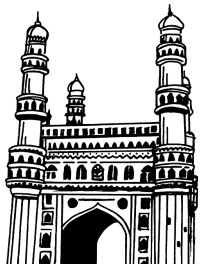
Rahul's ✔
Topper'sVoice

NEW SYLLABUS



B.C.A.

II Year IV Sem

Latest 2023 Edition

ARTIFICIAL INTELLIGENCE

- ☞ **Study Manual**
- ☞ **Important Questions**
- ☞ **Short Question & Answers**
- ☞ **Choose the Correct Answers**
- ☞ **Fill in the blanks**
- ☞ **Solved Model Papers**
- ☞ **Previous Question Paper**

- by -

WELL EXPERIENCED LECTURER

.199/-

Rahul Publications™

Hyderabad. Cell : 9391018098, 9505799122

B.C.A.

II Year IV Sem

ARTIFICIAL INTELLIGENCE

Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.

Price ` 199

Sole Distributors : **Cell : 9391018098, 9505799122**

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.

Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

ARTIFICIAL INTELLIGENCE

CONTENTS

STUDY MANUAL

SOLVED MODEL PAPERS

PREVIOUS QUESTION PAPER

SYLLABUS

UNIT - I

Introduction & Problem Solving: AI problems, AI Technique, Defining problem as a State Space Search, Production Systems, Problem Characteristics, Production System Characteristics. Heuristic Search Techniques: Generate – and – test, Hill Climbing, Best – First Search, Problem Reduction, Constraint Satisfaction, Means-ends Analysis.

UNIT - II

Game Playing: Overview, Min-Max search Procedure, Adding Alpha-beta Cutoffs, Additional Refinements, Iterative Deepening. Knowledge Representation Issues: Approaches, Issues, Frame Problem, Using Predicate Logic: Representing simple facts in logic, Representing Instance and ISA Relationships, Computable Functions and predicates, Resolution, Natural Deduction.

UNIT - III

Uncertainty and Reasoning Techniques:Non monotonic reasoning, Logics for Non monotonic reasoning, Implementation issues, Augmenting a problem solver, implementation of Depth First Search and Breadth first search. Statistical reasoning:Probability and Bayes theorem, Certainty factors and Rule-based systems, Bayesian Networks, Dempster-Shafer Theory.

UNIT - IV

Learning: What is Learning, Rote learning, Learning by taking advice, Learning in problem solving, learning from examples: Induction, Learning by Decision trees. Expert System: Representing and Using Domain Knowledge, Expert systems shells, Explanation, Knowledge Acquisition.

UNIT - V

Perception and Action: Real Time Search, Vision, Speech Recognition, ACTION: Navigation, Manipulation, Robot architectures. Natural Language Processing: Introduction, Syntactic Processing, Semantic Analysis, Statistical NLP, Spell Checking.

Contents

Topic **Page No.**

UNIT - IV

UNIT - V

Important Questions

UNIT - I

1. What is Artificial Intelligence? Write about various approaches used to define AI.

Ans :

Refer Unit-I, Q.No. 1

---

2. Write about problem solving techniques in AI.

Ans :

Refer Unit-I, Q.No. 4

---

3. Explain briefly about problem characteristics.

Ans :

Refer Unit-I, Q.No. 7

---

4. What is heuristic search? Explain about generate and test algorithm.

Ans :

Refer Unit-I, Q.No. 9

---

5. Explain Best - Frist Search algorithm.

Ans :

Refer Unit-I, Q.No. 12

---

6. Explain CSP with the example of SEND + MORE = MONEY.

Ans :

Refer Unit-I, Q.No. 15

---

7. Write about Planning - Goal Stack Algorithm.

Ans :

Refer Unit-I, Q.No. 18

UNIT - II

1. Write about the importance of game playing in AI.

Ans :

Refer Unit-II, Q.No. 1

---

2. Explain about the working of alpha beta pruning with example.

Ans :

Refer Unit-II, Q.No. 4

3. Explain Iterative Deepening Depth-First Search algorithm.

Ans :

Refer Unit-II, Q.No. 6

4. What is knowledge representation? What to represent as a knowledge.

Ans :

Refer Unit-II, Q.No. 8

5. Explain the relationship between knowledge and intelligence.

Ans :

Refer Unit-II, Q.No. 10

6. Explain the Problems Related to the Frame Problem.

Ans :

Refer Unit-II, Q.No. 13

7. What is Logic? Explain briefly about predicate logic in AI.

Ans :

Refer Unit-II, Q.No. 15

UNIT - III

1. What is monotonic and non monotonic reasoning? Explain.

Ans :

Refer Unit-III, Q.No. 1

2. What is augmenting a problem solver in AI.

Ans :

Refer Unit-III, Q.No. 4

3. What is the Breadth-First Search Algorithm?

Ans :

Refer Unit-III, Q.No. 5

4. What is uncertainty and state the causes of uncertainty.

Ans :

Refer Unit-III, Q.No. 7

5. What is certainty factor and explain about rule based systems.

Ans :

Refer Unit-III, Q.No. 10

6. Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Ans :

Refer Unit-III, Q.No. 12

7. Explain about Dempster- shafer theory.

Ans :

Refer Unit-III, Q.No. 13

UNIT - IV

1. What is learning? What are various learning aspects in AI ?

Ans :

Refer Unit-IV, Q.No. 1

2. What is rote learning? Explain about it.

Ans :

Refer Unit-IV, Q.No. 3

3. Explain how learning are used in problem solving?

Ans :

Refer Unit-IV, Q.No. 5

4. Explain about inductive learning algorithm.

Ans :

Refer Unit-IV, Q.No. 6

5. What is knowledge? Explain, how knowledge is used in expert system.

Ans :

Refer Unit-IV, Q.No. 8

6. Define expert system? Explain the components of an expert systems.

Ans :

Refer Unit-IV, Q.No. 9

7. Explain in detail about expert system usages.

Ans :

Refer Unit-IV, Q.No. 11

8. Explain briefly about knowledge acquisition.

Ans :

Refer Unit-IV, Q.No. 14

UNIT - V

1. Explain the concept of real time search?

Ans :

Refer Unit-V, Q.No. 1

2. Discribe challenges in working with speech recognition AI.

Ans :

Refer Unit-V, Q.No. 7

3. Explain brifly about various speech recognition techniques.

Ans :

Refer Unit-V, Q.No. 8

4. What is navigation? Explain the applications of navigation in AI.

Ans :

Refer Unit-V, Q.No. 9

5. Explain about manipulation technique in AI.

Ans :

Refer Unit-V, Q.No. 11

6. What is a Robot? Explain, how artificial intelligence is used for it.

Ans :

Refer Unit-V, Q.No. 12

7. Explain the architectures of robot system.

Ans :

Refer Unit-V, Q.No. 14

8. Explain the components of NLP.

Ans :

Refer Unit-V, Q.No. 16

9. Explain about statistical NLP

Ans :

Refer Unit-V, Q.No. 25

UNIT I

Introduction & Problem Solving: AI problems, AI Technique, Defining problem as a State Space Search, Production Systems, Problem Characteristics, Production System Characteristics.

Heuristic Search Techniques: Generate – and – test, Hill Climbing, Best - First Search, Problem Reduction, Constraint Satisfaction, Means-ends Analysis.

1.1 Introduction and Problem Solving

1.1.1 AI Problems

Q1. What is Artificial Intelligence? Write about various approaches used to define AI.

Ans : **(Imp.)**

Meaning

Artificial Intelligence is the branch of computer science concerned with making computers behave like humans.

The definitions of AI according to some text books are categorized into four approaches and are summarized in the table below :

- **Systems that Think like Humans**

  "The exciting new effort to make computers think machines with minds, in the full and literal sense."

- **Systems that Think Rationally**

  The study of mental faculties through the use of computer models."

- **Systems that act like Humans**

  The art of creating machines that perform functions that require intelligence when performed by people."

- **Systems that act Rationally**

  "Computational intelligence is the study of the design of intelligent agents."

The four approaches in more detail are as follows :

(a) Acting Humanly : The Turing Test Approach

- Test proposed by Alan Turing in 1950
- The computer is asked questions by a human interrogator.

The computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or not. Programming a computer to pass ,the computer need to possess the following capabilities :

- Natural language processing to enable it to communicate successfully in English.
- Knowledge representation to store what it knows or hears
- Automated reasoning to use the stored information to answer questions and to draw new conclusions.
- Machine learning to adapt to new circumstances and to detect and extrapolate patterns
- To pass the complete Turing Test,the computer will need.

- Computer vision to perceive the objects, and
- Robotics to manipulate objects and move about.

(b) Thinking humanly : The Cognitive Modelling Approach

We need to get inside actual working of the human mind :

- through introspection – trying to capture our own thoughts as they go by;
- through psychological experiments

Allen Newell and Herbert Simon, who developed GPS , the "General Problem Solver" tried to trace the reasoning steps to traces of human subjects solving the same problems.

The interdisciplinary field of cognitive science brings together computer models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind.

(c) Thinking rationally : The "laws of thought approach"

The Greek philosopher Aristotle was one of the first to attempt to codify "right thinking" ,that is irrefuatable reasoning processes. His syllogism provided patterns for argument structures that always yielded correct conclusions when given correct premises for example, "Socrates is a man; all men are mortal; therefore Socrates is mortal.".

These laws of thought were supposed to govern the operation of the mind; their study initiated a field called logic.

(d) Acting rationally : The rational agent Approach

An agent is something that acts. Computer agents are not mere programs ,but they are expected to have the following attributes also : (a) operating under autonomous control, (b) perceiving their environment, (c) persisting over a prolonged time period, (e) adapting to change.

A rational agent is one that acts so as to achieve the best outcome.

Q2. Explain the applications of AI.

Ans :

AI has been dominant in various fields such as:

- **Gaming**

AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

- **Natural Language Processing**

It is possible to interact with the computer that understands natural language spoken by humans.

- **Expert Systems**

There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

- **Vision Systems**

These systems understand, interpret, and comprehend visual input on the computer. For example,

- A spying aeroplane takes photographs, which are used to figure out spatial information or map of the areas.
- Doctors use clinical expert system to diagnose the patient.

- Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

- **Speech Recognition**

Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

- **Handwriting Recognition**

The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

- **Intelligent Robots**

Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

1.1.2 AI Technique

Q3. What is AI Technique?

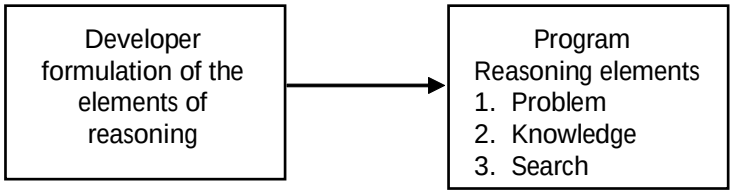
Ans :

AI problems are of many varieties and they appear to have very little in common. But there are techniques appropriate for the selection of variety of those problems, AI researches have show that "Intelligence requires knowledge", and knowledge itself posses some less desirable properties.

- It voluminous
- It is hard characterize accurately
- It is constantly changing
- It differs from data
- It is organized data

AI techniques EXPLOIT knowledge and for this knowledge must be represented as follows.

AI techniques must be designed keeping in mind the above constraints imposed by AI problems. AI techniques EXPLOIT knowledge and for this knowledge must be represented as follows.



1. Knowledge captures generalizations instead of representing individual situations separately , situation that share important properties grouped together . this avoids wastage of ,memory and unnecessary updation.
2. In many AI domains, most of the knowledge a program has, must be provided by people in terms of they understand.

3. It can be easily modified to correct errors.
4. It can be used in several situations even if it is not totally accurate or complete.
5. It can be used to help overcome its own sheer bulk, by helping to narrow the range of possibilities that must be considered.

AI techniques must be designed keeping in mind the above constraints imposed by AI problems.

1.1.3 Defining Problem As A State Space Search

Q4. Write about problem solving techniques in AI.

Ans : **(Imp.)**

The steps that are required to build a system to solve a particular problem are:

1. Problem Definition that must include precise specifications of what the initial situation will be as well as what final situations constitute acceptable solutions to the problem.
2. Problem Analysis , this can have immense impact on the appropriateness of varies possible techniques for solving the problem.
3. Selection of the best technique(s) for solving the particular problem.

Define the problem as a state Space Search

Consider the problem of "Playing Chess" .to build a program that could play chess, we have to specify the starting position of the chess board, the rules that define legal moves. And the board position that represent a win. The goal of the winning the game, if possible, must be made explicit.

The starting position can be described by an 8 X 8 array square in which each element square (x,y), (x varying from 1to 8 & y varying from1 to 8) describes the board position of an appropriate chess coin, the goal is any board position in which the opponent does not have a legal move and his or her "king" is under attack. The legal moves provide the way of getting from initial state of final state.

The legal moves can be described as a set of rules consisting of two parts: A left side that gives the current position and the right side that describes the change to be made to the board position. An example is shown in the following figure.

Current Position	Changing Board Position
While pawn at square (5 , 2) AND Square (5, 3)is empty AND Square (5, 4) is empty.	Move pawn from Square (5 , 2) to Square (5 , 4)

The current position of a coin on the board is its STATE and the set of all possible STATES is STATE SPACE. One or more states where the problem terminates is FINAL STATE or GOAL STATE . The state space representation forms the basis of most of the AI methods. It allows for a formal definition of the problem as the need to convert some given situation into some desired situation using a set of permissible operations. It permits the problem to be solved with the help of known techniques and control strategies to move through the problem space until goal state is found.

1.1.4 Production Systems

Q5. What is Production System? State is components.

Ans :

Meaning

Production system or production rule system is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behaviour but it also includes the mechanism necessary to follow those rules as the system responds to states of the world.

Components

The major components of Production System in Artificial Intelligence are:

1. Global Database

The global database is the central data structure used by the production system in Artificial Intelligence.

2. Set of Production Rules

The production rules operate on the global database. Each rule usually has a precondition that is either satisfied or not by the global database. If the precondition is satisfied, the rule is usually be applied. The application of the rule changes the database.

3. A Control System

The control system then chooses which applicable rule should be applied and ceases computation when a termination condition on the database is satisfied. If multiple rules are to fire at the same time, the control system resolves the conflicts.

Control/Search Strategies

How would you decide which rule to apply while searching for a solution for any problem? There are certain requirements for a good control strategy that you need to keep in mind, such as:

- The first requirement for a good control strategy is that it should cause motion.
- The second requirement for a good control strategy is that it should be systematic.
- Finally, it must be efficient in order to find a good answer.

Production System Rules

Production System rules can be classified as:

- Deductive Inference Rules
- Abductive Inference Rules

You can represent the knowledge in a production system as a set of rules along with a control system and database. It can be written as:

If(Condition) Then (Condition)

The production rules are also known as condition-action, antecedent-consequent, pattern-action, situation-response, feedback-result pairs.

Classes of Production System in Artificial Intelligence

There are four major classes of Production System in Artificial Intelligence:

- **Monotonic Production System**

  It's a production system in which the application of a rule never prevents the later application of another rule, that could have also been applied at the time the first rule was selected.

- **Partially Commutative Production System**

  It's a type of production system in which the application of a sequence of rules transforms state X into state Y, then any permutation of those rules that is allowable also transforms state x into state Y. Theorem proving falls under the monotonic partially communicative system.

- **Non-Monotonic Production Systems**

  These are useful for solving ignorable problems. These systems are important from an implementation standpoint because they can be implemented without the ability to backtrack to previous states when it is discovered that an incorrect path was followed. This production system increases efficiency since it is not necessary to keep track of the changes made in the search process.

- **Commutative Systems**

  These are usually useful for problems in which changes occur but can be reversed and in which the order of operation is not critical.

  Production systems that are not usually not partially commutative are useful for many problems in which irreversible changes occur, such as chemical analysis. When dealing with such systems, the order in which operations are performed is very important and hence correct decisions must be made at the first attempt itself.

Q6. Explain the advantages production system in artificial intelligence.

Ans :

Some of the advantages of Production system in artificial intelligence are:

1. Provides excellent tools for structuring AI programs
2. The system is highly modular because individual rules can be added, removed or modified independently
3. Separation of knowledge and Control-Recognises Act Cycle
4. A natural mapping onto state-space research data or goal-driven
5. The system uses pattern directed control which is more flexible than algorithmic control
6. Provides opportunities for heuristic control of the search.
7. A good way to model the state-driven nature of intelligent machines.
8. Quite helpful in a real-time environment and applications.

Production System in Artificial Intelligence: Example

Problem Statement

We have two jugs of capacity 5l and 3l (liter), and a tap with an endless supply of water. The objective is to obtain 4 liters exactly in the 5-liter jug with the minimum steps possible.

Production System

- Fill the 5 liter jug from tap
- Empty the 5 liter jug
- Fill the 3 liter jug from tap
- Empty the 3 liter jug
- Then, empty the 3 liter jug to 5 liter
- Empty the 5 liter jug to 3 liter
- Pour water from 3 liters to 5 liter
- Pour water from 5 liters to 3 liters but do not empty

Solution :

1,8,4,6,1,8 or 3,5,3,7,2,5,3,5;

1.1.5 Problem Characteristics

Q7. Explain briefly about problem characteristics.

Ans : **(Imp.)**

Heuristic search is a very general method applicable to a large class of problem . 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 ?

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: $\int x2 + 3x + \sin 2x \cos 2x\, dx$

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.

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.

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 out come 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 can not 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 sales man 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 rues 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

7. Does the task requires interaction with the person.

The problems can again be categorized under two heads.

i) Solitary in which the computer will be given a problem description and will produce an answer, with no intermediate communication and with he 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 & laws to computer)

ii) 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 can not 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.

1.1.6 Production System Characteristics

Q8. Explain about the characteristics of the production system.

Ans :

Here are some characteristics of production systems in AI:

1. Scalability

Production systems in AI should be designed to handle large volumes of data and workloads, and should be able to scale up or down to meet changing demands.

2. Fault-tolerance

Production systems should be resilient to errors and faults, and should be able to continue operating even if one or more components fail.

3. Real-time Processing

In many AI applications, data must be processed and analyzed in real-time to provide timely insights and recommendations. Production systems should be capable of processing data quickly and efficiently.

4. Low-latency

Many AI applications require low-latency processing to provide real-time responses to user requests. Production systems should be designed to minimize latency and provide fast response times.

5. Automation

Production systems should be highly automated, with minimal human intervention required to manage and operate the system. This helps to reduce the risk of errors and improve efficiency.

6. Security

Production systems should be designed with strong security features to protect against unauthorized access, data breaches, and other security threats.

7. Interoperability

Production systems should be designed to integrate easily with other systems and tools used in the AI ecosystem, to enable data sharing and facilitate collaboration.

8. Extensibility

Production systems should be designed to be easily extensible, so that new AI models and algorithms can be added as they become available.

9. Reproducibility

Production systems should be designed to ensure that results are reproducible, so that the same results can be obtained consistently over time and across different environments.

10. Monitoring and Maintenance

Production systems should include monitoring and maintenance tools to ensure that the system is running smoothly and to quickly identify and resolve any issues that arise.

1.2 Heuristic Search Techniques

1.2.1 Generate-and-Test

Q9. What is heuristic search? Explain about generate and test algorithm.

Ans : **(Imp.)**

Meaning

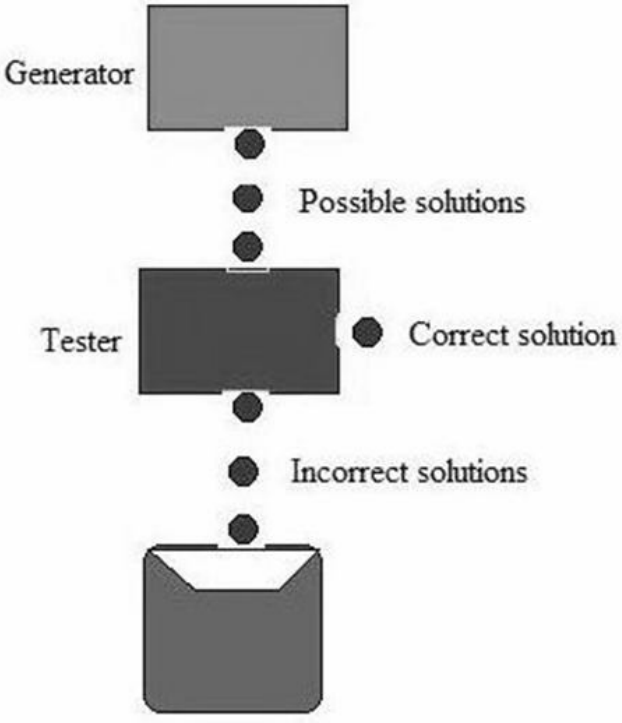
Heuristic search is an AI search technique that employs heuristic for its moves. Heuristic is a rule of thumb that probably leads to a solution. Heuristics play a major role in search strategies because of exponential nature of the most problems. Heuristics help to reduce the number of alternatives from an exponential number to a polynomial number. In Artificial Intelligence, heuristic search has a general meaning, and a more specialized technical meaning. In a general sense, the term heuristic is used for any advice that is often effective, but is not guaranteed to work in every case. Within the heuristic search architecture, however, the term heuristic usually refers to the special case of a heuristic evaluation function.

Generate and Test Search Algorithm

Generate-and-test search algorithm is a very simple algorithm that guarantees to find a solution if done systematically and there exists a solution.

Algorithm: Generate-And-Test

1. Generate a possible solution.
2. Test to see if this is the expected solution.
3. If the solution has been found quit else go to step



Potential solutions that need to be generated vary depending on the kinds of problems. For some problems the possible solutions may be particular points in the problem space and for some problems, paths from the start state.

Generate-and-test, like depth-first search, requires that complete solutions be generated for testing. In its most systematic form, it is only an exhaustive search of the problem space. Solutions can also be generated randomly but solution is not guaranteed. This approach is what is known as British Museum algorithm: finding an object in the British Museum by wandering randomly.

1.2.2 Hill Climbing

Q10. Explain briefly about Hill climbing algorithm.

Ans : **(Imp.)**

Hill climbing search algorithm is simply a loop that continuously moves in the direction of increasing value. It stops when it reaches a "peak" where no neighbour has higher value. This algorithm is considered to be one of the simplest procedures for implementing heuristic search. The hill climbing comes from that idea if you are trying to find the top of the hill and you go up direction from where ever you are. This heuristic combines the advantages of both depth first and breadth first searches into a single method.

The name hill climbing is derived from simulating the situation of a person climbing the hill. The person will try to move forward in the direction of at the top of the hill. His movement stops when it reaches at the peak of hill and no peak has higher value of heuristic function than this. Hill climbing uses knowledge about the local terrain, providing a very useful and effective heuristic for eliminating much of the unproductive search space. It is a branch by a local evaluation function.

The hill climbing is a variant of generate and test in which direction the search should proceed. At each point in the search path, a successor node that appears to reach for exploration.

Algorithm

Step 1

Evaluate the starting state. If it is a goal state then stop and return success.

Step 2

Else, continue with the starting state as considering it as a current state.

Continue step-4 until a solution is found i.e. until there are no new states left to be applied in the current state.

Step 4

a) Select a state that has not been yet applied to the current state and apply it to produce a new state.

b) Procedure to evaluate a new state.

i) If the current state is a goal state, then stop and return success.

ii) If it is better than the current state, then make it current state and proceed further.

iii) If it is not better than the current state, then continue in the loop until a solution is found.

Step 5: Exit.

Q11. State the advantages and disadvantages of Hill Climbing.

Ans :

Advantages

- Hill climbing technique is useful in job shop scheduling, automatic programming, designing, and vehicle routing and portfolio management.
- It is also helpful to solve pure optimization problems where the objective is to find the best according to the objective function.
- It requires much less conditions than other search techniques.

Disadvantages

- The question that remains on hill climbing search is whether this hill is the highest hill possible.
- Unfortunately without further extensive exploration, this question cannot be answered.
- This technique works but as it uses local information that's why it can be fooled.
- The algorithm doesn't maintain a search tree, so the current node data structure need only record the state and its objective function value.
- It assumes that local improvement will lead to global improvement.

1.2.3 Best – First Search

Q12. Explain Best - Frist Search algorithm.

Ans : **(Imp.)**

Best first search is an instance of graph search algorithm in which a node is selected for expansion based on evaluation function f(n). Traditionally, the node which is the lowest evaluation is selected for the explanation because the evaluation measures distance to the goal. Best first search can be implemented within general search frame work via a priority queue, a data structure that will maintain the fringe in ascending order off values. This search algorithm serves as combination of depth first and breadth first search algorithm. Best first search algorithm is often referred greedy algorithm this is because they quickly attack the most desirable path as soon as its heuristic weight becomes the most desirable.

Concept

Step 1: Traverse the root node

Step 2: Traverse any neighbour of the root node, that is maintaining a least distance from the root node and insert them in ascending order into the queue.

Step 3: Traverse any neighbour of neighbour of the root node, that is maintaining a least distance from the root node and insert them in ascending order into the queue

Step 4: This process will continue until we are getting the goal node

Algorithm:

Step 1: Place the starting node or root node into the queue.

Step 2: If the queue is empty, then stop and return failure.

Step 3: If the first element of the queue is our goal node, then stop and return success.

Step 4: Else, remove the first element from the queue. Expand it and compute the estimated goal distance for each child. Place the children in the queue in ascending order to the goal distance.

Step 5: Go to step-3

Step 6: Exit.

1.2.4 Problem Reduction

1.2.4.1 Problem Reduction with AO* Algorithm

Q13. Explain how problem reduction can be done by using AO* algorithm.

Ans : **(Imp.)**

When a problem can be divided into a set of sub problems, where each sub problem can be solved separately and a combination of these will be a solution, AND-OR graphs or AND - OR trees are used for representing the solution. The decomposition of the problem or problem reduction generates AND arcs. One AND are may point to any number of successor nodes. All these must be solved so that the arc will rise to many arcs, indicating several possible solutions. Hence the graph is known as AND - OR instead of AND. Figure shows an AND - OR graph.

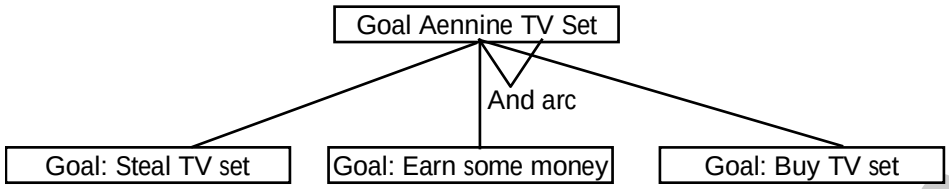
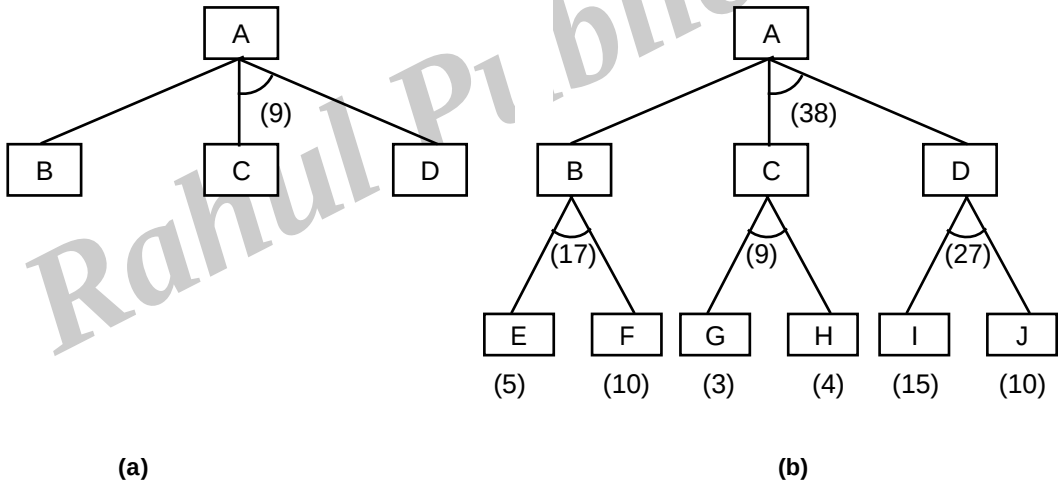


Fig.: AND-OR graph - an example

An algorithm to find a solution in an AND - OR graph must handle AND area appropriately. A* algorithm can not search AND - OR graphs efficiently. This can be understand from the give figure.



(a) **(b)**

Fig. : AND - OR graph

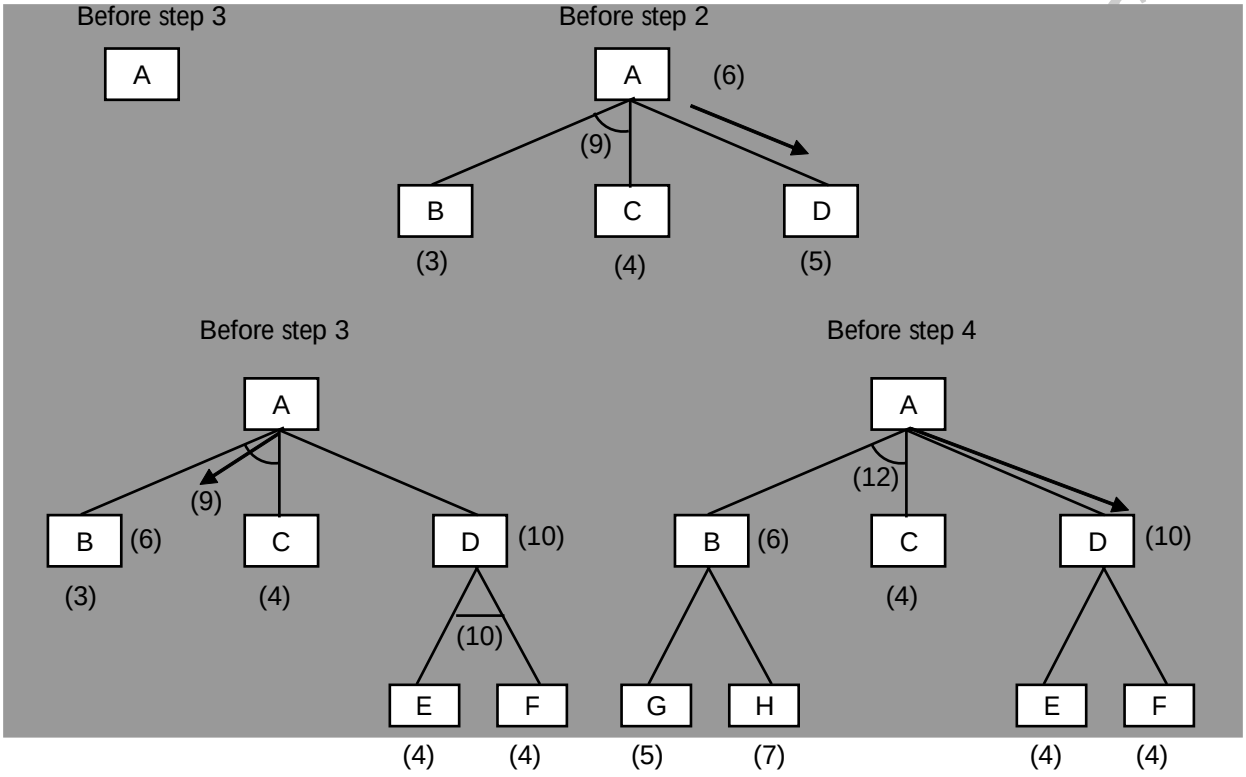
In figure (a) the top node A has been expanded producing two area one leading to B and leading to C-D .the numbers at each node represent the value of f ' at that node (cost of getting to the goal state from current state). For simplicity, it is assumed that every operation(i.e. applying a rule) has unit cost, i.e., each are with single successor will have a cost of 1 and each of its components. With the available information till now , it appears that C is the most promising node to expand since its f ' = 3 , the lowest but going through B would be better since to use C we must also use D' and the cost would be 9(3 + 4 + 1 + 1). Through B it would be 6(5 + 1).

Thus the choice of the next node to expand depends not only n a value but also on whether that node is part of the current best path form the initial mode. Figure (b) makes this clearer. In figure the node G appears to be the

most promising node, with the least f ' value. But G is not on the current beat path, since to use G we must use GH with a cost of 9 and again this demands that arcs be used (with a cost of 27). The path from A through B, E-F is better with a total cost of (17 + 1 = 18). Thus we can see that to search an AND-OR graph, the following three things must be done.

1. traverse the graph starting at the initial node and following the current best path, and accumulate the set of nodes that are on the path and have not yet been expanded.
2. Pick one of these unexpanded nodes and expand it. Add its successors to the graph and computer f ' (cost of the remaining distance) for each of them.
3. Change the f ' estimate of the newly expanded node to reflect the new information produced by its successors. Propagate this change backward through the graph. Decide which of the current best path.

The propagation of revised cost estimation backward is in the tree is not necessary in A* algorithm. This is because in AO* algorithm expanded nodes are re-examined so that the current best path can be selected. The working of AO* algorithm is illustrated in figure as follows:



Referring the figure. The initial node is expanded and D is Marked initially as promising node. D is expanded producing an AND arc E-F. f ' value of D is updated to 10. Going backwards we can see that the AND arc B-C is better .it is now marked as current best path. B and C have to be expanded next. This process continues until a solution is found or all paths have led to dead ends, indicating that there is no solution. An A* algorithm the path from one node to the other is always that of the lowest cost and it is independent of the paths through other nodes.

The algorithm for performing a heuristic search of an AND - OR graph is given below. Unlike A* algorithm which used two lists OPEN and CLOSED, the AO* algorithm uses a single structure G. G represents the part of the

search graph generated so far. Each node in G points down to its immediate successors and up to its immediate predecessors, and also has with it the value of h' cost of a path from itself to a set of solution nodes. The cost of getting from the start nodes to the current node "g" is not stored as in the A* algorithm. This is because it is not possible to compute a single such value since there may be many paths to the same state. In AO* algorithm serves as the estimate of goodness of a node. Also a there should value called FUTILITY is used. The estimated cost of a solution is greater than FUTILITY then the search is abandoned as too expansive to be practical.

For representing above graphs AO* algorithm is as follows

Q14. Explain the concept of AO* algorithm.

Ans :

1. Let G consists only to the node representing the initial state call this node INTT. Compute

   h'(INIT).

2. Until INIT is labeled SOLVED or hi (INIT) becomes greater than FUTILITY, repeat the following procedure.

   (I) Trace the marked arcs from INIT and select an unbounded node NODE.

   (II) Generate the successors of NODE .if there are no successors then assign FUTILITY as h'(NODE). This means that NODE is not solvable. If there are successors then for each one called SUCCESSOR, that is not also an ancester of NODE do the following

   (a) add SUCCESSOR to graph G

   (b) if successor is not a terminal node, mark it solved and assign zero to its h ' value.

   (c) If successor is not a terminal node, compute it h' value.

   (III) propagate the newly discovered information up the graph by doing the following . letS be a set of nodes that have been marked SOLVED. Initialize S to NODE. Until S is empty repeat the following procedure;

   (a) select a node from S call if CURRENT and remove it from S.

