es easy-solutions

New Syllabus SPPU

As per the New Credit System Syllabus (2019 course) of
Savitribai Phule Pune University w.e.f. academic year 2021-2022

ARTIFICIAL INTELLIGENCE

(Code : 310253)

"QUICK READ SERIES"

Semester VI
Computer Engineering

Chapterwise Solved Paper Solution
For End Semester Examination

TechKnowledge™ Publications

easy-solutions

easy – solutions

Savitribai Phule Pune University

As per New Credit System Syllabus(Rev. 2019) of Savitribai Phule Pune University with effective from Academic Year 2021-2022

Artificial Intelligence

(Code : 310253)

"Quick Read Series"

Semester VI - Computer Engineering

TechKnowledge™ Publications

EPE133A Price ₹ 120/-

Artificial Intelligence (Code : 310253)

(Semester VI - Computer Engineering) (SPPU)

Edition 2022

This edition is for sale in India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan, Sri Lanka and designated countries in South-East Asia. Sale and purchase of this book outside of these countries is unauthorized by the publisher.

Published By

TECHKNOWLEDGE PUBLICATIONS

Printed @

37/2, Ashtavinayak Industrial Estate,
Near Pari Company,
Narhe, Pune, Maharashtra State, India.
Pune - 411041

Head Office

B/5, First floor, Maniratna Complex, Taware Colony,
Aranyeshwar Corner, Pune - 411 009.
Maharashtra State, India
Ph : 91-20-24221234, 91-20-24225678.
Email : info@techknowledgebooks.com,
Website : www.techknowledgebooks.com

Subject Code : 310253

Book code : EPE133A

SYLLABUS

In-Sem. Exam

Unit I : Introduction

07 hrs

Introduction to Artificial Intelligence, Foundations of Artificial Intelligence, History of Artificial Intelligence, State of the Art, Risks and Benefits of AI, Intelligent Agents, Agents and Environments, Good Behavior: Concept of Rationality, Nature of Environments, Structure of Agents.

Unit II : Problem-solving

07 hrs

Solving Problems by Searching, Problem-Solving Agents, Example Problems, Search Algorithms, Uninformed Search Strategies, Informed (Heuristic) Search Strategies, Heuristic Functions, Search in Complex Environments, Local Search and Optimization Problems

End-Sem. Exam

Unit III : Adversarial Search and Games

07 hrs

Game Theory, Optimal Decisions in Games, Heuristic Alpha–Beta Tree Search, Monte Carlo Tree Search, Stochastic Games, Partially Observable Games, Limitations of Game Search Algorithms, Constraint Satisfaction Problems (CSP), Constraint Propagation: Inference in CSPs, Backtracking Search for CSPs.

Unit IV : Knowledge

07 hrs

Logical Agents, Knowledge-Based Agents, The Wumpus World, Logic, Propositional Logic: A Very Simple Logic, Propositional Theorem Proving, Effective Propositional Model Checking, Agents Based on Propositional Logic, First-Order Logic, Representation Revisited, Syntax and Semantics of First-Order Logic, Using First-Order Logic, Knowledge Engineering in First-Order Logic.

Unit V : Reasoning

07 hrs

Inference in First-Order Logic, Propositional vs. First-Order Inference, Unification and First-Order Inference, Forward Chaining, Backward Chaining, Resolution, Knowledge Representation, Ontological Engineering, Categories and Objects, Events, Mental Objects and Modal Logic, Reasoning Systems for Categories, Reasoning with Default Information

Unit VI : Planning

07 hrs

Automated Planning, Classical Planning, Algorithms for Classical Planning, Heuristics for Planning, Hierarchical Planning, Planning and Acting in Nondeterministic Domains, Time, Schedules, and Resources, Analysis of Planning Approaches, Limits of AI, Ethics of AI, Future of AI, AI Components, AI Architectures.

□□□

Table of Contents

□□□

Savitribai Phule Pune University
Semester VI (Computer Engineering)

ARTIFICIAL INTELLIGENCE

Unit III : Adversarial Search and Games

Q. 1 What is adversarial search? **(2 Marks)**

Ans. : Adversarial Search

- Adversarial Search Problem is having competitive activity which involves 'n' players and it is played according to certain set of protocols.
- Game is called adversarial because there are agents with conflicting goals and the surrounding environment in a game is competitive as there are 'n' players or agents participating.
- Goals are conflicting and environment is competitive because every participant wants to win the game.
- From above explanation it is understandable that we are dealing with a competitive multi agent environment.
- As the actions of every agent are unpredictable there are many possible moves/actions.
- In order to play a game successfully every agent in environment has to first analyze the action of other agents and how other agents are contributing in its own wining or loosing. After performing this analysis agent executes.

Q. 2 Write a short notes on : Features of AI game. **(5 Marks)**

Ans. :

AI Game - Features

Under artificial intelligence category, there are few special features as shown in Table 3.1, which make the game more interesting.

Table 3.1 : AI game features

Features	Explanation	Example
Two player	When there are two opponents/agents playing the game it is called 2-play game. To increase difficulty of the game Intelligence is added to agents in AI games. In case of AI Games must have at least two players. (i.e. single player games don't come under AI games category).	Chess
Multi-agent	When there are two or more opponents/agents playing the game it is called multi agent environment where action of every agents affects the action of other agents.	Monopoly
Non-cooperative environment	When the surrounding environment is not helpful for winning the game it is called as non-cooperative or competitive.	Card games
Turn taking	In a multi-agent environment when the agent/ player performs a move and has to wait for the next player to make the next move.	Any board game, Chess, carom, etc.
Time limit	One more constraint can be incorporated in a game i.e. keeping a limitation on time. Every player will get a finite amount of time to take an action.	Time bound chess games
Unpredictable opponent	In AI games action of opponent agent is fuzzy which makes the game challenging and unpredictable. Players are called unpredictable when the next step depends upon an input set which is generated by the other player.	Card game with multiplayer

Q. 3 Draw game tree of tic-tac-toe problem. **(10 Marks)**

Ans. :

Game Tree of tic-tac-toe problem

- Game tree is defined as a directed graph with nodes and edge. Here nodes indicate positions in a game and edges indicate next actions.
- A game tree is with the help of Tic-Tac-Toe example. Tic-Tac-Toe is a 2-player game, it is deterministic and every player plays when his/her turn come. Game tree has a Root Node which give the starting position of the game board, which is a blank 3 × 3 grid.
- Say in given example player 1 takes 'X' sign. Then MAX(X) indicates the board for the best single next move. Also it indicates that it is player 1's turn. (Remember that initial move is always indicated with a MAX).
- If the node is labeled as MIN then it means that it is opponents turn (i.e. player 2's turn) and the board for the best single next move is shown.
- Possible moves are represented with the help of lines. Last level shows terminal board position, which illustrates end of the game instance, Here, get zero as sum of all the play offs. Terminal state gives winning board position. Utility indicates the play off points gained by the player (– 1, 0, + 1).
- In a similar way can draw a game tree for any artificial intelligence based games.

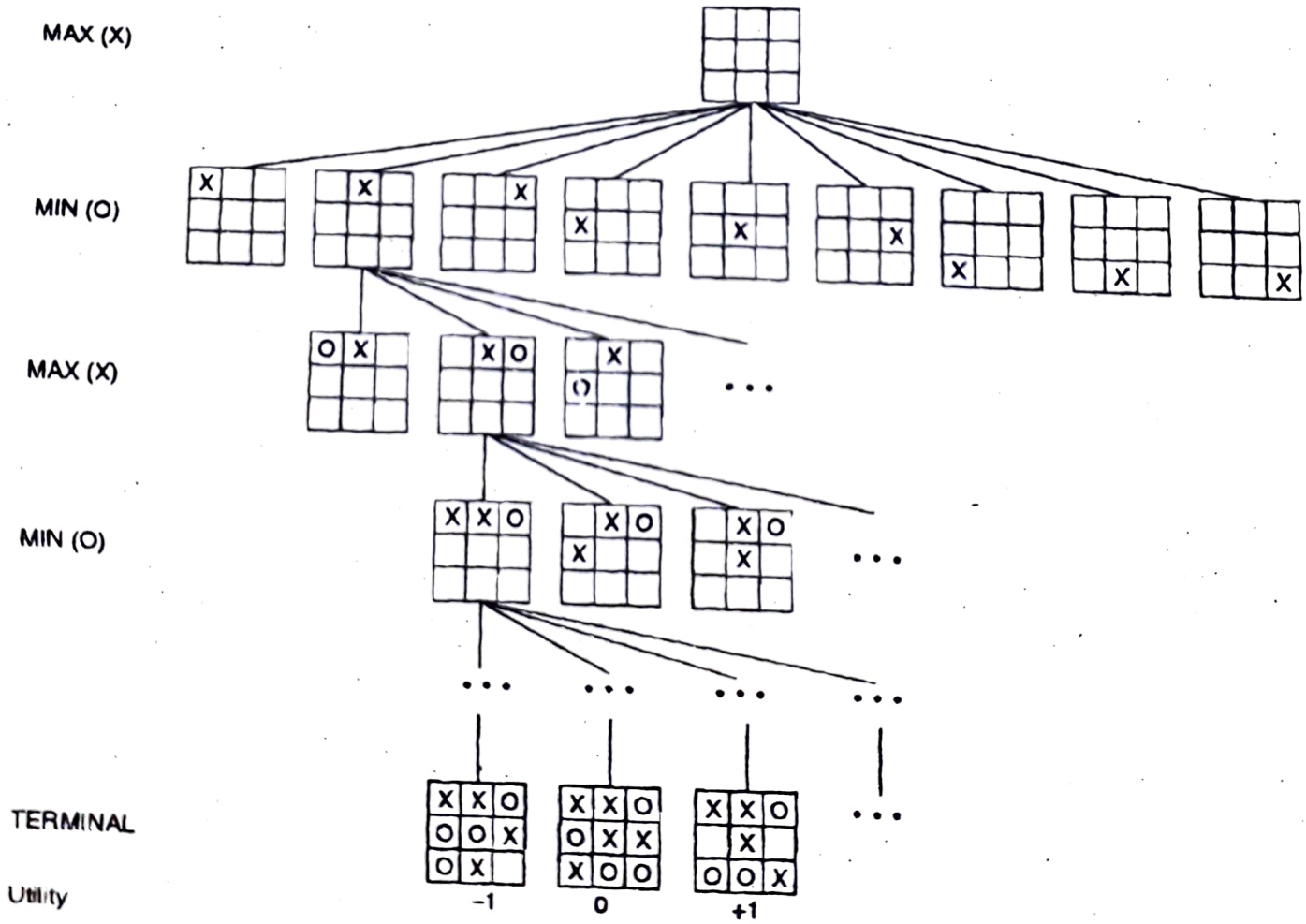


Fig. 3.1 : Tic-Tac-Toe Game Tree

Formulate a game as a search problem as follows :

Initial state	It gives the starting position of the game board.
Operators	They define the next possible actions.
Terminal state	Which indicates that all instance of game are over.
Utility	It displays a number which indicates if the game was won or lost or it was draw.

