to solve complex problems.

5.17.3 A.I and Deep Learning

- A.I means getting a computer to copy human behaviour in some ways. Deep learning is a subset of machine learning that enables computers to solve more complex problems.
- Deep Learning uses huge neural networks with many layers of processing units. It takes advantage of many improved training techniques in computing power to learn complex patterns in large amounts of data. Common applications include image and speech recognition.

Module 5

Scanned with CamScanner

5.17.4 Techniques of Deep Learning

- Deep learning theory is a subfield of AI and devoted to studying the design and analysis of machine algorithm.
- In statistical learning, algorithm is given samples that are labelled in some useful way. For example, the samples might be description of mushrooms and labels could be whether or not the mushrooms are edible.
- The algorithm takes these previously labelled samples and uses them to induce a classifier. This classifier is a function that assigns labels to samples, that have not been **seen** previously by the algorithm.
- The goal of statistical learning algorithm is to optimise some measure of performance such as minimising the number of mistakes made on new samples.
- In addition to performance bounds, computational learning theory studies the time complexity and feasibility of learning.
- In computational learning theory a computation is considered feasible if it can be done in polynomial time.

  There are two types of complexity results :

  (1) **Positive results :** It shows that a certain class of functions is learnable in polynomial time.

  (2) **Negative results :** It shows that certain classes cannot be learned in polynomial time.

Remark : Polynomial time

An algorithm is said to be of polynomial time if its running time is upper bound by a polynomial expression in the size of the input algorithm, that is

$T(n) = 0\ (n^k)$; for some positive constant K

Cobhom's thesis states that polynomial time is a synonym for 'tractable', 'feasible' 'efficient' or 'fast'.

5.18 PAC LEARNING

- In computational learning theory, 'Probably Approximately Correct' (PAC) learning is a framework for mathematical analysis of machine learning. PAC analyses the generalization error of a learning algorithm in terms of its error on a training set. The goal is to show that an algorithm achieves low generalization error with high probability.
- Probably approximately correct (PAC) learning theory helps analyse whether and under what conditions a learner L will probably output an approximately correct classifier.

5.18.1 Analysis of PAC

PAC analysis is used to compute transfer functions for circuits that exhibit frequency translation. It is a small signal analysis lie AC analysis, except that the circuits is first linearised about a periodically varying operating point as oppoced to a simple DC operating point.

5.18.2 Epsilon in PAC Learning

- Probability [error (h) > $\in$] < δ ; we are now in a position to say when a learned concept is good , [i.e. degree of goodness]
- 'When the [probability that its error is greater than accuracy $\in$] is less than the confidence level δ'
- Different degrees of 'goodness' will correspond to different values of epsilon and delta. The smaller epsilon and delta are, the better the learned concept will be.
- Our problem, for a given concept to be learned, and given epsilon and delta, is to determine the size of the training set. This may or may not depend on the algorithm used to derive the learned concept.
- Now, we want to learn the concept "medium-built person" from examples. We are given the height and weight of 'm' individuals, the training set. We are told for each [height, weight] pair if it is or not of medium built.
- We want to study this concept. We produce an algorithm that answers correctly if a pair [height, weight] represents/not a medium built person. We want to find the value of m to use if we want to learn the concepts.
- First to understand the concept; the probability of error of the learned concept is at most epsilon.
- Suppose C is the (true) concept we are learning, h is the concept we have learned, then

  error [h] = probability [C(x) < > h(x) for an individual x] < epsilon

Now, we set a bound delta on the probability that this error is greater than epsilon, that is,

Probability [error (h) > epsilon] < delta

Now, we assume that the concept is represented as a rectangle, with sides parallel to the axes height/weight, and with dimensions height-min, height-max; weight-min, weight-max.

Scanned with CamScanner

Now the hypothesis will take the form of a rectangle with the sides parallel to the axes. To build the learned concept, we build the algorithm.

(i) If there are no positive individuals in the training set, the learned concept is null.

(ii) We choose epsilon and delta and determine a value for m that will satisfy the PAC learning conditions.

5.18.3 Agnostic PAC Learning

- A learner that does not assume that it contains an error-free hypothesis and that simply finds the hypothesis with minimum training error is called as an agnostic learner.
- An important innovation of PAC frame work is the introduction of computational complexity theory concepts to machine learning.
- In particular the learner is expected to find efficient functions (time and space requirement bounded to a polynomials of the example size) , and the learner itself must implement an efficient procedure.

5.19 REINFORCEMENT LEARNING (R.L)

- We have already seen the meaning of Reinforcement learning, in nutshell. Now we discuss R.L. in detail.
- Reinforcement learning is an area of machine learning. In R.L., reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it has to learn from its experience.

5.19.1 Example : We Consider a Problem

- We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best solution path to reach the reward.
- The following range 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 fire. 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 subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond.

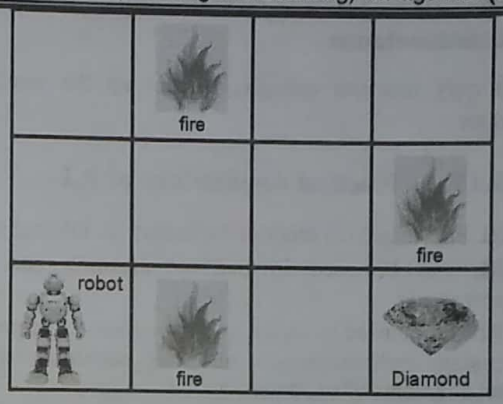


Fig. 5.19.1

5.19.2 Method of R.L.

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

(ii) **Output :** There are many possible output as there are varieties of solution to a given problem.

(iii) **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.

(iv) The model keeps on learning

(v) The best solution is decided based on the maximum reward.

5.19.3 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 RL

(i) Maximise performance

(ii) Sustain change for a long period of time

Disadvantages of RL

Too much Reinforcement can lead to overload of states which can diminish the results.

(2) **Negative :** Negative reinforcement is defined as strengthening of a behaviour because a negative condition is stopped or avoided.

Advantages

(i) Increases behaviour

(ii) Provide defiance to minimum standard of performance

Module 5

Scanned with CamScanner

Disadvantages

It only provides enough to meet up the minimum behaviour.

5.19.4 Practical Applications of R.L

(1) RL can be used in robotics for industrial automation.

(2) RL can be used in machine learning and data processing.

(3) RL can be used to create training systems that provide custom instruction and materials according to the requirements of students.

RL can be used in large environment in the following situations :

(i) A model of the environments is known, but an analytic solution is not available.

(ii) Only a simulation model of the environment is given (the subject of simulation based optimisation)

(iii) The only way to collect information about the environment is to interact with it.

(4) Training the models that control autonomous cars is an excellent example of a potential application of reinforcement learning.

- In an ideal situation, the computer should get no instructions on driving the car. The programmer would avoid hard-wiring anything connected with the task and allow the machine to learn the from its own errors. In a perfect situation, the only hard-wired element would be the reward function.
- **For example** in usual circumstances we would require an autonomous vehicle to put safety first, minimize ride time, reduce pollution offer passengers comfort and obey the rules of law.
- With an autonomous race car, we would emphasis, speed much more than the driver's comfort. The programmer cannot predict everything that could happen on the road. Instead of building lengthy "if-then" instructions, the programmer prepares the reinforcement learning agent to be capable of learning from the system of rewards and penalties. The agent, it is the another name for reinforcement learning algorithms performing the task gets rewards for reaching specific goals.
- In particular, if AI is going to drive a car, learning to play some Atari classics can be considered a meaningful intermediate milestone. A potential application of reinforcement learning in autonomous vehicles is the following interesting case :
- A developer is unable to predict all future road situations, so letting the model train itself with a system of penalties and rewards in a varied environment is possibly the most effective way for the AI to broaden the experience it both has and collects.

5.19.5 Challenges with RL

- The main challenges in RL is in preparing the simulation environment, which is highly dependent on the task to be performed. When the model has to go superhuman in chess, Go or Atari games, preparing the simulation environment is relatively simple. When it comes to building a model capable of driving on autonomous car, building a realistic simulator is crucial before letting the car ride on the street.
- The model has to figure out how to brake or avoid a collision in a safe environment, where sacrificing even a thousand cars comes at a minimum cost. Transferring the model out of the training environment and into the real world is where things get tricky. Scaling the neural network controlling the agent is another challenge. There is no way to communicate with the network other than through the system of rewards and penalties. This in particular may lead to catastrophic forgetting, where acquiring new knowledge causes some of the old to be erased from the network.
- Yet another challenge is reaching a local optimum that is the agent performs the task as it is; but not in the optimal or required way.

5.19.6 Challenges in Deep Reinforcement Learning Research and Applications

The existing challenges in deep reinforcement learning research and applications, including :

(1) The sample efficiency problem

(2) Stability of training

(3) The catastrophic interference problem

(4) The exploration problem

(5) Meta-learning and representation learning for the generality of reinforcement learning methods across tasks;

(6) Multi-agent reinforcement learning with other agents an part of the environment

(7) sim-to-real transfer for bridging the gaps between simulated environments and the real world.

Scanned with CamScanner

(8) Large-scale reinforcement learning with parallel training frameworks to shorten the wall-clock time for training etc.

Some key words to be noted

- Sample efficiency – stability
- Catastrophic interference - Exploration
- Meta learning – Representation learning Generality
- Multi agent reinforcement learning
- Sim2real-scalability

5.20 LEARNING FROM REWARDS AND PUNISHMENT

- The concept of Rewards and punishment was developed by D.F. skinner.
- Operant conditioning is a way of learning by means of rewards and punishments. This type of conditioning holds that a certain behaviour and a consequence, either a reward or punishment have a connection which brings about learning.
- Rewards and punishment motivate behaviour, but it is not clear how they impact skill performance and whether the effect varies across skills. Collectively these results suggest that punishment impacts skilled behaviour more than reward in a complex, task dependent fashion.
- If we are doing something that is rewarding, we continue to do it; and if it is punishing, we stop doing it. Therefore the reward and punishment centrers undoubtedly constitute one of the most important of all the controllers of our bodily activities, our drives, our aversions, our motivations.