   (b) compute h' of each of the arcs emerging from CURRENT , Assign minimum h' to CURRENT.

   (c) Mark the minimum cost path a s the best out of CURRENT.

   (d) Mark CURRENT SOLVED if all of the nodes connected to it through the new marked are have been labeled SOLVED.

   (e) If CURRENT has been marked SOLVED or its h ' has just changed, its new status must be propagate backwards up the graph .hence all the ancestors of CURRENT are added to S.

AO* Search Procedure

1. Place the start node on open.
2. Using the search tree, compute the most promising solution tree TP .
3. Select node n that is both on open and a part of tp, remove n from open and place it no closed.
4. If n is a goal node, label n as solved. If the start node is solved, exit with success where tp is the solution tree, remove all nodes from open with a solved ancestor.
5. If n is not solvable node, label n as unsolvable. If the start node is labeled as unsolvable, exit with failure. Remove all nodes from open ,with unsolvable ancestors.
6. Otherwise, expand node n generating all of its successor compute the cost of for each newly generated node and place all such nodes on open.
7. Go back to step(2)

Note: AO* will always find minimum cost solution.

1.2.5 Constraint Satisfaction

Q15. What is constraint satisfaction problem? Explain it with an example.

(OR)

Explain CSP with the example of SEND + MORE = MONEY.

Ans : **(Imp.)**

A constraint satisfaction problem (CSP) consists of

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

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

A finite CSP has a finite set of variables and a finite domain for each variable. Many of the methods considered in this chapter only work for finite CSPs, although some are designed for infinite, even continuous, domains.

The multidimensional aspect of these problems, where each variable can be seen as a separate dimension, makes them difficult to solve but also provides structure that can be exploited.

Given a CSP, there are a number of tasks that can be performed:

- Determine whether or not there is a model.
- Find a model.
- Find all of the models or enumerate the models.
- Count the number of models.
- Find the best model, given a measure of how good models are;
- Determine whether some statement holds in all models.

Constraint Satisfaction Problem1

Many problems in AI can be considered as problems of constraint satisfaction, in which the goal state satisfies a given set of constraint. constraint satisfaction problems can be solved by using any of the search strategies. The general form of the constraint satisfaction procedure is as follows:

Until a complete solution is found or until all paths have led to lead ends, do

1. select an unexpanded node of the search graph.
2. Apply the constraint inference rules to the selected node to generate all possible new constraints.
3. If the set of constraints contains a contradiction, then report that this path is a dead end.
4. If the set of constraints describes a complete solution then report success.
5. If neither a constraint nor a complete solution has been found then apply the rules to generate new partial solutions. Insert these partial solutions into the search graph.

Example: consider the crypt arithmetic problems.

```
  SEND
+ MORE
-------
 MONEY
-------
```

Assign decimal digit to each of the letters in such a way that the answer to the problem is correct to the same letter occurs more than once , it must be assign the same digit each time . no two different letters may be assigned the same digit. Consider the crypt arithmetic problem.

```
  SEND
+ MORE
-------
MONEY
-------
```

Constraints

- no two digit can be assigned to same letter.
- only single digit number can be assign to a letter.

1. no two letters can be assigned same digit.
2. Assumption can be made at various levels such that they do not contradict each other.

3. The problem can be decomposed into secured constraints. A constraint satisfaction approach may be used.
4. Any of search techniques may be used.
5. Backtracking may be performed as applicable us applied search techniques.
6. Rule of arithmetic may be followed.

Initial State of Problem

D=?

E=?

Y=?

N=?

R=?

O=?

S=?

M=?

C1=?

C2=?

C1 ,C 2, C3 stands for the carry variables respectively.

Goal State

The digits to the letters must be assigned in such a manner so that the sum is satisfied.

Solution Process

We are following the depth-first method to solve the problem.

1. initial guess m=1 because the sum of two single digits can generate at most a carry '1'.
2. When n=1 o=0 or 1 because the largest single digit number added to m=1 can generate the sum of either 0 or 1 depend on the carry received from the carry sum. By this we conclude that o=0 because m is already 1 hence we cannot assign same digit another letter(rule no.)
3. We have m=1 and o=0 to get o=0 we have s=8 or 9, again depending on the carry received from the earlier sum.

The same process can be repeated further. The problem has to be composed into various constraints.

Solution :

```
    9 5 6 7
+ 1 0 8 5
---------
1 0 6 5 2
---------
```

VALUES

S=9

E=5

N=6

D=7

M=1

O=0

R=8

Y=2

Q16. Explain, how to solve Constraint Satisfaction Problems using Search.

Ans :

Generate-and-test algorithms assign values to all variables before checking the constraints. Because individual constraints only involve a subset of the variables, some constraints can be tested before all of the variables have been assigned values. If a partial assignment is inconsistent with a constraint, any complete assignment that extends the partial assignment will also be inconsistent.

Example

In the delivery scheduling problem the assignments A=1 and B=1 are inconsistent with the constraint A"B regardless of the values of the other variables. If the variables A and B are assigned values first, this inconsistency can be discovered before any values are assigned to C, D, or E, thus saving a large amount of work.

An alternative to generate-and-test algorithms is to construct a search space from which the search strategies of the previous chapter can be used. The search problem can be defined as follows:

- The nodes are assignments of values to some subset of the variables.
- The neighbors of a node N are obtained by selecting a variable V that is not assigned in node N and by having a neighbor for each assignment of a value to V that does not violate any constraint.

Suppose that node N represents the assignment $X_1=v_1,...,X_k=v_k$. To find the neighbors of N, select a variable Y that is not in the set $\{X_1,...,X_k\}$. For each value y_i"dom(Y), such that $X_1=v_1,...,X_k=v_k,Y=y_i$ is consistent with the constraints, $X_1=v_1,...,X_k=v_k,Y=y_i$ is a neighbor of N.

- The start node is the empty assignment that does not assign a value to any variables.
- A goal node is a node that assigns a value to every variable. Note that this only exists if the assignment is consistent with the constraints.

In this case, it is not the path that is of interest, but the goal nodes.

Example

Suppose you have a CSP with the variables A, B, and C, each with domain {1,2,3,4}. Suppose the constraints are A<B and B<C. A possible search tree is shown in Figure.

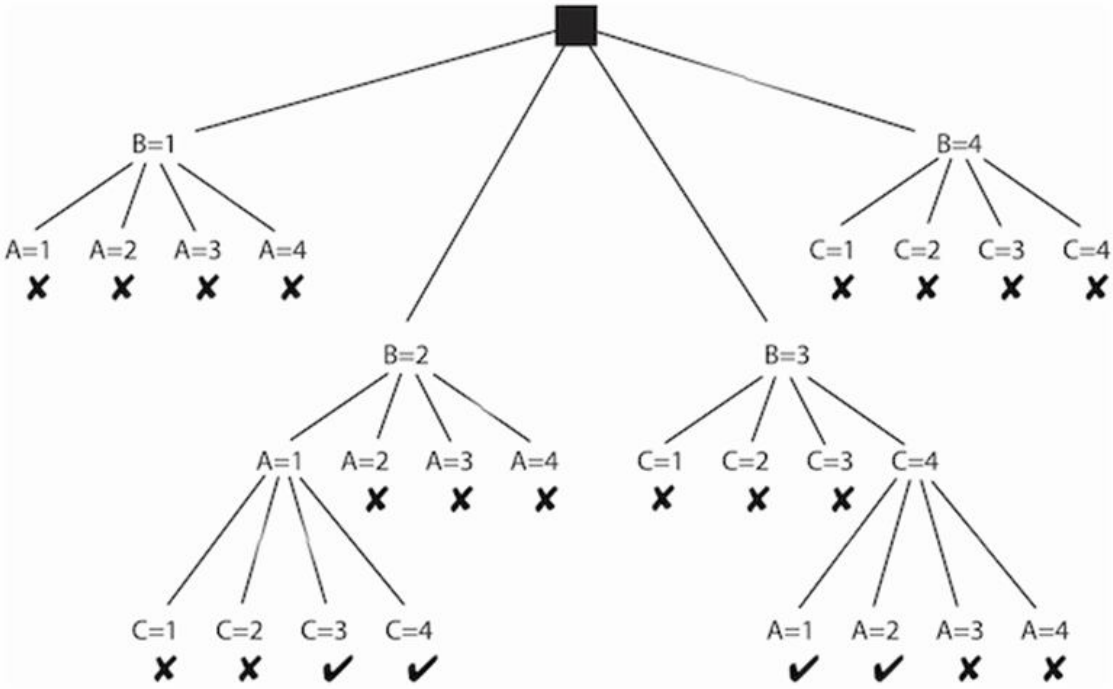


Figure: Search tree for the CSP of Example

In this figure, a node corresponds to all of the assignments from the root to that node. The potential nodes that are pruned because they violate constraints are labeled with ''. The leftmost × corresponds to the assignment A=1, B=1. This violates the A<B constraint, and so it is pruned.

This CSP has four solutions. The leftmost one is A=1, B=2, C=3. The size of the search tree, and thus the efficiency of the algorithm, depends on which variable is selected at each time. A static ordering, such as always splitting on A then B then C, is less efficient than the dynamic ordering used here. The set of answers is the same regardless of the variable ordering.

In the preceding example, there would be $4^3=64$ assignments tested in a generate-and-test algorithm. For the search method, there are 22 assignments generated.

Searching with a depth-first search, typically called backtracking, can be much more efficient than generate and test. Generate and test is equivalent to not checking constraints until reaching the leaves. Checking constraints higher in the tree can prune large subtrees that do not have to be searched.

1.2.6 Means-ends Analysis

Q17. Explain the concept of Means-Ends Analysis.

Ans : **(Imp.)**

Most of the search strategies either reason forward of backward however, often a mixture of the two directions is appropriate. Such mixed strategy would make it possible to solve the major parts of problem first and solve the smaller problems the arise when combining them together. Such a technique is called "Means - Ends Analysis".

The means -ends analysis process centers around finding the difference between current state and goal state. The problem space of means - ends analysis has an initial state and one or more goal state, a set of operate with a set of preconditions their application and difference functions that computes the difference between two state a(i) and s(j). A problem is solved using means - ends analysis by

1. Computing the current state s1 to a goal state s2 and computing their difference D12.
2. Satisfy the preconditions for some recommended operator op is selected, then to reduce the difference D12.
3. The operator OP is applied if possible. If not the current state is solved a goal is created and means- ends analysis is applied recursively to reduce the sub goal.
4. If the sub goal is solved state is restored and work resumed on the original problem.

(the first AI program to use means - ends analysis was the GPS General problem solver) means- ends analysis I useful for many human planning activities. Consider the example of planing for an office worker. Suppose we have a different table of three rules:

1. If in out current state we are hungry , and in our goal state we are not hungry , then either the "visit hotel" or "visit Canteen " operator is recommended.
2. If our current state we do not have money , and if in your goal state we have money, then the "Visit our bank" operator or the "Visit secretary" operator is recommended.
3. If our current state we do not know where something is , need in our goal state we do know, then either the "visit office enquiry" , "visit secretary" or "visit co worker " operator is recommended.

Q18. Write about Planning-Goal Stack Algorithm.

Ans : **(Imp.)**

Planning-Goal Stack Algorithm

One of the earliest techniques is planning using goal stack. Problem solver uses single stack that contains

- sub goals and operators both
- sub goals are solved linearly and then finally the conjoined sub goal is solved.

Plans generated by this method will contain complete sequence of operations for solving one goal followed by complete sequence of operations for the next etc.

Problem solver also relies on

- A database that describes the current situation.
- Set of operators with precondition, add and delete lists.

Let us assume that the goal to be satisfied is:

GOAL = G1 ⊥ G2 ⊥ … ⊥ Gn

Sub-goals G1, G2, …Gn are stacked with compound goal G1 ∧ G2 ∧ … ∧ Gn at the bottom.

Top G1
G2
:
Gn

Bottom G1 ⊥ G2 ⊥ … ⊥ G4

At each step of problem solving process, the top goal on the stack is pursued.

Algorithm

Find an operator that satisfies sub goal G1 (makes it true) and replace G1 by the operator.

- If more than one operator satisfies the sub goal then apply some heuristic to choose one.

In order to execute the top most operation, its preconditions are added onto the stack.

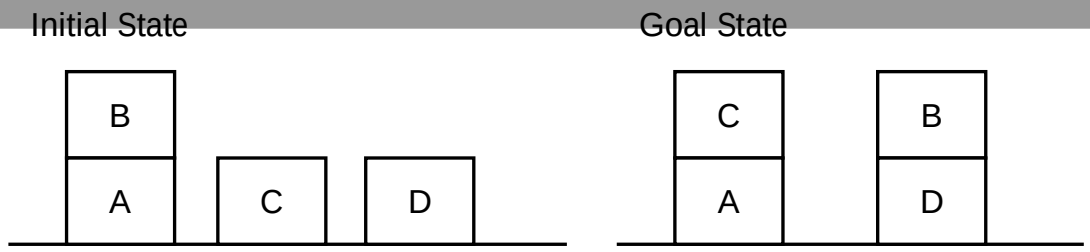
- Once preconditions of an operator are satisfied, then we are guaranteed that operator can be applied to produce a new state.
- New state is obtained by using ADD and DELETE lists of an operator to the existing database.

Problem solver keeps tract of operators applied.

- This process is continued till the goal stack is empty and problem solver returns the plan of the problem.

Goal Stack Example

With this example,let us explain the working method of Goal Stack Algorithm.



Initial State: ON(B, A) ∧ ONT(C) ⊥ ONT(A) ⊥ ONT(D) ⊥ CL(B) ∧ CL(C) ∧ CL(D) ∧ AE

Goal State: ON(C, A) ⊥ ON(B, D) ⊥ ONT(A) ⊥ ONT(D) ⊥ CL(C) ⊥ CL(B) ⊥ AE

We notice that following sub-goals in goal state are also true in initial state.

ONT(A) ∧ ONT(D) ⊥ CL(C) ⊥ CL(B) ⊥ AE

Represent for the sake of simplicity - TSUBG

Only sub-goals ON(C, A) &ON(B, D) are to be satisfied and finally make sure that TSUBG remains true.

Either start solving first ON(C, A) or ON(B, D). Let us solve first ON(C, A).

Goal Stack:

ON(C, A)

ON(B, D)

ON(C,A) ⊥ ON(B,D) ⊥ TSUBG

- To solve ON(C, A), operation S(C, A) could only be applied.
- So replace ON(C, A) with S(C, A) in goal stack.

Goal Stack:

S (C, A)

ON(B, D)

ON(C, A) ⊥ ON(B, D) ⊥ TSUBG

S(C, A) can be applied if its preconditions are true. So add its preconditions on the stack.

Goal Stack:

HOLD(C) Preconditions of STACK

CL(A)

CL(A) ⊥ HOLD(C)

S (C, A) Operator

ON(B, D)

ON(C, A) ⊥ ON(B, D) ∧ TSUBG

To do the S(C,A) operation all preconditions should be true. In the given problem CL(A) is not true.So,to make the state true ,replace CL(A) by U(B,A) and write the preconditions of Unstack operator.

Goal Stack:

ON(B, A)

CL(B) Preconditions of UNSTACK

AE

ON(B, A) ∧ CL(B) ⊥ AE

US(B, A) Operator

HOLD(C) Preconditions of STACK

CL(A)) ∧ HOLD(C)

S (C, A) Operator

ON(B, D)

ON(C, A) ∧ ON(B, D) ∧ TSUBG

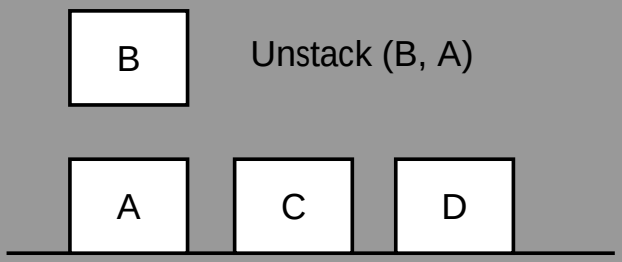
- ON(B, A), CL(B) and AE are all true in initial state, so pop these along with its compound goal.
- Next pop top operator US(B, A) and produce new state by using its ADD and DELETE lists.
- Add US(B, A) in a queue of sequence of operators.

SQUEUE = US (B, A)

State_1:

ONT(A) ⊥ ONT(C) ∧ ONT(D) ⊥ HOLD(B) ⊥ CL(A) ⊥ CL(C) ⊥ CL(D)

STATE 1



Goal Stack:

HOLD(C) Preconditions of STACK

CL(A)) ∧ HOLD(C)

S (C, A) Operator

ON(B, D)

ON(C, A) ∧ ON(B, D) ∧ TSUBG

To execute the S(C,A),all the preconditions of Stack operator should be true.But in this case HOLD(C) is not true .To make the state true use the operator S(B,D)

S(B,D) Operator

HOLD(C)

CL(A)) ∧ HOLD(C) Preconditions of STACK

S (C, A) Operator

ON(B, D)

ON(C, A) ∧ ON(B, D) ∧ TSUBG

Write down the preconditions of S(B,D)

Goal Stack

CL (D) ⊥ HOLD (B) Preconditions of STACK

S(B,D) Operator

HOLD(C)

CL(A)) ∧ HOLD(C) Preconditions of STACK

S(C, A) Operator

ON(B, D)

ON(C, A) ∧ ON(B, D) ∧ TSUBG

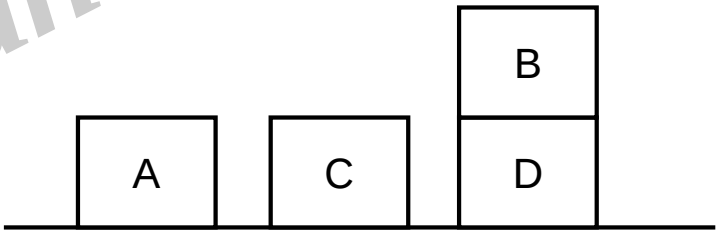
Add S(B, D) in a queue of sequence of operators.

SQUEUE = US (B, A), S (B, D)

State_2:

ONT(A) ⊥ ONT(C) ⊥ ONT(D) ⊥ ON(B, D) ⊥ CL(A) ⊥ CL(C) ⊥ CL(B) ⊥ AE

STATE 2



Goal Stack

HOLD(C)

CL(A)) ∧ HOLD(C) Preconditions of STACK

S (C, A) Operator

ON(B, D)

ON(C, A) ∧ ON(B, D) ∧ TSUBG

To execute S(C,A) all the preconditions should be true.here HOLD(C) is not true,to make the state true use the operator PU(C) and write the preconditions .

Goal Stack

ONT (C) ∧ CL (C) ∧ AE	Preconditions of PICKUP
PU (C)	Operator
HOLD(C)	
CL(A)) ∧ HOLD(C)	Preconditions of STACK
S (C, A)	Operator
ON(B, D)	
ON(C, A) ∧ ON(B, D) ∧ TSUBG	

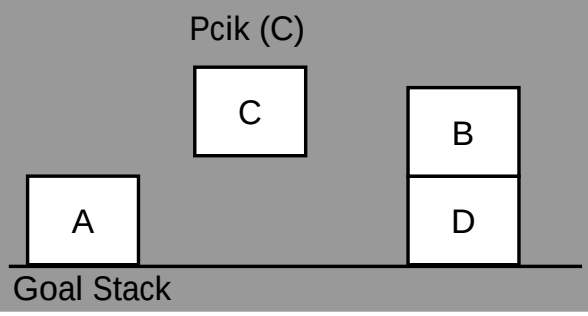
Here, all the preconditions of PU operator is true,so add PU(C) in a queue of sequence of operators.

SQUEUE = US (B, A), S (B, D), PU(C)

State_3:

ONT(A) ∧ HOLD(C) ∧ ONT(D) ∧ ON(B, D) ∧ CL(A) ∧ CL(B)

STATE 3



Goal Stack

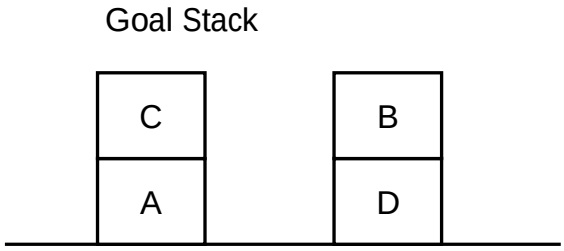
HOLD(C)	
CL(A)) ∧ HOLD(C)	Preconditions of STACK
S (C, A)	Operator
ON(B, D)	
ON(C, A) ∧ ON(B, D) ∧ TSUBG	

Here all the preconditions of S(C,A) is true ,so add S(C,A) in queue

SQUEUE = US (B, A), S (B, D),PU(C),S(C,A)

State_4:

$$ONT(A) \wedge ON(C, A) \wedge ONT(D) \wedge ON(B, D) \wedge CL(C) \wedge CL(B) \wedge AE$$



Finally ,we reached goal state after S(C,A) using Goal Stack algorithm,so the plan for the given problem is,

UnStack (B, A)

Stack (B, D)

PickUp(C)

Stack(C, A)

Short Question and Answers

1. What is Artificial Intelligence?

Ans :

Artificial Intelligence is the branch of computer science concerned with making computers behave like humans.

The definitions of AI according to some text books are categorized into four approaches and are summarized in the table below :

- **Systems that Think like Humans**

  "The exciting new effort to make computers think machines with minds, in the full and literal sense."

- **Systems that Think Rationally**

  The study of mental faculties through the use of computer models."

- **Systems that act like Humans**

  The art of creating machines that perform functions that require intelligence when performed by people."

- **Systems that act Rationally**

  "Computational intelligence is the study of the design of intelligent agents."

2. Applications of AI.

Ans :

AI has been dominant in various fields such as:

- **Gaming**

  AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

- **Natural Language Processing**

  It is possible to interact with the computer that understands natural language spoken by humans.

- **Expert Systems**

  There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

- **Vision Systems**

  These systems understand, interpret, and comprehend visual input on the computer. For example,

  - A spying aeroplane takes photographs, which are used to figure out spatial information or map of the areas.
  - Doctors use clinical expert system to diagnose the patient.
  - Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

- **Speech Recognition**

  Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

- **Handwriting Recognition**

  The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

3. Production System.

Ans :

Production system or production rule system is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behaviour but it also includes the mechanism necessary to follow those rules as the system responds to states of the world.

4. Characteristics of the production system.

Ans :

Here are some characteristics of production systems in AI:

i) **Scalability:** Production systems in AI should be designed to handle large volumes of data and workloads, and should be able to scale up or down to meet changing demands.

ii) **Fault-tolerance:** Production systems should be resilient to errors and faults, and should be able to continue operating even if one or more components fail.

iii) **Real-time Processing:** In many AI applications, data must be processed and analyzed in real-time to provide timely insights and recommendations. Production systems should be capable of processing data quickly and efficiently.

iv) **Low-latency:** Many AI applications require low-latency processing to provide real-time responses to user requests. Production systems should be designed to minimize latency and provide fast response times.

5. What is heuristic search?

Ans :

Heuristic search is an AI search technique that employs heuristic for its moves. Heuristic is a rule of thumb that probably leads to a solution. Heuristics play a major role in search strategies because of exponential nature of the most problems. Heuristics help to reduce the number of alternatives from an exponential number to a polynomial number. In Artificial Intelligence, heuristic search has a general meaning, and a more specialized technical meaning. In a general sense, the term heuristic is used for any advice that is often effective, but is not guaranteed to work in every case. Within the heuristic search architecture, however, the term heuristic usually refers to the special case of a heuristic evaluation function.

6. Hill climbing algorithm.

Ans :

Hill climbing search algorithm is simply a loop that continuously moves in the direction of increasing value. It stops when it reaches a "peak" where no neighbour has higher value. This algorithm is considered to be one of the simplest procedures for implementing heuristic search. The hill climbing comes from that idea if you are trying to find the top of the hill and you go up direction from where ever you are. This heuristic combines the advantages of both depth first and breadth first searches into a single method.

The name hill climbing is derived from simulating the situation of a person climbing the hill. The person will try to move forward in the direction of at the top of the hill. His movement stops when it reaches at the peak of hill and no peak has higher value of heuristic function than this. Hill climbing uses knowledge about the local terrain, providing a very useful and effective heuristic for eliminating much of the unproductive search space. It is a branch by a local evaluation function.

The hill climbing is a variant of generate and test in which direction the search should proceed. At each point in the search path, a successor node that appears to reach for exploration.

7. Best - Frist Search algorithm.

Ans :

Best first search is an instance of graph search algorithm in which a node is selected for expansion based on evaluation function f(n). Traditionally, the node which is the lowest evaluation is selected for the explanation because the evaluation measures distance to the goal. Best first search can be implemented within general search frame work via a priority queue, a data structure that will maintain the fringe in ascending order off values. This search algorithm serves as combination of depth first and breadth first search algorithm. Best first search algorithm is often referred greedy algorithm this is because they quickly attack the most desirable path as soon as its heuristic weight becomes the most desirable.

8. Constraint satisfaction

Ans :

A constraint satisfaction problem (CSP) consists of

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

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

A finite CSP has a finite set of variables and a finite domain for each variable. Many of the methods considered in this chapter only work for finite CSPs, although some are designed for infinite, even continuous, domains.

9. Means-Ends Analysis.

Ans :

Most of the search strategies either reason forward of backward however, often a mixture of the two directions is appropriate. Such mixed strategy would

make it possible to solve the major parts of problem first and solve the smaller problems the arise when combining them together. Such a technique is called "Means - Ends Analysis".

The means -ends analysis process centers around finding the difference between current state and goal state. The problem space of means - ends analysis has an initial state and one or more goal state, a set of operate with a set of preconditions their application and difference functions that computes the difference between two state a(i) and s(j). A problem is solved using means - ends analysis by.

10. Planning-Goal Stack Algorithm.

Ans :

One of the earliest techniques is planning using goal stack. Problem solver uses single stack that contains

- sub goals and operators both
- sub goals are solved linearly and then finally the conjoined sub goal is solved.

Plans generated by this method will contain complete sequence of operations for solving one goal followed by complete sequence of operations for the next etc.

Choose the Correct Answers

1. Which of the following is an application of Artificial Intelligence? [b]
   (a) It helps to exploit vulnerabilities to secure the firm
   (b) Language understanding and problem-solving (Text analytics and NLP)
   (c) Easy to create a website
   (d) It helps to deploy applications on the cloud
2. Which of the following is a component of Artificial Intelligence? [a]
   (a) Learning (b) Training
   (c) Designing (d) Puzzling
3. Among the following which is the component of the production ssytem [d]
   (a) A Control System (b) Global Database
   (c) Set of Production Rules (d) All the above
4. _____ number of informed search method are there in Artificial Intelligence. [a]
   (a) 4 (b) 3
   (c) 2 (d) 1
5. The available ways to solve a problem of state-space-search. [b]
   (a) 1 (b) 2
   (c) 3 (d) 4
6. Among the following which algorithm uses AND-OR graphs to find a solution to the problem [c]
   (a) Best first search algorithm
   (b) Hill climbing search algorithm
   (c) AO* algorithm
   (d) A* algorithm
7. Among the following which production system is useful for solving ignorable problems [b]
   (a) Monotonic Production System
   (b) Non-Monotonic Production System
   (c) Commutative Systems
   (d) Partially Commutative Production System
8. What is a rule in a production system? [c]
   (a) A statement of fact or piece of information in the knowledge base
   (b) An input value to the inference engine
   (c) A conditional statement that specifies a relationship between facts or information
   (d) An output value from the inference engine

9. Which of the following is not a heuristic search algorithm? [b]

(a) A* algorithm (b) Breadth-first search

(c) Depth-first search (d) Best-first search

10. Which technique is used to reduce the search space in a constraint satisfaction problem? [a]

(a) Forward checking (b) Arc consistency

(c) Backtracking search (d) Simulated annealing

Fill in the Blanks

1. _____ is the branch of computer science concerned with making computers behave like humans.
2. In _____ production system in which the application of a rule never prevents the later application of another rule.
3. _____ is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behaviour.
4. _____ search algorithm is simply a loop that continuously moves in the direction of increasing value. It stops when it reaches a "peak" where no neighbour has higher value.
5. _____ algorithm will always find minimum cost solution
6, _____ is a type of production system in which the application of a sequence of rules transforms state X into state Y
7. _____ allows for a formal definition of the problem as the need to convert some given situation into some desired situation using a set of permissible operations.
8. The algorithm which combines forward and backward is called _____
9. Which heuristic search algorithm guarantees finding the optimal solution.
10. A rule that restricts the possible values of one or more variables is called _____

ANSWERS

1. Artificial Intelligence
2. Monotonic Production System
3. Production system
4. Hill climbing
5. AO*
6. Partially Commutative Production System
7. State space search
8. Means –End analysis
9. A* algorithm
10. Constraint

UNIT II

Game Playing: Overview, Min-Max search Procedure, Adding Alpha-beta Cutoffs, Additional Refinements, Iterative Deepening. Knowledge Representation Issues: Approaches, Issues, Frame Problem, Using Predicate Logic: Representing simple facts in logic, Representing Instance and ISA Relationships, Computable Functions and predicates, Resolution, Natural Deduction.

2.1 GAME PLAYING

2.1.1 Overview

Q1. Write about the importance of game playing in AI.

Ans : **(Imp.)**

Game Playing is an important domain of artificial intelligence. Games don't require much knowledge; the only knowledge we need to provide is the rules, legal moves and the conditions of winning or losing the game.

Both players try to win the game. So, both of them try to make the best move possible at each turn. Searching techniques like BFS(Breadth First Search) are not accurate for this as the branching factor is very high, so searching will take a lot of time. So, we need another search procedures that improve –

- Generate procedure so that only good moves are generated.
- Test procedure so that the best move can be explored first.

Games are well-defined problems that are generally interpreted as requiring intelligence to play well.

- Introduces uncertainty since opponents moves can not be determined in advance.
- Search spaces can be very large.

For chess: -Branching factor: 35 -Depth: 50 moves each player -Search tree: 35100 nodes (~1040 legal positions)

- Despite this, human players do quite well without doing much explicit search. They seem to rely on remembering many patterns.
- Good test domain for search methods and development of pruning methods that ignore portions of the search tree that do not affect the outcome.

2.1.2 Min-Max Search Procedure

Q2. Explain about Min – Max Search algorithm.

Ans : **(Imp.)**

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.

1. Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
2. Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
3. The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
4. The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pseudo-code for MinMax Algorithm

Function minimax (node, depth, maximizing Player) is if depth = =0 or node is a terminal node then return static evaluation of node

```
if MaximizingPlayer then                    // for Maximizer Player
maxEva= -infinity
 for each child of node do
 eva= minimax(child, depth-1, false)
maxEva= max(maxEva,eva)                     //gives Maximum of the values
return maxEva
   else                                     // for Minimizer player
 minEva= +infinity
 for each child of node do
 eva= minimax(child, depth-1, true)
 minEva= min(minEva, eva)                   //gives minimum of the values
 return minEva
```

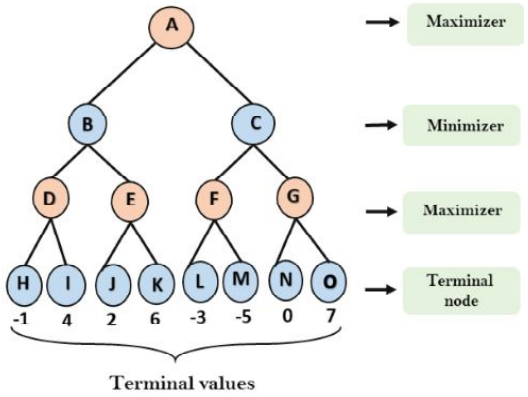
Initial call:

Minimax(node, 3, true)

Working of Min-Max Algorithm

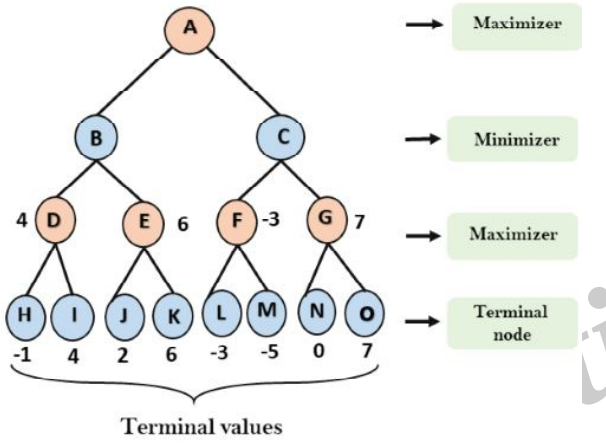
- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



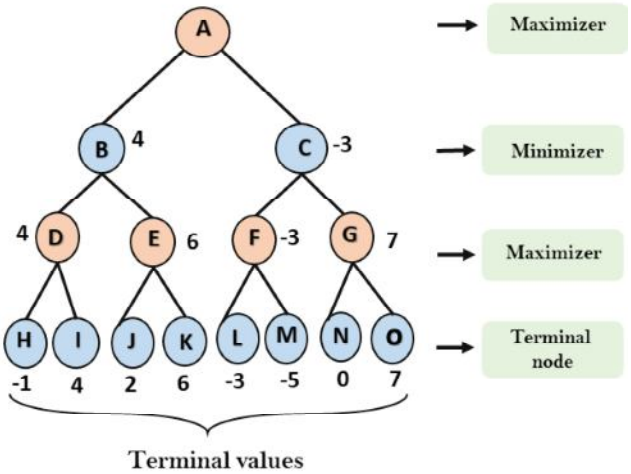
Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -", so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

- For node D max(-1,- ∞) = > max(-1,4)= 4
- For Node E max(2, - ∞) = > max(2, 6)= 6
- For Node F max(-3, - ∞) = > max(-3,-5) = -3
- For node G max(0, - ∞) = max(0, 7) = 7



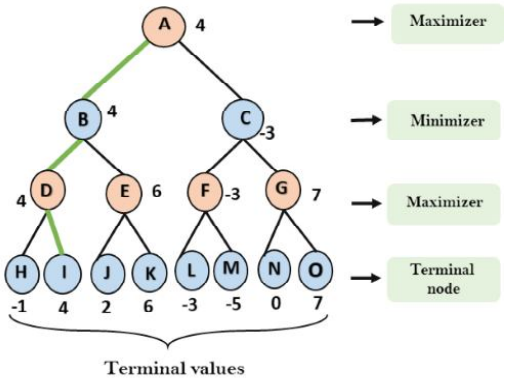
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +", and will find the 3rd layer node values.

- For node B = min(4,6) = 4
- For node C = min (-3, 7) = -3



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

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



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

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

Limitation of the minimax Algorithm

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

2.1.3 Adding Alpha-Beta Cutoffs

Q3. Explain about alpha beta pruning technique.

Ans :

1. Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
2. As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.
3. Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
4. The two-parameter can be defined as:
   a) Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -".
   b) Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +".