- From Tic-Tac-Toe game's example you can understand that for a 3 × 3 grid, two player game, where the game tree is relatively small (It has 9! terminal nodes), still cannot draw it completely on one single page.
- Imagine how difficult it would be to create a game tree for multi player games or for the games with bigger grid size.
- Many games have huge search space complexity. Games have limitation over the time and the amount of memory space it can consume. Finding an optimal solution is not feasible most of the times, so there is a need for approximation.
- Therefore there is a need for an algorithm which will reduce the tree size and eventually will help in reducing the processing time and in saving memory space of the machine.
- One method is **pruning** where, only the required parts, which improve quality of output, of the tree are kept and reaming parts are removed.
- Another method is **heuristic method** (it makes use of an **evaluation function**) it does not require exhaustive research. This method depends upon readily available information which can be used to control problem solving.

Q. 4 Explain minimax algorithm with an example. **SPPU : May 17, 5 Marks**

Ans. :

MiniMax Algorithm

- Minimax algorithm evaluates decision based on the present status of the game. This algorithm needs deterministic environment with perfect/exact information.
- Minimax algorithm directly implements the defining equation. Every time based on the successor state minimax value is calculated with the help of simple recursive computation.
- In case of minimax algorithm the selected action with highest minimax value should be equal to the best possible payoff (outcome) against best play.
- Take example of tic-tac toe game to understand minimax algorithm. We will take a random stage.

Step 1 : *Create an entire game tree including all the terminal states.*

Start action : 'O'

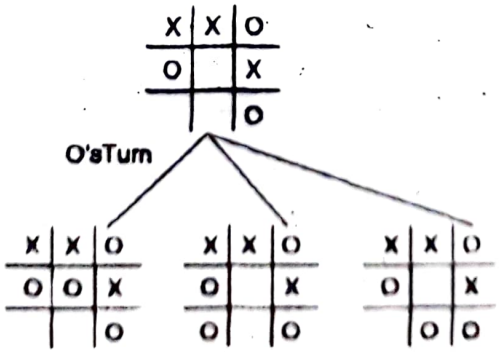


Fig. 3.2

Next action : 'X'

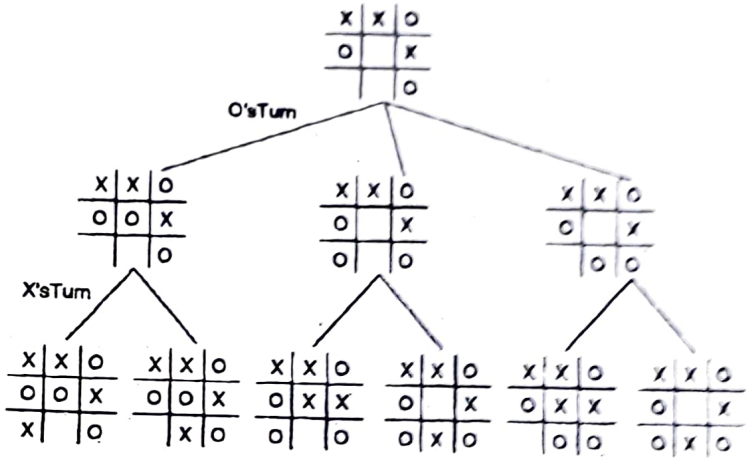


Fig. 3.3

Next action 'O'

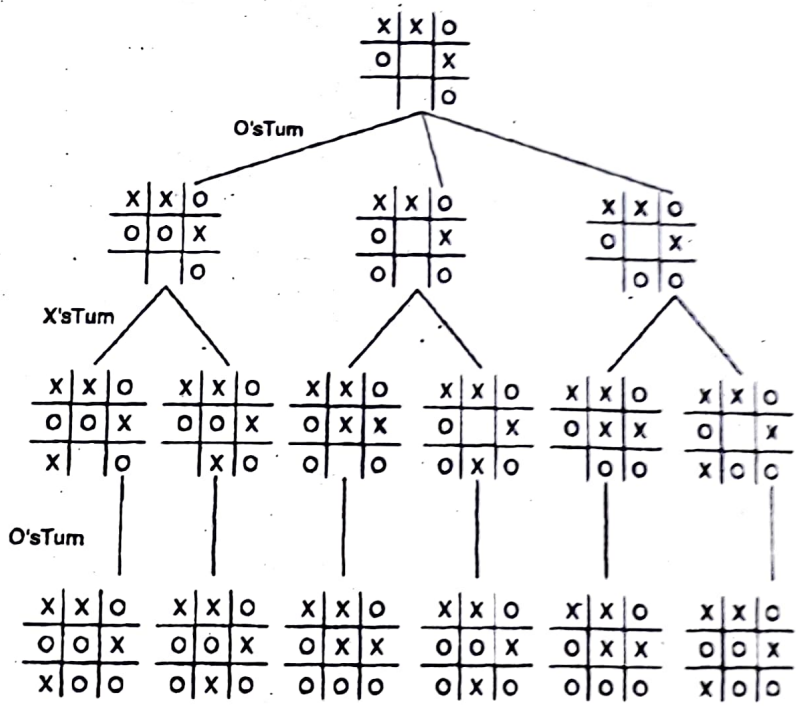


Fig. 3.4

Step 2 : For every terminal state find out utility (playoff points gained by every terminal state). Terminal position where 1 means win and 0 means draw.

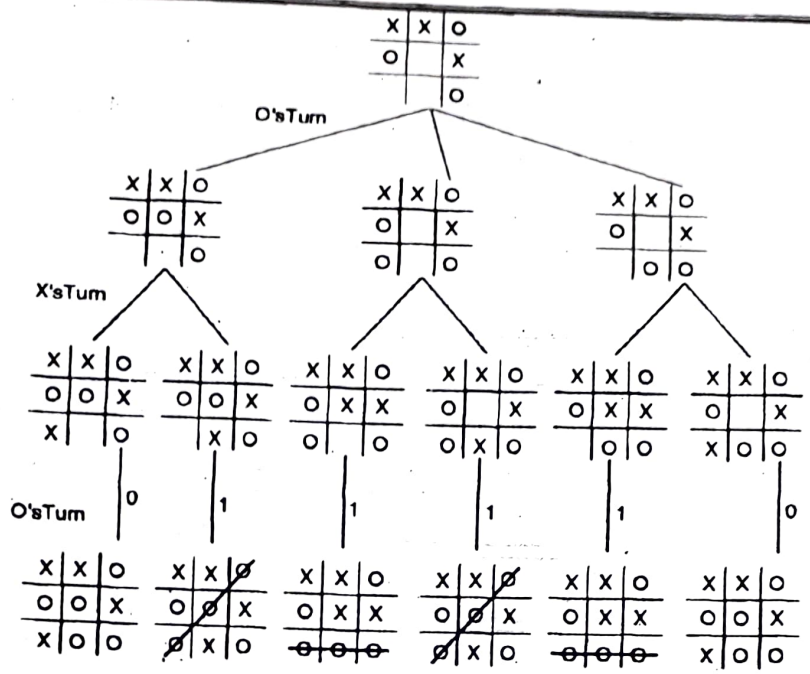


Fig. 3.5

Step 3 : *Apply MIN and MAX operators on the nodes of the present stage and propagate the utility values upward in the three.*

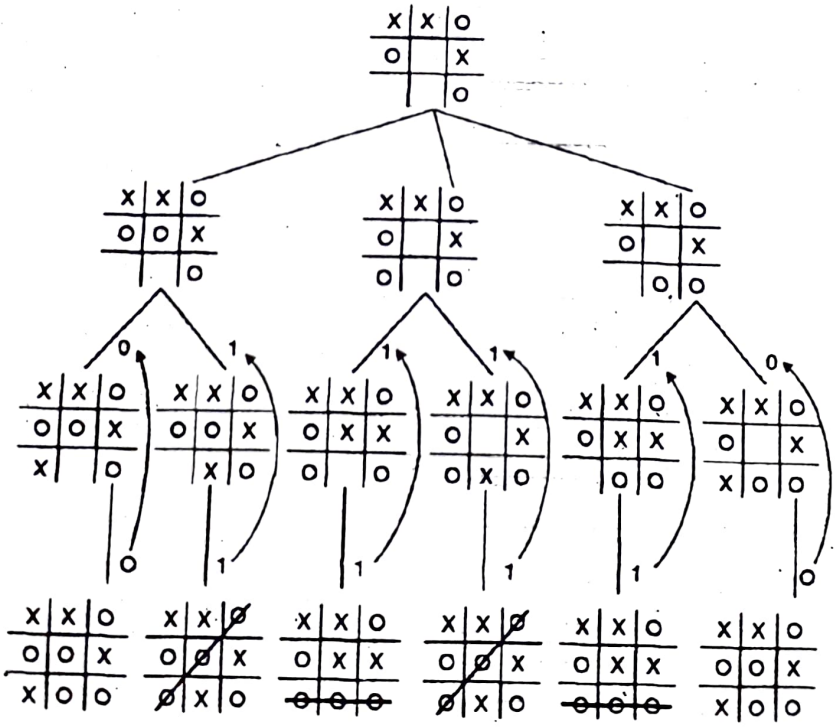


Fig. 3.6

Step 4 : With the max (of the min) utility value (payoff value) select the action at the root node using minimax decision.

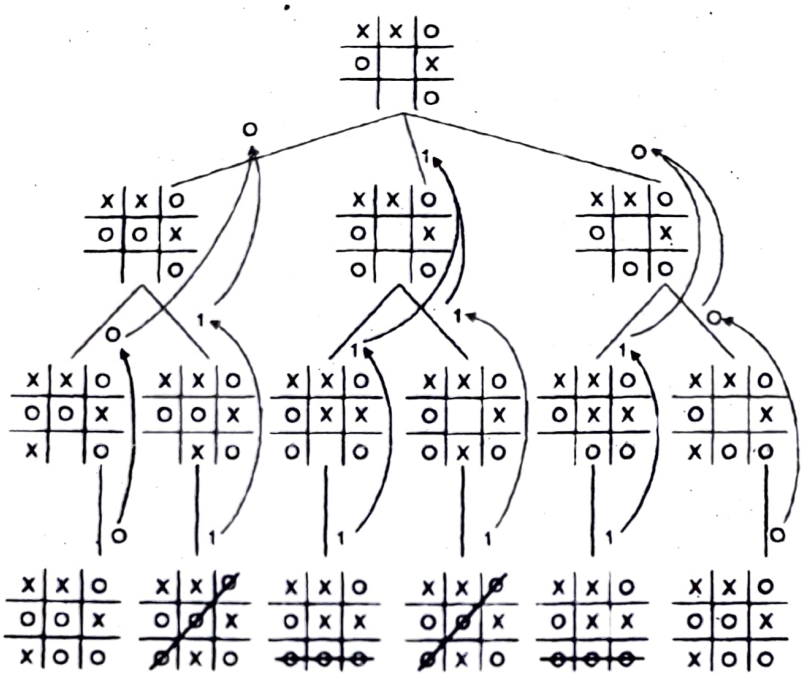


Fig. 3.7

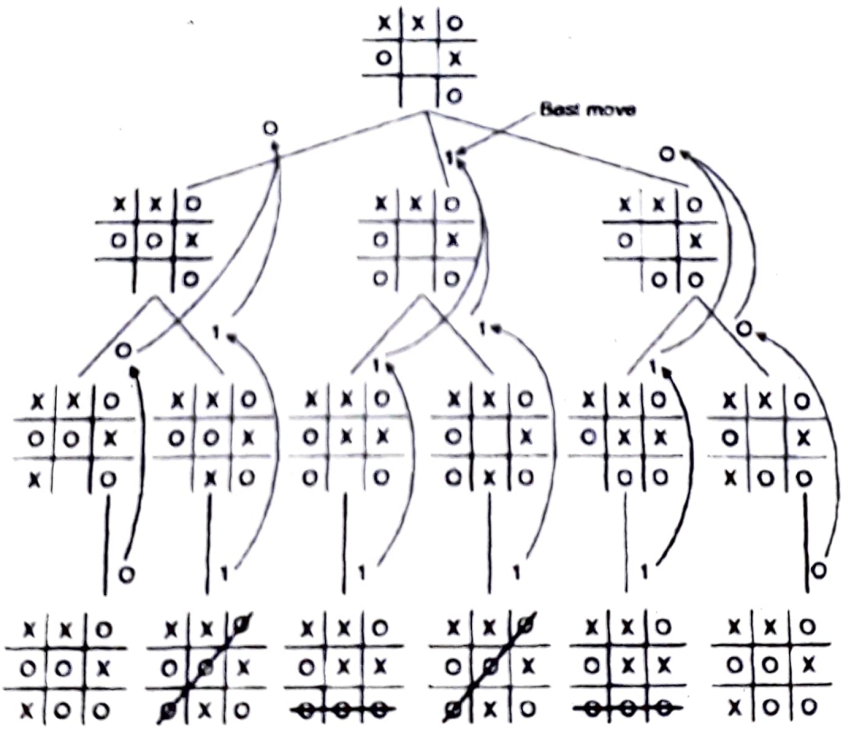


Fig. 3.8

(In case of Steps 2 and 3 assuming that the opponent will play perfectly as per our expectation)