5.20.1 Effectiveness of Rewards and Punishments

- Neuroscience suggests that it comes to motivating action (e.g. getting people to work longer hours or producing star reports etc.) rewards may be more effective than punishment.
- So our brain has evolved to accomodate our environment in which often the best way to gain rewards, is to take actions.
- The agent is rewarded for correct moves and punished for the wrong ones. In doing so, the agent tries to minimize wrong moves and maximize the right ones.

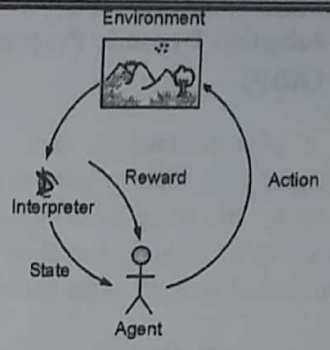


Fig.5.20.1

5.21 PASSIVE REINFORCEMENT LEARNING AND ACTIVE REINFORCEMENT LEARNING

- In case of passive reinforcement learning, the agent's policy is fixed which means that it is told what to do. Hence the goal of passive RL agent is to execute a fixed policy (i.e. sequence of actions) and evaluate it while that of an active RL agent is to act and learn an optimal policy.
- In contrast to this, in active RL an agent needs to decide what to do an there is no fixed policy that it can act on.

5.22 PASSIVE REINFORCEMENT LEARNING

As the goal of the agent is to evaluate how good an optimal policy is the agent needs to learn the expected utility u $\pi(s)$ for each state s. This can be done in 3 ways :

5.22.1 Direct Utility Estimation

- In this method, the agent executes a sequence of trials or runs (sequence of states action transitions that continue until the agent reaches the terminal state). Each trial gives a sample value and the agent estimates the utility based on the sample value. And it can be calculated as running averages of sample values.
- The main drawback is that this method makes a wrong assumption that state utilities are independent while in reality they are Markovian. Also it is slow to converge.

Module 5

Scanned with CamScanner

5.22.2 Adaptive Dynamic Programming (ADP)

- ADP is a smarter method than Direct Utility Estimation as it runs trials to learn the model of the environment by estimating the utility of a state as a sum of rewards for being in that state. And expected discounted reward for being in the next state :

$$U^{\pi}(s) = R(s) + r \sum_{s'} P(s' \mid s, \pi(s))\, U^{\pi}(s')$$

Where $R(s)$ = reward for being instate s

$P(S'/S, \pi(s))$ = transition model,

ν = discounted factor

And $U^{\pi}(s)$ = utility of being in state S

- It can be solved using value-iteration algorithm. The algorithm converges fast but can become quite costly to compute for large state spaces.
- ADP is a model based approach and requires the transition model of the environment.
- A model free approach is temporal difference learning.

5.22.3 Temporal Difference Learning (TDL)

- TDL does not require the agent to learn the transition model. The updates occur between successive states and agents only updates states that are directly affected

Here , $u^{\pi}(s) \leftarrow u^{\pi}(s') + \alpha\,[R(s) + \nu\,(u^{\pi}(s') - u^{\pi}(s))]$

Where,

α = learning rate which determines the convergence to true utilization

- While ADP adjusts the utility of s with all its successor states, TD learning adjusts it with that of a single successor state s′. TDL is slower in convergence of computation.

5.22.4 Active Reinforcement Learning

- As the goal of an active agent is to learn an optimal policy, the agent needs to learn the expected utility of each state and update its policy.
- It can be done using a passive ADP agent and then using value or policy iteration, it can learn optimal actions. Bu this approach results into a greedy agent.
- Hence, we use an approach that gives higher weights to unexplored actions and lower weights to actions with lower utilities.

$$U_{i+1}(s) \leftarrow R(s) + \nu \max_{a \in A} f\left(\sum_{s'} P(s'/s,a)\, u_i(s'), N(s,a)\right)$$

Where f(u,n) is the exploration function that increases with expected value u and decreases with number of trails n.

$$f(u, n) = \begin{cases} R^{+}, & \text{if } n < Ne \\ u, & \text{otherwise} \end{cases}$$

where R^{+} is an optimistic reward and N_e is the number of times we want an agent to be forced to pick an action in every state. The exploration function converts a passive agent into active one.

While ADP adjusts the utility of s with all its successive states, TD learning adjusts it with that of a single successor state S′.

TD is slower in convergence of computation.

5.23 Q-LEARNING

- Q-learning is a model free reinforced learning algorithm to learn the value of an action in a particular state.
- It does not require a model of the environment (hence 'model-free') and it can handle problems with stochastic transitions and rewards without requiring adaptations.
- For any finite Markov decision process (FMDP), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state.
- Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly-random policy. 'Q' refers to the function that the algorithm computes – the expected rewards for an action taken in a given state.

5.23.1 Goal of the Agent

- R.L. involves an agent, a set of states S and a set A of actions per state. By performing an action $a \in A$, the agent transitions from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score)
- The goal of the agent is to maximise its total reward. It does this by adding the maximum reward attainable from future states to the reward for achieving its current state, effectively influencing the current action by the potential future reward. This potential reward is

Scanned with CamScanner

a weighted sum of the expected values of the rewards of all future steps starting from the current state.

- As an example, consider the process of boarding a train in which the reward is measured by the negative of the total time spent boarding (alternatively, the cost of boarding the train is equal to the boarding time)
- One strategy is to enter the train door as soon as they open, minimising the initial wait time for yourself. If the train is crowded then you will have a slow entry after the initial action of entering the door as people are fighting you to depart the train as you attempt to board. The total boarding time, or cost, is then 'O seconds wait time +15 seconds fight time'.
- On the next day, by random chance (exploration), you decide to wait and let other people depart first. This results in a larger wait time initially. But the time fighting other people is less. Overall, this path has a higher reward than that of the previous day, since the total boarding time is now.

'5 seconds wait time +0 seconds fight time'

- Through exploration, despite the initial action resulting in a larger cost (or negative reward) than in the forceful strategy, the overall cost is lower, and it reveals a more rewarding strategy.

5.23.2 Influence of Variables : Learning Rate

- The learning rate or step size determines to what extent acquired information overrides old information.
- A factor of O makes the agent learn nothing (exclusively exploiting prior knowledge) while a factor 1 makes the agent consider only the most recent information (ignoring prior knowledge to explore possibilities). In fully deterministic environments, a learning rate of $\alpha_t = 1$ is optimal. When the problem is stochastic, the algorithm converges under some technical conditions on the learning rate that requires it to decrease to zero. In practice, often a constant learning rate is used, such as $\alpha_t = 0.1$ for all t.

5.23.3 Discount Factor

- The discount factor ν determines the importance of future rewards. A factor of O will make the agent short-sighted (myopic) by only considering current rewards, i.e. T_1 while a factor approaching 1 will make it strive for a long term high reward. If the discount factor meets or exceeds 1, the action values may diverge, for $\gamma = 1$, without a terminal state, all environment histories become infinitely long and utilities with additive, undiscounted rewards generally become infinite.
- Even with a discount factor only slightly lower than 1, Q-function learning leads to propagation of errors and instabilities when the value function is approximated with an artificial neural network.
- In that case, starting with a lower discount factor and increasing it towards its final value accelerates learning.

5.23.4 Initial Conditions (Q_0)

- Since Q-learning is an iterative algorithm, it assumes an initial condition before the first update occurs. High initial conditions before the first update occurs. High initial values, also known as **'optimistic initial conditions'** can encourage exploration.
- No matter what action is selected, the update rule will cause it to have lower values than the other alternative and that increases the choice of probability.
- The first reward γ can be used to reset the initial conditions. According to this idea, the first time an action is taken the reward is used to set the value of Q. This allows immediate learning in case of fixed deterministic rewards. A model that incorporates reset of initial conditions (RIC) is expected to predict participants behaviour better than a model that assumes any arbitrary initial condition (AIC). RIC seems to be consistent with human behaviour in repeated binary choice experiment.

5.23.5 Implementation

- Q-learning can be combined with function approximation. This makes it possible to apply the algorithm to larger problems, even when the state-space is continuous.
- One solution is to use an artificial neural network as a function approximator. Function approximation may speed up learning in finite problems, due to the fact that the algorithm can generalise earlier experiences to previously unseen states.

Module 5

Scanned with CamScanner

5.23.6 Quantization

- Another technique to decrease the state/action space quantizes possible values.
- Consider an example of learning to balance a stick on a finger. To describe a state at a certain point in time involves the position of the finger in space, its velocity, the angle of the stick, and the angular velocity of the stick. This gives a four element vector that describes **one state,** i.e. a snapshot of one state encoded into four values.
- The problem is that infinitely many possible states are present.
- The exact distance of the finger from its starting position ($-\infty$ to ∞) is not known. But rather it is far away or not can be located

5.23.7 Variants (Deep Q-learning)

- When a non-linear function approximator such as a neural network is used to represent Q, then reinforced learning becomes unstable or divergent. The reason for this instability is due to the correlation present in the sequence of observations.
- The small updates to Q may change the policy of the agent and the data distribution, and the correlations between Q and the target values.
- Here the technique uses a random sample of prior actions instead of the most recent action to proceed. This removes correlation in the observation sequence and smooths changes in the data distribution.
- Iterative updates adjust Q towards target values that are only periodically updated further reducing correlations with the target.

5.23.8 Delayed Q-learning

- Delayed Q-learning is an alternative implementation of the online Q-learning algorithm, with probably approximately correct (PAC) learning.
- (Greedy) GQ is a variant of Q-learning to use in combination with (linear) function approximation. The advantages of Greedy GQ is that convergence is guaranteed even when function approximation is used to estimate the action values.
- Distributed Q-learning is a variant of Q-learning which seeks to model the distribution of returns rather than the expected return of each action. It has been observed to facilitate estimate by deep neural networks and can enable alternative control methods, such as risk sensitive control.