The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

The main condition which required for alpha-beta pruning is:

$\alpha >= \beta$

Pseudo-code for Alpha-beta Pruning:

```
function minimax(node, depth, alpha, beta, maximizingPlayer) is
if depth ==0 or node is a terminal node then
return static evaluation of node
if MaximizingPlayer then                    // for Maximizer Player
    maxEva= -infinity
    for each child of node do
    eva= minimax(child, depth-1, alpha, beta, False)
    maxEva= max(maxEva, eva)
    alpha= max(alpha, maxEva)
    if beta<=alpha
    break
    return maxEva
else                                        // for Minimizer player
    minEva= +infinity
    for each child of node do
    eva= minimax(child, depth-1, alpha, beta, true)
    minEva= min(minEva, eva)
    beta= min(beta, eva)
    if beta<=alpha
    break
return minEva
```

Q4. Explain about the working of alpha beta pruning with example.

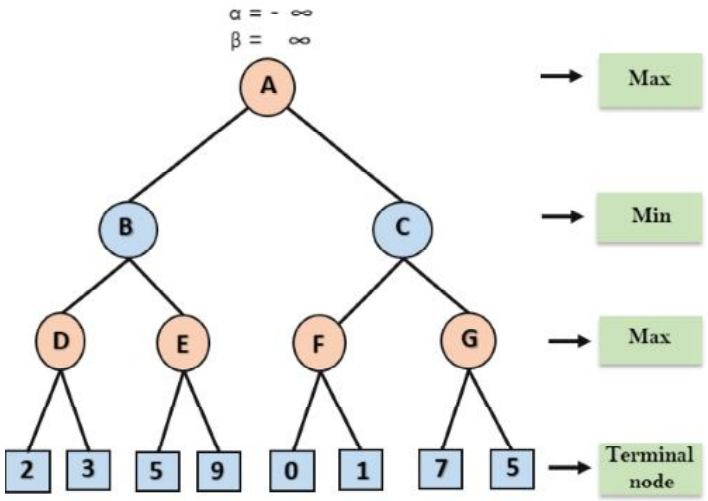
Ans : **(Imp.)**

Working of Alpha-Beta Pruning

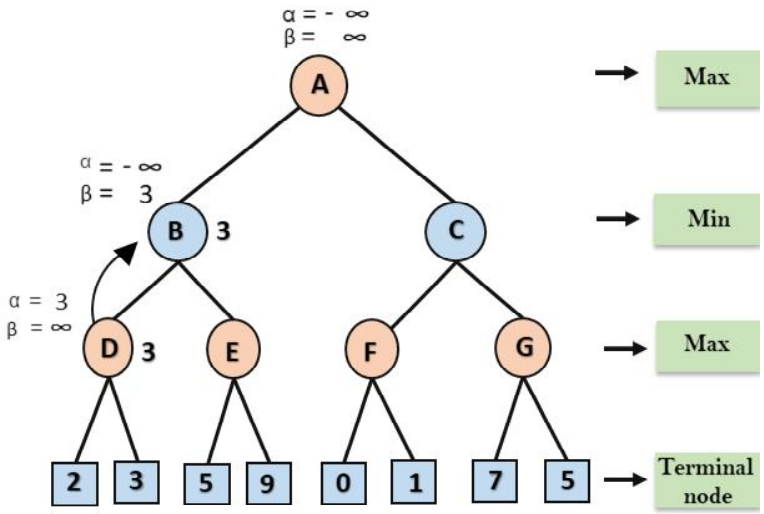
Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

Step 1: At the first step the, Max player will start first move from node A where $\alpha = -\infty$ and $\beta = +\infty$, these value of alpha and beta passed down to node B where again $\alpha = -\infty$ and $\beta = +\infty$, and Node B passes the same value to its child D.

Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.



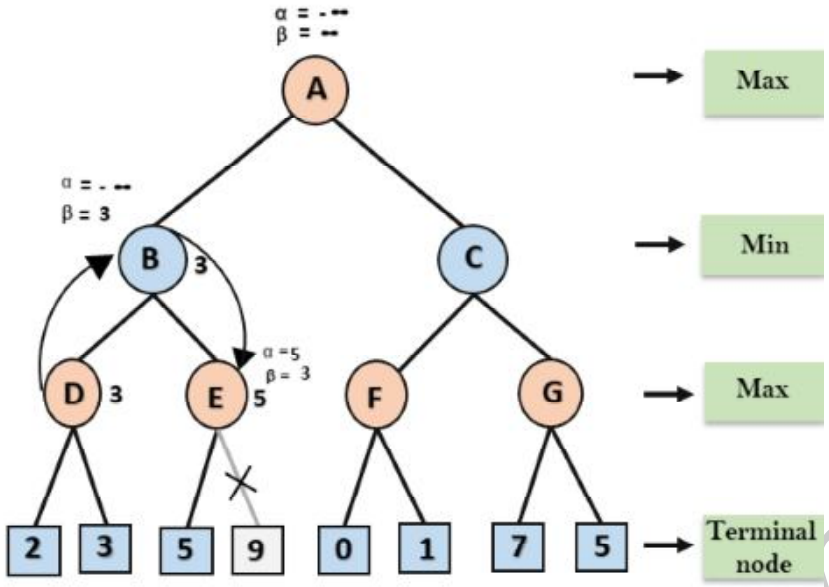
Step 3: Now algorithm backtrack to node B, where the value of â will change as this is a turn of Min, Now $\beta = +\infty$, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now $\alpha = -\infty$, and $\beta = 3$.



In the next step, algorithm traverse the next successor of Node B which is node E, and the values of $\alpha = -\infty$, and $\beta = 3$ will also be passed.

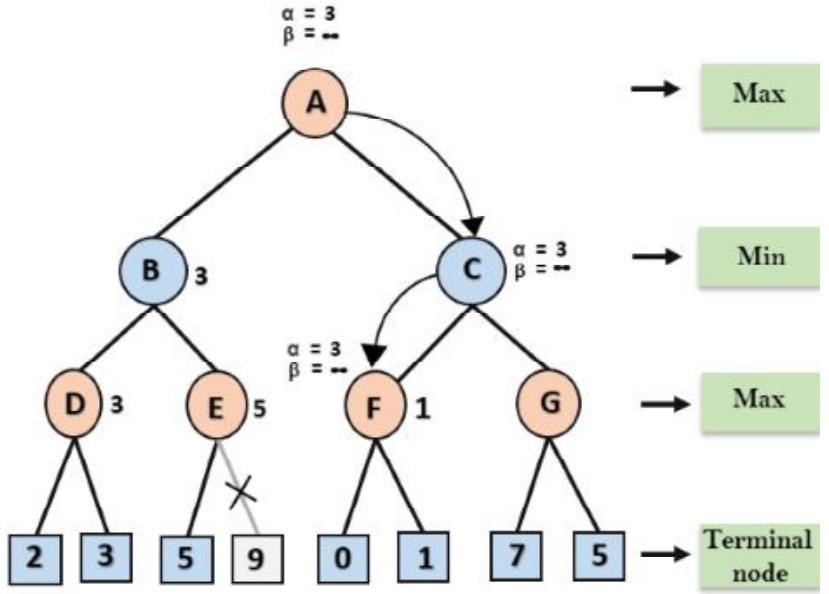
Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max ($-\infty$, 5) = 5, hence at node E $\alpha = 5$ and $\beta = 3$, where $\alpha >= \beta$, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.

Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max ($-\infty$, 3) = 3, and $\beta = +\infty$, these two values now passes to right successor of A which is Node C.

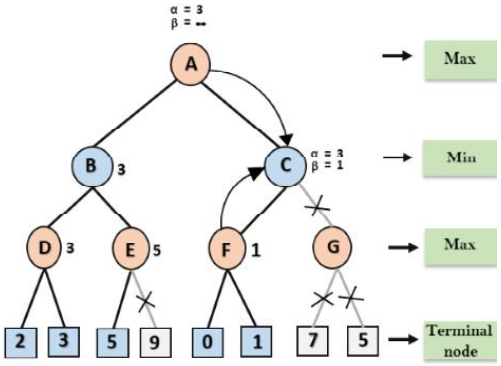


At node C, $\alpha = 3$ and $\beta = +\infty$, and the same values will be passed on to node F.

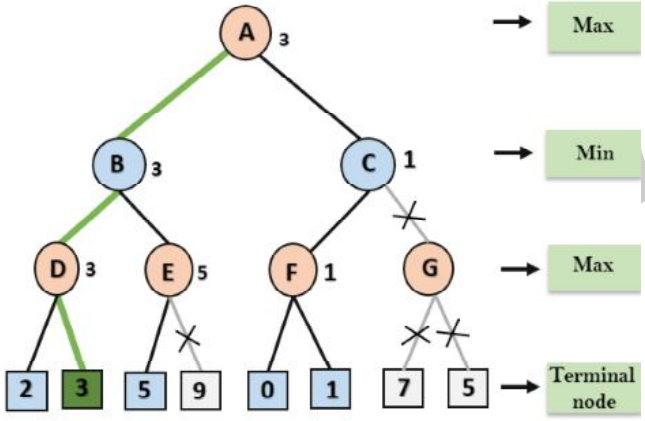
Step 6: At node F, again the value of α will be compared with left child which is 0, and max(3,0) = 3, and then compared with right child which is 1, and max(3,1) = 3 still á remains 3, but the node value of F will become 1.



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



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



2.1.4 Additional Refinements

Q5. Write about the additional refinements are used in AI.

Ans :

Here are some additional refinements in AI:

1. Transfer Learning

Transfer learning is a refinement in AI that involves training a model on one task and then using the knowledge gained to improve the performance on a different task. This can significantly reduce the amount of data and time required to train new models.

2. Reinforcement Learning

Reinforcement learning is a refinement in AI that involves training an agent to interact with an environment and learn from its actions and feedback. The agent receives rewards for positive behaviour and penalties for negative behaviour, which helps it learn to make better decisions.

3. Generative Adversarial Networks (GANs)

GANs are a refinement in AI that involves training two neural networks simultaneously: one to generate new data and another to evaluate the generated data for authenticity. This technique is often used to generate new images, videos, and other types of media.

4. Attention Mechanisms

Attention mechanisms are a refinement in AI that involve giving different levels of importance to different parts of input data. This technique is often used in natural language processing (NLP) to improve the accuracy of language models.

5. Convolutional Neural Networks (CNNs)

CNNs are a refinement in AI that are specifically designed for processing images and other types of spatial data. They use convolutional layers to extract features from the input data, and are widely used in applications such as computer vision and image recognition.

6. Long Short-Term Memory (LSTM) Networks

LSTMs are a refinement in AI that are specifically designed for processing sequential data, such as time series data or natural language text. They are able to remember long-term dependencies and are widely used in applications such as speech recognition and language translation.

7. Autoencoders

Autoencoders are a refinement in AI that are used for unsupervised learning. They are neural networks that learn to reconstruct input data from a compressed representation. They are often used for dimensionality reduction, feature extraction, and anomaly detection.

8. Federated Learning

Federated learning is a refinement in AI that involves training models on decentralized data sources without requiring the data to be centrally stored or processed. This technique is often used in applications where data privacy is a concern, such as in healthcare or financial services.

9. Hyperparameter Optimization

Hyperparameter optimization is a refinement in AI that involves searching for the optimal hyperparameters of a model, which are the parameters that control the learning process. This technique can significantly improve the performance of a model.

10. Explainable AI (XAI)

XAI is a refinement in AI that involves designing models that are transparent and explainable. This enables humans to understand how the model is making decisions and can increase trust in AI systems.

2.1.5 Iterative Deepening

Q6. Explain Iterative Deepening Depth-First Search algorithm.

Ans : **(Imp.)**

Iterative Deepening Depth-First Search (IDDFS) is a search algorithm that performs depth-first search (DFS) in a tree or graph with an increasing depth limit until the goal state is found. The working algorithm of IDDFS can be described as follows:

1. Initialize the depth limit to 0.
2. Repeat the following steps for increasing values of the depth limit until the goal state is found or there are no more nodes to explore: a. Perform a depth-first search on the tree or graph with the current depth limit. b. If the goal state is found, return the solution. c. If there are no more nodes to explore, return failure.
3. If the goal state is not found after exploring all nodes up to a certain depth limit, increase the depth limit and repeat steps 2a-2c.

The advantage of IDDFS over a traditional DFS is that it gradually increases the depth limit of the search, rather than using a fixed depth limit. This means that IDDFS can find the shortest path to the goal state in a tree or graph with an unknown depth.

The time complexity of IDDFS is O(b ^ d), where b is the branching factor of the search tree or graph, and d is the depth of the goal state. However, the space complexity of IDDFS is only O(d) since it only keeps track of the current path being explored.

Pseudo-code for IDDFS

```
1  IDDFS(T)
2  for d = 0 to infinity:
3  if (DLS(T, d)):
4  return 1
5  else
6  return 0
```

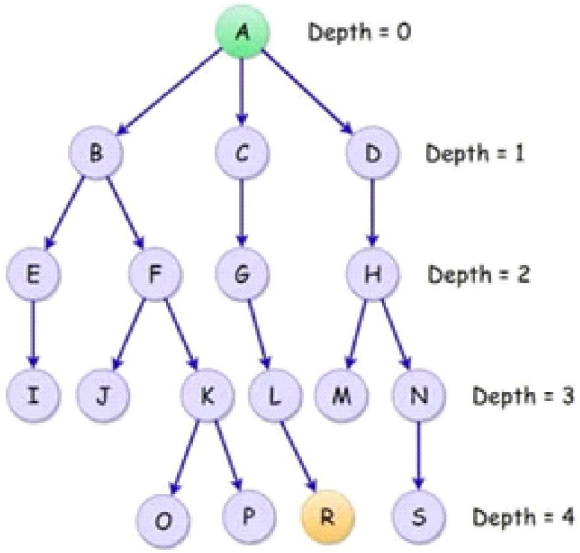
In order to implement the iterative deepening search we have to mark differences among:

1. Breakdown as the depth limit bound was attained.
2. A breakdown where depth bound was not attained.

While in the case once we try the search method multiple times by increasing the depth limit each time and in the second case even if we keep on searching multiple times since no solution exists then it means simply the waste of time. Thus we come to the conclusion that in the first case failure is found to be failing unnaturally, and in the second case, the failure is failing naturally.

Example of Iterative Deepening Depth-First Search

Let us take an example to understand this



Here in the given tree, the starting node is A and the depth initialized to 0. The goal node is R where we have to find the depth and the path to reach it. The depth from the figure is 4. In this example, we consider the tree as a finite tree, while we can consider the same procedure for the infinite tree as well. We knew that in the algorithm of IDDFS we first do DFS till a specified depth and then increase the depth at each loop. This special step forms the part of DLS or Depth Limited Search. Thus the following traversal shows the IDDFS search.

The tree can be visited as:ABEFCGDH

DEPTH = {0, 1,2, 3,4}

DEPTH LIMITS	IDDFS
0	A
1	ABCD
2	ABEFCGDH
3	ABEIFJKCGLDHMN
4	ABEIFJKOPCGLRDHMNS

This gives us a glimpse of the IDDFS search pattern.

Q7. State the advantages and disadvantages of Iterative Deepening Depth-First Search algorithm.

Ans :

Advantages

- IDDFS gives us the hope to find the solution if it exists in the tree.
- When the solutions are found at the lower depths say n, then the algorithm proves to be efficient and in time.

- The great advantage of IDDFS is found in-game tree searching where the IDDFS search operation tries to improve the depth definition, heuristics, and scores of searching nodes so as to enable efficiency in the search algorithm.
- Another major advantage of the IDDFS algorithm is its quick responsiveness. The early results indications are a plus point in this algorithm. This followed up with multiple refinements after the individual iteration is completed.
- Though the work is done here is more yet the performance of IDDFS is better than single BFS and DFS operating exclusively.
- Space and time complexities are expressed as: O(d) and here d is defined as goal depth.
- Let us consider the run time of IDDFS. Let say b > l where b is branching factor and l is the depth limit. Then next we search the goal node under the bound k. On the depth k, we say there may be b^k nodes that are only generated once. Similarly, the nodes at the depth limit k-1 is twice and thrice for k-2 depth. Thus the node generated at depth l is k times.

Disadvantages

- The time taken is exponential to reach the goal node.
- The main problem with IDDFS is the time and wasted calculations that take place at each depth.
- The situation is not as bad as we may think of especially when the branching factor is found to be high.
- The IDDFS might fail when the BFS fails. When we are to find multiple answers from the IDDFS, it gives back the success nodes and its path once even if it needs to be found again after multiple iterations. To stop the depth bound is not increased further.

2.2 KNOWLEDGE REPRESENTATION ISSUES

2.2.1 Approaches

Q8. What is knowledge representation? What to represent as a knowledge.

Ans : **(Imp.)**

Meaning

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning. Hence we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

What to Represent

Following are the kind of knowledge which needs to be represented in AI systems:

- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describebehavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).
- **Knowledge:** Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

Q9. Explain about different types of knowledge.

Ans :

Types of Knowledge

Following are the various types of knowledge:

1. Declarative Knowledge

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarativesentences.
- It is simpler than procedural language.

2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

3. Meta Knowledge

- Knowledge about the other types of knowledge is called Meta-knowledge.

4. Heuristic knowledge

- Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

5. Structural Knowledge

- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

Q10. Explain the relationship between knowledge and intelligence.

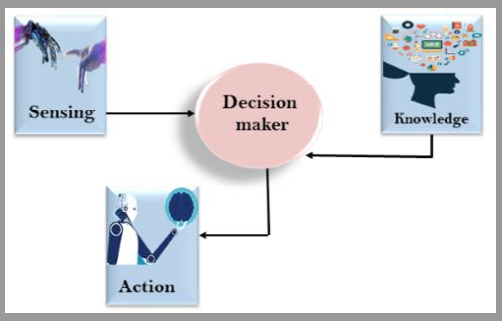
Ans : **(Imp.)**

The relation between knowledge and intelligence:

Knowledge of real-worlds plays a vital role in intelligence and same for creating artificial intelligence. Knowledge plays an important role in demonstrating intelligent behavior in AI agents. An agent is only able to accurately act on some input when he has some knowledge or experience about that input.

Let's suppose if you met some person who is speaking in a language which you don't know, then how you will able to act on that. The same thing applies to the intelligent behavior of the agents.

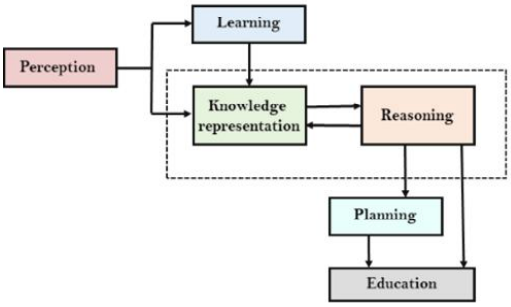
As we can see in below diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.



AI knowledge Cycle

An Artificial intelligence system has the following components for displaying intelligent behavior:

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



The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input. The learning component is responsible for learning from data captured by Perception comportment. In the complete cycle, the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of Knowledge representation and reasoning.

Approaches to Knowledge Representation

There are mainly four approaches to knowledge representation, which are givenbelow:

1. Simple Relational Knowledge

- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.

Example:

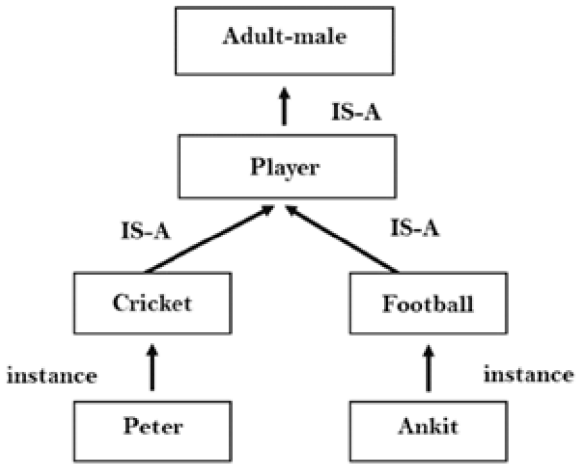
The following is the simple relational knowledge representation.

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

2. Inheritable Knowledge

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.
- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.

Example:



3. Inferential knowledge

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.

Example:

Let's suppose there are two statements:

a) Marcus is a man

b) All men are mortal

Then it can represent as;

man(Marcus)

"x = man (x) ----------> mortal (x)s

4. Procedural Knowledge

- Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.
- In this approach, one important rule is used which is If-Then rule.

- In this knowledge, we can use various coding languages such as LISP language and Prolog language.
- We can easily represent heuristic or domain-specific knowledge using this approach.
- But it is not necessary that we can represent all cases in this approach.

2.2.2 ISSUES

Q11. Explain about the issues in knowledge representation.

Ans :

- The fundamental goal of knowledge Representation is to facilitate inference (conclusions) from knowledge.
- The issues that arise while using KR techniques are many.

Some of these are explained below.

(i) Important Attributed

Any attribute of objects so basic that they occur in almost every problem domain?

There are two attributed "instance" and "isa", that are general significance. These attributes are important because they support property inheritance.

(ii) Relationship Among Attributes

Any important relationship that exists among object attributed?

The attributes we use to describe objects are themselves entities that we represent.

The relationship between the attributes of an object, independent of specific knowledge they encode, may hold properties like:

1. Inverse

This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways.

2. Existence in an is a Hierarchy

This is about generalization-specification, like, classes of objects and specialized subsets of those classes, there are attributes and specialization of attributes. For example, the attribute height is a specialization of general attribute physical-size which is, in turn, a specialization of physical-attribute. These generalization-specialization relationships are important for attributes because they support inheritance.

3. Technique for Reasoning about Values

This is about reasoning values of attributes not given explicitly. Several kinds of information are used in reasoning, like, height: must be in a unit of length, Age: of a person cannot be greater than the age of person's parents. The values are often specified when a knowledge base is created.

4. Single valued Attributes

This is about a specific attribute that is guaranteed to take a unique value. For example, a baseball player can at time have only a single height and be a member of only one team. KR systems take different approaches to provide support for single valued attributes.

(iii) Choosing Granularity

At what level of detail should the knowledge be represented?

Regardless of the KR formalism, it is necessary to know:

- At what level should the knowledge be represented and what are the primitives?
- Should there be a small number or should there be a large number of low-level primitives or High-level facts.
- High-level facts may not be adequate for inference while Low-level primitives may require a lot of storage.

Example of Granularity

- Suppose we are interested in following facts:

John spotted Sue.

This could be represented as

Spotted (agent(John),object (Sue))

Such a representation would make it easy to answer questions such are:

- Who spotted Sue?

Suppose we want to know:

- Did John see Sue?

Given only one fact, we cannot discover that answer.

We can add other facts, such as

Spotted(x, y) -> saw(x, y)

We can now infer the answer to the question.

(iv) Set of objects:

How should sets of objects be represented?

There are certain properties of objects that are true as member of a set but not as individual;

Example:

Consider the assertion made in the sentences: "there are more sheep than people in Australia", and "English speakers can be found all over the world."

To describe these facts, the only way is to attach assertion to the sets representing people, sheep, and English.

The reason to represent sets of objects is: if a property is true for all or most elements of a set, then it is more efficient to associate it once with the set rather than to associate it explicitly with every elements of the set.

This is done,

- in logical representation through the use of universal quantifier, and
- in hierarchical structure where node represent sets and inheritance propagate set level assertion down to individual.

(v) Finding Right Structure

Given a large amount of knowledge stored in a database, how can relevant parts are accessed when they are needed?

This is about access to right structure for describing a particular situation.

This requires, selecting an initial structure and then revising the choice.

While doing so, it is necessary to solve following problems:

- How to perform an initial selection of the most appropriate structure.
- How to fill in appropriate details from the current situations.
- How to find a better structure if the one chosen initially turns out not to be appropriate.
- What to do if none of the available structures is appropriate.
- When to create and remember a new structure.

There is no good, general purpose method for solving all these problems. Some knowledge representation techniques solve some of these issues.

2.2.3 Frame Problem

Q12. Define frame problem in AI.

Ans :

The frame problem is a well-known problem in artificial intelligence that arises when attempting to model changes in a system over time. It is named after the philosopher and cognitive scientist, Herbert A. Simon, who first introduced the concept in 1969.

The frame problem refers to the difficulty of determining which information is relevant when updating a knowledge representation after a change in the state of the world. In other words, it is the problem of how to represent and reason about the effects of actions on a dynamic environment while avoiding the explosion of irrelevant information.

For example, consider a robot tasked with opening a door. In order to perform this task, the robot needs to know which actions are necessary to achieve the goal, such as approaching the door, turning the doorknob, and pushing the door. However, it also needs to know what information is not relevant to the task, such as the color of the door or the position of a lamp in the room. If the robot were to consider all of the information in its environment, it would be overwhelmed with irrelevant information and unable to efficiently perform its task.

The frame problem has been a challenge for AI researchers because it requires a way to represent and reason about the effects of actions on the world in a way that is both comprehensive and computationally efficient. Several approaches have been proposed to address the frame problem, such as non-monotonic reasoning, situation calculus, and event calculus.

Q13. Explain the Problems Related to the Frame Problem.

Ans : **(Imp.)**

There are several problems related to the Frame Problem that make it difficult to solve:

1. The Relevance Problem

One of the main challenges of the Frame Problem is determining which information is relevant when updating a knowledge representation after a change in the state of the world. For example, if

a robot is tasked with opening a door, it needs to know which actions are necessary to achieve the goal, such as approaching the door, turning the doorknob, and pushing the door. However, it also needs to know what information is not relevant to the task, such as the color of the door or the position of a lamp in the room.

2. The Qualification Problem

Another problem related to the Frame Problem is how to represent knowledge about what remains unchanged after an action is performed. In other words, how can an agent distinguish between what has changed and what has remained the same after an action is performed? For example, if a robot moves a chair, it needs to know that the color and material of the chair remain unchanged, while its position has changed.

3. The Ramification Problem

The Ramification Problem refers to the issue of representing the indirect effects of actions. For example, if a robot moves a chair, it may indirectly affect other objects in the room, such as blocking a pathway or knocking over a lamp. These indirect effects can be difficult to represent and reason about, especially when they are complex or unpredictable.

4. The Frame Representation Problem

The Frame Representation Problem refers to the difficulty of representing knowledge about complex and abstract concepts. For example, representing knowledge about emotions or social norms can be challenging because these concepts are difficult to define and quantify.

Q14. Explain about various approaches to frame problem solution.

Ans :

Approaches to a Frame Problem Solution

There have been several approaches proposed to address the Frame Problem in artificial intelligence. Here are some of the most prominent ones:

1. Non-monotonic Reasoning

This approach involves using default rules or assumptions that can be overridden when new information becomes available. By using these default rules, an agent can avoid the need to constantly update its knowledge base in response to every new piece of information. Non-monotonic reasoning has been applied to a variety of domains, including robotics, natural language understanding, and knowledge-based systems.

2. Situation Calculus

Situation Calculus is a formal language for representing the effects of actions on the world. It provides a way to represent the knowledge an agent has about the state of the world before and after an action is performed. By using Situation Calculus, an agent can reason about the effects of an action without having to consider all possible changes to the environment.

3. Event Calculus

Event Calculus is a variant of Situation Calculus that focuses on representing events rather than states. It provides a way to represent the knowledge an agent has about the occurrence of events in the world and how they affect the world. By using Event Calculus, an agent can reason about the effects of actions and events without having to consider all possible changes to the environment.

4. Default Logic

Default Logic is a logical framework that allows for reasoning with incomplete information. It involves using default rules or assumptions to fill in missing information, and allows for a more efficient representation of knowledge by avoiding the need to represent every possible scenario.

5. Circumscription

Circumscription is a logical framework that allows for reasoning about exceptions to general rules. It involves specifying a set of exceptions to a general rule, and then using this knowledge to reason about specific cases. Circumscription has been applied to a variety of domains, including planning, diagnosis, and natural language understanding.

6. Abductive Reasoning

Abductive reasoning involves generating possible explanations for a given set of observations. By using this approach, an agent can reason about the effects of actions and events by generating possible explanations for the observed changes in the world.

2.3 USING PREDICATE LOGIC

2.3.1 Representing Simple Facts in Logic

Q15. What is Logic? Explain briefly about predicate logic in AI.

Ans : **(Imp.)**

Logic is a systematic way of representing and analyzing the structure of arguments and reasoning. It is concerned with understanding how reasoning works and how to evaluate the truth or validity of arguments.

Logic is a formal method for reasoning. Many concepts which can be verbalized can be translated into symbolic representations which closely approximate the meaning of these concepts. These symbolic structures can then be manipulated in programs to deduce various facts, to carry out a form of automated reasoning .in predicate logic statements from a natural language like English are translated into symbolic structures comprised of predicates, functions, variables, constants, quantifiers and logical connectives. The symbols form the basic building blocks for the knowledge, and their combination into valid structures is accomplished using the syntax of FOPL. Once structures have been created to represent basic facts, inference rules may then be applied to compare, combine and transform these "assumed" structures into new "deduced" structures. This is how automated reasoning or inferencing is performed.

Predicate logic is a type of formal logic that uses predicates and quantifiers to represent and reason about relationships between objects and concepts. It is a fundamental component of many AI systems, as it provides a way to express and reason about knowledge and information in a formal and precise way.

In predicate logic, a predicate is a statement that describes a relationship between one or more objects. For example, the predicate "Man(x)" describes the concept that "x is a man." Similarly, the predicate "Married(x,y)" describes the relationship between two objects, where "x is married to y."

Quantifiers are used to express statements that involve all or some objects. The symbol $\forall$ (for all) is used to indicate that a statement applies to all objects in a given domain, while the symbol $\exists$ (there exists) is used to indicate that there exists at least one object that satisfies a given statement.

For example, the statement "All men are mortal" can be expressed in predicate logic as $\forall$ x (Man(x) $\rightarrow$ Mortal(x)), where the predicate "Man(x)" indicates that x is a man, and the predicate "Mortal(x)" indicates that x is mortal.

Logical connectives, such as $\wedge$ (and), ($\vee$ (or), and $\neg$ (not), can be used to combine predicates and form more complex statements. For example, the statement "John is a man and Mary is a woman" can be expressed as Man(John) $\wedge$ Woman(Mary), where the $\wedge$ symbol indicates that both predicates must be true.

Predicate logic is used in various AI applications, including natural language processing, expert systems, and automated reasoning. It provides a way to represent and reason about complex relationships and knowledge in a formal and precise way, making it a powerful tool for building intelligent systems.

There are several standard logic symbols used in predicate logic, including:

1. **$\forall$ (for all)**

   This symbol is used to indicate that a statement holds for all values of a given variable. For example, $\forall$ x(P(x)) would mean "For all x, P(x) is true."

2. **$\exists$ (there exists)**

   This symbol is used to indicate that a statement holds for at least one value of a given variable. For example, $\exists$ x(P(x)) would mean "There exists an x such that P(x) is true."

3. **$\rightarrow$ (implies)**

   This symbol is used to indicate that one statement implies another. For example, P(x) $\rightarrow$ Q(x) would mean "If P(x) is true, then Q(x) is also true."

4. **$\wedge$ (and)**

   This symbol is used to indicate that two statements are both true.

   For example, P(x) $\wedge$ Q(x) would mean "Both P(x) and Q(x) are true."

5. **$\vee$ (or)**

   This symbol is used to indicate that at least one of two statements is true. For example, P(x) ($\vee$ Q(x) would mean "Either P(x) or Q(x) (or both) is true."

6. **$\neg$ (not)**

   This symbol is used to indicate the negation of a statement.

For example, ¬P(x) would mean "P(x) is not true."

These symbols are used to build complex logical statements in predicate logic, which is used to represent and reason about statements in natural language. Here are some examples of how the standard logic symbols are used in predicate logic:

1. ∀ (for all):

- ∀ x(P(x) → Q(x)) would mean "For all x, if P(x) is true, then Q(x) is also true."
- ∀ y ∃ x(Friend(x,y)) would mean "For all y, there exists an x such that x is a friend of y."

2. ∃ (there exists):

- ∃ x(P(x) ∧ Q(x)) would mean "There exists an x such that P(x) and Q(x) are both true."
- ∃ y ∀ x(Child(x,y)) would mean "There exists a y such that all x are children of y."

3. → (implies):

- P(x) → Q(x) would mean "If P(x) is true, then Q(x) is also true."
- (P(x) ∧ Q(x)) → R(x) would mean "If both P(x) and Q(x) are true, then R(x) is also true."

4. ∧ (and):

- P(x) ∧ Q(x) would mean "Both P(x) and Q(x) are true."
- (P(x) ∧ Q(x)) '" R(x) would mean "P(x), Q(x), and R(x) are all true."

5. ∨ (or):

- P(x) ∨ Q(x) would mean "Either P(x) or Q(x) (or both) is true."
- (P(x) ∨ Q(x)) ∨ R(x) would mean "Either P(x), Q(x), or R(x) (or more than one) is true."

6. ¬ (not):

- ¬ P(x) would mean "P(x) is not true."
- ¬ (P(x) '" Q(x)) would mean "It is not true that both P(x) and Q(x) are true."

Q16. How to represent simple facts in logic?

Ans :

Representation of Simple Facts in Logic

Simple facts can be represented in logic using statements or propositions that express a relationship between objects or concepts. In predicate logic, the basic unit of representation is a predicate, which is a statement that describes a relationship between one or more objects.

For example, consider the following simple fact: "John is a man." This fact can be represented in logic using a predicate, such as "Man(John)," where "Man" is the predicate and "John" is the object being described.

Similarly, the fact "The sky is blue" can be represented using the predicate "Blue(Sky)."

In logic, predicates can be combined using logical operators, such as "and" and "or," to represent more complex facts. For example, the fact "John is a man and Mary is a woman" can be represented using the predicates "Man(John) ∧ Woman(Mary)," where " ∧ " is the logical operator for "and."

We can also use quantifiers, such as "for all" (∀) and "there exists" (∃), to express statements that involve all or some objects. For example, the statement "All men are mortal" can be represented using the predicate " ∀ x (Man(x) → Mortal(x))," where " ∀ x" means "for all x," and " → " is the logical operator for "implies."

In summary, simple facts can be represented in logic using predicates, which describe a relationship between one or more objects. Predicates can be combined using logical operators and quantifiers to express more complex statements.

2.3.2 Representing Instance and is a Relationships

Q17. Explain the representation of instance and is a relationship.

Ans :

In AI, instance and ISA relationships are commonly used to represent objects and their relationships in a structured way. An instance is a particular occurrence or example of an object, while an ISA relationship indicates that one object is a subtype or subclass of another object.

One common way to represent instance and ISA relationships is through the use of a class hierarchy, where objects are organized into classes based on their attributes and relationships. The class hierarchy represents the ISA relationship, with each class representing a subtype of a more general superclass.

For example, in a class hierarchy for animals, we might have a superclass called "Animal," with subtypes such as "Mammal," "Bird," and "Reptile." Each of these subtypes might have additional subtypes, such as "Dog" and "Cat" for the Mammal class.

Instances of these classes can be represented by assigning values to their attributes. For example, an instance of the "Dog" class might have attributes such as "Breed," "Color," and "Age," with specific values assigned to each attribute.