Q. 5 Explain α-β pruning algorithm with an example. **SPPU : May 16, May 17, 10 Marks**

Ans. :

Alpha Beta Pruning

- Pruning means cutting off. In game search it resembles to clipping a branch in the search tree, probably which is not so fruitful.
- At any choice point along the path for max, α is considered as the value of the best possible choice found i.e., highest-value. For each "X", if "X" is worse i.e. lesser value than α value then, MAX will avoid it. Similarly we can define β value for MIN.
- α-β pruning is an extension to minimax algorithm where, decision making process need not consider each and every node of the game tree.
- Only the important nodes for quality output are considered in decision making. Pruning helps in making the search more efficient.
- **Pruning** keeps only those parts of the tree which contribute in improving the quality of the result remaining parts of the tree are removed.
- Consider the following game tree :

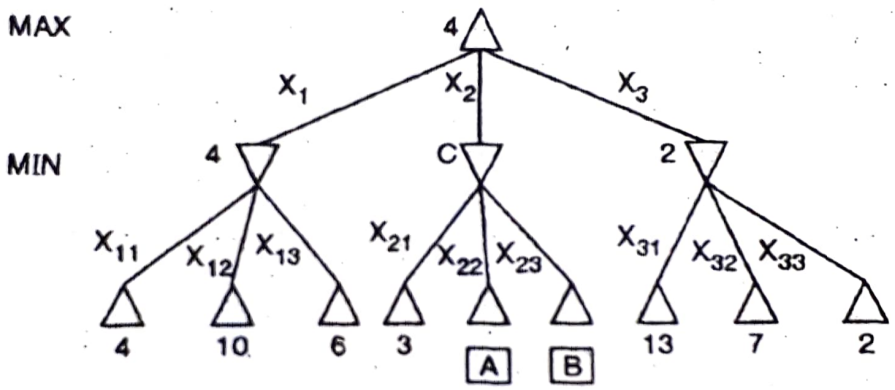


Fig. 3.9

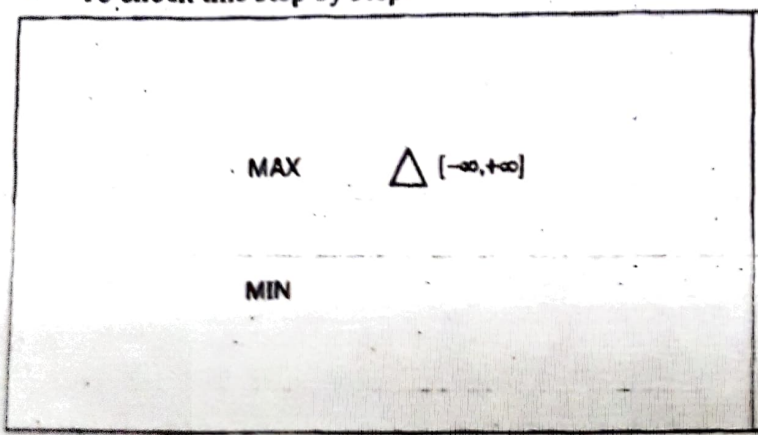
- For which we have to calculate minimax values for root node.

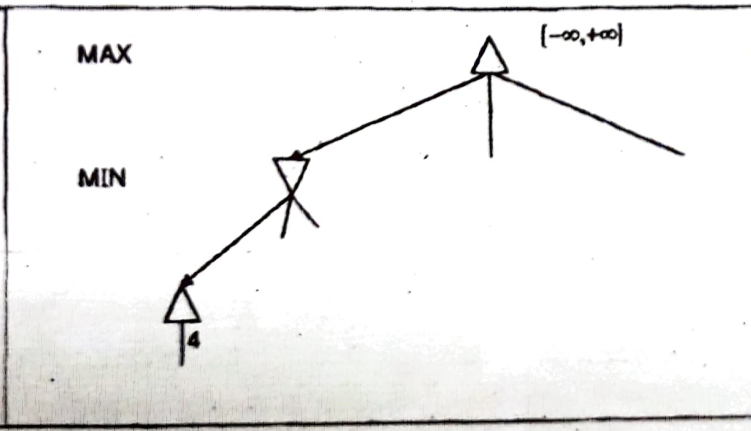
Minimax value of root node

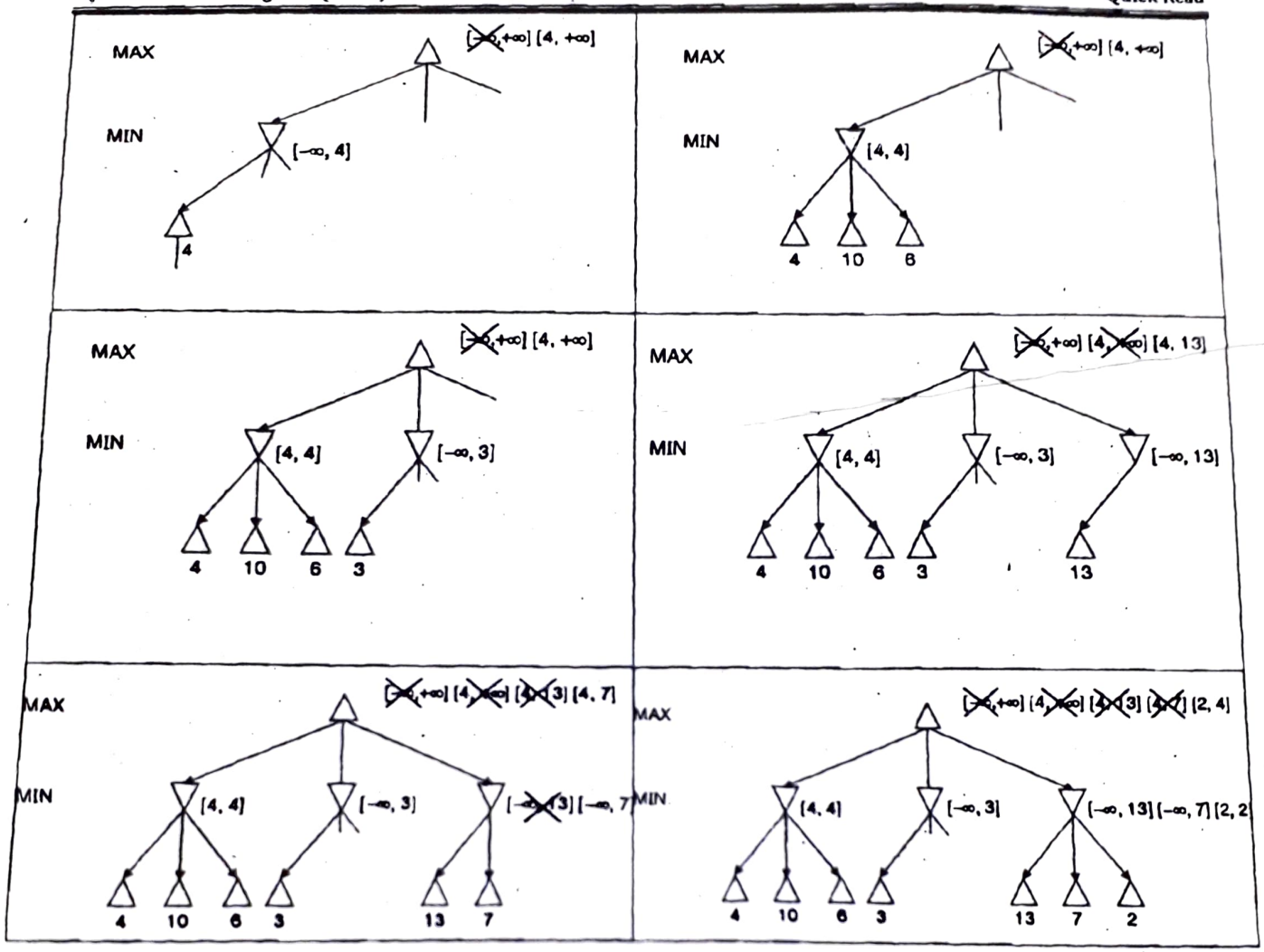
= Max(min(4,10,6), min(3,A,B), min(13,7,2))

= Max (4, C, 2) (C <=3) = 4

- To check this step by step.







So in this example pruned 2 β and 0 α branches. As the tree is very small, you may not appreciate the effect of branch pruning; but as consider any real game tree, pruning creates a significant impact on search as far as the time and space is concern.