5.23.9 Limitations

- The standard Q-learning algorithm (using Q-table) applies only to discrete action and state spaces. Discretisation of these values leads to inefficient learning, largely due to the dimensionality.
- However, there are adaptations of Q-learning that attempts to solve this problem such as wire-fitted Neural Q-learning.

5.23.10 Temporal Difference Learning (TD)

- TD learning refers to a class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of the value function. These methods sample from the environment, like Monte Carlo methods, and perform updates based on current estimates, like dynamic programming methods.
- While Monte-Carlo methods only adjust their estimates once the final outcome is known, TD methods adjust predictions to match later, more accurate, predictions about the future before the final outcomes is known. This is a form of bootstrapping. We illustrate this with the following example :

"Suppose you wish to predict the weather for Saturday, and you have some model that predicts Saturday's weather given the weather of each day in the week. In the standard case, you would wait until Saturday and then adjust all your models. However, when it is, for example, Friday, you should have a pretty good idea of what the weather would be on Saturday- and thus be able to change, say, Saturdays model before Saturday arrives. "

5.23.11 TD Algorithm in Neuroscience

- Researchers have discovered that the firing rate of dopamine neurorns in the ventral tegmental area (VTA) and substantia nigra (SIVc) appear to reduce the error function in the algorithm. The error function reports back the difference between the estimated reward at any given state or time step and the actual reward received.

Scanned with CamScanner

- The larger the error function, the larger the difference between the expected and actual reward. When this is paired with a stimulus that accurately reflects a future reward, the error can be used to associate the stimulus with the future reward.
- The relationship between the model and potential neurological function has produced research attempting to use TD to explain many aspects of behavioural research.

Remark

(1) The 'q' in q-learning stands for quality. Quality in this case represents how useful a given action is in gaining some future reward

(2) Gamma in Q-learning is the discount factor. It quantifies how much importance we give for future rewards. It is also handy to approximate the noise in future rewards Gamma varies from 0 to 1. If Gamma is closer to 0, the agent will tend to consider only immediate rewards.

5.24 INTRODUCTION TO STATISTICAL LEARNING

- In statistical learning a set of rules is used to understand the given data.
- Statistical learning theory is a framework for machine learning. It is developed using statistics and functional analysis.
- Statistical learning theory finds predictive function using given data to solve the problem.
- Statistical learning is understanding from training data and predicting on unseen data.
- Statistical learning is estimating a function f from the given data. Using f_c we can better grasp the relation between the dependent variable Y on X. Thus f gives a the systematic information that X provides about Y. Generally X is a vector value quantity of the types $X = (X_1, X_2, \ldots, X_m)$
- Statistical learning theory has practical applications in the fields of computer vision, speech recognition and bioinformatics.
- Statistical learning theory begins with a class of hypothesis and uses empirical data to select one hypothesis from that class. If the data generating mechanism is soft, then the difference between the training error of a hypothesis from the class is small (and hence may be negligible).
- Statistical learning theory comes under different classes : supervised learning and unsupervised learning on line learning and reinforcement learning.
- Supervised learning is predicting an output based on one or more inputs. Supervised learning involves learning from a training set of data. Every point in the training set is an input-output pair, where input maps the output. The learning method infers the function that maps between the input and the output and the learned function is used to predict the output from future input.

Types of supervised learning problems

- Supervised learning problems are problems of regression or problems of classification. And that depends on the type of output.If the output takes continuous values, it is a regression problem. For example, consider Ohm's law. If voltage is taken as input and current as output then regression will be given by the functional relation V = I-R.
- If the output takes discrete set of values then it is a classification problem. Classification is very common for machine learning applications.In facial recognition, we regard a person's face as input and person's name as output. Here the input is a large multidimensional vector whose elements are pixels in the picture.

Module 5

Chapter Ends...

Scanned with CamScanner

Module 6

CHAPTER 6

AI Applications

Syllabus

A. Introduction to NLP- Language models, Grammars, Parsing
B. Robotics - Robots, Robot hardware, Problems Robotics can solve
C. AI applications in Healthcare, Retail, Banking

6.1 Introduction 6-3
6.2 Overview of NLP Task 6-3
6.2.1 Common NLP Tasks in Software Programs 6-3
6.2.2 Evolution of NLP Systems or History of NLP 6-3
6.2.3 Components of NLP 6-4
6.2.4 Two Major Methods of NLP Analysis 6-4
6.2.5 The NLP Tasks 6-4
6.2.6 Different Levels of NLP 6-4
What are the levels of knowledge used in language understanding? Also write down the techniques used in NLP. (MU - Q. 6(b), Dec. 15, 10 Marks) 6-4
UQ. Write short note on : Steps in Natural Language Processing. (MU - Q. 6, May 17, 10 Marks) 6-4
UQ. What are steps involved in natural language processing (NLP) of an English sentence ? (MU - Q. 6(i), Dec. 17, 10 Marks) 6-4
UQ. What are steps involved in natural language processing (NLP) of an English sentence ? Explain with an example sentence. (MU - Q. 5(b), May 18, Q. 6(a), Dec. 18, Q. 6(b), May 19, 10 Marks) 6-4
6.2.7 Basic Steps of Natural Language Processing 6-5
6.2.8 Applications of NLP 6-5
6.3 NLP-Language Models 6-6
6.3.1 Challenges with Language Modelling 6-6
6.3.2 Types of Language Models 6-6
6.3.3 Statistical Language Models 6-6
6.3.4(I) N-Gram 6-6
6.3.5 (II) Neural Language Models 6-7
6.3.6 Some Common Examples of Language Models 6-7
6.3.7 Synthetic and Semantic Analysis in NLP 6-7
6.3.8 Plan to Use Language Models 6-8
6.4 Bert Model 6-8
6.4.1(I) Masked Language Model (MLM) 6-8

Scanned with CamScanner

6.4.2(II) Next Sentence Prediction (NSP) 6-8
6.4.3 Method of 'BERT ' Works 6-8
6.4.4 Uses of 'BERT' 6-8
6.5 Grammars in a.i. 6-9
6.5.1 Different Types of Grammars Languages 6-9
6.5.2 Types of Grammar 6-9
6.5.3 Grammars and Languages 6-9
6.6 Parsing : Syntactic Analysis 6-10
6.6.1 Bottom-up Parsing 6-10
6.6.2 Comparison between Top-down and Bottom-up Parsing 6-11
6.7 Robotics 6-11
6.7.1 Robotics Aspects 6-11
6.8 Applications of Robotics 6-12
6.9 Future Applications of Robotics 6-13
6.9.1 Robotics in Public Security 6-14
6.9.2 Robots in Education 6-14
6.9.3 Robots at Home 6-14
6.9.4 Robots as Co-workers 6-14
6.9.5 Robots might take our Jobs 6-14
6.9.6 They Create Jobs, Too 6-14
6.9.7 Autonomous Cars 6-14
6.9.8 Healthcare Robots 6-15
6.9.9 Robotics for Entertainment 6-15
6.9.10 Robots will Boost our Standard of Living 6-15
6.10 Field Programmable Gate Arrays (FPGA) in robotics 6-15
6.10.1 Technology in FPGA 6-15
6.10.2 Planning Algorithms 6-15
6.10.3 Four Dimensions 6-16
6.10.4 Components of Robots 6-16
6.11 Robot Hardware 6-16
6.11.1 Software "Robots" V/S Hardware Robots in RPA 6-16
6.11.2 Difference between Software Robots and Hardware Robots 6-17
6.12 Applications of a.i. in Health Care 6-17
6.12.1 Health Monitoring 6-17
6.12.2 Medical Imaging 6-18
6.12.3 AI Robot assisted Surgery 6-18
6.13 Other Applications 6-18
6.13.1 Retail 6-18
6.13.2 AI Applications in Banking 6-18
6.13.3 Prediction of Future Trends 6-18
6.13.4 Robotic Automation of Processes 6-18
• **Chapter Ends** **6-18**

Scanned with CamScanner

6.1 INTRODUCTION

Communication is an important act that agent can perform so as to exchange information with the environment. Communication can be carried out by producing and percepting certain signs drawn from a shared system of conventional signs.

In a partially observable world, communication can help agents to learn information that is observed or inferred by others. This information can make agent more successful.

Language is meant for communicating about the world. By studying language, we can come to understand more about the world. We can test our theories about the world by how well they support our attempt to understand language.

And, if we can succeed at building a computational model of language, we will have a powerful tool for communicating about the world. In this chapter, we look at how we can exploit knowledge about the world, in combination with linguistic facts, to build computational natural language systems."

6.2 OVERVIEW OF NLP TASK

GQ. Give general approaches to natural language process. **OR** Write short note on NLP.

(1) Natural language processing (NLP) is the ability of a computer program to understand human speech as it is spoken. NLP is a component of artificial intelligence (AI).

(2) The development of NLP applications is challenging because computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured or, perhaps through a limited number of clearly-enunciated voice commands. Human speech, however, is not always precise - it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

(3) Current approaches to NLP are based on machine learning, a type of artificial intelligence that examines and uses patterns in data to improve a program's own understanding. Most of the research being done on natural language processing revolves around search, especially enterprise search.

6.2.1 Common NLP Tasks in Software Programs

- Sentence segmentation, part-of-speech tagging and parsing.
- Deep analytics.
- Named entity extraction.
- Co-reference resolution.

The advantage of natural language processing can be seen when considering the following two statements :

"Cloud computing insurance should be part of every service level agreement (SLA)"and "A good SLA ensures an easier night's sleep -- even in the cloud".

If you use national language processing for search, the program will recognize that cloud computing is an entity, that cloud is an abbreviated form of cloud computing and that SLA is an industry acronym for service level agreement. The ultimate goal of NLP is to do away with computer programming languages altogether. Instead of specialized languages such as Java or Ruby or C, there would only be "human."

6.2.2 Evolution of NLP Systems or History of NLP

GQ. Discuss the evolution of NLP systems **Or** Given a brief history of NLP.