Another common way to represent instance and ISA relationships is through the use of frames, which are structured representations of objects that include both their attributes and their relationships to other objects. A frame for an object might include slots for attributes such as "Name," "Size," and "Color," as well as slots for relationships to other objects, such as "Parent" or "PartOf."

For example, a frame for a car might include slots for attributes such as "Make," "Model," and "Year," as well as slots for relationships to other objects, such as "Engine" or "Wheel."

Representing instance and ISA relationships is a fundamental component of many AI applications, as it provides a structured and systematic way to organize and reason about objects and their relationships.

Example:

Marcus was a man let me explain the use of instance and ISA relationships using the statement "Marcus was a man" as an example.

In this case, we can represent "Marcus" as an instance of the class "Man." The class "Man" represents the general category of human males, and Marcus is an instance of that class because he is a specific example of a male human.

To represent this in a more structured way, we could use a class hierarchy with "Man" as a subtype of a more general superclass such as "Human." This would allow us to represent additional subtypes of humans, such as "Woman" or "Child."

We could also use a frame to represent Marcus as an object with attributes such as "Name," "Gender," and "Age." The attribute "Gender" would be set to "Male," indicating that Marcus is a male, and the attribute "Age" would represent his age.

Overall, the use of instance and ISA relationships allows us to represent objects and their relationships in a structured and systematic way. In the case of "Marcus was a man," we can use these relationships to represent Marcus as an instance of the class "Man" or as an object with attributes that include his gender and age.

1. Man(Marcus).
2. Pompeian(Marcus).
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman(x).
4. ruler(Caesar).
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. $\forall x$: instance(x, Pompeian) $\rightarrow$ instance(x, Roman).
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. isa(Pompeian, Roman)
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).
6. $\forall x$: $\forall y$: $\forall z$: instance(x, y) $\wedge$ isa(y, z)$\rightarrow$ instance(x, z).

Figure: Three ways of representing class membership: ISA Relationships

2.3.3 Computable Functions And Predicates

Q18. What are computable functions and predicates ? explain them.

Ans :

Computable Functions and Predicates

- To express simple facts, such as the following greater-than and less-than relationships: gt(1,O) It(0,1) gt(2,1) It(1,2) gt(3,2) It(2,3)
- It is often also useful to have computable functions as well as computable predicates.
- Thus we might want to be able to evaluate the truth of gt(2 + 3,1)
- To do so requires that we first compute the value of the plus function given the arguments2 and 3, and then send the arguments 5 and 1 to gt.

Consider the following set of facts, again involving Marcus

1) Marcus was a man.

   man(Marcus)

2) Marcus was a Pompeian.

   Pompeian(Marcus)

3) Marcus was born in 40 A.D.

   born(Marcus, 40)

4) All men are mortal.

   x: man(x) $\rightarrow$ mortal(x)

5) All Pompeians died when the volcano erupted in 79 A.D. erupted(volcano, 79)' $\wedge\forall$ x : [Pompeian(x) $\rightarrow$ died(x, 79)]

6) No mortal lives longer than 150 years.

   x: t1: At2: mortal(x) born(x, t1) gt(t2 – t1,150) $\rightarrow$ died(x, t2)

7) It is now 1991.

   now = 1991

   So, above example shows how these ideas of computable functions and predicates can be useful.

   It also makes use of the notion of equality and allows equal objects to be substituted for each other whenever it appears helpful to do so during a proof.

   So, Now suppose we want to answer the question "Is Marcus alive?"

   The statements suggested here, there may be two ways of deducing an answer.

   Either we can show that Marcus is dead because he was killed by the volcano or we can show that he must be dead because he would otherwise be more than 150 years old, which we know is not possible.

   Also, As soon as we attempt to follow either of those paths rigorously, however, we discover, just as we did in the last example, that we need some additional knowledge. For example, our statements talk about dying, but they say nothing that relates to being alive, which is what the question is asking.

   So we add the following facts:

8) Alive means not dead.

   x: t: [alive(x, t) $\rightarrow \neg$ dead(x, t)] [$\neg$ dead(x, t) $\rightarrow$ alive(x, t)]

9) If someone dies, then he is dead at all later times.

   x: t1: At2: died(x, t1) gt(t2, t1) $\rightarrow$ dead(x, t2)

   So, Now let's attempt to answer the question "Is Marcus alive?" by proving: $\neg$ alive(Marcus, now)

2.3.4 Resolution

Q19. Explain about proposition resolution and unification algorithm.

Ans : **(Imp.)**

Propositional Resolution

1. Convert all the propositions of F to clause form.
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in step 1.
3. Repeat until either a contradiction is found or no progress can be made:

1. Select two clauses. Call these the parent clauses.
2. Resolve them together. The resulting clause, called the resolvent, will be the disjunction of all of the literals of both of the parent clauses with the following exception: If there are any pairs of literals L and $\neg$ L such that one of the parent clauses contains L and the other contains $\neg$ L, then select one such pair and eliminate both L and $\neg$ L from the resolvent.

3. If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of classes available to the procedure.

The Unification Algorithm

- In propositional logic, it is easy to determine that two literals cannot both be true at thesame time.
- Simply look for L and ¬ L in predicate logic, this matching process is more complicatedsince the arguments of the predicates must be considered.
- For example, man(John) and ¬man(John) is a contradiction, while the man(John) and ¬ man(Spot) is not.
- Thus, in order to determine contradictions, we need a matching procedure that comparestwo literals and discovers whether there exists a set of substitutions that makes themidentical.
- There is a straightforward recursive procedure, called the unification algorithm, that doesit.

Algorithm: Unify(L1, L2)

1. If L1 or L2 are both variables or constants, then:
2. If L1 and L2 are identical, then return NIL.
3. Else if L1 is a variable, then if L1 occurs in L2 then return {FAIL}, else return (L2/L1).
4. Also, Else if L2 is a variable, then if L2 occurs in L1 then return {FAIL}, else return (L1/L2). d. Else return {FAIL}.
5. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.
6. If LI and L2 have a different number of arguments, then return {FAIL}.
7. Set SUBST to NIL. (At the end of this procedure, SUBST will contain all the substitutions used to unify L1 and L2.)
8. For I $\rightarrow$ 1 to the number of arguments in L1 :
    1. Call Unify with the ith argument of L1 and the ith argument of L2, putting the result in S.
    2. If S contains FAIL then return {FAIL}.
    3. If S is not equal to NIL then:
    4. Apply S to the remainder of both L1 and L2.
    5. SUBST: = APPEND(S, SUBST).
    6. Return SUBST.

Q20. Explain about resolution in propositional logic.

Ans :

Resolution in Predicate Logic

We can now state the resolution algorithm for predicate logic as follows, assuming a set of givenstatements F and a statement to be proved P:

Algorithm: Resolution

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 a contradiction found, no progress can make, or a predetermined amount of effort has expanded.
1. Select two clauses. Call these the parent clauses.
2. Resolve them together. The resolvent will the disjunction of all the literals of both parent clauses with appropriate substitutions performed and with the following exception: If there is one pair of literals T1 and ¬T2 such that one of the parent clauses contains T2 and the other contains T1 and if T1 and 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 theresolvent. If there is more than one pair of complementary literals, only one pair should omit from the resolvent.
3. If the resolvent is an empty clause, then a contradiction has found. Moreover, If it is not, then add it to the set of classes available to the procedure.

Resolution Procedure

Resolution is a procedure, which gains its efficiency from the fact that it operates on statements that have been converted to a very convenient standard form. Resolution produces proofs by refutation.

In other words, to prove a statement (i.e., to show that it is valid), resolution attempts to show that the negation of the statement produces a contradiction with the known statements (i.e., that it is unsatisfiable).

The resolution procedure is a simple iterative process: at each step, two clauses, called the parent clauses, are compared (resolved), resulting in a new clause that has inferred from them. The new clause represents ways that the two parent clauses interact with each other. Suppose that there are two clauses in the system:

winterV summer

¬ winterV cold

- Now we observe that precisely one of winter and ¬ winter will be true at any point.
- If winter is true, then cold must be true to guarantee the truth of the second clause. If ¬ winter is true, then summer must be true to guarantee the truth of the first clause.
- Thus we see that from these two clauses we can deduce summer V cold
- This is the deduction that the resolution procedure will make.
- Resolution operates by taking two clauses that each contains the same literal, in thisexample, ***winter***.
- Moreover, The literal must occur in the positive form in one clause and in negative formin the other. The resolvent obtained by combining all of the literals of the two parentclauses except the ones that cancel.
- If the clause that produced is the empty clause, then a contradiction has found.

For example, the two clauses

Winter

¬ winter

will produce the empty clause.

2.3.5 Natural Deduction

Q21. What is natural deduction? Explain about various rules of inference.

Ans : **(Imp.)**

Testing whether a proposition is a tautology by testing every possible truth assignment is expensive—there are exponentially many. We need a deductive system, which will allow us toconstruct proofs of tautologies in a step-by-step fashion.

The system we will use is known as natural deduction. The system consists of a set of rules of Inference for deriving consequences from premises. One builds a proof tree whose root is theproposition to be proved and whose leaves are the initial assumptions or axioms.

For example, one rule of our system is known as modus ponens. Intuitively, this says that if weknow P is true, and we know that P implies Q, then we can conclude Q.

P P ⇒ Q

---------- (modus ponens)

Q

The propositions above the line are called premises; the proposition below the line is theconclusion. Both the premises and the conclusion may contain metavariables (in this case, P and Q) representing arbitrary propositions. When an inference rule is used as part of a proof, themetavariables are replaced in a consistent way with the appropriate kind of object .

Most rules come in one of two flavors: introduction or elimination rules. Introduction rules introduce the use of a logical operator, and elimination rules eliminate it. Modus ponens is an elimination rule for ⇒ . On the right-

hand side of a rule, we often write the name of the rule. Thisis helpful when reading proofs. In this case, we have written (modus ponens). We could alsohave written ($\Rightarrow$-elim) to indicate that this is the elimination rule for $\Rightarrow$.

Rules for Conjunction

Conjunction ($\wedge$) has an introduction rule and two elimination rules:

$$\frac{P\ Q}{P \wedge Q}(\wedge - \text{intro})\ \frac{P \wedge Q}{P}(\wedge - \text{elim} - \text{left})\ \frac{P \wedge Q}{Q}(\wedge - \text{elim} - \text{right})$$

Rule for T

The simplest introduction rule is the one for T. It is called "unit". Because it has no premises, this rule is an axiom: something that can start a proof. a proof.

_____ (unit)

T

Rules for Implication

In natural deduction, to prove an implication of the form $P \Rightarrow Q$, we assume P, then reason under that assumption to try to derive Q. If we are successful, then we can conclude that $P \Rightarrow Q$. In a proof, we are always allowed to introduce a new assumption P, then reason under that assumption. We must give the assumption a name; we have used the name x in the example below. Each distinct assumption must have a different name.

_____ (assum)

[x : P]

Because it has no premises, this rule can also start a proof. It can be used as if the proposition P were proved. The name of the assumption is also indicated here.

However, you do not get to make assumptions for free! To get a complete proof, all assumptions must be eventually *discharged*. This is done in the implication introduction rule. This rule introduces an implication $P \Rightarrow Q$ by discharging a prior assumption [x : P]. Intuitively, if Q can be proved under the assumption P, then the implication $P \Rightarrow Q$ holds without any assumptions.

We write x in the rule name to show which assumption is discharged. This rule and modus ponens are the introduction and elimination rules for implications.

$$\frac{\begin{array}{c}[x : p] \\ \vdots \\ Q\end{array}}{P \Rightarrow Q}(\Rightarrow \text{intro/x}) \qquad \frac{P \qquad P \Rightarrow Q}{Q}$$

A proof is valid only if every assumption is eventually discharged. This must happen in the proof tree below the assumption. The same assumption can be used more than once.

Rules for Disjunction

$$\frac{P}{P \vee Q}(\vee - \text{intro} - \text{left}) \qquad \frac{Q}{P \vee Q}(\vee - \text{intro} - \text{right})\ \frac{P \vee Q\ P \Rightarrow R\ Q \Rightarrow R}{R}(\vee - \text{elim})$$

Rules for Negation

A negation $\neg P$ can be considered an abbreviation for $P \Rightarrow \bot$

$$\frac{P \Rightarrow \bot}{\neg P}(\neg - intro) \quad \frac{\neg P}{P \Rightarrow \bot}(\neg - elim)$$

Rules for Falsity

$$\begin{array}{c}[x : \neg P] \\ \vdots \\ \bot \end{array} \over P (reduction\ ad\ absurdum, RAA / x) \quad \frac{\bot}{P}(ex\ falso\ quodlibet, EFQ)$$

Reductio ad absurdum (RAA) is an interesting rule. It embodies proofs by contradiction. It says that if by assuming that P is false we can derive a contradiction, then P must be true. The assumption x is discharged in the application of this rule. This rule is present in classical logicbut not in intuitionistic (constructive) logic. In intuitionistic logic, a proposition is not considered true simply because its negation is false.

Excluded Middle

Another classical tautology that is not intuitionistically valid is the the law of the excluded middle, $P \vee \neg P$. We will take it as an axiom in our system. The Latin name for this rule istertium non datur, but we will call it magic.

$$\frac{}{P \vee \neg P.}(magic)$$

Proofs

A proof of proposition P in natural deduction starts from axioms and assumptions and derives P with all assumptions discharged. Every step in the proof is an instance of an inference rule with meta variables substituted consistently with expressions of the appropriate syntactic class.

Example

For example, here is a proof of the proposition $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \wedge B \Rightarrow C)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[y : A \wedge B]}(A)}{A}(\wedge E) \quad \dfrac{\dfrac{}{[x : A \Rightarrow B \Rightarrow C]}(A)}{B \Rightarrow C}(\Rightarrow E) \quad \dfrac{\dfrac{}{[y : A \wedge B]}(A)}{B}(\wedge E)}{C}(\Rightarrow E)}{A \wedge B \Rightarrow C}(\Rightarrow I, y)}{(A \Rightarrow B \Rightarrow C) \Rightarrow (A \wedge B \Rightarrow C)}(\Rightarrow I, x)$$

The final step in the proof is to derive $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \wedge B \Rightarrow C)$ from $(A \wedge B \Rightarrow C)$, which is done using the rule ($\Rightarrow$-intro), discharging the assumption $[x : A \Rightarrow B \Rightarrow C]$. To see how this rule generates the proof step, substitute for the metavariables P, Q, x in the rule as follows: $P = (A \Rightarrow B \Rightarrow C)$, $Q = (A \wedge B \Rightarrow C)$, and $x = x$. The immediately previous step uses the same rule, but with a different substitution: $P = A \wedge B$, $Q = C$, $x = y$.

Short Question and Answers

1. Game playing in AI.

Ans :

Game Playing is an important domain of artificial intelligence. Games don't require much knowledge; the only knowledge we need to provide is the rules, legal moves and the conditions of winning or losing the game.

2. Min - Max Search algorithm.

Ans :

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.

3. Explain about alpha beta pruning technique.

Ans :

i) Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.

ii) As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.

iii) Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

4. Iterative Deepening.

Ans :

Iterative Deepening Depth-First Search (IDDFS) is a search algorithm that performs depth-first search (DFS) in a tree or graph with an increasing depth limit until the goal state is found. The working algorithm of IDDFS can be described as follows:

i) Initialize the depth limit to 0.

ii) Repeat the following steps for increasing values of the depth limit until the goal state is found or there are no more nodes to explore: a. Perform a depth-first search on the tree or graph with the current depth limit. b. If the goal state is found, return the solution. c. If there are no more nodes to explore, return failure.

iii) If the goal state is not found after exploring all nodes up to a certain depth limit, increase the depth limit and repeat steps 2a-2c.

5. What is knowledge representation?

Ans :

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning. Hence we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.

6. Frame problem in AI.

Ans :

The frame problem is a well-known problem in artificial intelligence that arises when attempting to model changes in a system over time. It is named after the

philosopher and cognitive scientist, Herbert A. Simon, who first introduced the concept in 1969.

The frame problem refers to the difficulty of determining which information is relevant when updating a knowledge representation after a change in the state of the world. In other words, it is the problem of how to represent and reason about the effects of actions on a dynamic environment while avoiding the explosion of irrelevant information.

7. What is Logic?

Ans :

Logic is a systematic way of representing and analyzing the structure of arguments and reasoning. It is concerned with understanding how reasoning works and how to evaluate the truth or validity of arguments.

Logic is a formal method for reasoning. Many concepts which can be verbalized can be translated into symbolic representations which closely approximate the meaning of these concepts. These symbolic structures can then be manipulated in programs to deduce various facts, to carry out a form of automated reasoning .in predicate logic statements from a natural language like English are translated into symbolic structures comprised of predicates, functions, variables, constants, quantifiers and logical connectives. The symbols form the basic building blocks for the knowledge, and their combination into valid structures is accomplished using the syntax of FOPL. Once structures have been created to represent basic facts, inference rules may then be applied to compare, combine and transform these "assumed" structures into new "deduced" structures. This is how automated reasoning or inferencing is performed.

8. Natural deduction

Ans :

Testing whether a proposition is a tautology by testing every possible truth assignment is expensive - there are exponentially many. We need a deductive system, which will allow us toconstruct proofs of tautologies in a step-by-step fashion.

The system we will use is known as natural deduction. The system consists of a set of rules of Inference for deriving consequences from premises. One builds a proof tree whose root is theproposition to be proved and whose leaves are the initial assumptions or axioms.

9. Disadvantages of IDDFS

Ans :

- The time taken is exponential to reach the goal node.
- The main problem with IDDFS is the time and wasted calculations that take place at each depth.
- The situation is not as bad as we may think of especially when the branching factor is found to be high.
- The IDDFS might fail when the BFS fails. When we are to find multiple answers from the IDDFS, it gives back the success nodes and its path once even if it needs to be found again after multiple iterations. To stop the depth bound is not increased further.

10. Advantages to IDDFS

Ans :

- IDDFS gives us the hope to find the solution if it exists in the tree.
- When the solutions are found at the lower depths say n, then the algorithm proves to be efficient and in time.
- The great advantage of IDDFS is found in-game tree searching where the IDDFS search operation tries to improve the depth definition, heuristics, and scores of searching nodes so as to enable efficiency in the search algorithm.
- Another major advantage of the IDDFS algorithm is its quick responsiveness. The early results indications are a plus point in this algorithm. This followed up with multiple refinements after the individual iteration is completed.
- Though the work is done here is more yet the performance of IDDFS is better than single BFS and DFS operating exclusively.

Choose the Correct Answers

1. Which of the following is an example of a perfect information game? [d]
   (a) Tic-tac-toe (b) Poker
   (c) Chess (d) Both a and c
2. In game theory, what is the minimax algorithm used for? [a]
   (a) To minimize the maximum loss (b) To maximize the minimum gain
   (c) To minimize the minimum loss (d) To maximize the maximum gain
3. What is alpha-beta pruning used for? [b]
   (a) To increase the number of nodes explored in a search tree
   (b) To reduce the number of nodes explored in a search tree
   (c) To evaluate the quality of a move
   (d) To select the best move in a game
4. Which of the following is not a commonly used knowledge representation technique? [d]
   (a) Logical representation (b) Semantic networks
   (c) Bayesian networks (d) Reinforcement learning
5. Which of the following knowledge is representing knowledge of some experts in a filed or subject. [b]
   (a) Meta-knowledge (b) Heuristic knowledge
   (c) Structural knowledge (d) Declarative Knowledge
6. Among the following in which knowledge approach, all data must be stored into a hierarchy of classes. [b]
   (a) Simple relational knowledge (b) Inheritable knowledge
   (c) Inferential knowledge (d) Procedural knowledge
7. Which of the following is the correct way to represent "All dogs are mammals" in predicate logic? [a]
   (a) $\forall x$ (Dog(x) $\rightarrow$ Mammal(x)) (b) $\forall x$ (Dog(x) $\wedge$ Mammal(x))
   (c) $\exists x$ (Dog(x) $\wedge$ Mammal(x)) (d) $\exists x$ (Dog(x) $\rightarrow$ Mammal(x))
8. Which of the following is an example of a quantifier in predicate logic? [c]
   (a) + (b) ->
   (c) $\exists$ (d) $\wedge$
9. Which of the following is a propositional variable? [d]
   (a) p (b) q
   (c) r (d) All of the above
10. Which of the following is a valid rule of inference? [a]
    (a) Modus Ponens (b) Modus Operandi
    (c) Abductive Reasoning (d) Appeal to Emotion

Fill in the Blanks

1. _____ algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory.
2. The time complexity of min- max algorithm is _____
3. _____ algorithm is an optimization technique for the minimax algorithm
4. _____ is a refinement in AI that involves training a model on one task and then using the knowledge gained to improve the performance on a different task.
5. _____ indicates that the current node is worse than the best option for the maximizing player, and therefore it can be pruned.
6. _____ is also known as imperative knowledge.
7. _____ knowledge approach represents knowledge in the form of formal logics
8. The fundamental goal of knowledge Representation is _____
9. Which AI technique enables the computers to understand the associations and relationships between objects and events _____
10. In predicate logic, the basic unit of representation is a _____

ANSWERS

1. Mini-max
2. $O(b^m)$
3. Alpha beta pruning.
4. Transfer learning
5. Beta cutoffs
6. Procedural Knowledge
7. Inferential
8. To facilitate inference from knowledge
9. Pattern Matching
10. Predicate

UNIT III

Uncertainty and Reasoning Techniques:Non monotonic reasoning, Logics for Non monotonic reasoning, Implementation issues, Augmenting a problem solver, implementation of Depth First Search and Breadth first search. Statistical reasoning:Probability and Bayes theorem, Certainty factors and Rule-based systems, Bayesian Networks, Dempster-Shafer Theory.

3.1 Uncertainty and Reasoning Techniques

3.1.1 Non Monotonic Reasoning

Q1. What is monotonic and non monotonic reasoning? Explain.

Ans : **(Imp.)**

(i) Monotonic Reasoning

In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.

To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.

Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.

Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic. Any theorem proving is an example of monotonic reasoning.

Example

Earth Revolves Around the Sun

It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

Advantages of Monotonic Reasoning

- In monotonic reasoning, each old proof will always remain valid.
- If we deduce some facts from available facts, then it will remain valid for always.

Disadvantages of Monotonic Reasoning

- We cannot represent the real world scenarios using Monotonic reasoning.
- Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be true.
- Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

(ii) Non-monotonic Reasoning

In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.

Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.

Non-monotonic reasoning deals with incomplete and uncertain models.

"Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

Example

Let suppose the knowledge base contains the following knowledge:

- Birds can fly
- Penguins cannot fly
- Pitty is a bird

So from the above sentences, we can conclude that Pitty can fly.

However, if we add one another sentence into knowledge base "Pitty is a penguin", which concludes "Pitty cannot fly", so it invalidates the above conclusion.

Advantages

- For real-world systems such as Robot navigation, we can use non-monotonic reasoning.
- In Non-monotonic reasoning, we can choose probabilistic facts or can make assumptions.

Disadvantages

- In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.
- It cannot be used for theorem proving.

3.1.2 Logics For Non Monotonic Reasoning

Q2. Explain various types of logics in non monotonic reasoning.

Ans :

Non-monotonic reasoning is a type of reasoning in which conclusions may be revised or withdrawn in the light of new information. This is in contrast to monotonic reasoning, where the addition of new information can only lead to the strengthening of previously reached conclusions.

The logics for non-monotonic reasoning are formal systems designed to handle the uncertainties and inconsistencies that arise in non-monotonic reasoning. These logics are used to model reasoning processes that involve incomplete or uncertain information and allow for the retraction or revision of conclusions based on new evidence.

There are several types of logics used in non-monotonic reasoning, each designed to handle different types of uncertainty and inconsistency. Here are some examples:

1. Default Logic

This logic is used to express rules that are assumed to be true unless there is evidence to the contrary. Default logic allows for the retraction or revision of conclusions based on new evidence.

2. Autoepistemic Logic

This logic is used to model situations in which an agent has incomplete knowledge about its own beliefs. It allows for the agent to update its beliefs based on new information.

3. Circumscription

This logic is used to simplify the representation of a domain by defining the domain in terms of a smaller set of properties or constraints.

4. Logic programming

This is a form of non-monotonic logic that is used to represent knowledge in the form of rules. Logic programming allows for the addition of new rules that can modify or retract previously held beliefs.

5. Probabilistic Logic

This logic is used to reason under uncertainty by assigning probabilities to propositions. Probabilistic logic allows for the updating of probabilities based on new evidence.

6. Modal Logic

This logic is used to reason about different possible worlds or states of affairs. Modal logic allows for the representation of beliefs that are only true in some possible worlds but not others.

Non-monotonic reasoning involves the use of various types of logics to handle uncertainty and inconsistency. Each logic is designed to handle different types of problems and can be used to model different types of reasoning processes.

3.1.3 Implementation Issues

Q3. What are the implementation issues in non-monotonic reasoning.

Ans :

Non-monotonic reasoning is a type of reasoning where new information can change the conclusions that

were previously drawn. Here are some implementation issues that are relevant in non-monotonic reasoning:

1. Conflict Resolution

In non-monotonic reasoning, it is possible for different pieces of information to conflict with each other, leading to inconsistent conclusions. Conflict resolution techniques must be implemented to resolve these conflicts and ensure that the reasoning process is consistent.

2. Reasoning with Uncertainty

Non-monotonic reasoning often involves dealing with uncertain or incomplete information. Techniques such as probabilistic reasoning, Bayesian networks, and fuzzy logic can be used to reason with uncertainty and provide more accurate conclusions.

3. Inference Control

In non-monotonic reasoning, the reasoning process may need to be controlled in order to ensure that the conclusions drawn are consistent with the domain knowledge. Inference control techniques, such as default logic, circumscription, and abduction, can be used to control the reasoning process and ensure that the conclusions are appropriate.

4. Integration with Other Reasoning Methods

Non-monotonic reasoning may need to be integrated with other reasoning methods, such as deductive reasoning or abductive reasoning, in order to provide a more complete picture of the domain knowledge. Integration techniques must be implemented to ensure that the different reasoning methods are compatible and consistent.

5. Scalability

Non-monotonic reasoning can become computationally intensive as the amount of knowledge and the complexity of the reasoning increases. Techniques such as knowledge compilation, caching, and incremental reasoning can be used to improve the scalability of the reasoning process.

6. Explanation and Visualization

Non-monotonic reasoning can often produce complex and hard-to-understand conclusions. Explanation and visualization techniques can be used to help users understand the reasoning process and the conclusions that are drawn.

7. Knowledge Representation

Non-monotonic reasoning requires an appropriate knowledge representation in order to reason effectively with uncertain or incomplete information. Knowledge representation techniques, such as frames, semantic networks, and ontologies, can be used to represent the domain knowledge in a way that is suitable for non-monotonic reasoning.

3.1.4 Augmenting a Problem Solver

Q4. What is augmenting a problem solver in AI.

Ans : **(Imp.)**

Augmenting a problem solver refers to the process of adding additional capabilities or knowledge to an existing problem-solving system in order to improve its performance or extend its functionality.

Here are some examples of how a problem solver can be augmented in AI:

1. Adding New Rules or Knowledge

One way to augment a problem solver is by adding new rules or knowledge to its existing knowledge base. For example, a medical diagnosis system can be augmented by adding new rules that incorporate new medical research or treatment guidelines.

2. Integrating with External Data Sources

Problem solvers can be augmented by integrating them with external data sources, such as databases or web services. For example, a financial decision-making system can be augmented by integrating it with real-time stock market data or news feeds.

3. Improving Reasoning Mechanisms

Problem solvers can be augmented by improving their reasoning mechanisms, such as their search algorithms or optimization techniques. For example, a scheduling system can be augmented by improving its search algorithm to find more efficient schedules.

4. Incorporating Machine Learning Techniques

Machine learning techniques, such as supervised or unsupervised learning, can be used to augment problem solvers by improving their accuracy or reducing their reliance on hand-coded rules. For example, a spam filter can be augmented by using machine learning techniques to improve its accuracy in identifying spam emails.

5. Enabling Human-Machine Collaboration

Augmenting problem solvers can also involve enabling collaboration between humans and machines. For example, a customer service chatbot can be augmented by enabling it to transfer complex questions to a human agent for further assistance.

6. Adding Natural Language Processing Capabilities

Problem solvers can be augmented by adding natural language processing (NLP) capabilities, which can enable them to process and understand human language inputs. For example, a virtual assistant can be augmented by adding NLP capabilities to enable it to understand and respond to spoken commands.

7. Incorporating Expert Systems

Expert systems are AI systems that mimic the decision-making abilities of a human expert in a specific domain. Augmenting problem solvers with expert systems can enable them to make more accurate and informed decisions. For example, a medical diagnosis system can be augmented with an expert system that incorporates the knowledge and experience of a medical specialist.

Overall, augmenting problem solvers in AI can involve a wide range of techniques and strategies, and the choice of approach will depend on the specific problem-solving task and the desired improvements or extensions to the existing system.

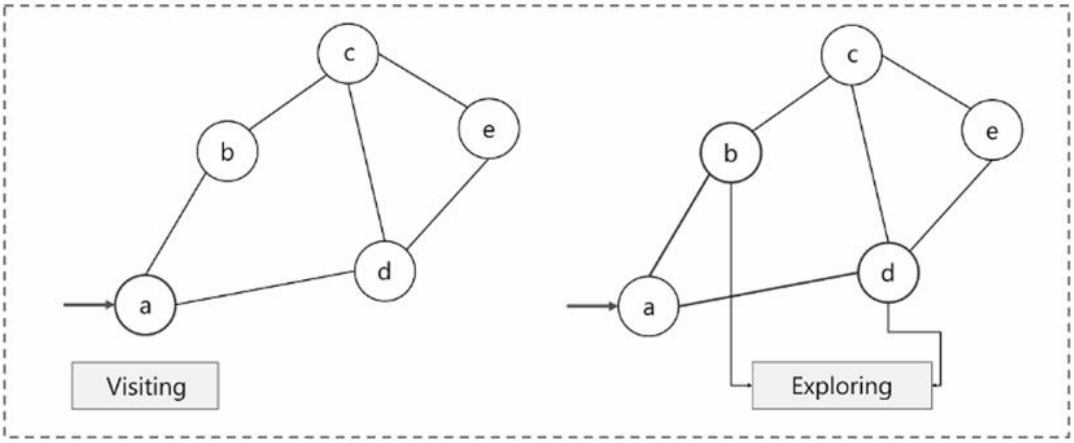
3.1.5 Implementation of Depth First Search and Breadth First Search

Q5. What is the Breadth-First Search Algorithm?

Ans : **(Imp.)**

Breadth-First Search algorithm is a graph traversing technique, where you select a random initial node (source or root node) and start traversing the graph layer-wise in such a way that all the nodes and their respective children nodes are visited and explored.

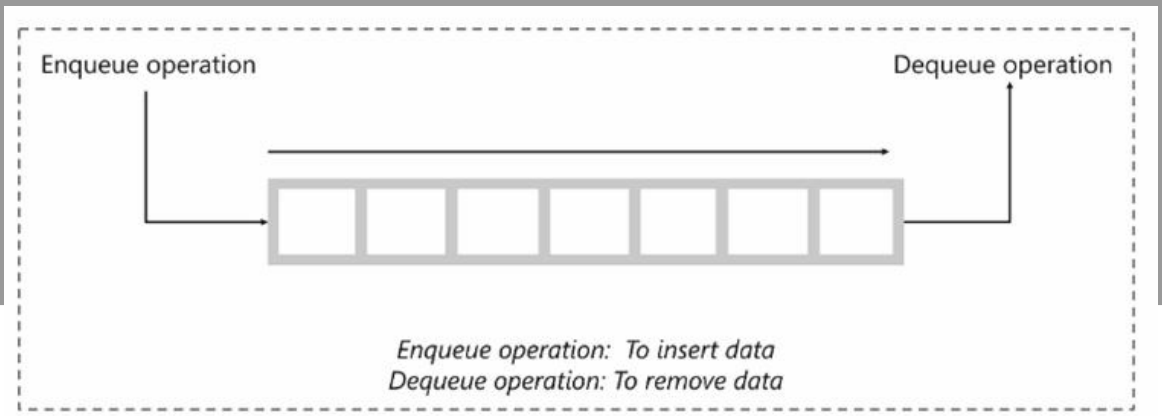
Before we move further and understand Breadth-First Search with an example, let's get familiar with two important terms related to graph traversal:



1. **Visiting a node :** Just like the name suggests, visiting a node means to visit or select a node.
2. **Exploring a node :** Exploring the adjacent nodes (child nodes) of a selected node.

Understanding the Breadth-First Search Algorithm with an example Breadth-First Search algorithm follows a simple, level-based approach to solve a problem. Consider the below binary tree (which is a graph). Our aim is to traverse the graph by using the Breadth-First Search Algorithm. Before we get started, you must be familiar with the main data structure involved in the Breadth-First Search algorithm.

A queue is an abstract data structure that follows the First-In-First-Out methodology (data inserted first will be accessed first). It is open on both ends, where one end is always used to insert data (enqueue) and the other is used to remove data (dequeue).



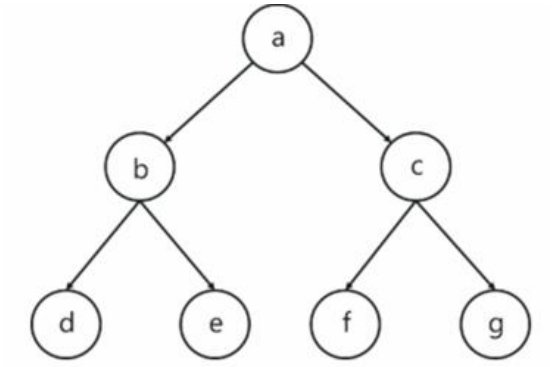
Now let's take a look at the steps involved in traversing a graph by using Breadth-First Search:

Step 1: Take an Empty Queue.