Q. 6 Explain Min-Max and Alpha Beta pruning algorithm with following example. **(10 Marks)**

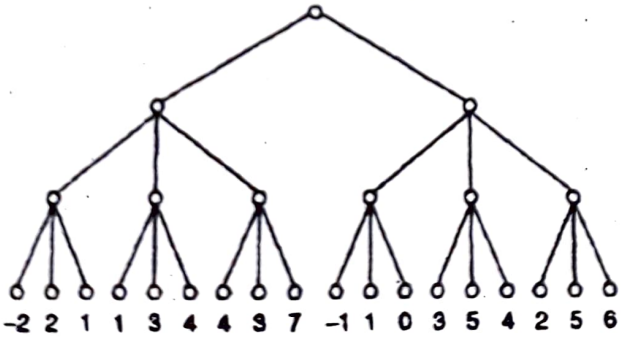
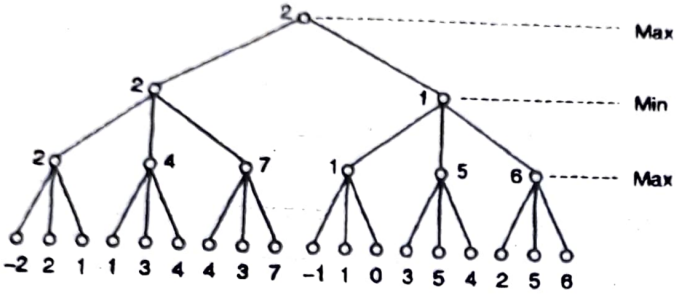
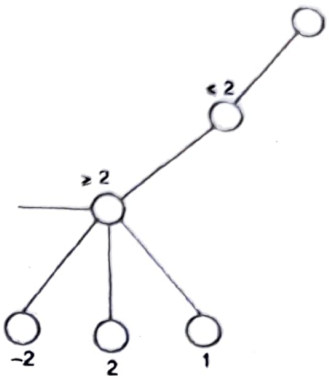


Fig. 3.10 : Game Tree

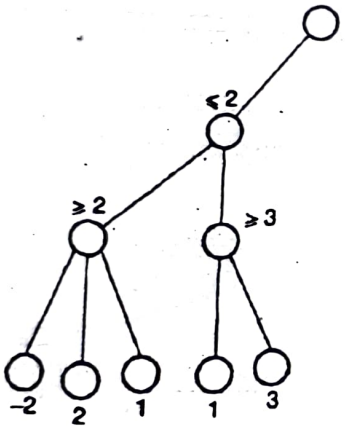
Ans. : Using min-max algorithm



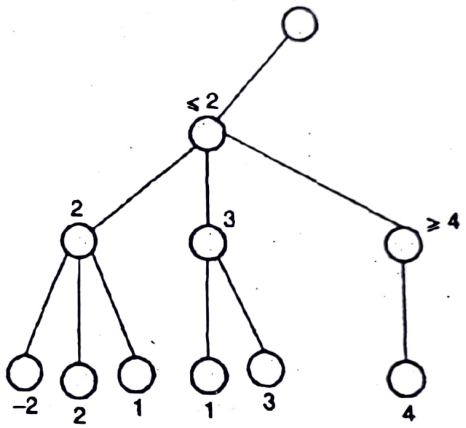
(a) Min-max solution with optimal path shown in bold



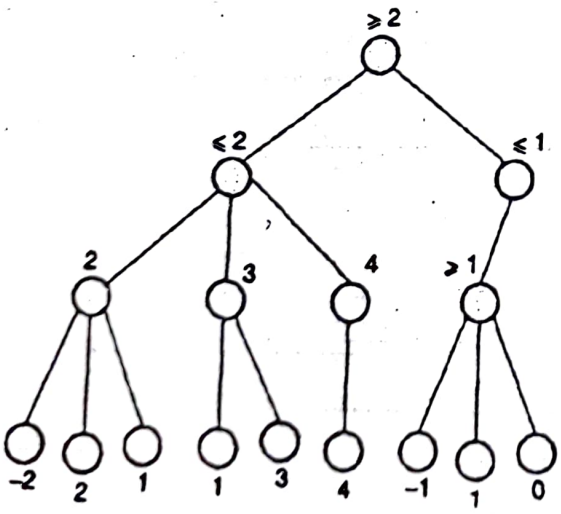
(b)



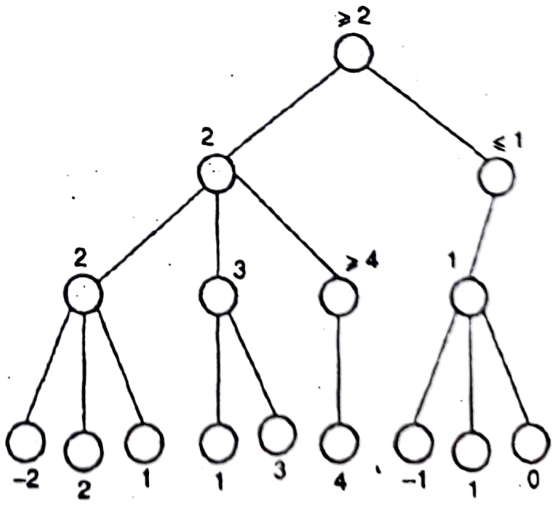
(c)



(d)

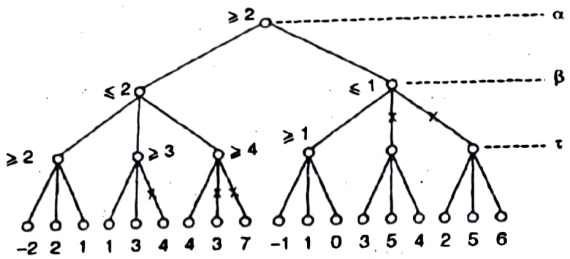


(e)



(f)

Fig. 3.11



(g) Solution by α-β pruning

Fig. 3.11

Total pruned branches

α - cuts = 2

β - cuts = 3

Q. 7 What is CSP? Give examples of real time CSPs. **(5 Marks)**

Ans. : Constraint :

Constraint is a logical relation among variables. Constraints arise in most areas of human endeavor. They are a natural medium for people to express problems in many fields.

Examples of constraints :

- The sum of three angles of a triangle is 180 degrees,
- The sum of the currents flowing into a node must equal zero.

Constraint satisfaction :

The Constraint satisfaction is a process of finding a solution to a set of constraints. Constraints articulate allowed values for variables, and finding solution is evaluation of these variables that satisfies all constraints.

Constraint Satisfaction Problems (CSPs) :

- In standard search problems, given with state space, heuristic function and goal state. In this case the state is represented in the form of any data structure that supports successor function, heuristic function, and goal test.
- Constraint Satisfaction Problems are special type of search in which, a state is defined by variables X_i with values from domain D_i and goal test is a set of constraints specifying allowable combinations of values for subsets of variables. Many real problems in AI can be modeled as Constraint Satisfaction Problems.CSP allows useful general purpose algorithms with more power than standard search algorithms CSPs are solved through search.

Examples of CSPs :

Some poplar puzzles like, map coloring problem, Latin Square, Eight Queens, and Sudoku are stated below.

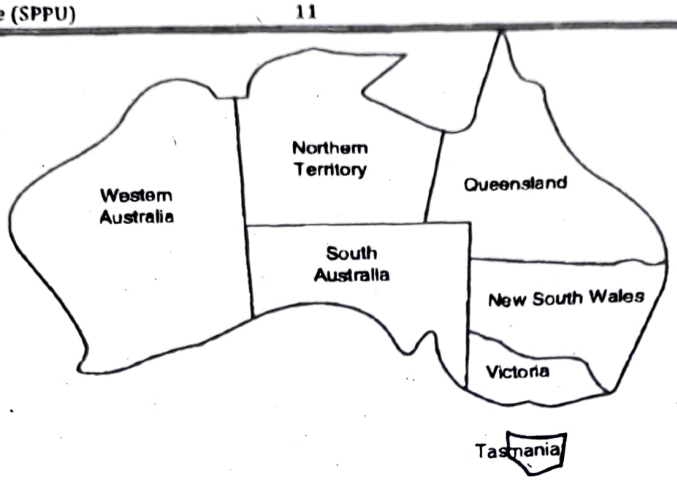
1. **Map coloring problem :**

In this example, Variables are WA, NT, Q, NSW, V, SA, T

Domains D_i = {red, green, blue}

Constraints : adjacent regions must have different colors.

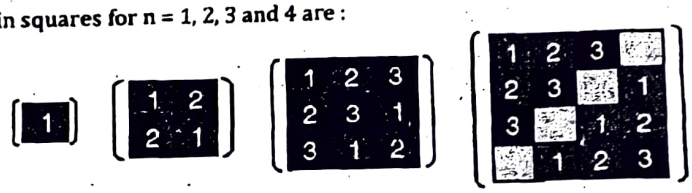
For example : WA ≠ NT, or (WA,NT) in {(red, green),(red, blue),(green, red), (green, blue), (blue, red), (blue, green)}



Solutions are complete and consistent assignments, e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

2. **Latin Square Problem :** How can one fill an n × n table with n different symbols such that each symbol occurs exactly once in each row and each column ?

   Solutions : The Latin squares for n = 1, 2, 3 and 4 are :



3. **Eight Queens Puzzle Problem :** How can one put 8 queens on a (8 × 8)chess board such that no queen can attack any other queen ?

   Solutions : The puzzle has 92 distinct solutions. If rotations and reflections of the board are counted as one, the puzzle has 12 unique solutions.

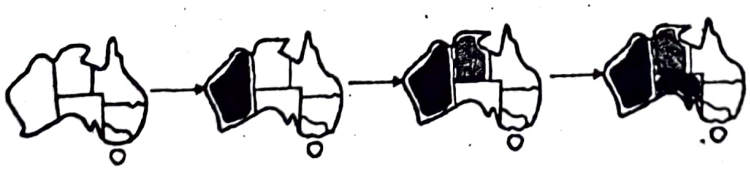
Q. 8 How to improve efficiency of backtracking in CSP? **(10 Marks)**

Ans. :

Improving Backtracking Efficiency

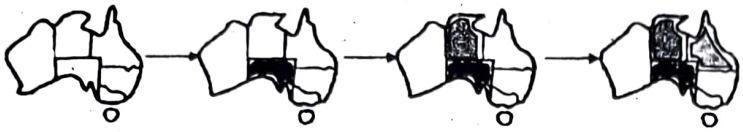
1. **Most constrained variable :**

   It choose the variable with the fewest legal values. By assigning this variable first, we can get a fair idea of other variable assignment. Also most of the constraints get satisfied by the assignment, hence chances of getting on wrong track are lowered down. This is also called as minimum remaining values (MRV) heuristic or Forced first heuristic.



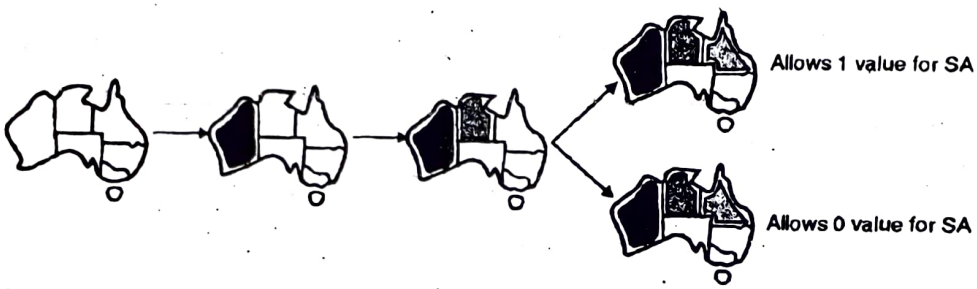
2. **Most constraining variable :**

This is the tie-breaker among most constrained variables. In Most constraining variable strategy first choose the variable with the most constraints on remaining variables. Example, the region surrounded by maximum number of regions. Once assign color to his part of the graph, it as good as decided the colors for all the surrounded parts.