(1) The work related to NLP was started with machine translation (MT) in 1950s. Alan Turing proposed is called the Turing test in 1950s. It is the testing ability of the machine program to have written conversation with human.

(2) This program should be written so well so that one would find it difficult to determine whether the conversation is with a machine or it is with the other person actually. During the same period, cryptography and language translation took place. Later on, syntactic structures came up along with linguistics. Further, the sentences were considered with knowledge augmentation and semantics. In 1960s, ELIZA (the most common NLP system) was developed that gained popularity.

(3) It was the simulation of a psychotherapist. At a very later stage, it was the case grammars that came up. Now, there has been a complete revolution in the NLP with the machine learning approaches coming up. Many NLP systems have been developed till today and a lot of competitions are being organized that are based on the Turing test.

Module 6

Scanned with CamScanner

6.2.3 Components of NLP

There are two components of NLP : Mapping the given input in the natural language into a useful representation. Different level of analysis required: morphological analysis, syntactic analysis, semantic analysis, discourse analysis.

(1) **Natural language generation :** Producing output in the natural language from some internal representation. Different level of synthesis required: deep planning (what to say), syntactic generation

(2) **NL understanding :** NL Understanding is much harder than NL Generation. But, still both of them are hard.

Planning : Planning problems are hard problems. They are certainly nontrivial. Method which we focus on ways of decomposing the original problem into appropriate subparts and on ways of handling interactions among the subparts during the problem-solving process are often called as planning. Planning refers to the process of computing several steps of a problem-solving procedure before executing any of them.

6.2.4 Two Major Methods of NLP Analysis

GQ. Write the name of two major method of NLP analysis. **Or** Eenumerale different approaches for exploring the semantics of parsed natural language sentences.

There are several techniques used in analyzing natural language processing. Some of them can be briefly described as follows :

1. **Pattern matching :** The idea here is an approach to natural language processing is to interpret input statements as a whole further than building up their interpretation by combining the structure and meaning of words or other lower level constituents. That means the interpretations are obtained by matching patterns of words against the input statements. For a deep level of analysis in pattern matching a large number of patterns are required even for a restricted domain. This problem can be improved by hierarchical pattern matching in which the input is gradually canonical through pattern matching against sub phrases. Another way to reduce the number of patterns is by matching with semantic primitives instead of words.
2. **Syntactically driven parsing :** Syntax means ways that words can fit together to form higher level units such as phrases, clauses and sentences. Therefore syntactically driven parsing means interpretation of larger groups of words are built up out of the interpretation of their syntactic constituent words or phrases. In a way this is the opposite of pattern matching as here the interpretation of the input is done as a whole.
3. **Semantic grammars :** Natural language analysis based on semantic grammar is bit similar to syntactically driven parsing except that in semantic grammar the categories used are defined semantically and syntactically. There here semantic grammar is also involved.
4. **Case frame instantiation :** Case frame instantiation is one of the major parsing techniques under active research today. It has some very useful computational properties such as its recursive nature and its ability to combine bottom-up recognition of key constituents with top-down instantiation of less structured constituents.

6.2.5 The NLP Tasks

NLP problem can be divided into two tasks :

1. Processing written text, using lexical, syntactic knowledge, and
2. Semantic knowledge of the language as well as the required real world information.

Processing spoken language, using all the information needed above plus additional knowledge about phonology as well as enough added information to handle the further ambiguities that arise in speech.

6.2.6 Different Levels of NLP

UQ. What are the levels of knowledge used in language understanding? Also write down the techniques used in NLP. **(MU - Q. 6(b), Dec. 15, 10 Marks)**

UQ. Write short note on : Steps in Natural Language Processing. **(MU - Q. 6, May 17, 10 Marks)**

UQ. What are steps involved in natural language processing (NLP) of an English sentence ? **(MU - Q. 6(i), Dec. 17, 10 Marks)**

UQ. What are steps involved in natural language processing (NLP) of an English sentence ? Explain with an example sentence. **(MU - Q. 5(b), May 18, Q. 6(a), Dec. 18, Q. 6(b), May 19, 10 Marks)**

1. **Morphology :** It is the analysis of individual words that consist of morphemes the smallest grammatical unit. Generally, words with 'ing', 'ed' change the meaning of the word. This analysis becomes necessary in the determination of tense as well.

Scanned with CamScanner

2. **Syntax :** Syntax is concerned with the rules. It includes legal formulation of the sentences to check the structures. (Some aspects are covered in compiler's phase of syntax analysis that you must have studied). For example, 'Hari is good not to'. The sentence structure is totally invalid here.
3. **Semantic :** During this phase, meaning check is carried out and the way in which the meaning is conveyed is analyzed. The previous example is syntactically as well as semantically wrong. Now, consider one more example, i.e., 'The table is on the ceiling.' This is syntactically correct, but semantically wrong.
4. **Discourse integration :** In communication or even in text formats, often the meaning of the current sentence is dependent on the one that is prior to it. Discourse analysis deals with the identification of discourse structure.
5. **Pragmatic :** In this phase, analysis of the response from the user with reference to what actually the language meant to convey is handled. So, it deals with the mapping for what the user has interpreted from the conveyed part arid what was actually expected. For a question like "Do you know how long it will take to complete the job?", the expected answer is the number of hours rather than a yes or no.
6. **Prosody :** It is an analysis phase that handles rhythm. This is the most difficult analysis that plays an important role in the poetry or shlokus (chants involving the name of God) that follow a rhythm.
7. **Phonology :** This involves analysis of the different kinds of sounds that are combined. It is concerned with speech recognition. Can the analysis levels discussed be overlapped or interrelated? Yes. It is very much possible to have an analysis actually forming a fuzzy structure. They can work in stages, where the second level makes use of the analysis or the outcomes of the first level. We now study them in detail.

6.2.7 Basic Steps of Natural Language Processing

GQ. Describe the basic steps of natural language processing.

There are general five steps :

- **Lexical analysis :** It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.
- **Syntactic analysis (Parsing) :** It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as "The school goes to boy" is rejected by English syntactic analyzer.

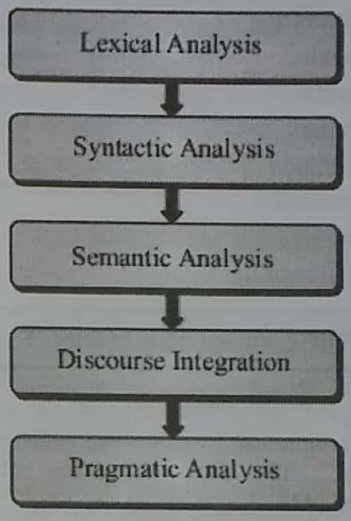


Fig. 6.2.1 : Basic steps of NLP

- **Semantic analysis :** It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as "hot ice-cream".
- **Discourse integration :** The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.
- **Pragmatic analysis :** During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

6.2.8 Applications of NLP

GQ. Explain application of NLP.

1. Spelling and grammar checking.
2. Optical character recognition (OCR).
3. Screen readers for blind and partially sighted users.
4. Augmentative and alternative communication (i.e., systems to aid people who have difficulty communicating because of disability).
5. Machine aided translation (i.e., systems which help a human translator, e.g., by storing translations of phrases and providing online dictionaries integrated with word processors, etc.).

Module 6

Scanned with CamScanner

6. Lexicographers tools.
7. Information retrieval.
8. Document classification (filtering, routing).
9. Document clustering.
10. Information extraction.
11. Question answering.
12. Summarization.
13. Text segmentation.
14. Exam marking.
15. Report generation (possibly multilingual).
16. Machine translation.
17. Natural language interfaces to databases.
18. E-mail understanding.

6.3 NLP-LANGUAGE MODELS

- Natural Language Processing (NLP) ability of a computer programme to understand human speech as it is spoken. NLP is a component of Artificial Intelligence (AI).
- Approaches to NLP are based on machine learning, a type of artificial intelligence that examines and uses patterns in data to improve a program's our understanding. Most of the research on natural language processing revolves around enterprise search.

6.3.1 Challenges with Language Modelling

- Formal language (like a programming language) are precisely defined. All the words and their usage are predefined in the system.
- Anyone who knows a specific programming language can understand what is written without any formal specification.
- Natural language, on the other hand is not designed. It evolves according to the convenience and learning of an individual. There are several terms in natural language that can be used in a number of ways. This introduces ambinguity but still can be understand by humans.
- Machines only understand the language of numbers for creating language models, it is necessary to convert all the words into a sequence of numbers. This is known as Encodings.
- Generally, a number is assigned to every word and this is called label-encoding in the sentence. "I love to play Tennis on weekends", every word is assigned a number [1, 2, 3, 4, 5, 6]. This is an example how encoding is done.

6.3.2 Types of Language Models

There are primarily two types of language models :

1. Statistical Language Models
2. Neural Language Models

6.3.3 Statistical Language Models

Statistical models include the development of probabilistic models that are able to predict the next word in the sequence, given the words that precede it. A number of statistical language models are in use already. We consider some of the popular models.

6.3.4(I) N-Gram

This is one of the simplest approaches to language modeling. Here, a probability distribution for a sequence of 'n' is created, where n can be any number and defines the size of the gram (or sequence of words being assigned a probability). If n = 4, a gram may look like : "can you help me". Basically, n is the amount of context that the model is trained to consider. There are different types of N-gram models such as unigrams, bigrams, trigrams, etc.

(1) **Unigram :** The unigram is the simplest type of language models. It does not look at any conditioning context in its calculations. It evaluates each word or term independently. Unigram models commonly handle language processing tasks such as information retrieval. The unigram is the foundation of more specific model variant called the query likelihood model, which uses information retrieval to examine a pool of documents and match the most relevant one to a specific query.

(2) **Bidirectional** : Unlike n-gram models, which analyse text in one direction (backwards), bidirectional models analyse text in both directions, backwards and forwards. These models can predict any word in a sentence or body of text by using every other word in the text. Examining text bidirectionally increases result accuracy. This type is often utilized in machine learning and speech generation applications. For example, Google uses a bidirectional model to process search queries.