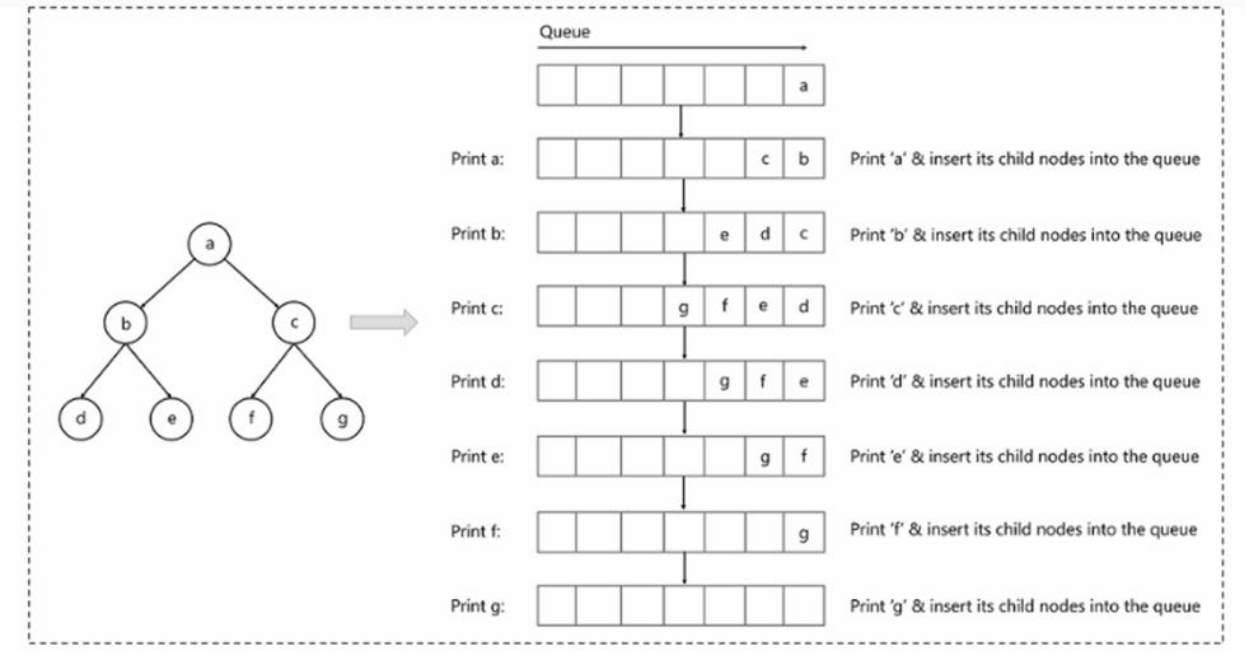
Step 2: Select a starting node (visiting a node) and insert it into the Queue.

Step 3: Provided that the Queue is not empty, extract the node from the Queue and insert its child nodes (exploring a node) into the Queue.

Step 4: Print the extracted node. Don't worry if you're confused, we shall understand this with an example. Take a look at the below graph, we will use the Breadth-First Search algorithm to traverse through the graph.



In our case, we'll assign node a' as the root node and start traversing downward and follow the steps mentioned above.



The above image depicts the end-to-end process of Breadth-First Search Algorithm. Let me explain this in more depth.

1. Assign a' as the root node and insert it into the Queue.
2. Extract node a' from the queue and insert the child nodes of a', i.e., b' and c'.
3. Print node a'.
4. The queue is not empty and has node b' and c'. Since b' is the first node in the queue, let's extract it and insert the child nodes of b', i.e., node d' and e'.
5. Repeat these steps until the queue gets empty. Note that the nodes that are already visited should not be added to the queue again.

Now let's look at the pseudocode of Breadth-First Search algorithm.

Breadth-First Search Algorithm Pseudocode

Here's the pseudocode to implement the Breadth-First Search Algorithm:

1. Input: s as the source node
2. BFS (G, s)
3. let Q be queue.
4. Q.enqueue(s)
5. mark s as visited
6. while (Q is not empty)
7. v = Q.dequeue( )
8. for all neighbors w of v in Graph G
9. if w is not visited
10. Q.enqueue(w)
11. markw as visited

In the above code, the following steps are executed:

1. (G, s) is input, here G is the graph and s is the root node
2. A queue Q' is created and initialized with the source node s'
3. All child nodes of s' are marked
4. Extract s' from queue and visit the child nodes
5. Process all the child nodes of v
6. Stores w (child nodes) in Q to further visit its child nodes
7. Continue till Q' is empty

Before we wrap up the blog, let's look at some applications of Breadth-First Search algorithm.

Q6. Explain Depth First Search algorithm.

Ans : **(Imp.)**

Depth First Search (DFS)

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows:

Pick a starting node and push all its adjacent nodes into a stack.

Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.

Repeat this process until the stack is empty. However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.

Pseudocode

```
DFS-iterative (G, s):
    //Where G is graph and s is source vertex
letS be stack
S.push( s )          //Inserting s in stack
mark s as visited.
while ( S is not empty):
    //Pop a vertex from stack to visit next
v  =S.top( )
S.pop( )
//Push all the neighbours of v in stack that are not visited
for all neighbours w of v in Graph G:
if w is not visited :
S.push( w )
markw as visited
  DFS-recursive(G, s):
mark s as visited
for all neighbours w of s in Graph G:
if w is not visited:
        DFS-recursive(G, w)
```

The following image shows how DFS works.

Time complexity O(V+E), when implemented using an adjacency list.

3.2 STATISTICAL REASONING

3.2.1 Probability and Bayes Theorem

Q7. What is uncertainty and state the causes of uncertainty.

Ans : **(Imp.)**

Meaning

Representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A'!B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

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

Causes of Uncertainty

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

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

Q8. Explain about probabilistic reasoning.

Ans :

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need of Probabilistic Reasoning in AI

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.

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

- Bayes' rule
- Bayesian Statistics

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, let's understand some common terms:

Probability

Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

$0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.

$P(A) = 0$, indicates total uncertainty in an event A.

$P(A) = 1$, indicates total certainty in an event A.

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

$$\text{Probabiltiy ofoccurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcoms}}$$

- $P(\neg A)$ = probability of a not happening event.
- $P(\neg A) + P(A) = 1$.
- **Event:** Each possible outcome of a variable is called an event.
- **Sample space:** The collection of all possible events is called sample space.
- **Random variables:** Random variables are used to represent the events and objects in the real world.
- **Prior probability:** The prior probability of an event is probability computed before observing new information.
- **Posterior Probability:** The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional Probability

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

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

Where

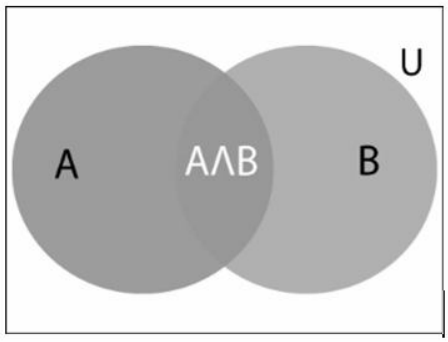
$P(A \wedge B)$ = Joint probability of a and B

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

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

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

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



Example:

In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

Sol :

Let, A is an event that a student likes Mathematics

B is an event that a student likes English.

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

Hence, 57% are the students who like English also like Mathematics.

Q9. Explain briefly about Bayes' theorem.

Ans :

Bayes' theorem is also known as Bayes' rule, Bayes' law, or Bayesian reasoning, which determines the probability of an event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two random events.

Bayes' theorem was named after the British mathematician Thomas Bayes. The Bayesian inference is an application of Bayes' theorem, which is fundamental to Bayesian statistics.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

Example:

If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.

Bayes' theorem can be derived using product rule and conditional probability of event A with known event B.

As from product rule we can write:

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

Similarly, the probability of event B with known event A:

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

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

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

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

It shows the simple relationship between joint and conditional probabilities. Here,

P(A|B) is known as posterior, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.

P(B|A) is called the likelihood, in which we consider that hypothesis is true, then we calculate the probability of evidence.

P(A) is called the prior probability, probability of hypothesis before considering the evidence

P(B) is called marginal probability, pure probability of an evidence.

In the equation (a), in general, we can write P (B) = P(A)*P(B|Ai), hence the Bayes' rule can be written as:

$$P(A_i|B) = \frac{P(A_i) * P(B \mid A_i)}{\sum_{i=1}^{k} P(A_i) * P(B \mid A_i)}$$

Where A_1, A_2, A_3,....., A_n is a set of mutually exclusive and exhaustive events.

Applying Bayes' Rule

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(B), and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect} \mid \text{cause})P(\text{cause})}{P(\text{effect})}$$

Example-1:

Application of Bayes' theorem in Artificial intelligence:

Following are some applications of Bayes' theorem:

- It is used to calculate the next step of the robot when the already executed step is given.
- Bayes' theorem is helpful in weather forecasting.
- It can solve the Monty Hall problem.

3.2.2 Certainty Factors and Rule-based Systems

Q10. What is certainty factor and explain about rule based systems.

Ans : **(Imp.)**

The Certainty Factor (CF) is a numeric value which tells us about how likely an event or a statement is supposed to be true. It is somewhat similar to what we define in probability, but the difference in it is that an agent after finding the probability of any event to occur cannot decide what to do. Based on the probability and other knowledge that the agent has, this certainty factor is decided through which the agent can decide whether to declare the statement true or false.

The value of the Certainty factor lies between -1.0 to +1.0, where the negative 1.0 value suggests that the statement can never be true in any situation, and the positive 1.0 value defines that the statement can

never be false. The value of the Certainty factor after analyzing any situation will either be a positive or a negative value lying between this range. The value 0 suggests that the agent has no information about the event or the situation.

A minimum Certainty factor is decided for every case through which the agent decides whether the statement is true or false. This minimum Certainty factor is also known as the threshold value. For example, if the minimum certainty factor (threshold value) is 0.4, then if the value of CF is less than this value, then the agent claims that particular statement false.

Rule Based Systems

A rule is an expression of the form "if A then B" where A is an assertion and B can be either an action or another assertion.

Example :

Trouble shooting of water pumps

- If pump failure then the pressure is low
- If pump failure then check oil level
- If power failure then pump failure
- Rule based system consists of a library of such rules.
- Rules reflect essential relationships within the domain.
- Rules reflect ways to reason about the domain.
- Rules draw conclusions and points to actions, when specific information about the domain comes in. This is called inference.

The inference is a kind of chain reaction like :

If there is a power failure then (see rules 1, 2, 3 mentioned above) Rule 3 states that there is a pump failure, and

- **Rule 1:** tells that the pressure is low, and
- **Rule 2 :** gives a (useless) recommendation to check the oil level.

It is very difficult to control such a mixture of inference back and forth in the same session and resolve such uncertainties.

How to deal such uncertainties ?

How to deal uncertainties in rule based system?

A problem with rule-based systems is that often the connections reflected by the rules are not absolutely certain (i.e. deterministic), and the gathered information is often subject to uncertainty.

In such cases, a certainty measure is added to the premises as well as the conclusions in the rules of the system.

A rule then provides a function that describes : how much a change in the certainty of the premise will change the certainty of the conclusion.

In its simplest form, this looks like :

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

This is a new rule, say rule 4, added to earlier three rules.

There are many schemes for treating uncertainty in rule based systems. The most common are :

- Adding certainty factors.
- Adoptions of Dempster-Shafer belief functions.
- Inclusion of fuzzy logic.

In these schemes, uncertainty is treated locally, means action is connected directly to incoming rules and uncertainty of their elements. Example : In addition to rule 4 , in previous slide, we have the rule

If C (with certainty x) then B (with certainty g(x))

Now If the information is that A holds with certainty a and C holds with certainty c, Then what is the certainty of B ?

3.2.3 Bayesian Networks

Q11. Explain about Bayesian networks.

Ans : **(Imp.)**

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

It is also called a Bayes network, belief network, decision network, or Bayesian model.

Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection.

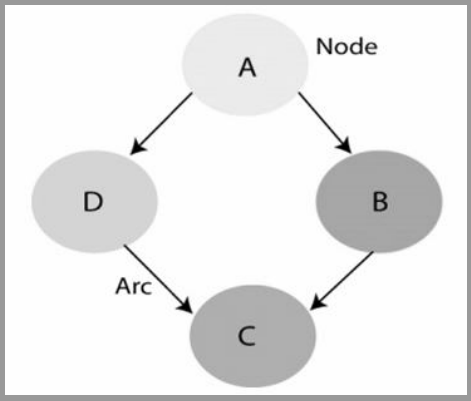
Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty.

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

- Directed Acyclic Graph
- Table of conditional probabilities.

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an Influence diagram.

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



- Each node corresponds to the random variables, and a variable can be continuous or discrete.
- Arc or directed arrows represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.

  These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
- In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.
- If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.
- Node C is independent of node A.

The Bayesian network has mainly two components:

- Causal Component
- Actual numbers

Each node in the Bayesian network has condition probability distribution $P(X_i \mid Parent(X_i))$, which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

Joint Probability Distribution

If we have variables $x_1, x_2, x_3, \ldots, x_n$, then the probabilities of a different combination of $x_1, x_2, x_3 \ldots x_n$, are known as Joint probability distribution.

$P[x_1, x_2, x_3, \ldots, x_n]$, it can be written as the following way in terms of the joint probability distribution.

$= P[x_1 \mid x_2, x_3, \ldots, x_n]P[x_2, x_3, \ldots, x_n]$

$= P[x_1 \mid x_2, x_3, \ldots, x_n]P[x_2 \mid x_3, \ldots, x_n] \ldots P[x_{n-1} \mid x_n]P[x_n]$.

In general for each variable Xi, we can write the equation as:

$P(X_i \mid X_{i-1}, \ldots, X_1) = P(X_i \mid Parents(X_i))$

Explanation of Bayesian Network

Let's understand the Bayesian network through an example by creating a directed acyclic graph:

Example

Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Q12. Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Sol : **(Imp.)**

- The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.
- The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.
- The conditional distributions for each node are given as conditional probabilities table or CPT.
- Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
- In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

List of all Events Occurring in this Network

- Burglary (B)
- Earthquake(E)
- Alarm(A)
- David Calls(D)
- Sophia calls(S)

We can write the events of problem statement in the form of probability: P[D, S, A, B, E], can rewrite the above probability statement using joint probability distribution:

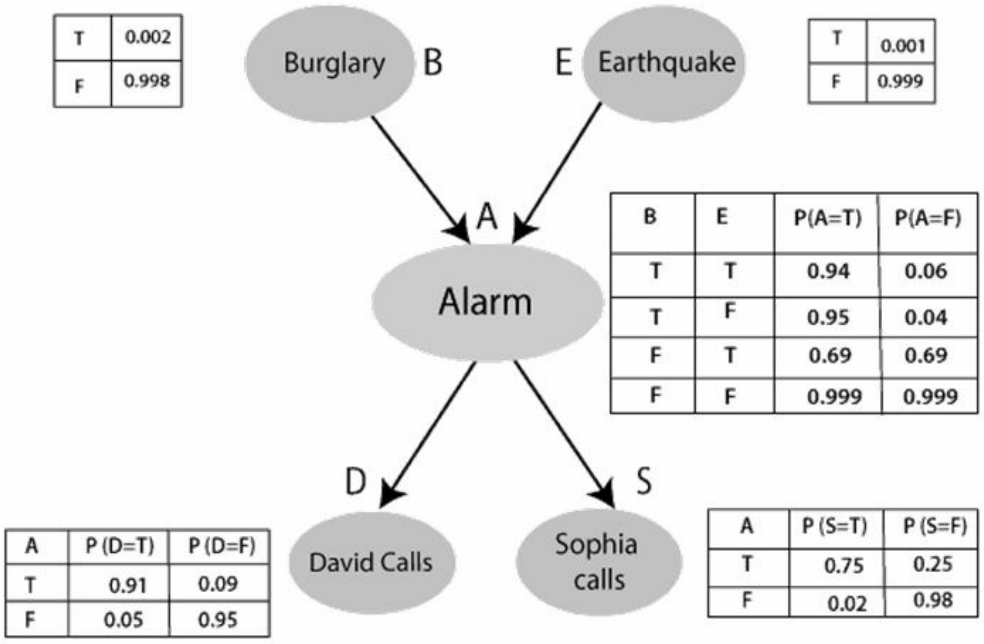
$$P[D, S, A, B, E] = P[D \mid S, A, B, E].\ P[S, A, B, E]$$

$$= P[D \mid S, A, B, E].\ P[S \mid A, B, E].P[A, B, E]$$

$$= P[D \mid A].\ P[S \mid A, B, E].\ P[A, B, E]$$

$$= P[D \mid A].\ P[S \mid A].\ P[A \mid B, E].\ P[B, E]$$

$$= P[D \mid A\ ].\ P[S \mid A].P[A \mid B, E].\ P[B \mid E].\ P[E]$$



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

Let's take the observed probability for the Burglary and earthquake component:

P(B = True) = 0.002, which is the probability of burglary.

P(B = False) = 0.998, which is the probability of no burglary.

P(E = True) = 0.001, which is the probability of a minor earthquake

P(E = False) = 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

Conditional Probability Table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Conditional probability table for David Calls

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

Conditional probability table for Sophia Calls:

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

$P(S, D, A, \neg B, \neg E) = P(S|A) * P(D|A) * P(A|\neg B \wedge \neg E) * P(\neg B) * P(\neg E)$.

$= 0.75 * 0.91 * 0.001 * 0.998 * 0.999$

$= 0.00068045$.

Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

The semantics of Bayesian Network

There are two ways to understand the semantics of the Bayesian network, which is given below:

1. To understand the network as the representation of the Joint probability distribution.

It is helpful to understand how to construct the network.

2. To understand the network as an encoding of a collection of conditional independence statements.

It is helpful in designing inference procedure.

3.2.4 Dempster-Shafer Theory

Q13. Explain about Dempster- shafer theory.

Ans : **(Imp.)**

DST is a mathematical theory of evidence based on belief functions and plausible reasoning. It is used to combine separate pieces of information (evidence) to calculate the probability of an event.

DST offers an alternative to traditional probabilistic theory for the mathematical representation of uncertainty.

DST can be regarded as, a more general approach to represent uncertainty than the Bayesian approach.

Bayesian methods are sometimes inappropriate

Example :

Let A represent the proposition "Moore is attractive". Then the axioms of probability insist that P(A) + P(¬ A) = 1.

- Now suppose that Andrew does not even know who "Moore" is, then
- We cannot say that Andrew believes the proposition if he has no idea what it means.
- Also, it is not fair to say that he disbelieves the proposition.
- It would therefore be meaningful to denote Andrew's belief B of

  B(A) and B(¬A) as both being 0.
- Certainty factors do not allow this.

Dempster-Shafer Model

The idea is to allocate a number between 0 and 1 to indicate a degree of belief on a proposal as in the probability framework.

However, it is not considered a probability but a belief mass. The distribution of masses is called basic belief assignment.

In other words, in this formalism a degree of belief (referred as mass) is represented as a belief function rather than a Bayesian probability distribution.

Example:

Belief assignment (continued from previous slide)

Suppose a system has five members, say five independent states, and exactly one of which is actual. If the original set is called S, |S| = 5, then the set of all subsets (the power set) is called $\mathbf{2^S}$.

If each possible subset as a binary vector (describing any member is present or not by writing 1 or 0), then 2^5 subsets are possible, ranging from the empty subset (0, 0, 0, 0, 0) to the "everything" subset (1, 1, 1, 1, 1).

The "empty" subset represents a "contradiction", which is not true in any state, and is thus assigned a mass of one ;

The remaining masses are normalized so that their total is 1.

The "everything" subset is labeled as "unknown"; it represents the state where all elements are present one , in the sense that you cannot tell which is actual.

Belief and Plausibility

Shafer's framework allows for belief about propositions to be represented as intervals, bounded by two values, belief (or support) and plausibility:

belief $\leq$ plausibility

Belief in a hypothesis is constituted by the sum of the masses of all sets enclosed by it (i.e. the sum of the masses of all subsets of the hypothesis). It is the amount of belief that directly supports a given hypothesis at least in part, forming a lower bound.

Plausibility is 1 minus the sum of the masses of all sets whose intersection with the hypothesis is empty. It is an upper bound on the possibility that the hypothesis could possibly happen, up to that value, because there is only so much evidence that contradicts that hypothesis.

Example :

A proposition say "the cat in the box is dead."

Suppose we have belief of 0.5 and plausibility of 0.8 for the proposition.

Example :

Suppose we have belief of 0.5 and plausibility of 0.8 for the proposition.

Evidence to state strongly, that proposition is true with confidence 0.5. Evidence contrary to hypothesis ("the cat is alive") has confidence 0.2.

Remaining mass of 0.3 (the gap between the 0.5 supporting evidence and the 0.2 contrary evidence) is "indeterminate," meaning that the cat could either be dead or alive. This interval represents the level of uncertainty based on the evidence in the system.

Hypothesis	Mass	Belief	Plausibility
Null (neither alive nor dead)	0	0	0
Alive	0.2	0.2	0.5
Dead	0.5	0.5	0.8
Either (alive or dead)	0.3	1.0	1.0

Null hypothesis is set to zero by definition, corresponds to "no solution". Orthogonal hypotheses "Alive" and "Dead" have probabilities of 0.2 and 0.5, respectively. This could correspond to "Live/Dead Cat Detector" signals, which have respective reliabilities of 0.2 and 0.5. All-encompassing "Either" hypothesis (simply acknowledges there is a cat in the box) picks up the slack so that the sum of the masses is 1. Belief for the "Alive" and "Dead" hypotheses matches their corresponding masses because they have no subsets;

- Belief for "Either" consists of the sum of all three masses (Either, Alive, and Dead) because "Alive" and "Dead" are each subsets of "Either".
- "Alive" plausibility is 1- m (Death) and "Dead" plausibility is 1- m (Alive). "Either" plausibility sums m(Alive) + m(Dead) + m(Either).
- Universal hypothesis ("Either") will always have 100% belief and plausibility; it acts as a checksum of sorts.

Short Question and Answers

1. Monotonic Reasoning

Ans :

In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.

To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.

Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.

2. Non-monotonic Reasoning

Ans :

In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.

Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.

Non-monotonic reasoning deals with incomplete and uncertain models.

"Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

3. Augmenting a problem solver in AI.

Ans :

Augmenting a problem solver refers to the process of adding additional capabilities or knowledge to an existing problem-solving system in order to improve its performance or extend its functionality.

Here are some examples of how a problem solver can be augmented in AI:

i) **Adding New Rules or Knowledge:** One way to augment a problem solver is by adding new rules or knowledge to its existing knowledge base. For example, a medical diagnosis system can be augmented by adding new rules that incorporate new medical research or treatment guidelines.

ii) **Integrating with External Data Sources:** Problem solvers can be augmented by integrating them with external data sources, such as databases or web services. For example, a financial decision-making system can be augmented by integrating it with real-time stock market data or news feeds.

iii) **Improving Reasoning Mechanisms:** Problem solvers can be augmented by improving their reasoning mechanisms, such as their search algorithms or optimization techniques. For example, a scheduling system can be augmented by improving its search algorithm to find more efficient schedules.

iv) **Incorporating Machine Learning Techniques:** Machine learning techniques, such as supervised or unsupervised learning, can be used to augment problem solvers by improving their accuracy or reducing their reliance on hand-coded rules. For example, a spam filter can be augmented by using machine learning techniques to improve its accuracy in identifying spam emails.

4. Depth First Search (DFS)

Ans :

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows:

Pick a starting node and push all its adjacent nodes into a stack.

5. What is uncertainty.

Ans :

Representation using first-order logic and propositional logic with certainty, which means we were

sure about the predicates. With this knowledge representation, we might write A'!B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

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

6. Need of Probabilistic Reasoning in AI.

Ans :

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.

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

- Bayes' rule
- Bayesian Statistics

7. Bayes' theorem.

Ans :

Bayes' theorem is also known as Bayes' rule, Bayes' law, or Bayesian reasoning, which determines the probability of an event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two random events.

Bayes' theorem was named after the British mathematician Thomas Bayes. The Bayesian inference is an application of Bayes' theorem, which is fundamental to Bayesian statistics.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

8. Dempster- shafer theory.

Ans :

DST is a mathematical theory of evidence based on belief functions and plausible reasoning. It is used to combine separate pieces of information (evidence) to calculate the probability of an event.

DST offers an alternative to traditional probabilistic theory for the mathematical representation of uncertainty.

DST can be regarded as, a more general approach to represent uncertainty than the Bayesian approach.

Bayesian methods are sometimes in appropriate

9. Various types of logics in non monotonic reasoning.

Ans :

i) Default Logic: This logic is used to express rules that are assumed to be true unless there is evidence to the contrary. Default logic allows for the retraction or revision of conclusions based on new evidence.

ii) Autoepistemic Logic: This logic is used to model situations in which an agent has incomplete knowledge about its own beliefs. It allows for the agent to update its beliefs based on new information.

iii) Circumscription: This logic is used to simplify the representation of a domain by defining the domain in terms of a smaller set of properties or constraints.

iv) Logic programming: This is a form of non-monotonic logic that is used to represent knowledge in the form of rules. Logic programming allows for the addition of new rules that can modify or retract previously held beliefs.

v) Probabilistic Logic: This logic is used to reason under uncertainty by assigning probabilities to propositions. Probabilistic logic allows for the updating of probabilities based on new evidence.

10. Breadth-First Search Algorithm.

Ans :

Breadth-First Search algorithm is a graph traversing technique, where you select a random initial node (source or root node) and start traversing the graph layer-wise in such a way that all the nodes and their respective children nodes are visited and explored.

Choose the Correct Answers

1. Which type of logic is used to model reasoning processes that involve incomplete or uncertain information? [d]
   a) Default logic b) Modal logic
   c) Autoepistemic logic d) Probabilistic logic
2. Which type of reasoning is used in deductive reasoning? [a]
   a) Monotonic reasoning b) Non-monotonic reasoning
   c) Both d) Neither
3. What is the main advantage of non-monotonic reasoning over monotonic reasoning? [b]
   a) Non-monotonic reasoning is faster than monotonic reasoning.
   b) Non-monotonic reasoning can handle incomplete or uncertain information and allows for the revision of conclusions based on new information.
   c) Non-monotonic reasoning is more precise than monotonic reasoning.
   d) Non-monotonic reasoning can only be used in inductive reasoning.
4. Which algorithm is used for traversing or searching a graph or tree data structure? [c]
   a) BFS (Breadth-First Search) b) DFS (Depth-First Search)
   c) Both A and B d) Neither A nor B
5. Which algorithm is used to find all the connected components in an undirected graph? [b]
   a) BFS (Breadth-First Search) b) DFS (Depth-First Search)
   c) Both A and B d) Neither A nor B
6. Which probabilistic reasoning method is used to update beliefs based on new evidence? [a]
   a) Bayesian inference b) Markov decision processes
   c) Dempster-Shafer theory d) Fuzzy logic
7. Which probability theory is commonly used in probabilistic reasoning? [c]
   a) Boolean probability theory b) Fuzzy probability theory
   c) Bayesian probability theory d) Possibility theory
8. Which of the following is not a component of a Rule-Based System? [d]
   a) Inference engine b) Knowledge base
   c) User interface d) Certainty factor
9. What is the purpose of a Bayesian Network? [a]
   a) To perform probabilistic inference and decision-making.
   b) To perform deductive reasoning.
   c) To perform inductive reasoning.
   d) To perform fuzzy reasoning.
10. What is a certainty factor range? [a]
    a) A range of values assigned to a rule to indicate its level of certainty.
    b) A range of values assigned to a fact to indicate its level of certainty.
    c) A range of values assigned to a conclusion to indicate its level of certainty.
    d) A range of values assigned to a system to indicate its level of accuracy.

Fill in the Blanks

1. In _____ reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base.
2. _____ reasoning deals with incomplete and uncertain models.
3. The technique of obtaining a sufficient explanation for known facts is known as _____ reasoning
4. _____ refers to the process of adding additional capabilities or knowledge to an existing problem-solving system in order to improve its performance or extend its functionality.
5. _____ algorithm is a graph traversing technique, where you select a random initial node and start traversing the graph layer-wise
6. The _____ algorithm is a recursive algorithm that uses the idea of backtracking.
7. _____ is a method of reasoning in which conclusions are drawn from a given set of data using probability theory.
8. The _____ is a numeric value which tells us about how likely an event or a statement is supposed to be true.
9. A _____ is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph
10. _____ is a theory of evidence that deals with uncertainty and ignorance in reasoning

ANSWERS

1. Monotonic
2. Non-monotonic
3. Abductive
4. Augmenting a problem solver
5. Breadth-First Search
6. DFS
7. Probabilistic reasoning
8. Certainty Factor (CF)
9. Bayesian network
10. Dempster-Shafer Theory

UNIT IV

Learning: What is Learning, Rote learning, Learning by taking advice, Learning in problem solving, learning from examples: Induction, Learning by Decision trees. Expert System: Representing and Using Domain Knowledge, Expert systems shells, Explanation, Knowledge Acquisition.

4.1 LEARNING

4.1.1 What is Learning

Q1. What is learning? What are various learning aspects in AI ?

Ans : **(Imp.)**

Meaning

Learning is the process of acquiring new knowledge, skills, or behavior through experience, study, or instruction. In the context of artificial intelligence, there are various types of learning that machines can perform:

1. Supervised Learning

In supervised learning, a machine learning algorithm is trained on a labeled dataset. The algorithm learns to make predictions based on input data, and the output is compared to the correct label. The algorithm adjusts its parameters to minimize the difference between its predictions and the correct labels.

2. Unsupervised Learning

In unsupervised learning, the algorithm is not provided with labeled data. The algorithm tries to find patterns and relationships in the data on its own. Unsupervised learning can be used for clustering, anomaly detection, and dimensionality reduction.

3. Reinforcement Learning

Reinforcement learning is a type of learning in which an agent learns to take actions in an environment to maximize a reward. The agent receives feedback in the form of rewards or penalties for each action it takes, and it learns to make decisions based on the expected reward.

4. Transfer Learning

Transfer learning involves using knowledge gained from one task to improve performance on a different task. For example, a model trained to recognize objects in photographs may be used as a starting point for a model that recognizes objects in videos.

5. Online Learning

Online learning involves updating the model with new data as it becomes available. This can be useful in scenarios where data is generated continuously over time.

6. Deep Learning

Deep learning is a type of machine learning that uses deep neural networks to learn representations of data. Deep learning models have shown state-of-the-art performance in a variety of tasks, including image recognition, speech recognition, and natural language processing.

These are some of the various learning aspects in AI, and each has its own strengths and weaknesses. The choice of learning approach depends on the specific problem and the available data.

Q2. Describe components of Learning System

Ans :

A learning system typically consists of several components, which work together to enable the machine to learn from data. Some of the key components of a learning system are:

1. Data

Data is the raw input that the machine learning algorithm uses to learn from. It can come in various forms, such as text, images, audio, or numerical data.

2. Feature Engineering

Feature engineering involves selecting or transforming the data into features that are meaningful for the task at hand. Feature engineering is a critical step in many machine learning tasks, as it can greatly affect the performance of the model.

3. Model Architecture

The model architecture determines how the data is transformed into predictions. This includes the type of algorithm used, the number of layers in the neural network, and the activation functions used in each layer.

4. Loss Function

The loss function measures how well the model's predictions match the true labels in the training data. The goal of the learning algorithm is to minimize the loss function.

5. Optimization Algorithm

The optimization algorithm is responsible for adjusting the model's parameters to minimize the loss function. Common optimization algorithms include gradient descent, stochastic gradient descent, and Adam.

6. Evaluation Metrics

Evaluation metrics measure how well the model performs on a test dataset. Common evaluation metrics include accuracy, precision, recall, and F1 score.

7. Deployment

Once the model is trained and evaluated, it can be deployed in a production environment. This may involve integrating the model with other systems, such as a web application or a mobile app.

These components work together in a learning system to enable the machine to learn from data and make predictions or decisions based on that data.

4.1.2 Rote Learning

Q3. What is rote learning? Explain about it.

Ans : **(Imp.)**

Meaning

The meaning of rote in 'rote learning' itself means learning by repetition. The process of repeating something over and over engages the short-term memory and allows us to quickly remember basic things like facts, dates, names, multiplication tables, etc. It differs from other forms of learning in that it doesn't require the learner to carefully think about something, and is rather dependent on the act of repetition itself.

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

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

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

Rote Learning Techniques

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

i) Read it Aloud

Read the text out loud with understanding. You can even try it before a mirror, ask a friend to listen to you, or read it out just under your breath. You can experiment with how slow or fast you want to read, how expressive you want to be, and internalize the rhythm of the text. Auditory learners will greatly benefit from this rote learning technique.

ii) Write it Down

Writing down the text information after reading is one of the best rote learning techniques. Doing so will help identify difficult passages and areas that need more practice. If you're preparing for a written exam, this kinesthetic rote technique will serve as a rehearsal and commit the information for easy retrieval.

iii) Sing it Out

There's a reason why songs commit to memory easier than text that is spoken simply and without any variation in pitch. So try putting the text that you want to learn with a melody that you like. Or, if you're feeling creative, come up with your own catchy tune to remember the text.

iv) Visualize

Humans are visual creatures and our brains are wired to remember things better with images. For every line and connected phrase, come up with ways to visualize it and remember it. The memory palace can be a useful trick for such rote learning techniques.

v) Free Association

Free association is one of the more interesting rote learning techniques, and a very useful way of remembering things quickly, especially if they are too messy for the traditional rote learning techniques. The main idea of this method is to combine new information with what you already know in a fun and personal way. For instance, if you're learning the 'Circle of Fifths' in music, you can associate each note to the numbers on the clock, one for each of the 12 notes in music. You are free to form your own associations as you see fit, as long as it helps you to recall the information.

Advantages

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

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

Disadvantages

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

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

Alternatives to Rote Learning

The meaning of rote doesn't mean much if it's done in isolation. The techniques make much more sense when they're combined with other forms of learning. Here are a few alternatives to rote learning that you should start incorporating in your studies today if you want to make the most of the learning process.

Associative Learning

This form of learning takes place when you learn an association between old and new information. The methods of classical and operant conditioning are at work here that pair different stimuli or events together.

Spaced Repetition

This form of learning is not too different from rote learning. The technique involves the use of flashcards wherein flashcards with new and difficult information are shown frequently to allow for the information to be internalized, while less difficult cards are shown less frequently. Doing so increases the rate of learning and recall without overwhelming the brain.

Active Learning

This is one form of learning that has taken off with the introduction of smart classes and technology. With active learning, students are actively and experientially involved in the learning process and their level of learning is reflected by how involved they are in the process.

Meaningful Learning

Meaningful learning is one of the deepest and most intimate forms of learning there are. Not only does it allow students to forge connections between old and new information, but also retain it through meaningful associations. The information can be retrieved from any starting point and can be applied to novel contexts as well.

4.1.3 Learning By Taking Advice

Q4. What is learning by advice? Explain.

Ans :

While performing experiments in lab you refer lab manual or teacher instructs you. After what is to be done and how you perform your experiment on your own. This is known as learning by advice.

Learning by advice requires more inference then rote learning. There are two main approaches for advice taking:

1. Automating all aspects of advice taking: In this method agent takes abstract high level advice from the teacher and then converts this advice into protocols which then help in guiding performance elements of the agent.
2. Using tools such as knowledge base editor and debuggers of advice taking: Here, an expert is a fundamental part of the learning agent. This type of advice taking is used to assist an expert to explain his expertise into detailed protocols. While storing protocols in the knowledge base, knowledge is transformed into an operational form.

4.1.4 Learning in Problem Solving

Q5. Explain how learning are used in problem solving?

Ans : **(Imp.)**

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

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

Problem-Solving Agent

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

Steps

- **Goal Formulation**

  It is the first and simplest step in problem-solving. It organizes the steps/sequence required to formulate one goal out of multiple goals as well as actions to achieve that goal. Goal formulation is based on the current situation and the agent's performance measure

- **Problem Formulation**

  It is the most important step of problem-solving which decides what actions should be taken to achieve the formulated goal. There are following five components involved in problem formulation:

- **Initial State**

  It is the starting state or initial step of the agent towards its goal.