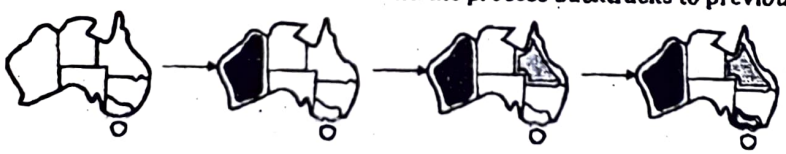
3. **Least constraining value :**

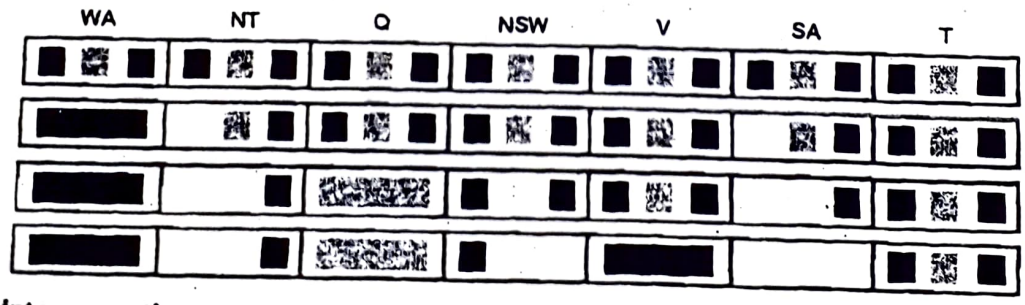
Given a variable, choose the least constraining value that is the one that rules out the fewest values for the remaining variables.



4. **Forward checking :**

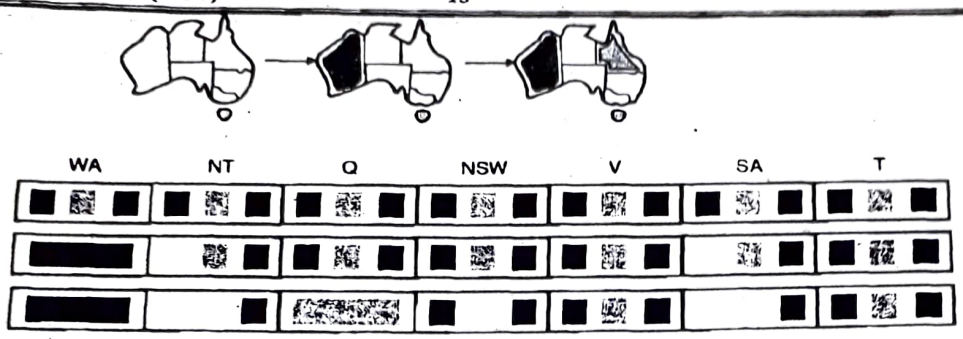
It keeps track of remaining legal values for unassigned variables. Terminate search when any variable has no legal values. As shown in the graph, as go on assigning colors to different regions, SA is not left with any valid color to assign. Hence the assignment will terminate and the process backtracks to previous state.





5. **Constraint propagation :**

Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures. In constraint propagation strategy, as we assign color to one of the part of the graph, the other parts are evaluated for the valid assignments. Hence we can detect the conflicts early. As shown the following example, As the region Q is getting assigned Green color, both NT and SA are left with blue color; but they both can't have same color as they are adjacent regions. Hence the wrong assignment of green to Q region is detected a step early.



NT and SA cannot both be blue!

Constraint propagation repeatedly enforces constraints locally.

6. **Arc consistency :**

In this technique, arcs are drawn from one variable to other if they have values satisfying the constraints. The aim is to make each arc consistent while propagating the constraints. The arc P → Q is consistent if and only if for every value p of 'P' there is some allowed value q. If 'P' loses a value, neighbors of 'P' need to be verified for permitted values and all arcs are again checked for consistency. Hence failures can be detected even early as compared to forward checking. We check arc consistency after every assignment.

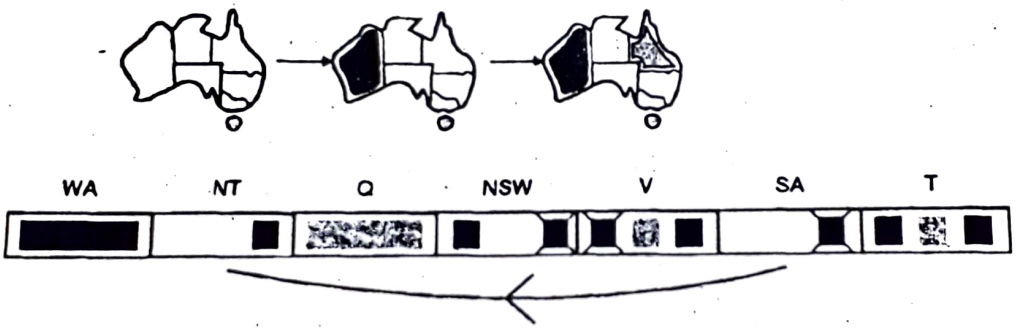


Fig. 3.12 : Arc Consistency

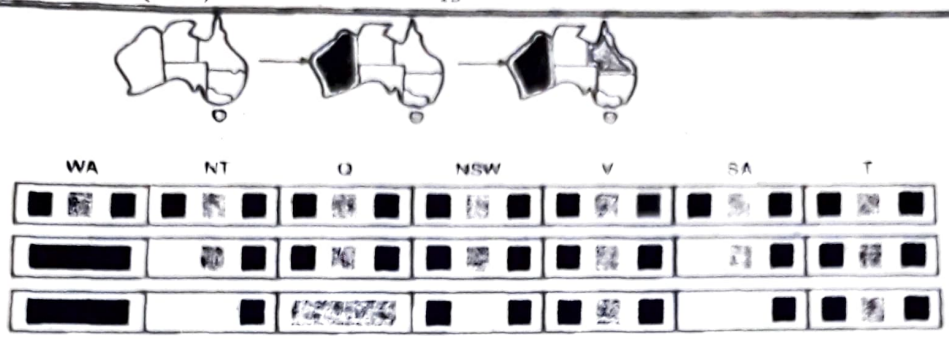
In Fig. 3.12 as WA and Q are assigned red and green colors respectively, only blue is left for NT and if it is assigned then there is no consistent value left for SA.

Q. 9 *How to solve crypt arithmetic problem? Explain with example.* **(5 Marks)**

Ans. :

Crypto-Arithmetic Problem

- A crypto-arithmetic problem is an example of Constraint Satisfaction Problem (CSP).
- It is a mathematical puzzle, where the numbers are represented by letters or symbols. Each letter represents a unique digit. The goal is to find the digits such that a given mathematical equation is verified. In general, there are few variables in form of letters which is to be assigned numeric values in the range of 0 to 9, such that the given equation hold true.
- To solve crypto-arithmetic problem, we have to generate the constraints by observing the given equation. Then we may be able to assign a single final value to one of the variable or can reduce the range of possibly allowed values for a particular variable. That helps in leading towards the solution.



NT and SA cannot both be blue!

Constraint propagation repeatedly enforces constraints locally.

6. **Arc consistency :**

In this technique, arcs are drawn from one variable to other if they have values satisfying the constraints. The aim is to make each arc consistent while propagating the constraints. The arc $P \rightarrow Q$ is consistent if and only if for every value p of 'P' there is some allowed value q. If 'P' loses a value, neighbors of 'P' need to be verified for permitted values and all arcs are again checked for consistency. Hence failures can be detected even early as compared to forward checking. We check arc consistency after every assignment.

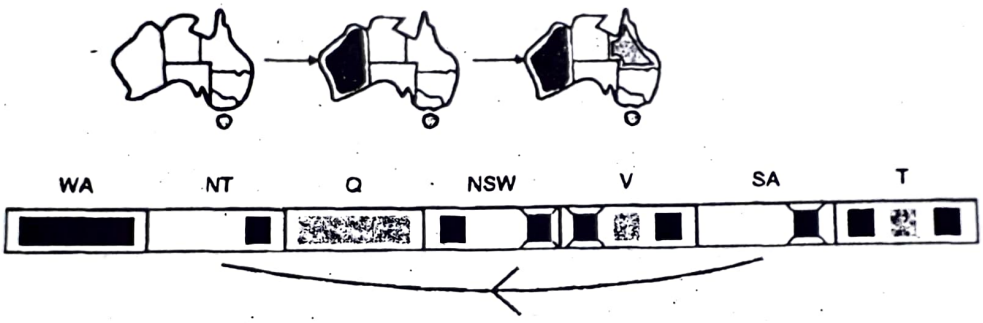


Fig. 3.12 : Arc Consistency

In Fig. 3.12 as WA and Q are assigned red and green colors respectively, only blue is left for NT and if it is assigned then there is no consistent value left for SA.

Q. 9 How to solve crypt arithmetic problem? Explain with example. **(5 Marks)**

Ans. :

Crypto-Arithmetic Problem

- A crypto-arithmetic problem is an example of Constraint Satisfaction Problem (CSP).
- It is a mathematical puzzle, where the numbers are represented by letters or symbols. Each letter represents a unique digit. The goal is to find the digits such that a given mathematical equation is verified. In general, there are few variables in form of letters which is to be assigned numeric values in the range of 0 to 9, such that the given equation hold true.
- To solve crypto-arithmetic problem, we have to generate the constraints by observing the given equation. Then we may be able to assign a single final value to one of the variable or can reduce the range of possibly allowed values for a particular variable. That helps in leading towards the solution.

- From the above problem one thing is clear that there is no specific algorithm to solve crypto-arithmetic, but it's a trial and error method based on backtracking strategy. Following example illustrates the procedure for solving crypto-arithmetic problems.

Example :

```
  I S
+ W T
+ S P M
-------
 B E I T
```

One of the possible solution is I = 3 S = 4 W = 7 T = 5 S = 9 P = 2 M = 6 B = 1 E = 0 as,

```
   3 4
+  7 5
+ 9 2 6
-------
 1 0 3 5
```

The above example has multiple solutions. Ideally, a good crypto-arithmetic puzzle must have only one solution.

Stage 1 : Describe

- In the describe stage explain the problem and the goal in natural language.
- For a complete problem description we need to answer following three questions :

1. What is the goal of the problem?

   In this example, the goal is to replace letters by digits such that the sum IS + WT + SPM = BEIT is verified.

2. Are there any unknowns or decision variables?

   The digits that the letters represent. In other words, for each letter we have one decision variable that can take any digit as value of that letter.

3. What are the constraints?

- The obvious constraint is the sum that has to be verified and all the variables must have different values in a feasible solution. But as the given equation, there are other implicit constraints as well.
- It is implicit that letters I, W, S and B cannot represent digit 0, as they are the first digit of a number.
- There are 9 distinct letters, so we need at least 9 different digits in the answer.
- By observation, we have B = 1, Consider three auxiliary variables, X, Y, Z. Hence we can write the constraint equations as,

$$S + T + M = 10X + T$$
$$I + W + P + X = 10Y + I$$
$$S + Y = 10B + E$$

- **Solve :** A quadratic equations representing the problem, we can solve them simultaneously to get the solution.

Example 2 :

```
  T W O
+ T W O
-------
 F O U R
```

E.g., setting F = 1, O = 4, R = 8, T = 7, W = 3, U = 6 gives 734 + 734 = 1468

Trick : introduce auxiliary variables X, Y

$O + O = 10X + R$; $W + W + X = 10Y + U$

$T + T + Y = 10F + O$

Also need pair wise constraints between original variables if they are supposed to be different.

Q. 10 *Write a short note on : Relevant aspects of AI games.* **(5 Marks)**

Ans. :

Relevant Aspects of AI Game

- **Accessible environments** : Games with accessible environments have all the necessary information handy. For example : Chess.
- **Search** : Also there are games which require search functionality which illustrates how players have to search through possible game positions to play a game. For example : minesweeper, battleships.
- **Unpredictable opponent** : In AI games opponents can be unpredictable, this introduces uncertainty in game play and thus game-playing has to deal with contingency / probability problems.