(3) **Exponential :** This type of statistical model evaluates text by using an equation which is a combination of n-grams and feature functions. Here the features and parameters of the desired results are already specified. The model is based on the principal of entropy, which states that probability distribution with the most entropy is the best choice. Exponential models have

Scanned with CamScanner

fewer statistical assumptions which mean the chances of having accurate results are more.

(4) **Continuous Space :** In this type of statistical model, words are arranged as a non-linear combination of weight in a neural network. The process of assigning weight to a word is known as word embedding. This type of model proves helpful in scenarios where the data set of words continues to become large and include unique words.

In case where the data set is large and consists of rarely used or unique words, linear models such as n-gram do not work. This is because, with increasing words, the possible word sequences increase, and thus the patterns predicting the next word become weaker.

6.3.5 (II) Neural Language Models

- These language models are based on neural networks and are often considered as an advanced approach to execute NLD tasks.
- Neural language models overcome the shortcomings of classical models such as n-gram and are used for complex tasks such as speech recognition or machine translation.
- Language is significantly complex and keeps on evolving. Therefore, the more complex the language is, the better it would be at performing NLP tasks. Comparing to the n-gram model, an exponential or continues space-model proves to be a better option for NLP tasks because they are designed to handle combiguity and language variation.
- Meanwhile, language models should be able to manage dependencies. For example, a model should be able to understand words derived from different languages.

6.3.6 Some Common Examples of Language Models

Language models are the cornerstone of Natural Language Processing (NLP) technology. We have made the best of language models in our routine, without even realising it. We consider some of the examples of language models.

1. **Speech Recognition :** Voice assistants such as siri and Alexa are examples of how language models help machines in processing speech audio.
2. **Machine Translation :** Google Translator and Microsoft translate are examples of how NLP models can help in translating one language to another.
3. **Sentiment Analysis :** This helps in analysing the sentiments behind a phrase. This use-case of NLP models is used in products that allow businesses to understand a customer's intent behind opinions or attitudes expressed in the text. Hubspot's Service Hub is an example of how language models can help in sentiment analysis.
4. **Text Suggestions :** Google services such as Gmail or Google Docs use language models to help users get text suggestions while they compose an email or create long text documents respectively.
5. **Parsing tools :** Parsing involves analyzing sentences or words that comply with syntax or grammar rules. Spell checking tools are perfect examples of language modeling and parsing.

6.3.7 Synthetic and Semantic Analysis in NLP

NLP Analysis (problems) can be written in two parts :

(1) Processing written text, using syntactic knowledge, and

(2) Semantic knowledge of the language as well as the required real world information.

Processing spoken language using all the information needed as well as added information to handle the ambiguities that arise in speech.

Levels of NLP

1. **Morphology :** It is the analysis of individuals words that consists of smallest grammatical unit. Generally, words with 'ing' 'red' 'change the meaning of the word'. This analysis is necessary in the determination of tense also.
2. **Syntax :** Syntax is concerned with rules. It includes legal formulation of the sentences to check the structures ('Hari is good not to'. The sentence structure is totally invalid here.)
3. **Semantic :** During this phase, meaning is checked and the way in which the meaning is conveyed is analysed (The above example is syntactically as well as semantically wrong.). Consider an example, 'The table is on the ceiling'. This is syntactically correct, but semantically wrong.
4. **Discourse integration :** In communication, often the meaning of the current sentence is dependent on the one that is prior to it. Discourse analysis deals with the identification of discourse structure.
5. **Pragmatic :** In this phase, analysis of the response from the user with reference to what actually the language meant to convey is handled. For example, a question like "Do you know how long it will take to complete the job?", the expected answer is the number of hours rather than a yes or no.

Scanned with CamScanner

6. **Prosody :** It is the analysis phase that handles rhythm. This is the most difficult analysis that plays an important role in the poetry, that follows a rhythm.
7. **Phonolgy :** This involves analysis of the different kinds of sounds that are combined. It is concerned with speech recognition. It is very much possible to have an analysis forming a fuzzy structure. They work in stages, where the second level makes the use of the analysis of the first level.

6.3.8 Plan to Use Language Models

There are several innovative ways in which language models can support NLP tasks. An AI expert can create language models for executing simple to complex NLP tasks.

6.4 BERT MODEL

BERT model is best for NLP. BERT stands for Bidirectional Encoder Representations from Transformers. BERT is a pre-trained model designed to drive the context of a word from both sides of it, the left and the right.

BERT represents a new era for NLP because, although very accurate, it is basically based on two simple ideas. Using bidirectional capability, BERT pre-trained on two different, but related NLP tasks (i) Masked Language Modelling and (ii) Next Sentence Prediction.

6.4.1(I) Masked Language Model (MLM)

The objective of MLM training is to hide a word in a sentence and then have the program predict what word has been hidden (masked) based on the hidden word's context.

6.4.2(II) Next Sentence Prediction (NSP)

The objective of NSP is to have the program predict whether two given sentences have a logical, sequential connection or whether their relationship is simply random.

6.4.3 Method of 'BERT ' Works

- The goal of any given NLP technique is to understand human language as it is spoken naturally.
- In BERT's case, this typically means predicting a word in a blank. To do this, models typically need to train using a large repository of specialized, labeled training data. This necessitates laborious manual **data labeling.**
- But BERT is pretrained using only an unlabelled, plain text corpus. It continues to learn unsupervised from the unlabelled text and improve even as its being used in practical applications (i.e. Google Search).
- It pretraining serves as a base layer of 'knowledge' to build from. From there, BERT can adapt to the ever-growing body of searchable content and queries and be fine-turned to a user's specifications. This process is known as 'transfer learning.'
- BERT is made possible by Google's research on Transformers. The transformer is the part of the model that gives BERT its increased capacity for understanding context and ambiguity in language.
- The transformer does this by processing any given word in relation to all other words in a sentence, rather than processing them one at a time.
- By looking at all surrounding words, the transformer allows the BERT model to understand the full context of the word, and therefore better understand searcher intent.
- This is contrasted against the traditional method of language processing known as word embedding, in which previous models would map only one dimension.
- BERT uses a method of masked language modelling to keep the word with a fixed meaning, independent of its context. BEST then identifies the masked. Word based on the context alone. In BERT words are defined by their surroundings, not by a pre-fixed identify.
- BERT relies solely on 'self-attention' mechanism. And this is possible by the bidirectional transformers at the centre of BERT's design.
- BERT accounts for the meaning by reading bidirectionally accounting for the effect of all other words in a sentence on the focus word.

6.4.4 Uses of 'BERT'

BERT is used at Google to optimize the interpretation of user search queries. BERT excels at several functions, including :

(i) Sequence-to-sequence based language generation tasks such as :
   - Question-answering
   - Abstract-summation
   - Sentence-prediction
   - Conventional response generation

(ii) Natural language
   - Understanding tasks such as :
   - Polysemi and coreference (words that sound or look the same but have different meanings resolution)
   - Word sense disambiguation
   - Natural language inference
   - Sentiment classification

Scanned with CamScanner

BERT is expected to have a large impact on voice search as well as text-based search, which is error-prone with Google's NLP - techniques.

Because of BERT's proficiency in understanding context, it can interpret patterns that different languages share without understanding the language completely.

Thus, BERT has the potential to improve artificial intelligence systems across the board. BERT is open source, meaning any one can use it.

6.5 GRAMMARS IN A.I.

A grammar is a declarative representation that defines the syntactic facts of a language. The most common way to represent grammars is as a set of production rules, and the simplest structure for them to build is a parse tree which records the rules and how they are matched.

6.5.1 Different Types of Grammars Languages

Grammar type	Grammar accepted	Language accepted
Type 0	Unrestricted grammar	Recursive enumerated language
Type 1	Context-sensitive grammar	Context-sensitive language
Type 2	Context-free grammar	Context-free language
Type 3	Regular grammar	Regular language

6.5.2 Types of Grammar

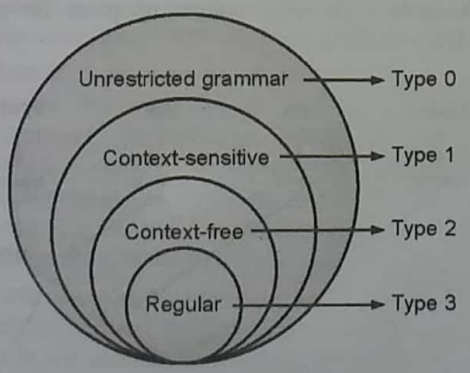


Fig. 6.5.1 : Types of Grammar

6.5.3 Grammars and Languages

- The types of grammars that exist are Noam Chomsky invented a hierarchy of grammars.
- The hierarchy consists of four main types of grammars.
- The simplest grammars are used to define regular languages.
- A regular language is one that can be understood by a finite state automation. Such languages are very simplistic and allow sentences such as aaabbbbcc"
- Recall that a finite state automation consists of a finite number of states, and rules that define how the automation can transit from one state to another.
- A finite state automation could be designed that defines the language and that consisted of a string of one or more occurrences of the letter a.
- Hence, the following strings would be valid strings in this language

aaa

a

aaaaaaaaaa

- Regular languages are of interest to computer scientists, but are not of great interest to the field of natural language processing because they are not so powerful to represent even simple formal language. And more complex natural languages are out of question.
- Sentences defined by a regular grammar are often known as regular expressions. The grammar that we defined above using rewrite rules is a context free grammar.
- It is context free because it does not specify the way that words should agree with each.

For example

- A state dog climbs mount Kanchanganga. Also, it follows the sentence, which is grammatically incorrect. They eats
- Thus, a content-free grammar can have only at most one terminal symbol on the right hand side of its rules.
- But for a context-sensitive grammar can have more than one terminal symbol on the right hand side. This enables the grammar to specify number, case, tense and gender agreement.

  Rewrite rules for context-sensitive grammars have the following form :

$$A X B \rightarrow A Y B$$

Module 6

Scanned with CamScanner

Which means that in the context of A and B, X can be rewritten as Y.