- **Actions**

  It is the description of the possible actions available to the agent.

- **Transition Model**

  It describes what each action does.

- **Goal Test**

  It determines if the given state is a goal state.

- **Path Cost**

  It assigns a numeric cost to each path that follows the goal. The problem-solving agent selects a cost function, which reflects its performance measure.

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

Note

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

Search

It identifies all the best possible sequence of actions to reach the goal state from the current state. It takes a problem as an input and returns solution as its output.

Solution:

It finds the best algorithm out of various algorithms, which may be proven as the best optimal solution.

- **Execution**

  It executes the best optimal solution from the searching algorithms to reach the goal state from the current state.

Example Problems

Basically, there are two types of problem approaches:

- **Toy Problem**

  It is a concise and exact description of the problem which is used by the researchers to compare the performance of algorithms.

- **Real-world Problem**

  It is real-world based problems which require solutions. Unlike a toy problem, it does not depend on descriptions, but we can have a general formulation of the problem.

Some Toy Problems

- **8 Puzzle Problem**

  Here, we have a 3x3 matrix with movable tiles numbered from 1 to 8 with a blank space. The tile adjacent to the blank space can slide into that space. The objective is to reach a specified goal state similar to the goal state, as shown in the below figure.

- In the figure, our task is to convert the current state into goal state by sliding digits into the blank space.

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

The problem formulation is as follows:

- **States**

  It describes the location of each numbered tiles and the blank tile.

- **Initial State**

  We can start from any state as the initial state.

- **Actions**

  Here, actions of the blank space is defined, i.e., either left, right, up or down

- **Transition Model**

  It returns the resulting state as per the given state and actions.

- **Goal test**

  It identifies whether we have reached the correct goal-state.

➢ **Path Cost**

The path cost is the number of steps in the path where the cost of each step is 1.

Note

The 8-puzzle problem is a type of sliding-block problem which is used for testing new search algorithms in artificial intelligence.

➢ **8-queens problem**

The aim of this problem is to place eight queens on a chessboard in an order where no queen may attack another. A queen can attack other queens either diagonally or in same row and column.

From the following figure, we can understand the problem as well as its correct solution.

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

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

Incremental Formulation

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

Following steps are involved in this formulation:

➢ **States**

Arrangement of any 0 to 8 queens on the chessboard.

➢ **Initial State**

An empty chessboard

➢ **Actions**

Add a queen to any empty box.

➢ **Transition Model**

Returns the chessboard with the queen added in a box.

➢ **Goal Test**

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

➢ **Path Cost**

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

In this formulation, there is approximately 1.8×10^{14} possible sequence to investigate.

Complete-state Formulation

It starts with all the 8-queens on the chessboard and moves them around, saving from the attacks.

Following steps are involved in this formulation

➢ **States**

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

➢ **Actions**

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

➢ This formulation is better than the incremental formulation as it reduces the state space from 1.8×10^{14} to 2057, and it is easy to find the solutions.

Some Real-world Problems

➢ **Traveling salesperson problem(TSP)**

It is a touring problem where the salesman can visit each city only once. The objective is to find the shortest tour and sell-out the stuff in each city.

➢ **VLSI Layout Problem**

In this problem, millions of components and connections are positioned on a chip in order to minimize the area, circuit-delays, stray-capacitances, and maximizing the manufacturing yield.

The layout problem is split into two parts:

➢ **Cell Layout**

Here, the primitive components of the circuit are grouped into cells, each performing its specific function. Each cell has a fixed shape and size. The task is to place the cells on the chip without overlapping each other.

- **Channel Routing**

  It finds a specific route for each wire through the gaps between the cells.

- **Protein Design**

  The objective is to find a sequence of amino acids which will fold into 3D protein having a property to cure some disease.

4.2 LEARNING FROM EXAMPLES

4.2.1 Induction

Q6. Explain about inductive learning algorithm.

Ans : **(Imp.)**

Meaning

Inductive learning is a type of machine learning that involves learning general rules or patterns from a specific set of examples. In other words, an inductive learning algorithm attempts to generalize from specific examples to make predictions about new, unseen examples.

The main goal of inductive learning is to find a hypothesis that can accurately predict the output for new, unseen inputs. Inductive learning algorithms typically work in the following way:

1. **Data Collection**

   The first step is to collect a dataset of examples that represent the problem we want to solve. The dataset is typically split into two parts - a training set and a test set.

2. **Hypothesis Space**

   The next step is to define a hypothesis space - a set of possible hypotheses that could explain the input-output relationships in the data.

3. **Hypothesis Selection**

   The inductive learning algorithm searches through the hypothesis space to find the hypothesis that best fits the training data.

4. **Evaluation**

   Finally, the algorithm evaluates the hypothesis on the test data to determine how well it generalizes to new, unseen examples.

The most common inductive learning algorithms are decision tree algorithms and rule-based algorithms. In decision tree algorithms, the hypothesis space consists of a set of decision trees, where each node in the tree represents a decision based on a specific feature of the input. Rule-based algorithms, on the other hand, represent the hypothesis space as a set of rules that capture the patterns in the data.

The ila algorithm: general requirements at start of the algorithm:

1. list the examples in the form of a table 'T' where each row corresponds to an example and each column contains an attribute value.
2. create a set of m training examples, each example composed of k attributes and a class attribute with n possible decisions.
3. create a rule set, R, having the initial value false.
4. initially all rows in the table are unmarked.

Steps in the Algorithm

Step 1: divide the table 'T' containing m examples into n sub-tables (t1, t2,.....tn). One table for each possible value of the class attribute. (repeat steps 2-8 for each sub-table)

Step 2: Initialize the attribute combination count ' j ' = 1.

Step 3: For the sub-table on which work is going on, divide the attribute list into distinct combinations, each combination with 'j ' distinct attributes.

Step 4: For each combination of attributes, count the number of occurrences of attribute values that appear under the same combination of attributes in unmarked rows of the sub-table under consideration, and at the same time, not appears under the same combination of attributes of other sub-tables. Call the first combination with the maximum number of occurrences the max-combination ' MAX'.

Step 5: If 'MAX' = = null , increase ' j ' by 1 and go to Step 3.

Step 6: Mark all rows of the sub-table where working, in which the values of 'MAX' appear, as classified.

Step 7: Add a rule (IF attribute = "XYZ" –> THEN decision is YES/ NO) to R whose left-hand side will have attribute names of the 'MAX' with their values separated by AND, and its right-hand side contains the decision attribute value associated with the sub-table.

Step 8: If all rows are marked as classified, then move on to process another sub-table and go to Step 2. else, go to Step 4. If no sub-tables are available, exit with the set of rules obtained till then.

An example showing the use of ILA suppose an example set having attributes Place type, weather, location, decision and seven examples, our task is to generate a set of rules that under what condition what is the decision.

Example No.	Place type	weather	location	decision
I)	hilly	winter	kullu	Yes
II)	mountain	windy	Mumbai	No
III)	mountain	windy	Shimla	Yes
IV)	beach	windy	Mumbai	No
V)	beach	warm	goa	Yes
VI)	beach	windy	goa	No
VII)	beach	warm	Shimla	Yes

step 1 subset 1

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

subset 2

S.No	place type	weather	location	decision
5	mountain	windy	Mumbai	No
6	beach	windy	Mumbai	No
7	beach	windy	goa	No
step (2-8)				

- **at iteration 1** row 3 & 4 column weather is selected and row 3 & 4 are marked. the rule is added to R IF weather is warm then a decision is yes.

- **at iteration 2** row 1 column place type is selected and row 1 is marked. the rule is added to R IF place type is hilly then the decision is yes.
- **at iteration 3** row 2 column location is selected and row 2 is marked. the rule is added to R IF location is Shimla then the decision is yes.
- **at iteration 4** row 5&6 column location is selected and row 5&6 are marked. the rule is added to R IF location is Mumbai then a decision is no.
- **at iteration 5** row 7 column place type & the weather is selected and row 7 is marked. rule is added to R IF place type is beach AND weather is windy then the decision is no.

Finally we get the rule set :- Rule Set

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

4.2.2 Learning by Decision Trees

Q7. Explain about learning by decision trees technique.

Ans : **(Imp.)**

Learning by decision trees is a popular machine learning technique used to model complex decision-making problems. Decision trees are tree-like structures that are constructed from a set of training data. Each node in the tree represents a decision or a test on a particular feature or attribute of the input data, and each branch represents the outcome of the decision or test. The leaves of the tree represent the final decision or prediction made by the algorithm.

The process of learning by decision trees involves three steps: tree construction, pruning, and testing.

1. Tree Construction

During the tree construction phase, the algorithm searches through the training data to find the best feature or attribute to split the data. The goal is to find the feature that best separates the data into pure subsets or homogeneous subsets, with respect to the target variable being predicted. To measure the quality of a split, a splitting criterion is used. The most commonly used splitting criterion is the Gini index or information gain.

The algorithm repeats this process recursively for each subset until it either achieves a perfect classification or reaches a stopping criterion, such as a minimum number of instances required to create a leaf node or a maximum depth of the tree.

2. Pruning

After the tree is constructed, the algorithm may perform a pruning process to reduce overfitting. Overfitting occurs when the tree is too complex and captures noise in the training data, leading to poor generalization on new data. Pruning involves removing branches or subtrees that do not contribute to the accuracy of the tree. The most commonly used pruning technique is reduced error pruning.

3. Testing

The final step in learning by decision trees is testing. Once the tree is constructed and pruned, the algorithm evaluates its performance on a separate set of test data to estimate its generalization performance. The accuracy of the tree is measured using metrics such as precision, recall, F1 score, or accuracy.

Learning by decision trees has many advantages in AI, including interpretability, simplicity, and scalability. Decision trees are easy to understand and can be visualized, making them useful for explaining the reasoning behind the decision-making process. They are also fast to train and can handle large datasets with high dimensionality.

However, decision trees have some limitations. They can be sensitive to noisy data and can easily overfit the training data if not properly pruned. They also cannot model complex relationships between features and may require the use of ensemble methods such as random forests to improve their accuracy.

Here are some examples of how learning by decision trees can be applied in different domains:

1. Medical Diagnosis

Decision trees can be used to assist medical professionals in diagnosing diseases based on a patient's symptoms and medical history. The decision tree can be trained on a large dataset of patient records, and the resulting model can be used to predict the most likely diagnosis for a new patient. The tree can also be used to identify the most important symptoms or risk factors that contribute to a particular diagnosis.

2. Fraud Detection

Decision trees can be used to detect fraudulent behavior in financial transactions. The decision tree can be trained on a dataset of past transactions, and the resulting model can be used to identify suspicious transactions in real-time. The tree can also be used to identify the most important features or patterns that indicate fraud.

3. Customer Segmentation

Decision trees can be used to segment customers based on their demographic, psychographic, or behavioral characteristics. The decision tree can be trained on a dataset of customer information, and the resulting model can be used to group customers into different segments with similar attributes. The tree can also be used to identify the most important features or characteristics that distinguish one segment from another.

4. Predictive Maintenance

Decision trees can be used to predict when machines or equipment will fail based on their usage and maintenance history. The decision tree can be trained on a dataset of past failures and maintenance records, and the resulting model can be used to predict when a machine is likely to fail in the future. The tree can also be used to identify the most important factors or variables that contribute to machine failures.

5. Recommender Systems

Decision trees can be used to recommend products or services to customers based on their past behavior and preferences. The decision tree can be trained on a dataset of customer interactions with a website or app, and the resulting model can be used to suggest new products or services that the customer is likely to be interested in. The tree can also be used to identify the most important features or factors that influence a customer's purchasing behavior.

4.2.3 Expert System

4.2.4 Representing and using Domain Knowledge

Q8. What is knowledge? Explain, how knowledge is used in expert system.

Ans : **(Imp.)**

Expert system is built around a knowledge base module.

Expert system contains a formal representation of the information provided by the domain expert. This information may be in the form of problem-solving rules, procedures, or data intrinsic to the domain. To incorporate these information into the system, it is necessary to make use of one or more knowledge representation methods. Some of these methods are described here.

Transferring knowledge from the human expert to a computer is often the most difficult part of building an expert system.

The knowledge acquired from the human expert must be encoded in such a way that it remains a faithful representation of what the expert knows, and it can be manipulated by a computer.

Three common methods of knowledge representation evolved over the years are IF-THEN rules, Semantic networks and Frames.

The first two methods were illustrated in the earlier lecture slides on knowledge representation therefore just mentioned here. The frame based representation is described more.

1. IF-THEN rules

Human experts usually tend to think along :

condition $\Rightarrow$ action or Situation $\Rightarrow$ conclusion

Rules **"if-then"** are predominant form of encoding knowledge in expert systems. These are of the form :

If a1 , a2 , ... , an

Then b1 , b2 , ... , bn where

eachai is a condition or situation, and

each bi is an action or a conclusion.

2. Semantic Networks

In this scheme, knowledge is represented in terms of objects and relationships between objects.

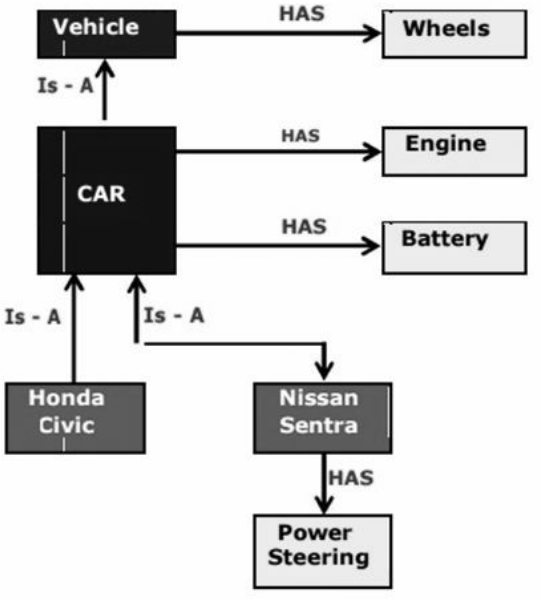
The objects are denoted as nodes of a graph. The relationship between two objects are denoted as a link between the corresponding two nodes.

The most common form of semantic networks uses the links between nodes to represent IS-A and HAS relationships between objects.

Example of Semantic Network

The figure below shows a car IS-A vehicle; a vehicle HAS wheels.

This kind of relationship establishes an inheritance hyhiera in the network, with the objects lower down in the network inheriting properties from the objects higher up.



3. Frames

In this technique, knowledge is decomposed into highly modular pieces called frames, which are generalized record structures. Knowledge consist of concepts, situations, attributes of concepts, relationships between concepts, and procedures to handle relationships as well as attribute values.

- Each concept may be represented as a separate frame.
- The attributes, the relationships between concepts, and the procedures are allotted to slots in a frame.
- The contents of a slot may be of any data type - numbers, strings, functions or procedures and so on.
- The frames may be linked to other frames, providing the same kind of inheritance as that provided by a semantic network.

A frame-based representation is ideally suited for objected-oriented programming techniques.

Example : Frame-based Representation of Knowledge

Two frames, their slots and the slots filled with data type are shown.

Frame	*Car*
Inheritance Slot	*Is-A*
Value	*Vehicle*
Attribute Slot	*Engine*
Value	*Vehicle*
Value	*1*
Value	
Attribute Slot	*Cylinders*
Value	*4*
Value	*6*
Value	*8*
Attribute Slot	*Doors*
Value	*2*
Value	*5*
Value	*4*

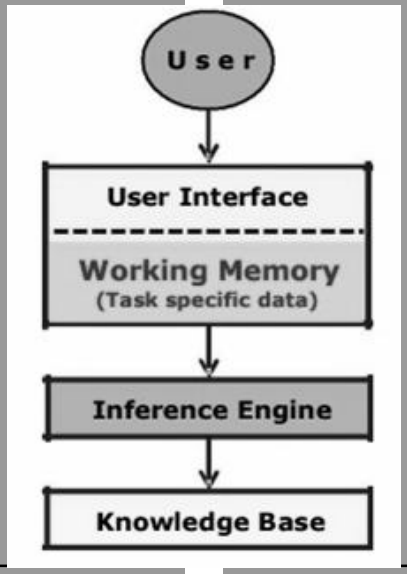
Frame	*Car*
Inheritance Slot	*Is-A*
Value	*Car*
Attribute Slot	*Make*
Value	*Honda*
Value	
Value	
Attribute Slot	*Year*
Value	*1989*
Value	
Value	
Attribute Slot	
Value	
Value	
Value	

Working Memory

Working memory refers to task-specific data for a problem. The contents of the working memory, changes with each problem situation. Consequently, it is the most dynamic component of an expert system, assuming that it is kept current.

- Every problem in a domain has some unique data associated with it.
- Data may consist of the set of conditions leading to the problem, its parameters and so on.
- Data specific to the problem needs to be input by the user at the time of using, means consulting the expert system. The Working memory is related to user interface.

Figure below shows how Working memory is closely related to user interface of the expert system.



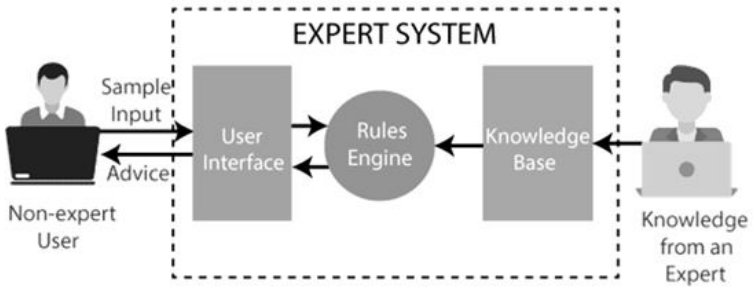
4.2.5 Expert Systems Shells

Q9. Define expert system? Explain the components of an expert systems.

Ans : **(Imp.)**

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

Below is the block diagram that represents the working of an expert system:



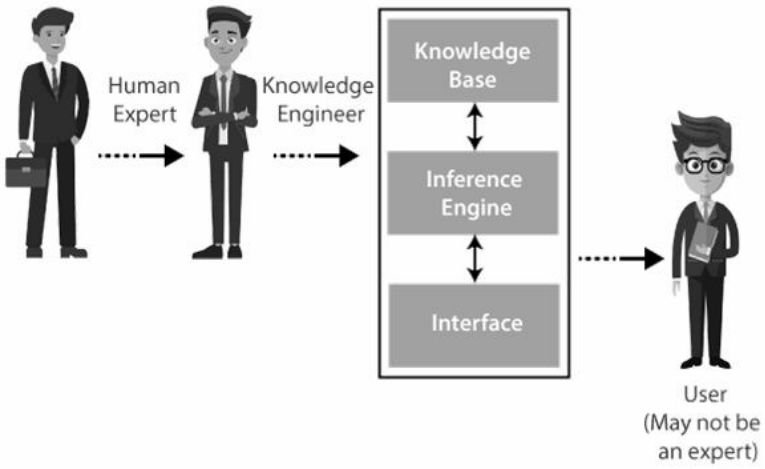
Below are some popular examples of the Expert System

- **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.
- **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.
- **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.
- **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.

Components of Expert System

An expert system mainly consists of three components:

- User Interface
- Inference Engine
- Knowledge Base



1. User Interface

With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.

2. Inference Engine(Rules of Engine)

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.

- With the help of an inference engine, the system extracts the knowledge from the knowledge base.

There are two types of inference engine:

- **Deterministic Inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on **facts** and **rules**.
- **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

- **Forward Chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- **Backward Chaining:** It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

3. Knowledge Base

- The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.
- It is similar to a database that contains information and rules of a particular domain or subject.
- One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base

- **Factual Knowledge**

The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

- **Heuristic Knowledge**

This knowledge is based on practice, the ability to guess, evaluation, and experiences.

- **Knowledge Representation**

It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

- **Knowledge Acquisitions**

It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

Q10. Explain about expert system shells.

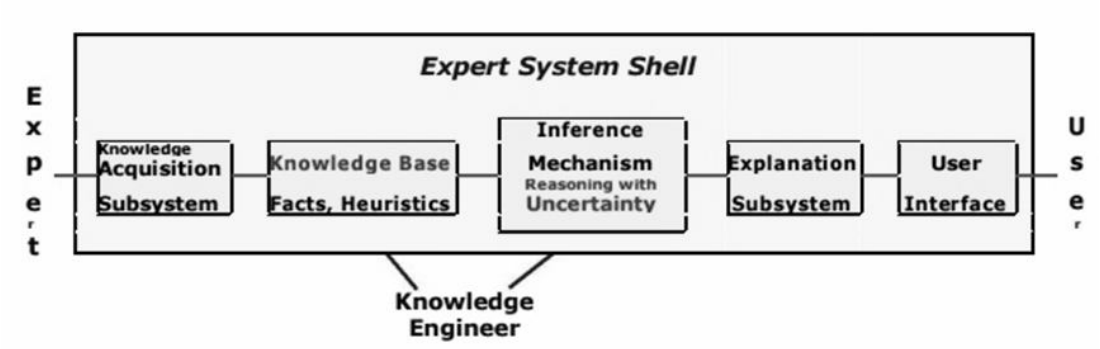
Ans :

Expert System Shells

An Expert system shell is a software development environment. It contains the basic components of expert systems. A shell is associated with a prescribed method for building applications by configuring and instantiating these components.

Shell components and Description

The generic components of a shell : the knowledge acquisition, the knowledge Base, the reasoning, the explanation and the user interface are shown below. The knowledge base and reasoning engine are the core components.



All these components are described in the next slide.

Knowledge Base

A store of factual and heuristic knowledge. Expert system tool provides one or more knowledge representation schemes for expressing knowledge about the application domain. Some tools use both Frames (objects) and IF-THEN rules. In PROLOG the knowledge is represented as logical statements.

Reasoning Engine

Inference mechanisms for manipulating the symbolic information and knowledge in the knowledge base form a line of reasoning in solving a problem. The inference mechanism can range from simple modus ponens backward chaining of IF-THEN rules to Case-Based reasoning.

Knowledge Acquisition Subsystem

A subsystem to help experts in build knowledge bases. However, collecting knowledge, needed to solve problems and build the knowledge base, is the biggest bottleneck in building expert systems.

Explanation Subsystem

A subsystem that explains the system's actions. The explanation can range from how the final or intermediate solutions were arrived at justifying the need for additional data.

User Interface

A means of communication with the user. The user interface is generally not a part of the expert system technology. It was not given much attention in the past. However, the user interface can make a critical difference in the peeived utility of an Expert system.

4.2.6 Explanation

Q11. Explain in detail about expert system usages.

Ans : **(Imp.)**

Expert systems are computer programs that mimic the decision-making ability of a human expert in a particular field. They use artificial intelligence (AI) techniques to provide advice, recommendations, and solutions to complex problems in various domains. Expert systems have been used in a wide range of applications, including:

1. Medical Diagnosis

Expert systems have been developed to assist medical professionals in diagnosing diseases and selecting appropriate treatment options. They can analyze patient symptoms, medical history, and test results to provide accurate diagnoses and recommend treatments.

2. Financial Analysis

Expert systems can analyze financial data, market trends, and investment opportunities to provide recommendations on investment strategies, stock selection, and risk management.

3. Manufacturing

Expert systems can be used to monitor and control complex manufacturing processes, such as chemical production, to optimize efficiency, reduce costs, and improve product quality.

4. Customer Service

Expert systems can be used to provide customer support and assistance in various industries, such as banking, insurance, and telecommunications. They can analyze customer queries and provide answers, troubleshoot problems, and suggest solutions.

5. Agriculture

Expert systems can assist farmers in making decisions related to crop management, irrigation, fertilization, and pest control. They can analyze environmental data, soil conditions, and weather forecasts to optimize crop yields and reduce losses.

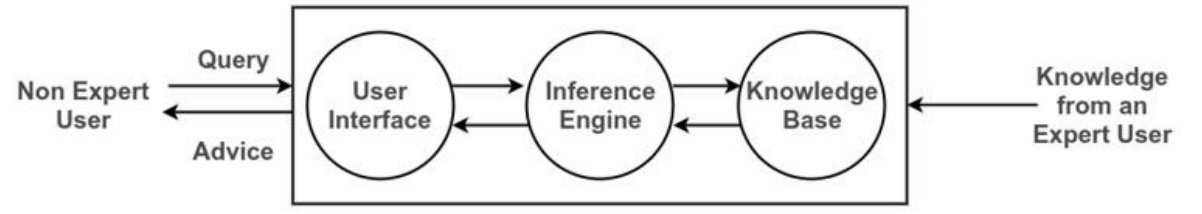
6. Legal Services

Expert systems can assist legal professionals in legal research, case analysis, and decision-making. They can analyze legal data, such as case law and statutes, to provide guidance on legal issues and strategies.

7. Education

Expert systems can be used to develop intelligent tutoring systems that provide personalized learning experiences to students. They can analyze student performance, identify learning gaps, and provide targeted feedback and recommendations.

Components of an Expert System



Architecture of an Expert System

- **Knowledge Base**

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

- **Inference Engine**

  The function of the inference engine is to fetch the relevant knowledge from the knowledge base, interpret it and to find a solution relevant to the user's problem. The inference engine acquires the rules from its knowledge base and applies them to the known facts to infer new facts. Inference engines can also include an explanation and debugging abilities.

- **Knowledge Acquisition and Learning Module**

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

- **User Interface**

  This module makes it possible for a non-expert user to interact with the expert system and find a solution to the problem.

- **Explanation Module**

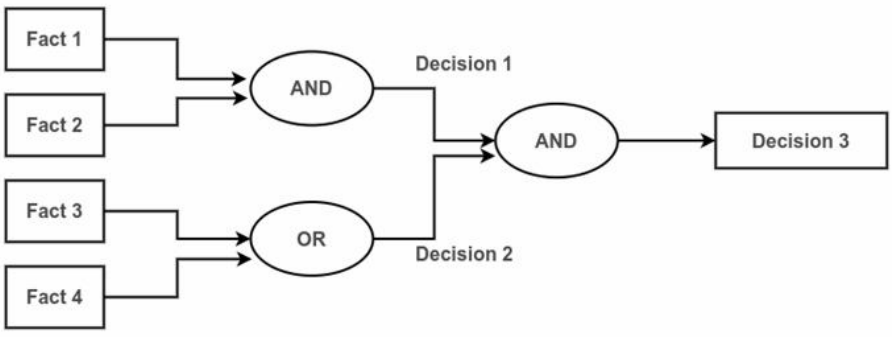
  This module helps the expert system to give the user an explanation about how the expert system reached a particular conclusion.

Q12. Explain forward chaining and backward chaining.

Ans : **(Imp.)**

- Forward chaining and backward chaining are two common inference strategies used in expert systems and rule-based systems.
- Forward chaining: In forward chaining, also known as data-driven reasoning, the inference engine begins with the available data and uses a set of rules to make inferences and draw conclusions. The inference engine evaluates the antecedents of each rule in the knowledge base against the available data, and if the antecedent is true, then the rule's consequent is executed. The output of one rule becomes the input for the next rule in the chain, and the process continues until a goal is achieved or no further rules can be executed.
- An example of forward chaining can be found in a medical diagnosis system. The system can begin with the patient's symptoms and evaluate a set of rules to diagnose the disease. If the patient has a fever, cough, and chest pain, the system may execute a rule that states that the patient may have pneumonia. The system may then execute further rules to confirm the diagnosis, such as checking for abnormal lung sounds, chest X-rays, or blood tests.

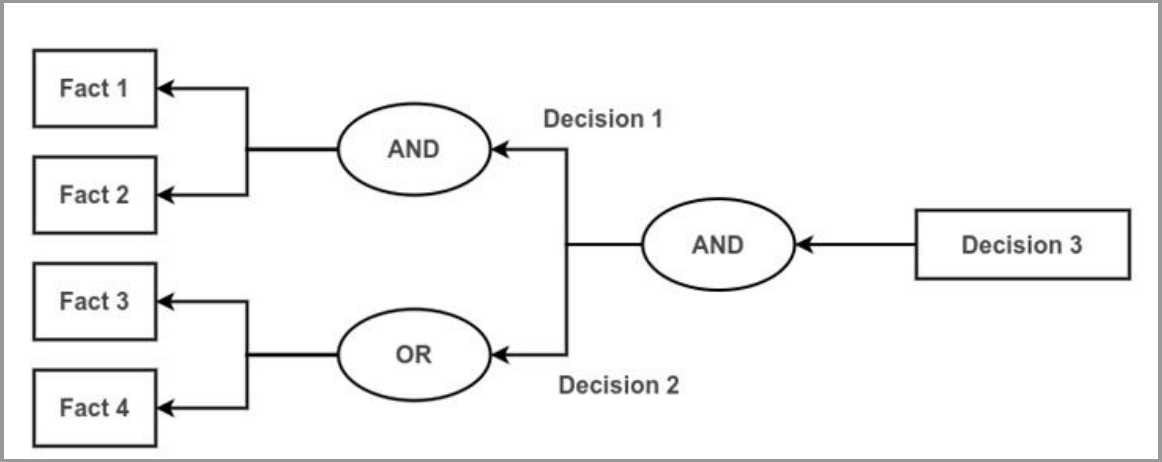
.



Forward Chaining

Backward chaining: In backward chaining, also known as goal-driven reasoning, the inference engine begins with a goal and works backward through the rules in the knowledge base to determine if the goal can be achieved. The inference engine evaluates the consequents of each rule in the knowledge base against the goal and, if the consequent matches the goal, the inference engine evaluates the antecedent of the rule to see if it is true. If the antecedent is not true, the inference engine will continue to work backward through the rules until it can find a rule that has a true antecedent, and then it will execute the rule's consequent.

An example of backward chaining can be found in a legal expert system. The system may begin with a legal question, such as "Is the defendant liable for damages?" The system would then work backward through the rules in the knowledge base to determine if the defendant's actions meet the criteria for liability. The system may check if the defendant had a duty of care, if the defendant breached that duty, and if the breach caused the plaintiff's damages.



Backward Chaining

4.2.7 Knowledge Acquisition

Q13. Define knowledge acquisition state in process.

Ans :

Knowledge acquisition (KA) is the process of obtaining knowledge from human experts or other sources, and transferring that knowledge into a knowledge base in order to build an expert system. KA is a critical part of the development of expert systems, as the knowledge base is the foundation of the system's intelligence.

Process

The KA process involves several steps, including:

1. Identification of Experts

The first step in KA is identifying subject matter experts who possess the knowledge required to build the expert system. These experts may be individuals with relevant domain knowledge or practitioners with hands-on experience.

2. Elicitation of Knowledge

The second step involves eliciting knowledge from the identified experts. This process can take several forms, including interviews, questionnaires, observations, and brainstorming sessions.

3. Knowledge Representation

Once the knowledge has been obtained, it must be organized and represented in a manner that can be processed by the expert system. The knowledge representation technique may vary depending on the nature of the knowledge, the type of expert system, and the desired functionality.

4. Validation and Verification

The next step is to validate and verify the knowledge that has been acquired. This involves testing the knowledge against real-world scenarios or test cases to ensure its accuracy and completeness.

5. Maintenance

Finally, the knowledge base must be updated and maintained as new knowledge is acquired, or as changes are made to the system's requirements or functionality.

Overall, KA is a critical aspect of expert system development, as the accuracy and completeness of the knowledge base directly impact the system's performance and effectiveness. KA requires expertise in both the subject matter domain and in the techniques of KA, including knowledge elicitation, knowledge representation, and knowledge validation.

Q14. Explain briefly about knowledge acquisition.

Ans : **(Imp.)**

There are several learning methods used in knowledge acquisition (KA), each with its own advantages and disadvantages. Some of the most commonly used methods are:

1. Interviews

Interviews involve direct questioning of subject matter experts to obtain their knowledge. Interviews can be structured or unstructured and can provide valuable insights into the expert's thought process.

2. Questionnaires

Questionnaires are pre-designed sets of questions used to elicit knowledge from experts. They are often used when the number of experts is large or when the experts are geographically dispersed.

3. Observations

Observations involve observing experts as they perform tasks or interact with systems. This method can provide valuable insights into the expert's behavior, decision-making process, and problem-solving strategies.

4. Case-based Reasoning

Case-based reasoning involves capturing knowledge in the form of cases or examples. The expert system can then use these cases to reason about new situations.

5. Neural Networks

Neural networks are machine learning algorithms that can be used to learn patterns in data. They can be used to learn from large datasets, such as sensor data or historical records.

6. Rule Induction

Rule induction involves automatically generating rules from data. This method is often used when the knowledge to be acquired is complex or difficult to express in a simple form.

7. Genetic Algorithms

Genetic algorithms are optimization algorithms inspired by the process of natural selection. They can be used to evolve solutions to problems or to optimize parameters in a system.

Short Question and Answers

1. What is learning?

Ans :

Learning is the process of acquiring new knowledge, skills, or behavior through experience, study, or instruction. In the context of artificial intelligence, there are various types of learning that machines can perform:

i) Supervised Learning

In supervised learning, a machine learning algorithm is trained on a labeled dataset. The algorithm learns to make predictions based on input data, and the output is compared to the correct label. The algorithm adjusts its parameters to minimize the difference between its predictions and the correct labels.

ii) Unsupervised Learning

In unsupervised learning, the algorithm is not provided with labeled data. The algorithm tries to find patterns and relationships in the data on its own. Unsupervised learning can be used for clustering, anomaly detection, and dimensionality reduction.

2. What is rote learning.

Ans :

The meaning of rote in 'rote learning' itself means learning by repetition. The process of repeating something over and over engages the short-term memory and allows us to quickly remember basic things like facts, dates, names, multiplication tables, etc. It differs from other forms of learning in that it doesn't require the learner to carefully think about something, and is rather dependent on the act of repetition itself.

3. Learning by advice.

Ans :

While performing experiments in lab you refer lab manual or teacher instructs you. After what is to be done and how you perform your experiment on your own. This is known as learning by advice.

Learning by advice requires more inference then rote learning. There are two main approaches for advice taking:

i) Automating all aspects of advice taking: In this method agent takes abstract high level advice from the teacher and then converts this advice into protocols which then help in guiding performance elements of the agent.

ii) Using tools such as knowledge base editor and debuggers of advice taking: Here, an expert is a fundamental part of the learning agent. This type of advice taking is used to assist an expert to explain his expertise into detailed protocols. While storing protocols in the knowledge base, knowledge is transformed into an operational form.

4. Inductive learning algorithm.

Ans :

Inductive learning is a type of machine learning that involves learning general rules or patterns from a specific set of examples. In other words, an inductive learning algorithm attempts to generalize from specific examples to make predictions about new, unseen examples.

The main goal of inductive learning is to find a hypothesis that can accurately predict the output for new, unseen inputs. Inductive learning algorithms typically work in the following way:

i) Data Collection

The first step is to collect a dataset of examples that represent the problem we want to solve. The dataset is typically split into two parts - a training set and a test set.

ii) Hypothesis Space

The next step is to define a hypothesis space - a set of possible hypotheses that could explain the input-output relationships in the data.

iii) Hypothesis Selection

The inductive learning algorithm searches through the hypothesis space to find the hypothesis that best fits the training data.