For example : Scrabble.

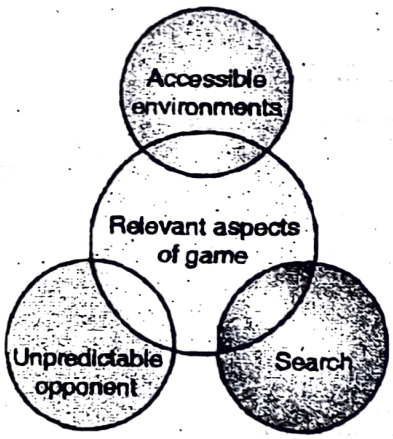


Fig. 3.13 : Relevant aspects of AI game

Q. 11 *Write a short notes on : Game types.* **(5 Marks)**

Ans. :

Type of Games :

Game can be classified under deterministic or probabilistic category.

1. **Deterministic**
   - It is a fully observable environment. When there are two agents playing the game alternatively and the final results of the game are equal and opposite then the game is called deterministic.
   - Take example of tic-tac-toe where two players play a game alternatively and when one player wins a game then other player losses game.
2. **Probabilistic**
   - Probabilistic is also called as non-deterministic type. It is opposite of deterministic games, where you can have multiple players and you cannot determine the next action of the player.
   - Another way of classification for games can be based on exact/perfect information or based on inexact / approximate information.

1. **Exact/perfect information** : Games in which all the actions are known to other player is called as game of exact or perfect information. For example tic-tac-toe or board games like chess, checkers.
2. **Inexact / approximate information** : Game in which all the actions are not known to other players (or actions are unpredictable) is called game of inexact or approximate information. In this type of game, player's next action depends upon who played last, who won last hand, etc. For example card games like hearts.

- Consider following games and they are classified into various types of games based on the parameters :

Table 3.2 : Types of game

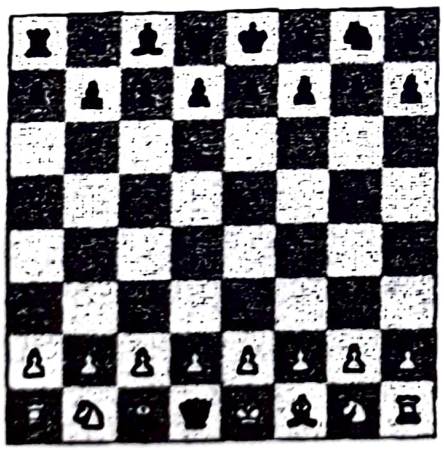
	Exact/perfect information	Inexact / approximate information
Deterministic	• Chess • Checkers	• Battleships • Card Game (Hearts)
Probabilistic	• Ludo • Monopoly	• Scrabble • Poker

Q.12 Write a short note on : 1. Chess and 2. Checkers. **(10 Marks)**

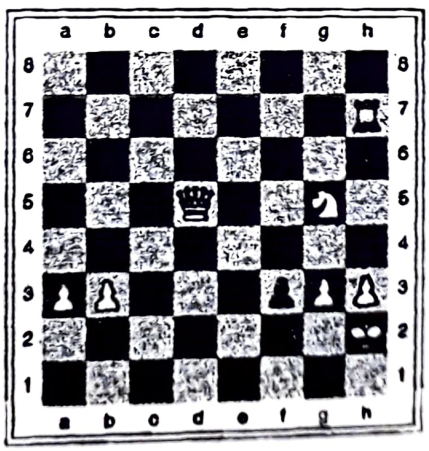
Ans.:

1. Chess

- Chess comes under deterministic and exact/perfect information category. This game is a two person, zero-sum game.
- In chess both players can see chess board positions so there is no secrecy and players don't play at the same time they play one after the other in an order.
- Thus this game has perfect environment to test artificial intelligence techniques. In 1990's a computer Deep Blue II defeated Garry Kasparov who was world champion at that time. This example is given to understand how artificial intelligence can be used in decision making.



(a) Chess board



(b) Deep Blue II vs Garry Kasparov final position game 1

Fig. 3.14

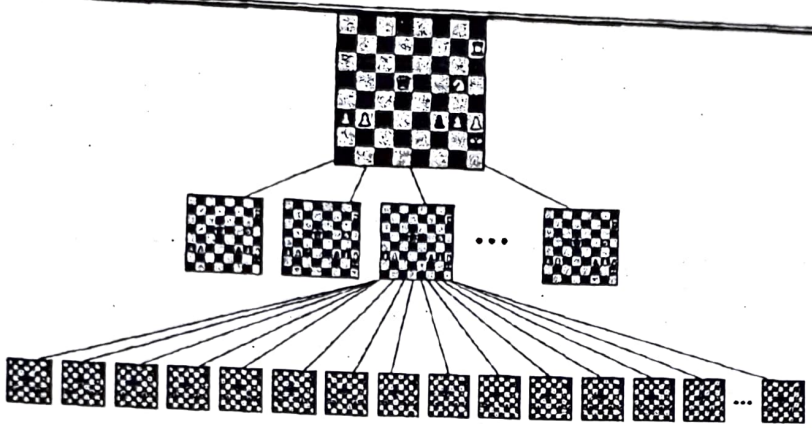


Fig. 3.15 : Chess game tree

2. Checkers

- Checkers comes under deterministic and exact/perfect information category. This game is a two person game where both players can see board positions, so there is no secrecy and players play one after the other in an order.
- In 1990's a computer program name Chinook (also called draughts) was developed which defeated human world champion Marion Tinsley.

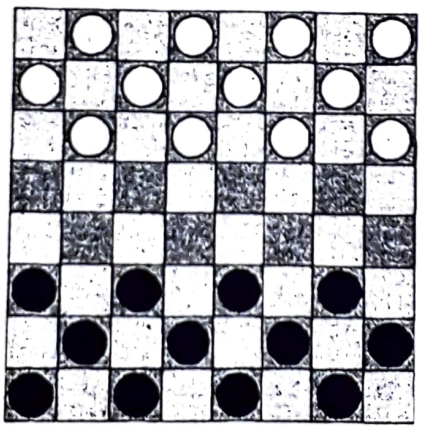


Fig. 3.16 : Checkers board

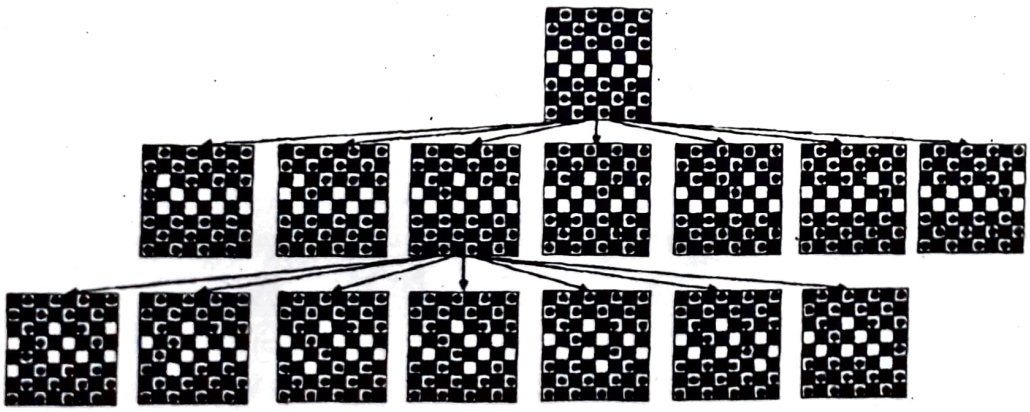


Fig. 3.17 : Checkers game tree

Q.13 Perform α - β cutoff on the following. **(5 Marks)**

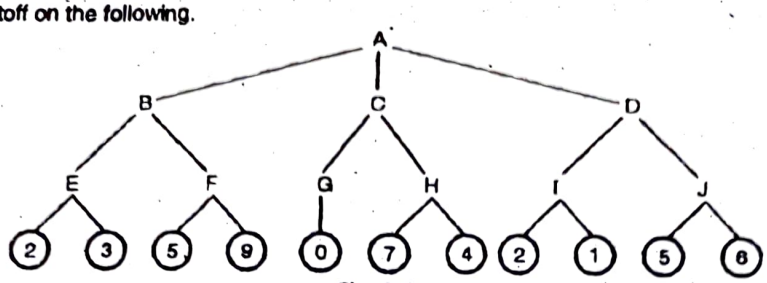


Fig. 3.18

Ans. :

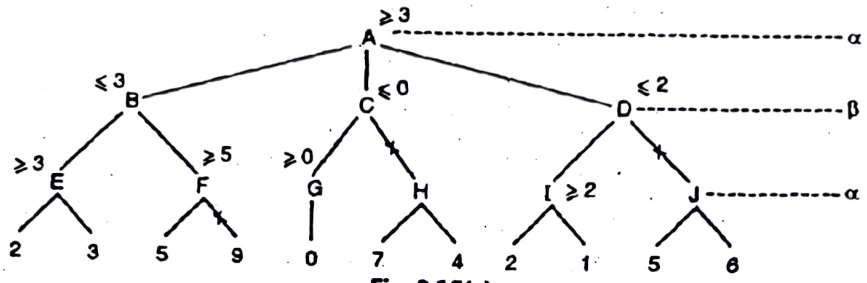


Fig. 3.18(a)

No. of α - cuts = 1 No. of β - cuts = 2

Q. 14 Apply alpha-beta pruning on example given in Fig. 3.19 considering first node as max. **(10 Marks)**

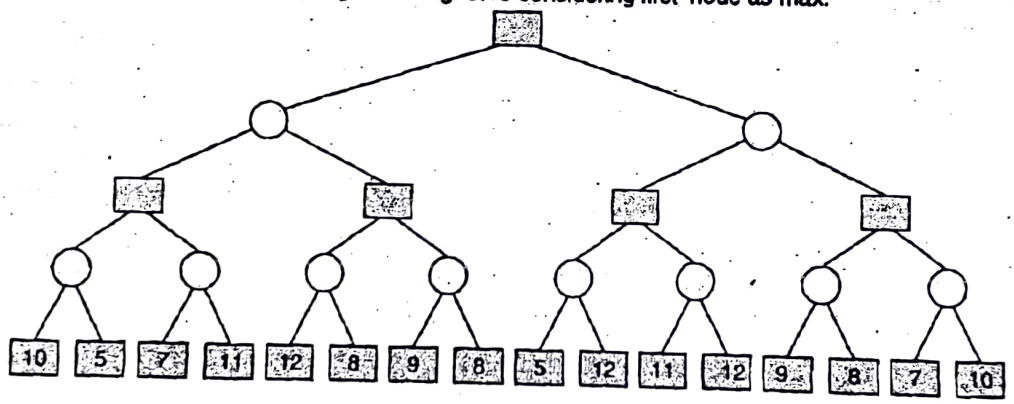


Fig. 3.19

Ans. :