Each of A, B, X and Y can be either a terminal or a non-terminal symbol.

- Context-sensitive grammars are most usually used for natural language processing because they use the kinds of grammars that natural languages use.
- But they involve a much larger number of rules and make them harder for human developers to design than context-free grammars.
- The final class of grammars in Chomsky's hierarchy consists of recursively enumerable grammars (also known as unrestricted grammar)
- An unrestricted grammar can define any language and it has no restrictions on the structure of its rewrite rules.
- Computer scientists are interested in such grammars but not of use to the study of natural language processing.

6.6 PARSING : SYNTACTIC ANALYSIS

- We have seen that morphologic analysis can be used to determine to which part of speech each word in a sentence belongs. Now we examine how this information is used to determine the syntactic structure of s sentence.
- The process, in which we convert a sentence into a tree that represents the sentences syntactic structure, is known as '**parsing**'. Parsing a sentence tells us whether it is a valid sentence, defined by our grammar. If a sentence is not a valid sentence, then it cannot be parsed.
- Again, parsing a sentence involves producing a tree. We consider an example. The black cat crossed the road.

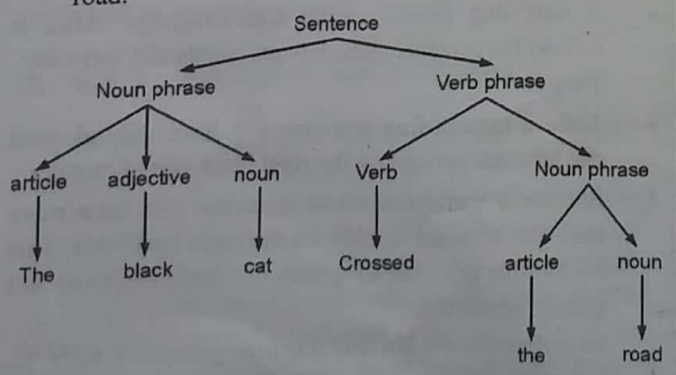
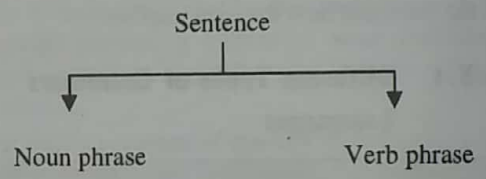


Fig. 6.6.1

- This tree shows how the sentence is made up of a noun phrase and a verb phrase. The noun phrase consists of an article, an adjective, and a noun.
- The verb phrase consists of a verb and a further noun phrase, which is turn consists of an article and a noun.
- Parse trees can be built in a bottom-up fashion or in a top-down fashion. When a parse tree is built from the top-down, it is known as a derivation tree.

Remark

- Building a parse tree from the top down involves starting from a sentence and determining which possible words can be applied to the sentence.
- In this case, sentence would be rewritten as



- Then the verb phrase and noun phrase would be broken down in the same way, until only terminal symbols are left.

6.6.1 Bottom-up Parsing

- Bottom up parsing is also known as shift-reduce parsing. Bottom up parsing is used to construct a parse tree for an input string.
- In the bottom-up parsing the parsing starts with the input symbol and construct the parse tree up to the start symbol by tracing out the rightmost derivations of string in reverse.

Example : To write bottom up parse for sentence 'Meera ate the apple'

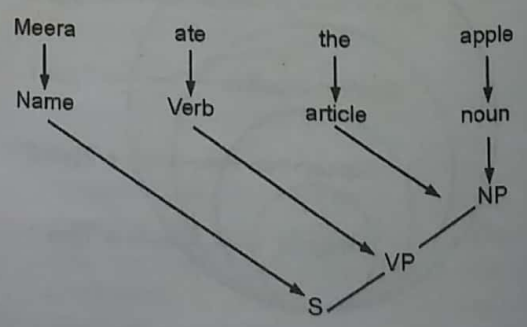


Fig. 6.6.2

Scanned with CamScanner

6.6.2 Comparison between Top-down and Bottom-up Parsing

Sr. no.	Top-down parser	Bottom-up parser
1.	Parser starts with the sentence S and rewrite into a sequence of terminal symbols that matches the classes of the words in the input sentence	A bottom up parser starts with the terminal symbols in the input sentence. The parser successively rewrite a sequence of terminal symbols and left hand side with the grammar rule.
2.	It starts at the root of derivation tree and fill in.	It starts at the leaves and fill in.
3.	It picks and tries to match the input.	It starts in a state valid for legal first tokens.
4.	The top-down strategy never wastes time in exploring trees that cannot result in S.	In bottom-up strategy, no lead to S.

6.7 ROBOTICS

- Robotics is 'an interdisciplinary branch of computer science and engineering'. Robotics involves design, construction, operation and use of robots. The goal of robotics is to design machines than can help and assist humans.
- Robotics covers fields of mechanical engineering, electrical engineering, information engineering, mechatronics, electronics bioengineering, computer engineering, control engineering, software engineering, mathematics etc.
- Robots can be used in many situations for many purposes, but today many are used in dangerous environment (including inspection of radioactive materials, bomb detection and deactivation), manufacturing processes or where humans cannot survive (e.g. in space, underwater, in high heat, and containment of hazardous materials and radiation).
- Robots can take on any form, but some are made to resemble humans in appearance. This is claimed to help in the acceptance of robots in certain behaviours which are usually performed by people. Such robots attempt to replicate walking, lifting, speech or any other human activity.
- Certain robots require user input to operate while other robots function automatically. The concept of creating robots that can operate automatically is to serve various practical purposes, whether domestically, commercially or militarily. Many robots are built to do jobs that are hazardous to people, such as defusing bombs, finding survivors in unstable ruins, exploring mines and shipwrecks. Robotics is also used in STEM. (Science, technology, engineering and mathematics) as a teaching aid.

6.7.1 Robotics Aspects

There are many types of robots; they are used in many different environment and for many different uses. Although being very diverse in application and form, they all share three basic similarities when it comes to their construction

1. **Robots all have some kind of mechanical construction**, a frame, form or shape designed to achieve a particular task. For example, a robot designed to travel across heavy dirt or mud, might use caterpillar tracks. The mechanical aspect is mostly the creators solution to completing the assigned task and dealing with the physics of the environment around it. Form follows function.
2. **Robots have electrical components** that power and control machinery. For example, the robot with caterpillar tracks would need some kind of power to move the tracker treads. That power comes in the form of electricity, which will have to travel through a wire and originate from a battery, a basic electrical circuit.

   Even petrol powered machines, that get their power mainly from petrol still require an electric current to start the combustion process which is why most petrol-powered machines like cars, have batteries.

   The electrical aspect of robots is used for movement (through motors), sensing (where electrical signals are used to measure things like heat, sound, position and energy status) and operation (robots need some level of electrical energy supplied to their motors and sensors in order to activate and perform basic operation)
3. **All robots contain some level of computer programming code**. A robot that needs to move across a muddy road may have the correct mechanical construction and receive the correct amount of power from its battery, but must not 90 anywhere without a program telling it to move.

   Programs are the core essence of a robot, it could have excellent mechanical and electrical construction, but if its program is poorly constructed, its performance will be very poor.

Module 6

Scanned with CamScanner

There are three different type, of robotic programs : remote control, artificial intelligence and hybrid.

A robot with remote-control programming has a preexisting set of commands, that it will only perform if and when it receives a signal from a control source, typically a human being with a remote control. It is perhaps more appropriate to view devices; controlled primarily by human commands as falling in the discipline of automation rather than robotic. Robots that use artificial intelligence interact with their environment on their own without a control source, and can determine reactions to objects and problems they encounter using their preexisting programming.

- Hybrid is a form of programming that incorporates AI and RC functions in them.

6.8 APPLICATIONS OF ROBOTICS

- As more and more robots are designed for specific tasks, the method of classification is more relevant. For example, many robots are designed for assembly work, which may not be able to function for other applications. They are termed as **'assembly robots'**
- For seam welding, some suppliers provide complete welding systems with the robot i.e. the welding equipment along with other material handling facilities like turntables etc. as an integrated unit. Such an integrated robotic system is called a **'welding robot'**
- Some robots are specifically designed for heavy load manipulation and labelled as **'heavy-duty robots.'**

Current and Potential Applications are as follows :

(1) Industrial Robots

Robots are increasingly used in manufacturing. Automotive industry is the main customer of industrial robots with 52% of sales in the auto industry, they can amount for more than half of the 'labour'. There are even "lights off" factories such as an IBM keyboard manufacturing factory in Texas that is fully automated.

(2) Construction Robots

- Construction robots can be separated into three types :
  (a) traditional robots (b) robotic arm
  (c) robotic exoskeleton
- Construction robots are a sub-set of industrial robots used for building and infrastructure construction at site. These robots have to be able to move and fix itself to the working zone, handle construction materials and interact with humans and other machineries. These robots have been successful to do works such as finishing the exterior, steel placement, construction of masonary wall reinforcement concrete etc.
- The main challenges to use robots in site is due to limitation in work-space.

(3) Military Robots

- Military robots are autonomous robots or remote controlled mobile robots designed for military applications, from transport to search and rescue and attack. Some such systems are currently in use and many are under development.
- Artillergy has seen promising research with an experimental weapons system named "Dragon fire II" which automates loading and ballistics calculations required for accurate predicted fire, providing a 12-seconds response time to fire support requests. However, military weapons are prevented from being fully autonomous; they require human input at certain intervention points to ensure that targets are not within restricted fire areas.
- There have been some developments towards developing autonomous fighter jets and bombers. The use of autonomous fighters and bombers to destroy enemy target is especially promising because of the lack of training required for robotic pilots, autonomous planes are capable of performing tasks which otherwise cannot be done by pilots. Plane design does not require a life support system and loss of a plane does not mean a loss of a pilot.
- But the largest drawback to robotics is their inability to accommodate for non-standard conditions.