5. Decision trees technique.

Ans :

Learning by decision trees is a popular machine learning technique used to model complex decision-making problems. Decision trees are tree-like structures that are constructed from a set of training data. Each node in the tree represents a decision or a test on a particular feature or attribute of the input data, and each branch represents the outcome of the decision or test. The leaves of the tree represent the final decision or prediction made by the algorithm.

6. Expert systems.

Ans :

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

7. Components of Knowledge Base

Ans :

- **Factual Knowledge**

  The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

- **Heuristic Knowledge**

  This knowledge is based on practice, the ability to guess, evaluation, and experiences.

- **Knowledge Representation**

  It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

- **Knowledge Acquisitions**

  It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

8. Expert System Shells

Ans :

An Expert system shell is a software development environment. It contains the basic components of expert systems. A shell is associated with a prescribed method for building applications by configuring and instantiating these components.

9. Expert system usages.

Ans :

Expert systems are computer programs that mimic the decision-making ability of a human expert in a particular field. They use artificial intelligence (AI) techniques to provide advice, recommendations, and solutions to complex problems in various domains. Expert systems have been used in a wide range of applications, including:

i) Medical Diagnosis

Expert systems have been developed to assist medical professionals in diagnosing diseases and selecting appropriate treatment options. They can analyze patient symptoms, medical history, and test results to provide accurate diagnoses and recommend treatments.

ii) Financial Analysis

Expert systems can analyze financial data, market trends, and investment opportunities to provide recommendations on investment strategies, stock selection, and risk management.

iii) Manufacturing

Expert systems can be used to monitor and control complex manufacturing processes, such as chemical production, to optimize efficiency, reduce costs, and improve product quality.

iv) Customer Service

Expert systems can be used to provide customer support and assistance in various industries, such as banking, insurance, and telecommunications. They can analyze customer queries and provide answers, troubleshoot problems, and suggest solutions.

10. Components of an Expert System

Ans :

- **Knowledge Base**

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

- **Inference Engine**

  The function of the inference engine is to fetch the relevant knowledge from the knowledge base, interpret it and to find a solution relevant to the user's problem. The inference engine acquires the rules from its knowledge base and applies them to the known facts to infer new facts. Inference engines can also include an explanation and debugging abilities.

- **Knowledge Acquisition and Learning Module**

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

- **User Interface**

  This module makes it possible for a non-expert user to interact with the expert system and find a solution to the problem.

Choose the Correct Answers

1. Which type of learning requires labeled data for training? [a]
   (a) Supervised Learning (b) Unsupervised Learning
   (c) Reinforcement Learning (d) Transfer Learning
2. Which type of learning involves using knowledge gained from one task to improve performance on a different task? [d]
   (a) Supervised Learning (b) Unsupervised Learning
   (c) Reinforcement Learning (d) Transfer Learning
3. What is the main goal of inductive learning? [b]
   (a) To learn specific examples from general rules. (b) To learn general rules from specific examples.
   (c) To memorize the training data (d) To overfit the training data
4. Which of the following is a commonly used splitting criterion for decision trees? [b]
   (a) Mean squared error (b) Gini index
   (c) K-nearest neighbors (d) Linear regression
5. Which of the following domains is NOT suitable for learning by decision trees? [d]
   (a) Medical diagnosis (b) Fraud detection
   (c) Customer segmentation (d) Image recognition
6. What is the role of the inference engine in an expert system? [c]
   (a) To store and retrieve data
   (b) To present the results to the user
   (c) To apply the rules to the data and make decisions
   (d) To generate new rules based on the data
7. Which of the following is NOT a typical component of an expert system shell? [d]
   (a) Knowledge acquisition subsystem (b) Inference engine
   (c) User interface (d) Expert knowledge base
8. Which of the following is an example of an expert system shell? [c]
   (a) Microsoft Excel (b) MATLAB
   (c) CLIPS (d) Adobe Photoshop
9. In which system would backward chaining be most appropriate? [a]
   (a) Medical diagnosis system (b) Traffic control system
   (c) Manufacturing process control system (d) None of the above
10. Which of the following is not a method of knowledge acquisition? [d]
    (a) Interviews (b) Observations
    (c) Experimentation (d) Testing

Fill in the Blanks

1. The process of repeating something over and over engages the short-term memory and allows us to quickly remember basic things like facts, dates, names, multiplication tables, etc is called _____.
2. The _____ are known as the simplest agents because they directly map states into actions.
3. The 8-puzzle problem is a type of _____ problem which is used for testing.
4. The _____ is called as a touring problem
5. _____ is a popular machine learning technique used to model complex decision-making problems
6. A system that mimics the decision-making ability of a human expert is called _____
7. The set of rules used to make decisions is called as _____
8. A type of software used to develop expert systems is called _____
9. In _____ inference strategy does the system begin with available data and work forward through the rules to achieve a goal?
10. The process of acquiring knowledge from human experts is called _____

ANSWERS

1. Rote learning
2. Reflex agents
3. Sliding-block
4. Travelling sales man problem
5. Learning by decision trees
6. Expert system
7. Knowledge base
8. Expert system shell
9. Forward chaining
10. Knowledge acquisition

UNIT V

Perception and Action: Real Time Search, Vision, Speech Recognition, ACTION: Navigation, Manipulation, Robot architectures. Natural Language Processing: Introduction, Syntactic Processing, Semantic Analysis, Statistical NLP, Spell Checking.

5.1 Perception And Action

5.1.1 Real Time Search

Q1. Explain the concept of real time search?

Ans : **(Imp.)**

Real-time search is a type of search algorithm used in artificial intelligence and computer science to solve problems where the solution needs to be found in real-time or near real-time. In real-time search, the agent continuously receives new information about the state of the environment and updates its search accordingly.

Real-time search algorithms are used in various applications such as video games, robotics, and decision support systems. They are designed to efficiently explore the search space and find a solution within a limited amount of time.

One common approach used in real-time search is called depth-limited search, which limits the depth of the search tree to a fixed value. The search then proceeds by exploring the search tree to the specified depth and then evaluating the nodes at that depth. The agent then selects the best path based on the evaluation function.

Another approach used in real-time search is called iterative deepening depth-first search. In this approach, the agent starts with a depth limit of one and gradually increases it until a solution is found or a maximum depth is reached. This approach has the advantage of finding solutions quickly and can be applied to large search spaces.

Real-time search algorithms often employ heuristics to guide the search towards promising paths in the search tree. Heuristics can be used to estimate the cost of reaching the goal state from each node, which helps the agent to prioritize its search.

Real-time search is a useful technique for solving problems that require quick responses, and it is often used in real-time decision-making systems. However, it has some limitations, such as the inability to handle uncertainty and the need for a well-defined search space.

5.1.2 Vision

Q2. Write about vision in AI.

Ans :

In the context of AI, a vision refers to the ability of an AI system to perceive and understand visual information from the world around it, similar to how humans process visual information. Vision is a crucial component of many AI applications, including autonomous vehicles, robotics, and computer vision.

Computer vision is a subfield of AI that focuses on the development of algorithms and techniques that enable machines to interpret and understand visual information. Computer vision algorithms are designed to extract information from visual data, such as images and videos, and use that information to make decisions or perform tasks.

Computer vision algorithms can be trained using machine learning techniques, such as deep learning, to recognize patterns and features in visual data. For example, a computer vision system may be trained to identify objects in an image or detect changes in a video feed.

There are many applications of computer vision in AI, including:

1. Object Recognition

Identifying objects in an image or video feed, such as vehicles, pedestrians, or road signs.

2. Facial Recognition

Identifying and recognizing individuals based on their facial features.

3. Image and Video Search

Searching for specific images or videos based on visual features, such as color or shape.

4. Medical Imaging

Analyzing medical images to identify abnormalities or diagnose diseases.

5. Robotics

Enabling robots to perceive and interact with their environment using visual information.

6. Autonomous Vehicles

Computer vision is used to enable self-driving cars to detect and recognize objects such as other cars, pedestrians, and traffic signs.

7. Augmented Reality

Computer vision is used to track the position and movement of objects in real-time, allowing for the overlay of digital information on top of the real world.

8. Robotics

Computer vision is used to enable robots to perceive and interact with their environment, allowing them to perform tasks such as object detection, grasping, and manipulation.

9. Medical Imaging

Computer vision is used to analyze medical images such as X-rays, CT scans, and MRIs to help diagnose diseases and identify abnormalities.

10. Surveillance

Computer vision is used to monitor and analyze video feeds from security cameras to detect suspicious activity and identify individuals.

11. Agriculture

Computer vision is used to monitor crop growth and identify plant diseases, enabling farmers to optimize their yield and minimize the use of pesticides.

12. Retail

Computer vision is used to enable smart checkout systems that automatically detect and scan items, reducing checkout times and improving the shopping experience.

13. Entertainment

Computer vision is used to enable facial recognition and emotion detection in virtual and augmented reality applications, improving user engagement and immersion.

Vision is a critical component of AI and computer vision plays a vital role in enabling machines to perceive and understand visual information from the world around them.

Q3. Explain the vision algorithms used in AI.

Ans :

Here are some common vision algorithms used in AI:

1. Convolutional Neural Networks (CNNs)

CNNs are a type of deep learning algorithm that are widely used for image and video recognition. They work by extracting features from the input image or video using a series of convolutional layers, and then using these features to classify the image or video.

2. Object Detection

Object detection algorithms are used to identify and locate objects within an image or video. They typically use a combination of feature extraction and classification techniques, such as CNNs, to identify the presence and location of objects.

3. Semantic Segmentation

Semantic segmentation algorithms are used to classify each pixel in an image into a specific object class. These algorithms typically use CNNs to extract features from the image and then assign each pixel a class label.

4. Optical Flow

Optical flow algorithms are used to track the motion of objects within a video. They work by analyzing the movement of pixels between consecutive frames of the video and estimating the motion vectors of each pixel.

5. Generative Adversarial Networks (GANs)

GANs are a type of deep learning algorithm that are used to generate new images or videos that are similar to a given set of input images or videos. They work by training two neural networks, a generator and a discriminator, to produce and evaluate new images or videos

5.1.3 Speech Recognition

Q4. What is speech recognition AI?

Ans :

Meaning

Speech recognition enables computers, applications and software to comprehend and translate human speech data into text, for business solutions. The speech recognition model works by using artificial intelligence (AI) to analyse your voice and language, identify by learning the words you are saying, and then output those words with transcription accuracy as model content or text data on a screen.

Q5. How does speech recognition AI work?

Ans :

Speech recognition or voice recognition is a complex process that involves audio accuracy over several steps and data or language solutions, including:

- Recognizing the words, models and content in the user's speech or audio. This business accuracy step requires training the model to identify each word in your vocabulary or audio cloud.
- Converting those audios and language into text. This step involves converting recognized audios into letters or numbers (called phonemes) so that other parts of the AI software solutions system can process those models.
- Determining what was said. Next, AI looks at which content and words were spoken most often and how frequently they were used together to determine their meaning (this process is known as "predictive modelling").
- Parsing out commands from the rest of your speech or audio content (also known as disambiguation).

Speech Recognition AI and Natural Language Processing

Natural Language Processing is a part of artificial intelligence that involves analyzing data related to natural language and converting it into a machine-comprehendible format. Speech recognition and AI play a pivotal role in NLPs in improving the accuracy and efficiency of human language recognition.

A lot of businesses now include speech-to-text software or speech recognitionAI to enhance their business applications and improve customer experience. By using speech recognition AI and natural language processing together, companies can transcribe calls, meetings etc. Giant companies like Apple, Google, and Amazon are leveraging AI-based speech or voice recognition applications to provide a flawless customer experience.

Q6. State the applications of speech recognition AI.

Ans :

Speech recognition AI is being used as business solutions in many industries and applications. From ATMs to call centres and voice-activated audio content assistants, AI is helping people interact with technology and software more naturally with better data transcription accuracy than ever before.

(i) Call Centers

Speech recognition is one of the most popular uses of speech AI in call centers. This technology allows you to listen to what customers are saying and then use that information via cloud models to respond appropriately.

You can also use speech recognition technology for voice or audio biometrics, which means using voice patterns as proof of identity or authorization for access solutions or services without relying on passwords or other traditional methods or models like fingerprints or eye scans. This can eliminate business issues like forgotten passwords or compromised security codes in favor of something more secure: your voice!

(ii) Banking

Banking and financial institutions are using speech AI applications to help customers with their business queries. For example, you can ask a bank about your account balance or the current interest rate on your savings account. This cuts down on the time it takes for customer service representatives to answer questions they would typically have to research and look at cloud data, which means quicker response times and better customer service.

(iii) Telecommunications

Speech-enabled AI is a technology that's gaining traction in the telecommunications industry. Speech recognition technology models enable calls to be analysed and managed more efficiently. This allows agents to focus on their highest-value tasks to deliver better.

(iv) Customer Service

Customers can now interact with businesses in real-time 24/7 via voice transcription solutions or text messaging applications, which makes them feel more connected with the company and improves their overall experience.

(v) Healthcare

Speech AI is a learning technology used in many different areas as transcription solutions. Healthcare is one of the most important, as it can help doctors and nurses better care for their patients. Voice-activated devices use learning models that allow patients to communicate with doctors, nurses, and other health care professionals without using their hands or typing on a keyboard.

Doctors can use speech recognition AI via cloud data to help patients understand their feelings and why they feel that way. It's much easier than having them read through a brochure or pamphlet—and it's more engaging. Speech AI can also take down patient histories and help with medical transcriptions.

(vi) Media and Marketing

Tools such as dictation software use speech recognition and AI to help users type or write more in much less time. Roughly speaking, copywriters and content writers can transcribe as much as 3000-4000 words in as less as half an hour on an average.

Accuracy, though, is a factor. These tools don't guarantee 100% foolproof transcription. Still, they are extremely beneficial in helping media and marketing people in composing their first drafts.

Q7. Discribe challenges in working with speech recognition AI.

Ans : **(Imp.)**

Challenges in Working with Speech Recognition AI

- There are many challenges in working with speech AI. For example, both technology and cloud are new and developing rapidly. As a result, it isn't easy to make accurate predictions about how long it will take for a company to build its speech-enabled product.
- Another challenge with speech AI is getting the right tools to analyse your data. Most people need access to this technology or cloud, so finding the right tool for your requirements may take time and effort.
- You must use the correct language and syntax when creating your algorithms on cloud. This can be difficult because it requires understanding how computers and humans communicate. Speech recognition still needs improvement, and it can be difficult for computers to understand every word you say.

- If you use speech recognition software, you will need to train it on your voice before it can understand what you're saying. This can take a long time and requires careful study of how your voice sounds different from other people's.
- The other concern is that there are privacy laws surrounding medical records. These laws vary from state to state, so you'll need to check with your jurisdiction before implementing speech AI technology.
- Educating your staff on the technology and how it works is important if you decide to use speech AI. This will help them understand what they're recording and why they're recording it.

Q8. Explain brifly about various speech recognition techniques.

Ans : **(Imp.)**

There are several techniques that can be used for speech recognition, some of which include:

1. Hidden Markov Models (HMM)

This technique models the probability of a sequence of acoustic features corresponding to speech sounds. It is a statistical model that uses a sequence of hidden states to represent the acoustic properties of speech. HMM is widely used in speech recognition systems due to its effectiveness.

2. Artificial Neural Networks (ANN)

ANNs are another popular technique used in speech recognition. ANNs are designed to learn from examples and can be used to classify speech sounds. These networks consist of layers of interconnected nodes that process the input data and produce the output.

3. Gaussian Mixture Models (GMM)

This technique is based on the assumption that the probability density function of the acoustic features can be modeled as a mixture of Gaussian distributions. GMMs are used to model the different speech sounds and their variations.

4. Deep Learning

Deep learning is a type of machine learning that uses deep neural networks to learn from large amounts of data. It has been shown to be effective in speech recognition tasks, especially when combined with other techniques such as HMM and GMM.

5. Dynamic Time Warping (DTW)

DTW is a technique used for matching sequences of speech sounds. It is often used in speech recognition systems to compare an unknown speech signal with a pre-defined set of speech templates.

6. Convolutional Neural Networks (CNN)

CNNs are a type of neural network that are commonly used in image processing, but have also been used in speech recognition. They can be used to extract relevant features from the speech signal and classify speech sounds.

These techniques can be used individually or in combination to improve the accuracy of speech recognition systems.

5.2 ACTION

5.2.1 Navigation

Q9. What is navigation? Explain the applications of navigation in AI.

Ans : **(Imp.)**

Meaning

Navigation in artificial intelligence refers to the ability of an AI system to plan and execute a path to a goal location in a given environment. Navigation is a critical component of many AI applications, including autonomous vehicles, robotics, and virtual assistants.

Navigation algorithms typically involve three main components: perception, planning, and control. Perception involves using sensors to perceive the environment and generate a map of the surroundings. Planning involves using the map and other information, such as the current location and goal location, to plan a

path to the destination. Control involves executing the planned path and adjusting the path as needed based on new sensor data.

Applications

Here are some uses and applications of navigation in AI:

1. Autonomous Vehicles

Navigation is a critical component of self-driving cars, which use sensors such as cameras, lidar, and radar to perceive the environment and plan a safe and efficient route to the destination.

2. Robotics

Navigation is essential for robots to navigate through unknown environments and perform tasks such as object manipulation and delivery.

3. Drones

Navigation is used to enable drones to navigate through complex and dynamic environments, such as urban areas or disaster zones, to perform tasks such as search and rescue, delivery, and surveillance.

4. Virtual Assistants

Navigation is used to enable virtual assistants such as Siri or Alexa to understand user commands and provide relevant information or perform actions.

5. Video Games

Navigation is used to enable characters and objects in video games to navigate through the game world, interact with the environment, and achieve game objectives.

6. Augmented Reality

Navigation is used to enable virtual objects to interact with the real world, such as by overlaying digital information on real-world objects.

In summary, navigation is a critical component of many AI applications, including autonomous vehicles, robotics, and virtual assistants. Navigation algorithms involve perceiving the environment, planning a path to the destination, and executing the planned path, and are essential for enabling machines to move and interact with the world around them.

Q10. Explain various forms of navigation algorithms.

Ans :

Here are some common navigation algorithms used in AI:

1. A* algorithm

A* is a popular pathfinding algorithm that is used to find the shortest path between two points in a graph or grid-based environment. The algorithm uses a heuristic function to estimate the cost of moving from one node to another, and chooses the path with the lowest estimated cost.

2. Dijkstra's Algorithm

Dijkstra's algorithm is another pathfinding algorithm that is used to find the shortest path between two points in a graph-based environment. The algorithm works by assigning a tentative distance to each node in the graph and iteratively updating the distances until the shortest path is found.

3. Probabilistic Roadmap (PRM)

PRM is a sampling-based algorithm used in robotics to plan collision-free paths in high-dimensional spaces. The algorithm works by randomly sampling the configuration space of the robot and constructing a roadmap of collision-free paths between the samples.

4. Rapidly-exploring Random Tree (RRT)

RRT is another sampling-based algorithm used in robotics to plan collision-free paths in high-dimensional spaces. The algorithm works by iteratively growing a tree of random configurations and connecting the tree to the goal configuration until a path is found.

5. Dynamic Window Approach (DWA)

DWA is a real-time motion planning algorithm used in robotics to navigate through dynamic environments. The algorithm works by evaluating the robot's current state, predicting its future state, and selecting a velocity command that avoids obstacles and minimizes the distance to the goal.

5.2.2 Manipulation

Q11. Explain about manipulation technique in AI.

Ans : **(Imp.)**

Manipulation in AI refers to the ability of an AI system to manipulate physical objects in the real world. Manipulation is a critical component of many AI applications, including robotics, manufacturing, and logistics.

AI systems that are capable of manipulation require advanced algorithms that can process visual and sensor data to detect and recognize objects, plan and execute actions, and learn from experience. Manipulation algorithms typically involve three main components: perception, planning, and control.

Perception involves using sensors such as cameras, lidar, or tactile sensors to perceive the environment and generate a model of the surrounding objects. This can involve object recognition, object tracking, and object pose estimation.

Planning involves using the model of the environment to plan a sequence of actions that will enable the AI system to manipulate the objects. This can involve tasks such as grasping, pushing, pulling, or placing objects.

Control involves executing the planned actions and adjusting them as needed based on new sensor data. This can involve feedback control and closed-loop control techniques to ensure precise and accurate manipulation of the objects.

Applications

Here are some uses and applications of manipulation in AI:

1. Robotics

Manipulation is a critical component of robots used in manufacturing, logistics, and healthcare. Robots are used to perform tasks such as pick-and-place, assembly, packaging, and inspection.

2. Warehousing and Logistics

Manipulation is used to automate tasks such as sorting, packing, and shipping in warehouses and distribution centers.

3. Agriculture

Manipulation is used in precision agriculture to perform tasks such as planting, pruning, and harvesting.

4. Healthcare

Manipulation is used in surgical robots to perform minimally invasive procedures with high precision and accuracy.

In summary, manipulation is a critical component of many AI applications, including robotics, manufacturing, and logistics. Manipulation algorithms involve perceiving the environment, planning a sequence of actions, and executing the planned actions, and are essential for enabling machines to manipulate objects in the real world.

Manipulation Algorithms

Manipulation algorithms are used in AI systems to enable machines to manipulate physical objects in the real world. These algorithms typically involve several steps, including perception, planning, and control.

Here is an overview of the three main components of manipulation algorithms:

1. Perception

Perception involves using sensors such as cameras, lidar, or tactile sensors to perceive the environment and generate a model of the surrounding objects. Perception algorithms can involve object recognition, object tracking, and object pose estimation.

2. Planning

Planning involves using the model of the environment generated by the perception algorithms to plan a sequence of actions that will enable the AI system to manipulate the objects. Planning algorithms can involve tasks such as grasping, pushing, pulling, or placing objects. These algorithms must take into account the shape, size, and weight of the objects, as well as any obstacles or other constraints in the environment.

3. Control

Control involves executing the planned actions and adjusting them as needed based on new sensor data. Control algorithms can involve feedback control and closed-loop control techniques to ensure precise and accurate manipulation of the objects. These algorithms must be able to handle uncertainty in the environment, such as unexpected obstacles or changes in the position of the objects.

Some common manipulation algorithms used in AI systems include:

1. Grasping Algorithms

Grasping algorithms are used to enable robots to grasp objects with their end-effectors, such as hands or grippers. These algorithms can involve determining the best grasp location and orientation for a given object, as well as adjusting the grip as needed during manipulation.

2. Motion Planning Algorithms

Motion planning algorithms are used to plan a path for a robot's end-effector to manipulate an object. These algorithms can involve generating a collision-free path that avoids obstacles in the environment.

3. Force Control Algorithms

Force control algorithms are used to enable robots to apply the appropriate amount of force during manipulation tasks. These algorithms can involve using sensors to measure the force applied by the robot and adjusting it as needed to achieve the desired level of force.

5.2.3 Robot Architectures

Q12. What is a Robot? Explain, how artificial intelligence is used for it.

Ans : **(Imp.)**

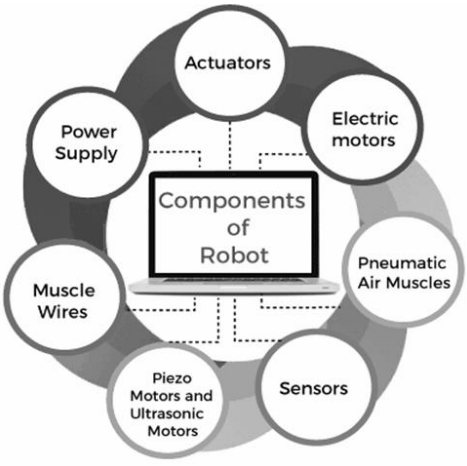
A robot is a machine that looks like a human, and is capable of performing out of box actions and replicating certain human movements automatically by means of commands given to it using programming.

Examples:

Drug Compounding Robot, Automotive Industry Robots, Order Picking Robots, Industrial Floor Scrubbers and Sage Automation Gantry Robots, etc.

Components of Robot

Several components construct a robot, these components are as follows:



- **Actuators**

  Actuators are the devices that are responsible for moving and controlling a system or machine. It helps to achieve physical movements by converting energy like electrical, hydraulic and air, etc. Actuators can create linear as well as rotary motion.

- **Power Supply**

  It is an electrical device that supplies electrical power to an electrical load. The primary function of the power supply is to convert electrical current to power the load.

- **Electric Motors**

  These are the devices that convert electrical energy into mechanical energy and are required for the rotational motion of the machines.

- **Pneumatic Air Muscles**

  Air Muscles are soft pneumatic devices that are ideally best fitted for robotics. They can contract and extend and operate by pressurized air filling a pneumatic bladder. Whenever air is introduced, it can contract up to 40%.

- **Muscles Wire**

  These are made up of nickel-titanium alloy called Nitinol and are very thin in shape. It can also extend and contract when a specific amount of heat and electric current is supplied into it. Also, it can be formed and bent into different shapes when it is in its martensitic form. They can contract by 5% when electrical current passes through them.

- **Piezo Motors and Ultrasonic Motors**

  Piezoelectric motors or Piezo motors are the electrical devices that receive an electric signal and apply a directional force to an opposing ceramic plate. It helps a robot to move in the desired direction. These are the best suited electrical motors for industrial robots.

- **Sensor**

  They provide the ability like see, hear, touch and movement like humans. Sensors are the devices or machines which help to detect the events or changes in the environment and send data to the computer processor. These devices are usually equipped with other electronic devices. Similar to human organs, the electrical sensor also plays a crucial role in Artificial Intelligence & robotics. AI algorithms control robots by sensing the environment, and it provides real-time information to computer processors.

Q13. State the applications of robotics.

Ans : **(Imp.)**

Applications of Robotics

Robotics have different application areas. Some of the important applications domains of robotics are as follows:

- **Robotics in Defence Sectors**

  The defence sector is undoubtedly the one of the main parts of any country. Each country wants their defence system to be strong. Robots help to approach inaccessible and dangerous zone during war. DRDO has developed a robot named Daksh to destroy life-threatening objects safely. They help soldiers to remain safe and deployed by the military in combat scenarios. Besides combat support, robots are also deployed in anti-submarine operations, fire support, battle damage management, strike missions, and laying machines.

- **Robotics in Medical Sectors**

  Robots also help in various medical fields such as laparoscopy, neurosurgery, orthopaedic surgery, disinfecting rooms, dispensing medication, and various other medical domains.

- **Robotics in Industrial Sector**

  Robots are used in various industrial manufacturing industries such as cutting, welding, assembly, disassembly, pick and place for printed circuit boards, packaging & labelling, palletizing, product inspection & testing, colour coating, drilling, polishing and handling the materials.

Moreover, Robotics technology increases productivity and profitability and reduces human efforts, resulting from lower physical strain and injury. The industrial robot has some important advantages, which are as follows:

- Accuracy
- Flexibility
- Reduced labour charge
- Low noise operation
- Fewer production damages
- Increased productivity rate.

- **Robotics in Entertainment**

Over the last decade, use of robots is continuously getting increased in entertainment areas. Robots are being employed in entertainment sector, such as movies, animation, games and cartoons. Robots are very helpful where repetitive actions are required. A camera-wielding robot helps shoot a movie scene as many times as needed without getting tired and frustrated. A big-name Disney has launched hundreds of robots for the film industry.

- **Robots in the Mining Industry**

Robotics is very helpful for various mining applications such as robotic dozing, excavation and haulage, robotic mapping & surveying, robotic drilling and explosive handling, etc. A mining robot can solely navigate flooded passages and use cameras and other sensors to detect valuable minerals. Further, robots also help in excavation to detect gases and other materials and keep humans safe from harm and injuries. The robot rock climbers are used for space exploration, and underwater drones are used for ocean exploration.

AI technology used in Robotics

- **Computer Vision**

Robots can also see, and this is possible by one of the popular Artificial Intelligence technologies named Computer vision. Computer Vision plays a crucial role in all industries like health, entertainment, medical, military, mining, etc.

Computer Vision is an important domain of Artificial Intelligence that helps in extracting meaningful information from images, videos and visual inputs and take action accordingly.

- **Natural Language Processing**

NLP (Natural Languages Processing) can be used to give voice commands to AI robots. It creates a strong human-robot interaction. NLP is a specific area of Artificial Intelligence that enables the communication between humans and robots. Through the NLP technique, the robot can understand and reproduce human language. Some robots are equipped with NLP so that we can't differentiate between humans and robots.

Similarly, in the health care sector, robots powered by Natural Language Processing may help physicians to observe the decease details and automatically fill in EHR. Besides recognizing human language, it can learn common uses, such as learn the accent, and predict how humans speak.

- **Edge Computing**

Edge computing in robots is defined as a service provider of robot integration, testing, design and simulation. Edge computing in robotics provides better data management, lower connectivity cost, better security practices, more reliable and uninterrupted connection.

- **Complex Event Process**

Complex event processing (CEP) is a concept that helps us to understand the processing of multiple events in real time. An event is described as a Change of State, and one or more events combine to define a Complex event. The complex event process is most widely used term in various industries such as healthcare, finance, security, marketing, etc. It is primarily used in credit card fraud detection and also in stock marketing field.

For example, the deployment of an airbag in a car is a complex event based on the data from multiple sensors in real-time. This idea is used in Robotics, for example, Event-Processing in Autonomous Robot Programming.

- **Transfer Learning and AI**

This is the technique used to solve a problem with the help of another problem that is already solved. In Transfer learning technique, knowledge gained from solving one problem can be implement to solve related problem. We can understand it with an example such as the model used for identifying a circle shape can also be used to identify a square shape.

Transfer learning reuses the pre-trained model for a related problem, and only the last layer of the model is trained, which is relatively less time consuming and cheaper. In robotics, transfer learning can be used to train one machine with the help of other machines.

- **Reinforcement Learning**

Reinforcement learning is a feedback-based learning method in machine learning that enables an AI agent to learn and explore the environment, perform actions and learn automatically from experience or feedback for each action. Further, it is also having feature of autonomously learn to behave optimally through hit-and-trail action while interacting with the environment. It is primarily used to develop the sequence of decisions and achieve the goals in uncertain and potentially complex environment. In robotics, robots explore the environment and learn about it through hit and trial. For each action, he gets rewarded (positive or negative). Reinforcement learning provides Robotics with a framework to design and simulate sophisticated and hard-to-engineer behaviours.

- **Affective Computing**

Affective computing is a field of study that deals with developing systems that can identify, interpret, process, and simulate human emotions. Affective computing aims to endow robots with emotional intelligence to hope that robots can be endowed with human-like capabilities of observation, interpretation, and emotion expression.

- **Mixed Reality**

Mixed Reality is also an emerging domain. It is mainly used in the field of programming by demonstration (PbD). PbD creates a prototyping mechanism for algorithms using a combination of physical and virtual objects.

Q14. Explain the architectures of robot system.

Ans : **(Imp.)**

The design of the robot's system architecture is important for enabling the robot to achieve its goal without requiring extremely complex software systems for implementation. In general, the system architecture is defined by two major parts: the structure and the style. The structure defines the way in which the system is broken down into components

Alternatively the style of the architecture refers to the computational concepts that define the implementation of the design

Architecture Structures

Robotic architecture refers to the design of architectural structures that incorporate robotic systems or elements. There are various architectural structures that can be used in robotic architecture, some of which include:

1. Exoskeletons

Exoskeletons are robotic structures that are designed to be worn by a human operator. They provide additional strength and mobility to the wearer, allowing them to perform tasks that would be difficult or impossible without the exoskeleton.

2. Kinetic Facades

Kinetic facades are building facades that incorporate moving or dynamic elements. These elements can be controlled by robotic systems, allowing the facade to adapt to changing environmental conditions or user needs.

3. Automated Shading Systems

Automated shading systems are robotic systems that control the amount of light entering a building. They can be programmed to adjust the shading throughout the day, reducing the amount of energy needed for cooling and lighting.

4. Mobile Robots

Mobile robots are autonomous or semi-autonomous robots that can move around a building or a site. They can be used for tasks such as security, cleaning, or transportation.

5. Robotic Building Systems

Robotic building systems are prefabricated building components that can be assembled or disassembled by robotic systems. They can be used to create complex structures quickly and efficiently.

6. Robotic Maintenance Systems

Robotic maintenance systems are robots that are designed to perform maintenance tasks on buildings or other structures. They can be used for tasks such as cleaning, painting, or repairing.

These architectural structures can be used individually or in combination to create innovative and efficient buildings that incorporate robotic systems.

Architecture Styles

Robotic architecture is a new field that incorporates robotic systems and elements in building design. The integration of robotics into architecture has led to the development of new architectural styles that are different from traditional styles. Some of the architectural styles used in robotic architecture are:

1. Parametric Architecture

Parametric architecture is a style that uses mathematical algorithms to generate complex shapes and forms. This style is well-suited for robotic architecture because robots can easily produce complex shapes and forms.

2. Digital Fabrication

Digital fabrication is a style that uses computer-controlled machines to manufacture building components. Robots can be used for digital fabrication, making this style particularly suitable for robotic architecture.

3. Adaptive Architecture

Adaptive architecture is a style that incorporates sensors and other devices to respond to changing environmental conditions. Robots can be used to control the adaptive elements of the building, making this style particularly suited to robotic architecture.

4. Kinetic Architecture

Kinetic architecture is a style that incorporates moving or dynamic elements into the building design. Robots can be used to control these elements, making this style well-suited for robotic architecture.

5. Organic Architecture

Organic architecture is a style that uses natural forms and shapes in building design. Robots can be used to produce complex organic shapes, making this style particularly suitable for robotic architecture.

6. Smart Architecture

Smart architecture is a style that incorporates sensors and other devices to monitor and control building systems. Robots can be used to control these systems, making this style particularly suited to robotic architecture.

These architectural styles are not mutually exclusive, and different styles can be combined to create innovative and efficient buildings that incorporate robotic systems.

5.3 Natural Language Processing

5.3.1 Introduction

Q15. What is NLP? Explain the advantages and disadvantages of NLP.

Ans :

Meaning

NLP stands for Natural Language Processing, which is a part of Computer Science, Human

language, and Artificial Intelligence. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation.

Advantages

- NLP helps users to ask questions about any subject and get a direct response within seconds.
- NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
- NLP helps computers to communicate with humans in their languages.
- It is very time efficient.
- Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Disadvantages

A list of disadvantages of NLP is given below:

- NLP may not show context.
- NLP is unpredictable
- NLP may require more keystrokes.
- NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

Q16. Explain the components of NLP.

Ans : **(Imp.)**

There are the following two components of NLP

1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks:

- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

Q17. State the applications of NLP.

Ans : **(Imp.)**

There are the following applications of NLP:

1. Question Answering

Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

2. Spam Detection

Spam detection is used to detect unwanted e-mails getting to a user's inbox.

3. Sentiment Analysis

Sentiment Analysis is also known as opinion mining. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (Natural Language Processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)

4. Machine Translation

Machine translation is used to translate text or speech from one natural language to another natural language.