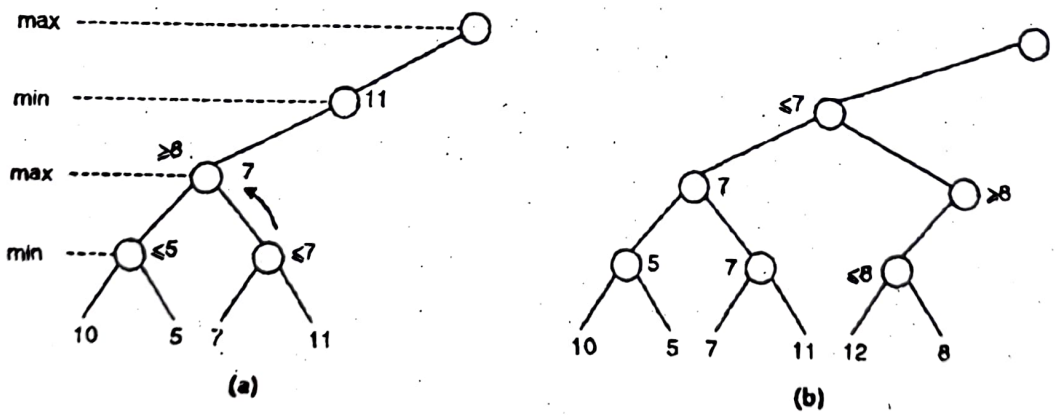


Fig. 3.19

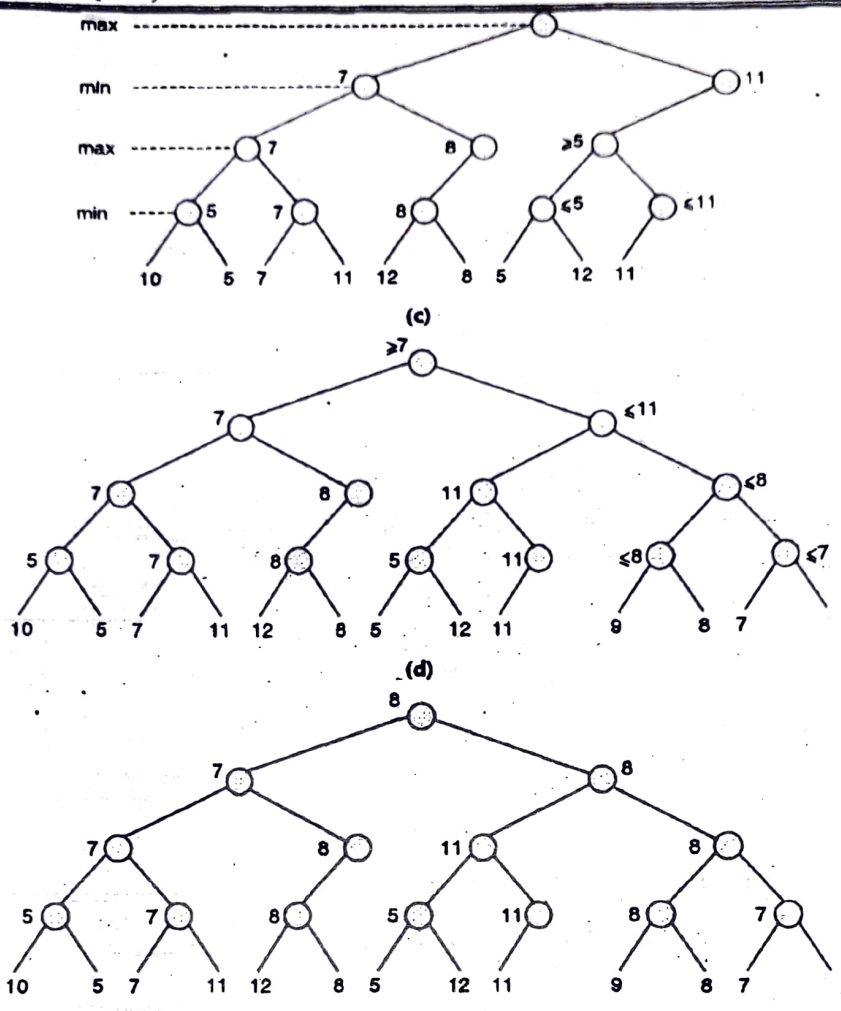


Fig. 3.19

Q. 15 Give α-β pruning algorithm properties. **(2 Marks)**

Ans.:

Properties of α-β

- Final results are not affected by pruning.
- Ordering of Good actions helps in improving effectiveness of pruning technique.
- If there is exact/perfect ordering then we can get time complexity as $O(b^{m/2})$.
- Depth of search is doubled with pruning.

Q.16 Show the solution for map coloring using constraint satisfaction for following states of India : Maharashtra, Madhya Pradesh, Andhra Pradesh, Karnataka, Goa and Gujarat. What is the optimal number of colors required for coloring? **(5 Marks)**

Ans. :

No. of regions = 6 (Guj, MP, MH, AP, KAR, Goa)

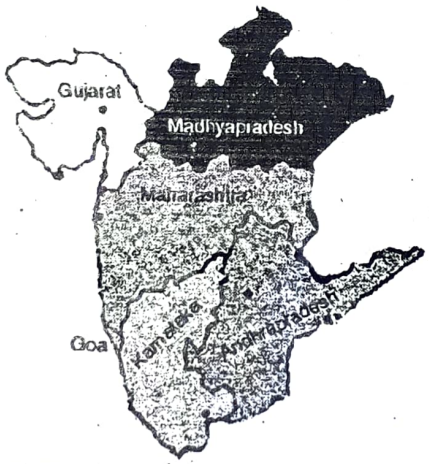
Constraints = Guj ≠ MP ≠ MAH

MAH ≠ GOA ≠ KAR

KAR ≠ AP ≠ MAH

GUJ	MP	MAH	GOA	KAR	AP
Red	Red	Red	Red	Red	Red
	Blue		Blue		Blue
			Green		Green
					Yellow

Hence, the optimal number of colours required to colour this map is '4'.



□□□

Unit IV : Knowledge

Q. 1 Explain in detail the knowledge based agent. **(10 Marks)**

Ans. :

Architecture of a KB Agent

Knowledge based agents can be implemented at three levels namely, knowledge level, logical level and implementation level.

1. Knowledge level
2. Logical level
3. Implementation level

1. Knowledge level :

- It is the most abstract level of agent implementation. The knowledge level describes agent by saying what it knows. That is what knowledge the agent has as the initial knowledge.
- Basic data structures and procedures to access that knowledge are defined in his level. Initial knowledge of knowledge base is called as background knowledge.
- Agents at the knowledge level can be viewed as an agent for which one only need to specify what the agent knows and what its goals are in order to specify its behavior, regardless of how it is to be implemented.
- **For example :** A taxi driving agent might know that the Golden Gate Bridge connects San Francisco with the marin county.

2. Logical level :

- At the logical level, the knowledge is encoded into sentences. This level uses some formal language to represent the knowledge the agent has. The two types of representations we have are propositional logic and first order or predicate logic.
- **For example :** Links (Golden Gate Bridge, San Francisco, Marin County).