(4) Space Exploration

- There are many things in space that are very dangerous for astronauts to do. Astronaut cannot room on mars all day to collect soil samples or work on repairing a space-ship from the outside while it is in deep space.
- In these situations, robots are a great choice because there are no chances for the loss of human life. So space institutions like NASA frequently use robots and autonomous vehicles to do things that humans cannot for example. Mars Rover is an autonomous Robot that travels on Mars and takes pictures of Martian Rock formation that are important and then sends them back on Earth for NASA scientists to study.

(5) Entertainment

- Robots also play important role in the entertainment industry. They can be used behind the sets in movies and serials to manage the camera, provide special effects, etc. They can be used for boring repetitive tasks that are not suitable for a human. Robots can also be used to do stunt work that is very dangerous for humans but looks pretty good in an action movie.

Scanned with CamScanner

- Theme parks like Disney World are also using autonomous robots to enhance the magical experience of their customers.

(6) Agriculture

- Agriculture is the sector that is the basis of human civilization. However, agriculture is also a seasonal sector that is dependent on ideal weather conditions, optimal soil etc.
- Moreover, there are many repetitive tasks in agriculture that are a waste of farmers time, and can be performed more suitable by robots. These include seeding used control, harvesting etc. robots are usually used for harvesting the crops which allow farmers to be more efficient.
- An example of a robot that is used to remove weeds in farms is the Ecorobotix. It is powered by solar energy and can be used to target and spray weeds using a complex camera system.

(7) Underwater Exploration

- Robots are a great option for exploring places that humans cannot reach easily, like the depths of the ocean. There is a lot of water pressure deep in the ocean which means humans cannot go that down and machines such as submarines can only go to a certain depth as well. A deep underwater is a mysterious place that can finally be explored using specially designed robots.
- These robots are remote-controlled, and they can go into depths of the ocean to collect data and images about the aquatic plant and animal life.

(8) Security

- If the security quards are robots, then even thieves would be scared ! That is why robots are being proposed as security agents as they can protect humans, and they would not be in danger like human security guards would be. Currently, robotics companies are working on pairing robot guards with human security consultants.
- The company Knight scope in the United states is a very famous company in this field. It has autonomous security robots capable assisting human security guards with real-time, actionable intelligence. These robots can help with crimes such as armed robberies, burglaries, domestic violence, fraud, hit and runs etc.

(9) Food Preparation

- There are robots that even can cook and create complete meals. These robots chefs can create food using hundreds of different recipes.
- All humans need to do is choose the recipe they want and provide the robot with prepackaged containers of all the ingredients that are needed for that recipe.
- The robot can then cook the food on its own. Moley Robotics is the company that has created a robotic kitchen with a robot and the robot can cook like a master chef. Thus, now a robot can cook food for you.

(10) Manufacturing

- There are many repetitive and common tasks in the manufacturing industry that don't require any usage of mind or imagination. The tasks like welding, assembly, packing etc. These tasks can be easily done by robots.
- These robots can be trained to perform these repetitive and monotonous tasks with precision under the guidance and supervision of a human.
- This option is also best for the manufacturing processes that are dangerous and may be harmful to humans.

(11) Customer Service

- There are robots that are developed to look exactly like humans for cosmetic purposes. These robots are primarily used in the field of customer service in high visibility area to promote robotics.
- One example is Nadine, a humanoid robot in Singapore that can recognize people from previous visits, make eye contact, shake hands, continue chatting based on previous meetings etc. Another such customer service robot is Junko Chihira in Japan, a humanoid robot working at the tourist information centre.
- So far we have seen various applications of robot, in different industries. Robots are used these days for everything, ranging from security guards, chets, doctor's assistants, customer service agents, and even a one – man army in war ! Because of their precision and programming to perform various tasks that are dangerous, boring or repetitive to humans.
- Robots can do all the physically demanding tasks that are difficult or impossible for humans, and they are becoming smarter with the advances in artificial intelligence. Thus Robots can be the perfect helper for humans and solve many problems in different industries.

6.9 FUTURE APPLICATIONS OF ROBOTICS

- Based on the current trends, robotics can transform our future. Here we discuss some of the ways.
- Even residents in rural countries in the U.S. have access to such robots. This is just one example of how AI affects, the way we live.

Module 6

Scanned with CamScanner

- Based on the current trends, here are 10 ways we can expect robotics to transform our future.

6.9.1 Robotics in Public Security

- Artificial technology for predicting and detecting crime might appear for away from possibility, but it is quite possible for the future we are looking at.
- Drone footage, for instances, will make that happen soon. In addition 'automatic recognition of suspicious activities' is already a reality for camera based security systems.
- This technology is going to change society in a very important way: it will allow law enforcement officials to act quickly whenever a suspicious behaviour is spotted.

6.9.2 Robots in Education

- In classroom learning, there is a gap between individual learning settings and the teacher. An educational expert Kendra Roberts explains "A single teacher does not have the capacity to meet the needs of personalized learning for every single student in the classroom".
- Computer based learning is already changing things in that matter. It is not replacing the teacher, but it enables students to learn at their own pace.
- Robots are supposed to boost the process of personalized learning. The humanoid robot NAO is already forming bonds with students from around the world. It comes with important senses of natural interaction, including moving, listening, speaking and connecting.

6.9.3 Robots at Home

- These robots are called as 'Cloud connected home robots'.
- These robots are already becoming part of our livers. We can set up the vacuum cleaner to do the work for us, and we can also schedule a warm-home cooked meal to be ready by the time we are finished with work. Multi-function robotic cookers are able to fry, steam, bake, slow cook, and perform any other action without our intervention. We have to just set them up.
- These 'cloud-connected robots' are likely to evolve into more advanced version. We expect to see speech comprehension and increased interactions with humans in upcoming years. These developments may end up changing the entire look and feel of our homes.

6.9.4 Robots as Co-workers

- Robots will have a profound effect on the workplace of the future. They shall become capable of taking on multiple roles in an organization, so it is time for us to start thinking about the way we shall interact with our new coworkers.
- The machines will likely evolve more in terms of voice recognition, so we shall be communicating with them through voice commands.
- Futurist Nikolas Badminton expects to develop things : "You are probably going to walk into an office and your system has been wandering over the last couple of hours considering what is being going on in business , your role, your job, what you need to do that day and probably offer up several ideas about the right direction of what to do". It sounds like a pretty achievable future" !

6.9.5 Robots might take our Jobs

- Robots have already replaced many people in their jobs. The jobs in office administration, logistics and transport are also at risk of being replaced.

  In the coming future, we might see the robots in the form of big trucks on the roads.
- A study of Ball state university predicted that many occupations are at risk of being automated, including insurance underwriters, telemarketers, and tax-return preparers. To be more precise, robots are expected to take over half of all low-skilled jobs.

6.9.6 They Create Jobs, Too

- Technology is changing fast and it does have economic imbalances. Driverless cars, for instance, are highly likely to replace cab drivers in the future.
- Hence 'Robots will take our jobs!' is perhaps the most common fear surrounding robotics development.
- In the near future 'artificial intelligence' will most likely replace tasks, not jobs. The good news is that it will also create new markets and hence jobs. We might need additional education and retraining for those jobs, but the opportunities will be definitely there.

6.9.7 Autonomous Cars

- Self driving cars still require some human intervention, but we are getting closer to the day when they wait.
- Wayno, the company that aroce from the self-driving car project by Google, no longer has a monopoly on this industry. On the contrary, every significant automobile producer is pursuing this technology.

Scanned with CamScanner

- Uber is one of the strongest players in this field. The users of this service can now get matched with a 'self-driving uber' when they request the service, so they can get a glimpse of the future.
- As companies continue to invest in this trend, we shall live to see a different face of public transport by 2030.

6.9.8 Healthcare Robots

- With robots, we shall be having a different future for healthcare. Instead of visiting a primary care physician who will give us check-up with a simple stethoscope, we shall have intelligent robots performing these tasks. They will interact with patients, check on their conditions, and evaluate the need for further appointments.
- Pharmabotics will bring more huge changes. They shall be like ATM's for medicines, so that we can have the medications we require, while avoiding the inconvenience of talking to a stronger about our health issues.

6.9.9 Robotics for Entertainment

Robots are getting more personalized and engaging than ever. With the growth of this industry, virtual reality will enter our homes in the near future. We shall be able to interact with our home entertainment systems through conversations, and they will respond to our attempts to communicate.

6.9.10 Robots will Boost our Standard of Living

Throughout the past we have witnessed that automation and mechanization boosts the overall standard of living. We have seen it with Industrial Revolution, and it is going to happen again. According to estimates from United Nations, poverty has been reduced to a greater extent over the past five decades than in the previous 50. That is because the global economy grew sevenfold, and technology played a huge part in that progress.

With software automating all kinds of work, we are looking at a more comfortable future for ourselves.

6.10 FIELD PROGRAMMABLE GATE ARRAYS (FPGA) IN ROBOTICS

- FPGA is Field Programmable Gate Arrays as semiconductor devices that are based around a matrix of configurable logic blocks (CLBS) connected via programmable interconnects.
- FPGA can be reprogrammed to desired application or functionally requirements after manufacturing.
- Increasingly sophisticated tools are enabling embedded control system designers to more quickly create and more easily adapt FDGA-based applications.
- In manufacturing and automation contents, FPGAs are well-suited for use in robotics and machine-tool applications, as well as for fan, pump, compressor and conveyer control.
- FPGA are often used to configure I/O module functionality.

Remark

I/O module stands for Input/Output module, which is a device that acts as the connective bridge between a computer system at one end and an I/O or peripheral device of some kind at the other, such as printer, webcam or scanner.

6.10.1 Technology in FPGA

For commercial FPGAs, the main switch technologies are antifuses (e.g. Actel) and static RAM cells (e.g. Xilinx). For commercial CPLDs, the main switch technologies are Erasable Programmable ROM (EPROM) transistors and Electrically Erasable PROM (EEO-ROM) transistors.