Example: Google Translator

5. Spelling correction

Microsoft Corporation provides word processor software like MS-word, PowerPoint for the spelling correction.

6. Speech Recognition

Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

7. Chatbot

Implementing the Chatbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.

8. Information extraction

Information extraction is one of the most important applications of NLP. It is used for extracting structured information from unstructured or semi-structured machine-readable documents.

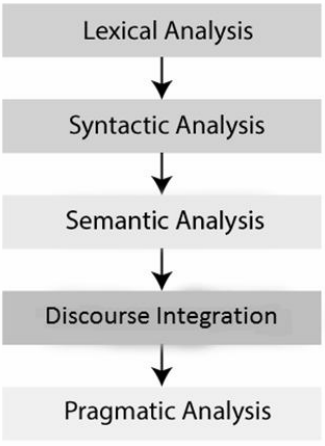
9. Natural Language Understanding (NLU)

It converts a large set of text into more formal representations such as first-order logic structures that are easier for the computer programs to manipulate notations of the natural language processing.

Q18. Explain the phases of NLP

Ans :

There are the following five phases of NLP:



1. Lexical Analysis and Morphological

The first phase of NLP is the Lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences, and words.

2. Syntactic Analysis (Parsing)

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

Example: Agra goes to the Poonam

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

3. Semantic Analysis

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences.

4. Discourse Integration

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

5. Pragmatic Analysis

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

For Example: "Open the door" is interpreted as a request instead of an order.

5.3.2 Syntactic Processing

Q19. Explain about Syntactic processing in NLP.

Ans : **(Imp.)**

Syntactic analysis is also known as Syntax analysis or Parsing. To implement the task of parsing, we use parsers.

About Parser

We already know parsers are used to implement parsing, but what is the definition of a parser? It is

described as a software component meant to take input text data and provide a structural representation of the data after validation for correct syntax using formal grammar.

It also creates a data structure, which is often in the form of a parse tree, an abstract syntax tree, or another hierarchical structure. After searching over the space of a variety of trees, it attempts to identify an ideal tree for a certain text.

Types of Parsing

Generally, there are two types of Parsing: Top-down parsing and Bottom-up parsing.

In top-down parsing, the parser builds the parse tree from the start symbol and then attempts to convert the start symbol to the input. The recursive technique is used to process the input in the most popular type of top-down parsing, but it has one major drawback: backtracking.

Whereas, In bottom-up parsing, the parser begins with the input symbol and works its way up to the start symbol, attempting to create the parser tree. Now, these types of parsings are used by different parsers.

The following are the types of parsers that are available:

1. Recursive Descent Parser

It is a to-the-point parser used frequently during parsing. It follows a top-down process where it checks if the syntax of the input is correct or not, by scanning the text from left to right.

For these sorts of parsers, the required operation is to read characters from the input stream and match them with the terminals using grammar. We will learn about grammar later in this article.

2. Shift-reduce Parser

Shift-reduce parsers use a bottom-up process, unlike recursive descent parsers. Its goal is to locate the words and phrases that correspond to the right-hand side of a grammatical production, replace them with the left-hand side, and try to find a word sequence that continues until the entire sentence is reduced.

Thus this parser starts with the input symbol and builds the parser tree all the way to the start symbol.

3. Chart Parser

Chart parser is mainly used for ambiguous grammars, like grammars of natural languages. It solves parsing difficulties using the dynamic programming idea. It saves partly theorized findings in a structure called a 'chart' as a consequence of dynamic programming. The 'chart' can also be utilized in a variety of situations.

4. Regexp Parser

It's one of the most popular parsers out there. On top of a POS-tagged string, it applies a regular expression defined in the form of grammar. Basically, it parses the input phrases using regular expressions and generates a parse tree as a result.

Parse Trees

A Parse tree is a graphical representation of a derivation. The root node of the parse tree is the start symbol of derivation, whereas the leaf nodes are terminals and the inner nodes are non-terminals. The most useful characteristic of the parse tree is that it produces the original input string when traversed in sequence.

About Grammar

Parsing is done to analyze the grammar of a sentence, so we must have a basic idea about the concept of grammar. To explain the syntactic structure of well-formed programs, grammar is highly significant. They imply syntactical norms for dialogue in natural languages in the literary sense.

Since the beginning of natural languages such as English, Hindi, and others, linguists have sought to define the grammar. The theory of formal languages is also useful in computer science, particularly in the areas of programming languages and data structures.

There are three types of grammar that we will list out here: Constituency grammar, dependency grammar, and context-free grammar.

1. Constituency Grammar

Constituency grammar is also known as phrase structure and is proposed by Noam Chomsky. It is based on constituency relation (hence, the name), and is completely the opposite of dependency grammar.

The sentence structure in this type of grammar is seen via the lens of constituency relations in all relevant frameworks. The constituency connection is derived from Latin and Greek grammar's subject-predicate division.

The noun phrase NP and verb phrase VP are used to understand the basic sentence structure. A parse tree that uses constituency grammar is known as a constituency-based parse tree.

2. Dependency Grammar

The following are the most important aspects of Dependency Grammar and Dependency Relationship:

- The linguistic units, i.e. words, are linked together via directed connections in DG.
- The verb takes center stage in the sentence structure.
- In terms of directed connection, all other syntactic elements are related to the verb. Dependencies are the syntactic components in question.

Parse trees that use dependency grammar are called dependency-based parse trees.

3. Context-free Grammar

Context-free grammar (CFG) is a superset of Regular grammar and a notation for describing languages. The following 4 components consisting of a finite set of grammar rules:

- **Set of Non-terminals**

  It is indicated by the letter V. The non-terminals are syntactic variables that represent groups of strings that the grammar generates to help define the language.

- **Set of Terminals**

  It's also known as tokens, and it's defined by Ó. The fundamental symbols of terminals are used to create strings.

- **Set of productions**

  P is the symbol for it. The set specifies the possible combinations of terminals and non-terminals. Non-terminals, an arrow, and terminals make up every production(P) (the sequence of terminals). Non-terminals are referred to as the left side of the production, whereas terminals are referred to as the right side.

- **Start Symbol**

  The production process starts with the start sign. The letter S stands for it. The start symbol is always a non-terminal symbol.

5.3.3 Semantic Analysis

Q20. What is semantic analysis in NLP.

Ans :

Humans interact with each other through speech and text, and this is called Natural language. Computers understand the natural language of humans through Natural Language Processing (NLP).

NLP is a process of manipulating the speech of text by humans through Artificial Intelligence so that computers can understand them. It has made interaction between humans and computers very easy.

Human language has many meanings beyond the literal meaning of the words. There are many words that have different meanings, or any sentence can have different tones like emotional or sarcastic. It is very hard for computers to interpret the meaning of those sentences.

Semantic analysis is a subfield of NLP and Machine learning that helps in understanding the context of any text and understanding the emotions that might be depicted in the sentence. This helps in extracting important information from achieving human level accuracy from the computers. Semantic analysis is used in tools like machine translations, chatbots, search engines and text analytics.

Q21. How does Semantic Analysis work?

Ans :

According to this source, Lexical analysis is an important part of semantic analysis. Lexical semantics is the study of the meaning of any word. In semantic analysis, the relation between lexical items are identified. Some of the relations are hyponyms, synonyms, Antonyms, Homonyms etc.

Let us learn in details about the relations:

- **Hyponymy**

  It illustrates the connection between a generic word and its occurrences. The generic term is known as hypernym, while the occurrences are known as hyponyms.

- **Homonymy**

  It may be described as words with the same spelling or form but diverse and unconnected meanings.

- **Polysemy**

  Polysemy is a term or phrase that has a different but comparable meaning. To put it another way, polysemy has the same spelling but various and related meanings.

- **Synonymy**

  It denotes the relationship between two lexical elements that have different forms but express the same or a similar meaning.

- **Antonymy**

  It is the relationship between two lexical items that include semantic components that are symmetric with respect to an axis.

- **Meronomy**

  It is described as a logical arrangement of letters and words indicating a component portion of or member of anything.

Through identifying these relations and taking into account different symbols and punctuations, the machine is able to identify the context of any sentence or paragraph.

Q22. Explain briefly about representation.

Ans :

Semantic analysis represents the meaning of any sentence. These are done by different processes and methods. Let us discuss some building blocks of the semantic system:

- **Entities**

  Any sentence is made of different entities that are related to each other. It represents any individual category such as name, place, position, etc. We will discuss in detail about entities and their correlation later in this blog.

- **Concepts**

  It represents the general category of individual, such as person, city etc.

- **Relations**

  It represents the relation between different entities and concepts in a sentence.

- **Predicates**

  It represents the verb structure of any sentence.

There are different approaches to Meaning Representations according, some of them are mentioned below:

- First-order predicate logic (FOPL)
- Frames
- Semantic Nets
- Case Grammar
- Rule-based architecture
- Conceptual graphs
- Conceptual dependency (CD)

Meaning Representation is very important in Semantic Analysis because:

1. It helps in linking the linguistic elements of a sentence to the non-linguistic elements.
2. It helps in representing unambiguous data at lexical level.
3. It helps in reasoning and verifying correct data.

Q23. Explain the processeof semantic analysis.

Ans : **(Imp.)**

The following are some of the processes of Semantic Analysis:

1. Word Sense Disambiguation

It is an automatic process of identifying the context of any word, in which it is used in the sentence. In natural language, one word can have many meanings. For eg- The word 'light' could be meant as not very dark or not very heavy. The computer has to understand the entire sentence and pick up the meaning that fits the best. This is done by word sense disambiguation.

2. Relationship Extraction

In a sentence, there are a few entities that are co-related to each other. Relationship extraction is the process of extracting the semantic relationship between these entities. In a sentence, "I am learning mathematics", there are two entities, 'I' and 'mathematics' and the relation between them is understood by the word 'learn'.

Q24. Explain the techniques of semantic analysis.

Ans :

There are two types of techniques in Semantic Analysis depending upon the type of information that you might want to extract from the given data. These are semantic classifiers and semantic extractors. Let us briefly discuss them.

1. Semantic Classification Models

These are the text classification models that assign any predefined categories to the given text.

- **Topic classification**

  It is a method for processing any text and sorting them according to different known predefined categories on the basis of its content.

  For eg: In any delivery company, the automated process can separate the customer service problems like 'payment issues' or 'delivery problems', with the help of machine learning. This will help the team notice the issues faster and solve them.

- **Sentiment Analysis**

  It is a method for detecting the hidden sentiment inside a text, may it be positive, negative or neural. This method helps in understanding the urgency of any statement. In social media, often customers reveal their opinion about any concerned company.

  For example, someone might comment saying, "The customer service of this company is a joke!". If the sentiment here is not properly analysed, the machine might consider the word "joke" as a positive word.

- **Latent Semantic Analysis**

  It is a method for extracting and expressing the contextual-usage meaning of words using statistical calculations on a huge corpus of text. LSA is an information retrieval approach that examines and finds patterns in unstructured text collections as well as their relationships.

- **Intent Classification**

  It is a method of differentiating any text on the basis of the intent of your customers. The customers might be interested or disinterested in your company or services. Knowing prior whether someone is interested or not helps in proactively reaching out to your real customer base.

2. Semantic Extraction Models

- **Keyword Extraction**

  It is a method of extracting the relevant words and expressions in any text to find out the granular insights. It is mostly used along with the different classification models. It is used to analyze different

keywords in a corpus of text and detect which words are 'negative' and which words are 'positive'. The topics or words mentioned the most could give insights of the intent of the text.

- **Entity Extraction**

  Any sentence or phrase is made up of different entities like names of people, places, companies, positions, etc. This method is used to identify those entities and extract them. It can be very useful for customer service teams of businesses like delivery companies as the machine can automatically extract the names of their customers, their location, shipping numbers, contact information or any other relevant or important data.

5.3.4 Statistical NLP

Q25. Explain about statistical NLP

Ans : **(Imp.)**

Statistical processing in natural language processing (NLP) involves the use of statistical algorithms and models to process and analyze language data. It is a key area of NLP that is used for a wide range of tasks, including text classification, sentiment analysis, named entity recognition, language translation, and speech recognition.

At a high level, statistical processing in NLP involves collecting and analyzing large sets of language data to identify patterns and relationships between words and phrases. This data is then used to train statistical models that can be used to make predictions or classifications about new language data.

Some of the key techniques used in statistical processing of NLP include:

1. **Corpus Analysis**

   The statistical processing of NLP starts with collecting and analyzing a large corpus of language data. This can include text data from sources such as books, articles, websites, and social media. The data is analyzed to identify patterns and relationships between words and phrases.

2. **Tokenization**

   Tokenization involves breaking down text data into individual words or tokens. This is a critical step in statistical processing, as it allows for the analysis of individual words and phrases.

3. **Part-of-speech Tagging**

   Part-of-speech tagging involves identifying the part of speech of each word in a text. This information is used to understand the meaning and context of a sentence.

4. **Named Entity Recognition**

   Named entity recognition involves identifying and classifying named entities such as people, organizations, and locations in a text. This information is used for tasks such as text classification and information retrieval.

5. **Sentiment Analysis**

   Sentiment analysis involves analyzing the tone and emotion of a text. This is typically done using machine learning algorithms trained on labeled data.

6. **Language Translation**

   Language translation involves using statistical models to translate text from one language to another. This is typically done using parallel corpora, which are collections of text in two or more languages that are aligned sentence by sentence.

Overall, statistical processing is an essential component of NLP, as it allows for the efficient and accurate analysis of large sets of language data. By using statistical models and algorithms, NLP researchers and practitioners can gain insights into the structure and meaning of language, and develop new applications and tools for processing and analyzing text data.

Here are some examples of how statistical processing is used in NLP:

1. Text Classification

Text classification is the task of categorizing text data into predefined categories. For example, a news article might be classified as "politics", "sports", or "entertainment". Statistical models such as Naive Bayes and Support Vector Machines (SVMs) can be trained on labeled data to perform text classification.

2. Sentiment Analysis

Sentiment analysis is the task of determining the tone and emotion of a piece of text. For example, a tweet might be classified as "positive", "negative", or "neutral". Statistical models such as Logistic Regression and Random Forests can be trained on labeled data to perform sentiment analysis.

3. Named entity Recognition

Named entity recognition is the task of identifying and classifying named entities in text data. For example, a sentence might contain named entities such as "New York City" or "Barack Obama". Statistical models such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) can be trained on labeled data to perform named entity recognition.

4. Language Translation

Language translation involves using statistical models to translate text from one language to another. For example, a sentence in English might be translated into French. Statistical models such as phrase-based and neural machine translation models can be trained on parallel corpora to perform language translation.

5. Speech Recognition

Speech recognition is the task of transcribing spoken language into written text. Statistical models such as Hidden Markov Models (HMMs) and Deep Neural Networks (DNNs) can be trained on speech data to perform speech recognition.

5.3.5 Spell Checking

Q26. What is spell checking? Explain spell checking techniques.

Ans : **(Imp.)**

Meaning

Spell checking is the process of identifying and correcting spelling errors in a piece of text. Spell checking is an important task in natural language processing (NLP) and is used in a wide range of applications such as word processors, search engines, and chatbots.

Techniques

There are several techniques for spell checking. Here are some of the most commonly used ones:

1. Dictionary-based Spell Checking

In this technique, a dictionary of correctly spelled words is used to compare against the text being checked. The spell checker scans the text and identifies any words that are not found in the dictionary. The spell checker then suggests possible corrections based on the words in the dictionary that are similar in spelling to the misspelled word.

Example

Let's say we have a text document with the word "acheive". A dictionary-based spell checker would scan the text and identify "acheive" as a misspelled word because it is not found in the dictionary. The spell checker would then suggest possible corrections based on similar words in the dictionary, such as "achieve".

2. Rule-based Spell Checking

In this technique, a set of rules is used to check the spelling of words. The rules are based on the orthography (spelling system) of the language being checked. For example, a rule might be that the letter "i" must always be followed by the letter "e", except after "c". Rule-based spell checking can be more accurate than dictionary-based spell checking, but it requires more computational power to implement.

Example

In the English language, there is a rule that the letter "i" must always be followed by the letter "e", except after "c". So if we have the word "recieve" in a text document, a rule-based spell checker would identify "recieve" as a misspelling because it violates the "i before e except after c" rule. The spell checker would then suggest a possible correction, such as "receive".

3. Statistical Spell Checking

In this technique, statistical models are used to identify and correct spelling errors. Statistical spell checkers use a large corpus of text data to identify common misspellings and to learn patterns of letter sequences that are likely to be misspelled. These models can also take into account the context of the text being checked, such as the frequency of certain words or the likelihood of certain spelling errors in a particular domain or genre of text.

Example: Let's say we have a large corpus of English text data, such as a collection of news articles. We can use this data to train a statistical language model to identify and correct spelling errors. The model would learn patterns of letter sequences that are likely to be misspelled, as well as the frequency of certain words and letter combinations in the English language. When we run the model on a text document, it can identify misspelled words and suggest possible corrections based on the statistical patterns it has learned.

4. Hybrid Spell Checking

Hybrid spell checking combines two or more of the above techniques to improve the accuracy of spell checking. For example, a hybrid spell checker might use a dictionary-based approach to identify possible corrections, but then use a statistical model to select the most likely correction based on the context of the text being checked.

Example

A hybrid spell checker might use a combination of a dictionary-based and rule-based approach. It would first identify misspelled words using the dictionary, and then apply a set of rules to filter out false positives and suggest possible corrections. For example, if the word "thier" is identified as a misspelling by the dictionary, the spell checker might apply a rule that suggests changing "thier" to "their", because "thier" violates the "i before e except after c" rule.

Overall, spell checking is an important task in NLP and relies on a combination of techniques to accurately identify and correct spelling errors in text.

Here are several algorithms used for spell checking in AI. Some of the commonly used ones are:

1. Levenshtein Distance

It is a measure of the difference between two strings, which is used to calculate the minimum number of edits required to transform one string into another. It is also known as the edit distance algorithm.

2. Dictionary-based Approach

This approach compares the input word with a pre-existing dictionary of correctly spelled words. If the input word is not found in the dictionary, it is flagged as a spelling error.

3. Rule-based Approach

This approach uses a set of rules to identify potential spelling errors. For example, a rule may identify that the word "recieve" is misspelled because the correct spelling is "receive".

4. Probabilistic Approach

This approach uses statistical models to predict the likelihood of a word being spelled correctly or incorrectly. The models are trained on large datasets of correctly spelled words.

5. Neural Network Approach

This approach uses neural networks to learn the patterns in correctly spelled words and then uses these patterns to predict whether a new word is spelled correctly or not.

Short Question and Answers

1. Real Time Search?

Ans :

Real-time search is a type of search algorithm used in artificial intelligence and computer science to solve problems where the solution needs to be found in real-time or near real-time. In real-time search, the agent continuously receives new ainformation about the state of the environment and updates its search accordingly.

Real-time search algorithms are used in various applications such as video games, robotics, and decision support systems. They are designed to efficiently explore the search space and find a solution within a limited amount of time.

2. Speech recognition AI?

Ans :

Speech recognition enables computers, applications and software to comprehend and translate human speech data into text, for business solutions. The speech recognition model works by using artificial intelligence (AI) to analyse your voice and language, identify by learning the words you are saying, and then output those words with transcription accuracy as model content or text data on a screen.

3. Navigation

Ans :

Navigation in artificial intelligence refers to the ability of an AI system to plan and execute a path to a goal location in a given environment. Navigation is a critical component of many AI applications, including autonomous vehicles, robotics, and virtual assistants.

Navigation algorithms typically involve three main components: perception, planning, and control. Perception involves using sensors to perceive the environment and generate a map of the surroundings. Planning involves using the map and other information, such as the current location and goal location, to plan a path to the destination. Control involves executing the planned path and adjusting the path as needed based on new sensor data.

4. Explain, how artificial intelligence is used for it.

Ans :

A robot is a machine that looks like a human, and is capable of performing out of box actions and replicating certain human movements automatically by means of commands given to it using programming.

Examples:

Drug Compounding Robot, Automotive Industry Robots, Order Picking Robots, Industrial Floor Scrubbers and Sage Automation Gantry Robots, etc.

5. Applications of Robotics

Ans :

- **Robotics in Defence Sectors**

  The defence sector is undoubtedly the one of the main parts of any country. Each country wants their defence system to be strong. Robots help to approach inaccessible and dangerous zone during war. DRDO has developed a robot named Daksh to destroy life-threatening objects safely. They help soldiers to remain safe and deployed by the military in combat scenarios. Besides combat support, robots are also deployed in anti-submarine operations, fire support, battle damage management, strike missions, and laying machines.

- **Robotics in Medical Sectors**

  Robots also help in various medical fields such as laparoscopy, neurosurgery, orthopaedic surgery, disinfecting rooms, dispensing medication, and various other medical domains.

- **Robotics in Industrial Sector**

  Robots are used in various industrial manufacturing industries such as cutting, welding, assembly, disassembly, pick and place for printed circuit boards, packaging & labelling, palletizing, product inspection & testing, colour coating, drilling, polishing and handling the materials.

6. Natural Language Processing

Ans :

NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation.

7. Disadvantages

Ans :

A list of disadvantages of NLP is given below:

- NLP may not show context.
- NLP is unpredictable
- NLP may require more keystrokes.
- NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

8. Components of NLP.

Ans :

There are the following two components of NLP

i) Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks:

- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

ii) Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

9. Applications of NLP.

Ans :

There are the following applications of NLP:

i) Question Answering: Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

ii) Spam Detection: Spam detection is used to detect unwanted e-mails getting to a user's inbox.

iii) Sentiment Analysis: Sentiment Analysis is also known as opinion mining. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (Natural Language Processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)

iv) Machine Translation: Machine translation is used to translate text or speech from one natural language to another natural language.

10. Semantic analysis in NLP.

Ans :

Humans interact with each other through speech and text, and this is called Natural language. Computers understand the natural language of humans through Natural Language Processing (NLP).

NLP is a process of manipulating the speech of text by humans through Artificial Intelligence so that computers can understand them. It has made interaction between humans and computers very easy.

Human language has many meanings beyond the literal meaning of the words. There are many words that have different meanings, or any sentence can have different tones like emotional or sarcastic. It is very hard for computers to interpret the meaning of those sentences.

Semantic analysis is a subfield of NLP and Machine learning that helps in understanding the context of any text and understanding the emotions that might be depicted in the sentence. This helps in extracting important information from achieving human level accuracy from the computers. Semantic analysis is used in tools like machine translations, chatbots, search engines and text analytics.

Choose the Correct Answers

1. The process of converting raw sensory data into meaningful information is known as: [b]
   a) Sensing b) Perception
   c) Action d) Feedback
2. Which of the following is not an example of perception in AI? [c]
   a) Object recognition b) Speech recognition
   c) Decision making d) Facial recognition
3. Which of the following is not an example of action in AI? [c]
   a) Moving a robot arm b) Playing a move in a game
   c) Sending a message d) Speaking
4. Which of the following factors can affect the accuracy of speech recognition systems? [d]
   a) Background noise b) Speaker accent
   c) Speech rate d) All of the above
5. Which of the following techniques is not used in speech recognition? [d]
   a) Hidden Markov Models (HMM) b) Artificial Neural Networks (ANN)
   c) Decision Trees (DT) d) Random Forests (RF)
6. Which of the following is a commonly used algorithm for path planning in robot navigation? [a]
   a) A* algorithm b) Depth-first search
   c) Breadth-first search d) Random search
7. Which of the following is not a type of robot manipulation system? [d]
   a) Teleoperation b) Kinesthetic teaching
   c) Autonomous d) Augmented reality
8. Which of the following techniques is used to convert text to numerical vectors? [c]
   a) Part-of-speech tagging b) Dependency parsing
   c) Word embedding d) Named entity recognition
9. Which of the following algorithms is commonly used for sentiment analysis? [d]
   a) Naive Bayes b) K-Means
   c) Support Vector Machines (SVM) d) All of the above
10. Which of the following is an example of a semantic error in language? [a]
    a) Using the wrong word in a sentence b) Using incorrect grammar
    c) Using incorrect punctuation d) None of the above

Fill in the Blanks

1. _____ is a type of search algorithm used in artificial intelligence and computer science to solve problems where the solution needs to be found in real-time or near real-time.
2. A _____ refers to the ability of an AI system to perceive and understand visual information from the world around it
3. _____in artificial intelligence refers to the ability of an AI system to plan and execute a path to a goal location in a given environment
4. The process of converting speech signals into text or other forms of data is known as _____
5. _____ is a critical component of robots used in manufacturing, logistics, and healthcare.
6. _____ algorithms are used to enable robots to grasp objects with their end-effectors, such as hands or grippers
7. _____is concerned with the meaning of words and sentences, while syntactic analysis is concerned with their structure.
8. A measure of the difference between two strings. Is called _____
9. _____in natural language processing (NLP) involves the use of statistical algorithms and models to process and analyze language data
10. It is a method of extracting the relevant words and expressions in any text to find out the granular insights is called _____

ANSWERS

1. Real-time search
2. Vision
3. Navigation
4. Speech recognition
5. Manipulation
6. Grasping
7. Semantic analysis
8. Levenshtein distance
9. Statistical processing
10. Keyword extraction

FACULTY OF INFORMATICS

BCA II Year IV Semester (CBCS) Examination

Model Paper - I

ARTIFICIAL INTELLIGENCE

Time : 3 Hours] **[Max. Marks : 70**

Note : Answer all questions from Part - A, & any five questions from Part - B Choosing one questions from each unit.

PART - A (10 × 2 = 20 Marks)

ANSWERS

1. (a) Production System. **(Unit-I, SQA-3)**
   (b) Means-Ends Analysis. **(Unit-I, SQA-9)**
   (c) Game playing in AI. **(Unit-II, SQA-1)**
   (d) Explain about alpha beta pruning technique. **(Unit-II, SQA-3)**
   (e) Bayes' theorem. **(Unit-III, SQA-7)**
   (f) Depth First Search **(Unit-III, SQA-4)**
   (g) Decision trees technique. **(Unit-IV, SQA-5)**
   (h) Expert system usages. **(Unit-IV, SQA-9)**
   (i) Speech recognition AI? **(Unit-V, SQA-2)**
   (j) Applications of Robotics **(Unit-V, SQA-5)**

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. (a) Write about problem solving techniques in AI. **(Unit-I, Q.No. 4)**
   (b) Explain the applications of AI. **(Unit-I, Q.No. 2)**

OR

3. (a) Write about Planning-Goal Stack Algorithm. **(Unit-I, Q.No. 18)**
   (b) What is Production System? State is components. **(Unit-I, Q.No. 5)**

UNIT - II

4. Explain about Min–Max Search algorithm. **(Unit-II, Q.No. 2)**

OR

5. Explain about various approaches to frame problem solution. **(Unit-II, Q.No. 14)**

UNIT - III

6. What is the Breadth-First Search Algorithm? **(Unit-III, Q.No. 5)**

OR

7. What is certainty factor and explain about rule based systems. **(Unit-III, Q.No. 10)**

UNIT - IV

8. (a) What is learning? What are various learning aspects in AI ? **(Unit-IV, Q.No. 1)**

   (b) What is rote learning? Explain about it. **(Unit-IV, Q.No. 3)**

OR

9. (a) Explain how learning are used in problem solving? **(Unit-IV, Q.No. 5)**

   (b) Explain forward chaining and backward chaining. **(Unit-IV, Q.No. 12)**

UNIT - V

10. (a) Write about vision in AI. **(Unit-V, Q.No. 2)**

    (b) Explain brifly about various speech recognition techniques. **(Unit-V, Q.No. 8)**

OR

11. (a) Explain the architectures of robot system. **(Unit-V, Q.No. 14)**

    (b) Explain about Syntactic processing in NLP. **(Unit-V, Q.No. 19)**

FACULTY OF INFORMATICS

BCA II Year IV Semester (CBCS) Examination

Model Paper - II

ARTIFICIAL INTELLIGENCE

Time : 3 Hours] **[Max. Marks : 70**

Note : Answer all questions from Part - A, & any five questions from Part - B Choosing one questions from each unit.

PART - A (10 × 2 = 20 Marks)

ANSWERS

1. (a) What is heuristic search? **(Unit-I, SQA-5)**
   (b) Applications of AI. **(Unit-I, SQA-2)**
   (c) Disadvantages of IDDFS **(Unit-II, SQA-9)**
   (d) Frame problem in AI. **(Unit-II, SQA-6)**
   (e) Augmenting a problem solver in AI. **(Unit-III, SQA-3)**
   (f) What is uncertainty. **(Unit-III, SQA-5)**
   (g) What is rote learning. **(Unit-IV, SQA-2)**
   (h) Components of an Expert System **(Unit-IV, SQA-10)**
   (i) Real Time Search? **(Unit-V, SQA-1)**
   (j) Explain, how artificial intelligence is used for it. **(Unit-V, SQA-4)**

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. (a) Explain briefly about problem characteri-stics. **(Unit-I, Q.No. 7)**
   (b) Explain how problem reduction can be done by using AO* algorithm. **(Unit-I, Q.No. 13)**

OR

3. (a) Explain CSP with the example of SEND + MORE = MONEY. **(Unit-I, Q.No. 15)**
   (a) Explain, how to solve Constraint Satisfac-tion Problems using Search. **(Unit-I, Q.No. 16)**

UNIT - II

4. Explain the representation of instance and is a relationship. **(Unit-II, Q.No. 17)**

OR

5. Explain about various approaches to frame problem solution. **(Unit-II, Q.No. 14)**

UNIT - III

6. What is augmenting a problem solver in AI. **(Unit-III, Q.No. 4)**

OR

7. Explain about Dempster- shafer theory. **(Unit-III, Q.No. 13)**

UNIT - IV

8. (a) What is learning by advice? Explain. **(Unit-IV, Q.No. 4)**

   (b) What is knowledge? Explain, how knowledge is used in expert system. **(Unit-IV, Q.No. 8)**

OR

9. (a) Explain in detail about expert system usages. **(Unit-IV, Q.No. 11)**

   (b) Explain briefly about knowledge acquisition. **(Unit-IV, Q.No. 14)**

UNIT - V

10. (a) Explain the concept of real time search? **(Unit-V, Q.No. 1)**

    (b) State the applications of speech recognition AI. **(Unit-V, Q.No. 6)**

OR

11. (a) Explain various forms of navigation algorithms. **(Unit-V, Q.No. 10)**

    (b) How does Semantic Analysis work? **(Unit-V, Q.No. 21)**

FACULTY OF INFORMATICS

BCA II Year IV Semester (CBCS) Examination

Model Paper - III

ARTIFICIAL INTELLIGENCE

Time : 3 Hours] **[Max. Marks : 70**

Note : Answer all questions from Part - A, & any five questions from Part - B Choosing one questions from each unit.

PART - A (10 × 2 = 20 Marks)

ANSWERS

1. (a) Hill climbing algorithm. **(Unit-I, SQA-6)**
   (b) Planning-Goal Stack Algorithm. **(Unit-I, SQA-10)**
   (c) Natural deduction **(Unit-II, SQA-8)**
   (d) Min - Max Search algorithm. **(Unit-II, SQA-2)**
   (e) Various types of logics in non monotonic reasoning. **(Unit-III, SQA-9)**
   (f) Need of Probabilistic Reasoning in AI. **(Unit-III, SQA-6)**
   (g) Learning by advice. **(Unit-IV, SQA-3)**
   (h) Components of Knowledge Base **(Unit-IV, SQA-7)**
   (i) Semantic analysis in NLP. **(Unit-V, SQA-10)**
   (j) Disadvantages **(Unit-V, SQA-7)**

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. (a) Explain the advantages production system in artificial intelligence. **(Unit-I, Q.No. 6)**
   (b) Write about problem solving techniques in AI. **(Unit-I, Q.No. 4)**

OR

3. (a) What is Artificial Intelligence? Write about various approaches used to define AI. **(Unit-I, Q.No. 1)**
   (b) What is heuristic search? Explain about generate and test algorithm. **(Unit-I, Q.No. 9)**

UNIT - II

4. Explain Iterative Deepening Depth-First Search algorithm. **(Unit-II, Q.No. 6)**

OR

5. Explain about proposition resolution and unification algorithm. **(Unit-II, Q.No. 19)**

UNIT - III

6. What is monotonic and non monotonic reasoning? Explain. **(Unit-III, Q.No. 1)**

OR

7. Explain about probabilistic reasoning. **(Unit-III, Q.No. 8)**

UNIT - IV

8. (a) Explain about learning by decision trees technique. **(Unit-IV, Q.No. 7)**

   (b) Define expert system? Explain the components of an expert systems. **(Unit-IV, Q.No. 9)**

OR

9. (a) Define knowledge acquisition state in process. **(Unit-IV, Q.No. 13)**

   (a) Explain about inductive learning algorithm. **(Unit-IV, Q.No. 6)**

UNIT - V

10. (a) Explain brifly about various speech recognition techniques. **(Unit-V, Q.No. 8)**

    (b) What is a Robot? Explain, how artificial intelligence is used for it. **(Unit-V, Q.No. 12)**

OR

11. (a) Discribe challenges in working with speech recognition AI. **(Unit-V, Q.No. 7)**

    (b) What is NLP? Explain the advantages and disadvantages of NLP. **(Unit-V, Q.No. 15)**

FACULTY OF INFORMATICS

BCA IV - Semester (CBCS) Examination

February - 2023

ARTIFICIAL INTELLIGENCE

Time : 3 Hours] [Max. Marks : 70

Note : I) Answer all questions from Part - A and answer any five questions from Part-B. Choosing one questions from each unit.

II) Missing data, if any, may be suitably assumed.

PART - A (10 × 2 = 20 Marks)

1. (a) Define Mundane Tasks.

   (b) Explain formal Tasks.

   (c) Explain view point level.

   (d) Define Predicate.

   (e) Define Random Variable.

   (f) Define Clustering.

   (g) Define Backward Chaining.

   (h) Explain Factual Knowledge.

   (i) Define Augmented Transition Networks.

   (j) Define NLU.

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. (a) Describe various problem Characteristics.

   (b) Explain Best first search algorithm.

OR

3. (a) Write about Means - Ends Analysis.

   (b) Discuss the various issues involved in the design of search program.

UNIT - II

4. (a) Write about Frame problem.

   (b) Write about Additional Refinements and iterative Deepening.

OR

5. (a) Explain First - order logic in Artificial Intelligence.

   (b) What is Predicate logic?

UNIT - III

6. (a) What is meant by Non-monotonic reasoning?

   (b) What are problem solving strategies?

OR

7. (a) How do you Calculate certainty factor in Artificial Intelligence?

   (b) Explain Bayesian network in Artificial Intelligence in detail.

UNIT - IV

8. (a) What are the Characteristics of rote Learning?

   (b) Explain learning by taking advice in Artificial intelligence.

OR

9. (a) Explain about Knowledge Acquisition.

   (b) Write about Expert systems.

UNIT - V

10. (a) What are the 4 components of Artificial Intelligence?

    (b) What is condition action rule in Artificial Intelligence?

OR

11. (a) List out Applications of Robot in Artificial Intelligence.

    (b) What are the advantages of speech recognition?