6.10.2 Planning Algorithms

- Planning algorithms for teams of robots fall into two categories :
  (i) Centralized algorithms, in which a single computer makes decisions for the whole team, and
  (ii) Decentralized algorithms, in which each robot makes its own decisions based on local observations.

  With centralized algorithms, if the central computer goes offline, the whole system falls down.
- Decentralized algorithms handle erratic communications better, but they are harder to design, because each robot is essentially guessing what the others will do.
- Most research on decentralized algorithms has focused on making collective decision-making more reliable and has deferred the problem of avoiding obstacles in the robots environment.
- In simulations involving squadrons of minihelicopters, the decentralised algorithm came up with the same flight plans that a centralized version did.

Module 6

Scanned with CamScanner

- The drones generally preserved an approximation of their preferred formation, a square at a fixed altitude - although to accomodate obstacles the square rotated and the distances between the drones contracted,
- In a typical decentralized group planning algorithm, each robot might broadcast its observations of the environment to its teammates, and all the robots would then execute the same planning algorithm, on the basis of same information.
- The main idea is that each robot on the basis of its own observations, maps out an obstacle-free region in its immediate environment and passes that map to its nearest neighbors. When a robot receives a map from a neighbor, it calculates the intersection of that map with its own and passes that on.
- Each robot ends up with a map that reflects all of the obstacles detected by all the team members.

6.10.3 Four Dimensions

- The maps have not three dimensions, but four-the fourth being time. This is how the algorithm accounts for moving obstacles.
- The four dimensional map describes how a three dimensional map would have to change to accommodate the obstacle's change of location, over a span of few seconds. And it does it in a mathematically compact manner.
- Each robot updates its map several times a second a short enough span of time, assuming that the velocity of an accelerating object will not change suddenly.
- On the basis of its latest map, each robot calculates the trajectory that will maximize both its local goal staying information and its global goal.

6.10.4 Components of Robots

Some of the important components of Robots are as follows

1. **Manipulator :** Just like the human arm, the robot consists of what is called a manipulator having several joints and links.
2. **Endeffector :** The base of the manipulator is fixed to base support and at its other free end, the endeffector is attached. The Endeffector is expected to perform tasks by the palm and finger arrangement of the human arm.
3. **The Locomotion Device :** In case of human-beings the power for the movement of the arm, the palm and finger is provided by muscles. For the robot the power for the movement (locomotion) is provided by the motors. The motors used for providing locomotion in robots are of three types depending on the source of energy : Electric, Hydraulic or Pneumatic.
4. **The controller :** The digital computer (both the hardware and the software) acts as a controller to the robots. The controller functions in a manner analogous to the human brain. With the help of this controller, the robot is able to carry out the assigned task. The controller directs and controls the movements of the manipulator and the Endeffector. In short, the controller controls the robot.
5. **The sensors :** Without the data supplied by the sense organs, the brain would be incapable of intelligence. In other words the controller of the robot cannot do any meaningful task, if the robot is not with a component analogous to the sense organs, of the human body. Thus, the most important component of the robot is the set of sensors. Sensors are nothing but measuring instruments which measures quantities such as position, velocity, force, torque, proximity and temperature etc.

6.11 ROBOT HARDWARE

- A hardware robot is an 'appliance working or implementing mechanical energy and having different characters, each with a specific function and together doing a special task.
- It is all about managing physical methods. It includes managing mechanical devices and control systems to automate jobs in a technical manner.
- Hardware side of Robotics includes 'processor, buses, memory and peripherals like co-processors, sensors, robotic arm, controllers etc.

6.11.1 Software "Robots" V/S Hardware Robots in RPA

- The world of computer science and technology is full of software robots and hardware robots which are responsible for bringing automation.
- RDA is a kind of business process automation that enables anyone to describe a set of directions for a robot to give error-free solutions.
- Hardware robots are employed to automate some real tasks, such as in construction. But many kinds of software robots in RPA have nothing to do with hardware robots. Also many parts of robotics have nothing to do with automation.

Scanned with CamScanner

6.11.2 Difference between Software Robots and Hardware Robots

Sr. No.	Software Robots	Hardware Robots
1.	Software robots use a blend of automation, computer perception, and machine learning to automate monotonous, high volume jobs that are custom focused.	Hardware robot is a piece of machinery utilising or implementing automatic power and having different sections each with a specific function and currently. .
2	Trigger pressed there are many varieties of automation or software robots, varying from the completely standardized to the completely implicit, and extremely easy to the mind blowing complex'	Executing a special task Hardware robot is a part of engineering which includes various methods to create, develop, program and apply robotic machines
3.	This requires using software to complete tasks which people normally perform when they are utilizing computer programs.	It includes a system of tools that uses the actuator data to perform a particular purpose or output of actions and undertaking.
4.	RPA has nothing to do with hardware robots. It applies to "software robots" which are coded to apply for computer programs in the same style as a human operator would. They are simpler to blend into the current business methods.	These comprise computer sensors resularly that monitor the execution of these hardware robots, that can design change in these different engineering systems.
5.	It can be described as an expansion of RPA which utilizes AI to determine how humans complete tasks when applying for a computer program. This enables "software robots" to work more effectively.	In this, microprocessor operate by having installed the software developed into them. Without that software, a microprocessor control application will not run.
6.	Software robots can log into applications, transfer files applications, transfer files and folder, copy and paste data, edit forms, derive structured and semi-structured information from documents and more.	A hardware robot is an appliance working or implementing mechanical energy, and having different characters, each with a specific function and together doing a special task.
7.	It is all about managing physical methods. It includes managing mechanical devices and control systems to automate jobs in a technical manner.	Hardware robots are heterogeneous systems that consist of fundamental components, and devices, control parts, interfaces for beneficial applications, anything from auto-mobiles and airplanes.
8.	Software robots employ the user interface to catch data and manage applications just like people do.	Robot element companies still need a basic set of policies to support when creating the interfaces of their robot hardware devices.

6.12 APPLICATIONS OF A.I. IN HEALTH CARE

AI is transforming health-care. AI in Health Technology applications encourage healthier behaviour in individuals and help with the proactive management of a healthy lifestyle.

Additionally, AI increases the ability for healthcare professionals to better understand the day-to-day patterns and needs of the people they care, for better feedback, guidance and support.

6.12.1 Health Monitoring

- Tools to Support interventions and Healthy Behaviours monitors heart rate and activity levels. They can send alerts to the user to get more exercise and can share this information to doctors. Medical chatbots can offer relevant high quality information, reassurance, answers and ways of thinking about the situations related to human behaviour.
- Artificial Intelligence in Health AI is already being used to detect disease, such as Cancer, more accurately and in their early stages. Using AI for digital retinopathy screening will allow non-clinicians to be trained on retinal imaging, obtaining interpretation of the images within minutes and thus giving patients instant feedback.

Module 6

Scanned with CamScanner

6.12.2 Medical Imaging

- Machine learning algorithms can process unimaginable amounts of information in the blink of an eye and provide more precise than humans in spotting even the smallest detail in medical imaging.
- It can find every sign of potential conditions such as osteoporosis, breast cancer and many more with a 90% accuracy rate. As people approach 'end of life' they feel loneliness. AI have conversations and other social interactions with people to keep aging minds sharp.

6.12.3 AI Robot assisted Surgery

Robots have been used in medicine for more than 40 years. Surgical robots can either aid a human surgeon or execute operations by themselves. They are also used in hospitals and labs for repetitive tasks, in rehabilitation, physical therapy and in support of those with long-term conditions.

6.13 OTHER APPLICATIONS

6.13.1 Retail

- Artificial Intelligence (AI) is reinventing the retail landscape from using computer vision to customize promotion in real time to applying machine learning for 'inventory management' retailers can harness AI to connect with their customers and operate more efficiently.
- AI is enabling retail systems to work together to optimise customer experiences, forecasting, inventory management, and more. AI technologies like computer vision bring near-real-time intelligence to brick and mortar stores.
- That same data, when analysed in the cloud, can provide additional business insights.
- Digital transformation in retail is about more than connecting things. It is about converting data into insights, which inform actions that drive better business outcomes.
- AI in retail including machine learning and deep learning – are key to generating these insights. For retailers, that leads to incredible customer experiences, opportunities to grow revenues, fast innovations and smart operations all of which help differentiate one from the competitions.
- A special type of AI deep learning in retail known as computer vision is gaining traction at brick and mortar. Computer vision "sees" and interprets visual data, giving one eyes where one needs them.
- And it is opening the door for new retail use cases across customer experience demand forecasting, inventory management, and more.

6.13.2 AI Applications in Banking

Artificial Intelligence is the future of banking as it brings the power of advanced data analytics to combat fraudulent transactions and improve compliance.AI accomplishes anti-money laundering activities in few seconds, which otherwise takes hours and days.

AI also enables banks to manage huge volumes of data at record speed to derive valuable insights from it. Features such as digital payment advisers and biometric fraud detection mechanisms lead to a higher quality of services to a wider customer base. All this translates to increased revenue, reduced costs and boots in profits.

6.13.3 Prediction of Future Trends

- With its power to predict future scenarios by analyzing past behaviours, AI helps banks predict future outcomes and trends. This helps banks to identify fraud, detect anti-money laundering pattern and make customer recommendations.
- Money launderers, through a series of actions, portray that the source of their illegal money is legal. With its power of Machine Learning and cognition, AI identifies these hidden actions and helps save millions banks.
- Similarly, AI is able to detect suspicious data patterns among volumes of data to carry out fraud management. Further, with its key recommendation engines, AI studies past to predict future behaviour of data points, which helps banks to successfully up-sell and cross-sell.

6.13.4 Robotic Automation of Processes

- AI reviews and transforms processes by applying Robotic Process Automation (RPA).
- This enables automation of about 80% of repetitive work processes, allowing knowledge workers to dedicate their time in value - add operations that require high level of human intervention.
- By adapting AI, leaders in the banking sector have already taken actions with due diligence to reap the above mentioned benefits.

Chapter Ends...

Scanned with CamScanner