3. Implementation level :

- In implementation level, the physical representation of logical level sentences is done. This level also describes data structures used in knowledge base and algorithms that used for data manipulation.
- **For example :** Links (Golden Gate Bridge, SanFrancisco, Marin County).

```
function KB – Agent (percept) returns an action
    static : KB, a knowledge base
             t, a counter, initially o, indicating time
    TELL (KB, MAKE – PERCEPT-SENTENCE(percept, t))
    action ← ASK (KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

Fig. 4.1 : General function of knowledge based agent

- Fig. 4.1 is the general implementation of knowledge based agent. TELL and ASK are the sub procedures implemented to perform the respective actions.
- The knowledge base agent must be able to perform following tasks :
  - Represent states, actions, etc.
  - Incorporate new precepts.
  - Update internal representations of the world.
  - Deduce hidden properties of the world.
  - Deduce appropriate actions.

Q. 2 Explain WUMPUS World Environment giving its PEAS description.

SPPU : Dec. 15, May 16, May 17, Dec. 17, May 18. 10 Marks

Ans. :

The WUMPUS World Environment

- Fig. 4.2 shows the WUMPUS world. WUMPUS is an early computer game also known as "Hunt the Wumpus". WUMPUS was developed by Gregory Yob in 1972/1973. It was originally written in **BASIC (Beginner's All-purpose Symbolic Instruction Code).**
- WUMPUS is a map-based game. Let's understand the game :
  - WUMPUS world is like a cave which represents number of rooms, rooms, which are connected by passage ways. Take a 4 × 4 grid to understand the game.
  - WUMPUS is a monster who lives in one of the rooms of the cave. WUMPUS eats the player (agent) if player (agent) comes in the same room. Fig. 4.2 shows that room (3, 1) where WUMPUS is staying.
  - Player (agent) starts from any random position in cave and has to explore the cave. We are starting from (1, 1) position.
- There are various sprites in the game like pit, stench, breeze, gold, and arrow. Every sprite has some feature. Let's understand this one-by-one :
  - Few rooms have bottomless pits which trap the player (agent) if he comes to that room. You can see in the Fig. 4.2 shows that room (1, 3), (3, 3) and (4, 4) have bottomless pit. Even WUMPUS can fall into a pit.

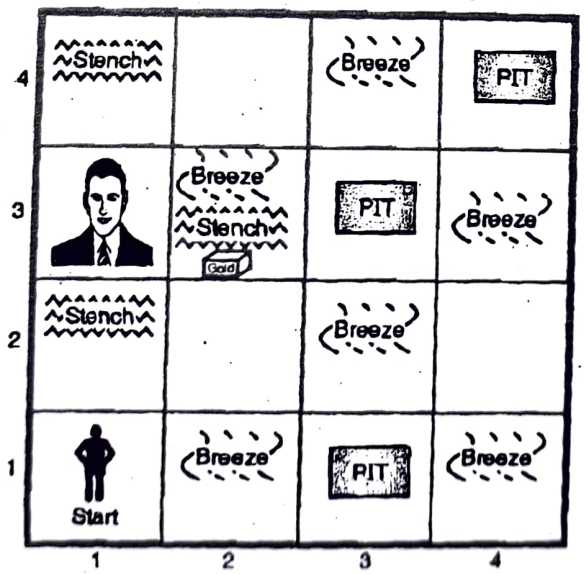


Fig. 4.2 : The WUMPUS World

- Stench experienced in a room which has a WUMPUS in its neighborhood room. See the Fig. 4.2, here room (2, 1),(3, 2) and (4, 1) have Stench.
- Breeze is experienced in a room which has a pit in its neighborhood room. Fig. 4.2 shows that room (1, 2), (1, 4), (2, 3), (3, 2), (3, 4) and (4, 3) consists of Breeze.
- Player (Agent) has arrows and he can shoot these arrows in straight line to kill WUMPUS.
- One of the rooms consists of gold, this room glitters. Fig. 4.2 shows that room (3, 2) has Gold.
- Apart from above features player (agent) can accept two types of percepts which are: Bump and scream. A bump is generated if player (agent) walks into a wall. While a sad scream created everywhere in the cave when the WUMPUS is killed.

Description of the WUMPUS World

- An agent receives percepts while exploring the rooms of cave. Every percepts can be represented with the help of five element list, which is [stench, breeze, glitter, bump, scream]. Note that player (agent) cannot perceive its own location.
- If the player (agent) gets percept as [Stench, Breeze, None, None, None]. Then it means that there is a stench and a breeze, but no glitter, no bump, and no scream in the WUMPUS world at that position in the game.
- Let's take a look at the actions which can be performed by the player (agent) in *WUMPUS World* :
  - Move : To move in forward direction,
  - Turn : To turn right by 90 degrees or left by 90 degrees,
  - Grab : To pick up gold if it is in the same room as the player (agent),
  - Shoot : To Shoot an arrow in a straight line in the direction faced by the player (agent).
- These actions are repeated till the player (agent) kills the WUMPUS or if the player (agent) is killed. If the WUMPUS is killed then it is a winning condition, else if the player (agent) is killed then it is a losing condition and the game is over.
- Game developer can keep a restriction on the number of arrows which can be used by the player (agent). So if we allow agent to have only one arrow, then only the first shoot action will have some effect. If this shoot action kills the WUMPUS then you win the game, otherwise it reduces the probability of winning the game.
- Lastly there is a die action : It takes places automatically if the agent enters in a room with a bottomless pit or in a room with WUMPUS. Die action is irreversible.

Q. 3 Give PEAS descriptors for WUMPUS world. **(5 Marks)**

Ans. :

PEAS Properties of WUMPUS World

1. **Performance measure**
   - + 100 for grabbing the gold and coming back to the starting position,
   - – 200 if the player (agent) is killed.
   - – 1 per action,
   - – 10 for using the arrow.
2. **Environment**
   - Empty Rooms.
   - Room with WUMPUS.

- Rooms neighbouring to WUMPUS which are smelly.
- Rooms with bottomless pits
- Rooms neighbouring to bottomless pits which are breezy.
- Room with gold which is glittery.
- Arrow to shoot the WUMPUS.

3. **Sensors (assuming a robotic agent)**
   - Camera to get the view
   - Odour sensor to smell the stench
   - Audio sensor to listen to the scream and bump.
4. **Effectors (assuming a robotic agent)**
   - Motor to move left, right
   - Robot arm to grab the gold
   - Robot mechanism to shoot the arrow.

The WUMPUS world agent has following characteristics :

1. Fully observable
2. Deterministic
3. Episodic
4. Static
5. Discrete
6. Single agent

Q. 4 What is reasoning ? What is its role in artificial intelligence ? **(5 Marks)**

Ans. :

Logic/Reasoning

- Logic can be called as reasoning which is carried out or it is a review based on strict rule of validity to perform a specified task.
- In case of intelligent systems we say that any of logic's particular form cannot bind logical representation and reasoning, they are independent of any particular form of logic.
- Make a note that logic is beneficial only if the knowledge is represented in small extent and when knowledge is represented in large quantity the logic is not considered valuable.
- Fig. 4.3 depicts that sentences are physical configurations of an agent, also it is shows that sentences need sentence. This means that reasoning is a process of forming new physical configurations from old ones.

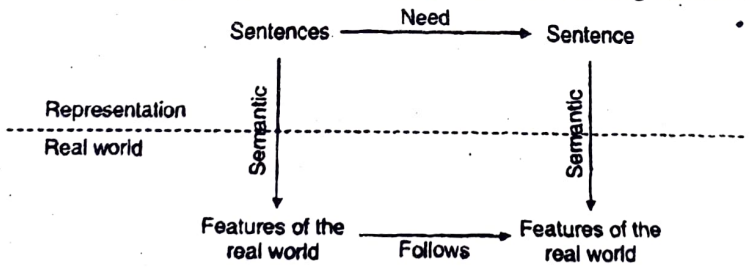


Fig. 4.3 : Correspondence between real world and its representation

- Logical reasoning should make sure that the new configurations represent features of the world that actually follow the features of the world that the old configurations represented.

Role of Reasoning in AI

- Fig. 4.4 shows how logic can be seen as a knowledge representation language. There are various levels to the logic and most fundamental type of logic is propositional logic.

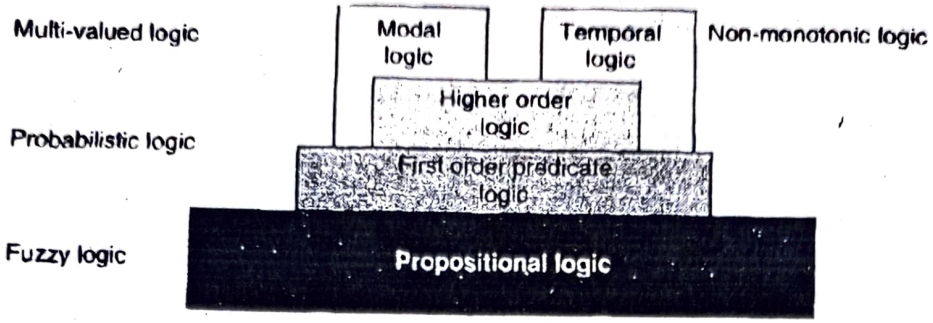


Fig. 4.4 : Logic as Knowledge Representation language

- Propositional logic can be considered at fuzzy logic level, where rules are values between range of 0 and 1. Next level is also called as probabilistic logic level using which first order predicate logic is implemented.
- In this Fig. 4.4 that there are two more levels above higher order logic which are multi-valued and non-monotonic logic levels and they consist of modal logic and temporal logic respectively. All these types of logic are basic building blocks of intelligent systems and they all use reasoning in order to represent sentences. Hence reasoning plays a very important role in AI.

Q. 5 Explain various method of knowledge representation techniques. **(10 Marks)**

Ans. :

Representation of Knowledge using Rules

- **Knowledge can be considered to be represented at generally two levels :**
  - **(i) Knowledge level :** This level describes the facts.
  - **(ii) Symbol level :** This level deals with using the symbols for representing the objects, which can be manipulated in programs.
- **Knowledge can be represented using the following rules :**

(a) Logical representations
(b) Production rule representations.
(c) Semantic networks
(d) Frame representations

(a) Logical representation

- The logical representations are mostly concerned with truth of statements regarding the world. These statements are most generally represented using statements like TRUE or FALSE.
- Logic is successfully used to define ways to infer new sentences from the existing ones. There are certain logics that are used for the representation of information, and range in terms of their expressiveness. There are logic that are more expressive and are more useful in translation of sentences from natural languages into the logical ones. There are several logics that are widely used :
  1. **Propositional logic :** These are restricted kinds that make use of propositions (sentences that are either true or false but not both) which can be either true or false. Proposition logic is also known as propositional calculus, sentential calculus or boolean algebra.

All propositions are either true or false,

For example :

(i) Leaves are green (ii) Violets are blue.

Sentence	Truth Value	Proposition
Sky is blue	true	yes
Roses are red	true	yes
2 + 2 = 5	false	yes

2. **First Order Predicate Logic :** These are much more expressive and make use of variables, constants, predicates, functions and quantifiers along with the connectives.
3. **Higher Order Predicate Logic :** Higher order predicate logic is distinguished from first order predicate logic by using additional quantifiers and stronger semantics.
4. **Fuzzy Logic :** These indicate the existence of in between TRUE and FALSE or fuzziness in all logics.
5. **Other Logic :** These include multiple valued logic, modal logics and temporal logics.

(b) Production Rule Representation

One of the widest used methods to represent knowledge is to use production rules, it is also known as IF-THEN rules.

Syntax :

IF condition THEN action

IF premise THEN conclusion

IF proposition p1 and proposition p2 are true

THEN proposition p3 is true

Example :

IF pressure is high, THEN volume is small.

IF the road is slippery, THEN driving is dangerous.

- Some of the benefits of IF-THEN rules are that they are modular, each defining a relatively small and, at least in principle, independent piece of knowledge. New rules may be added and old ones deleted usually independently of other rules.
- Production rules are simple but powerful forms of representing knowledge, they provide flexibility for combining procedural and declarative representations in a unified manner. The major advantage of production rules are that they are modular, independent of other rules with the provision for addition new rules and deleting older ones.

(c) Semantic networks

- These represent knowledge in the form of graphical networks, since graphs are easy to be stored inside programs as they are concisely represented by nodes and edges.
- A semantic network basically comprises of nodes that are named and represent concepts, and labelled links representing relations between concepts. Nodes represent both types and tokens.
- For example, the semantic network in Fig. 4.5 expresses the knowledge to represent the following data :
  - Bom is grey in color.
  - Tom is owned by Sam.

- Tom is a Mammal.
- Fish is an Animal.
- Cats love Milk.
- All mammals are animals.

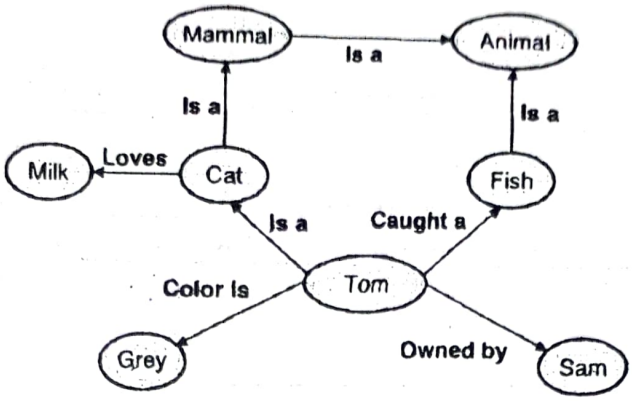


Fig. 4.5

- **Conceptual Graph** : It is a recent scheme used for semantic network, introduced by John Sowa, has a finite, connected, bipartite graph. The nodes represent either concepts or conceptual relations. It differs from the previous method that it does not use labelled arcs. **For example :** Ram, Laxman and Bharat are Brothers or cat color is grey can be represented as shown in Fig. 4.6.

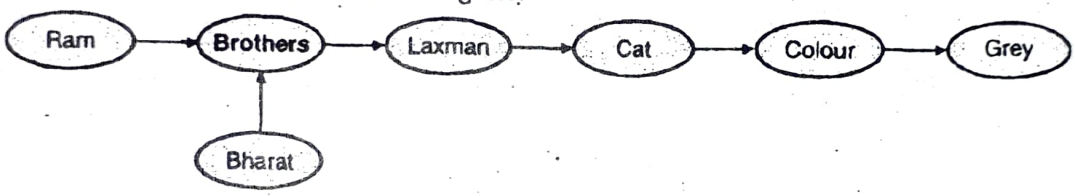


Fig. 4.6

(d) Frame Representation

- This concept was introduced by Marvin Minsky in 1975. They are mostly used when the task becomes quite complex and needs more structured representation. More structured the system becomes more would be the requirement of using frames which would prove beneficial. Generally frames are record like structures that consists of a collection of slots or attributes and the corresponding slot values.
- Slots can be of any size and type. The slots have names and values (subfields) called as facets. Facets can have names or numbers too. A simple frame is shown in the Fig. 4.6 for a person Ram,
  - (Ram)
  - (PROFESSION (VALUE professor)
  - (AGE(VALUE 50))
  - (WIFE(VALUE sita))
  - (CHILDREN(VALUE luv kush))
  - (ADDRESS (STREET(VALUE 4C gb road)))
  - CITY(VALUE banaras))
  - (STATE(VALUE mh))
  - (ZIP(VALUE400615))

Q. 6 What is prepositional logic ? Write syntax and semantics and example sentences for propositional logic. (6 Marks)

Ans.:

Propositional Logic (PL)

- **Propositional Logic (PL)** is simple but powerful for some artificial intelligence problems. You have learnt simple mathematical logic in which uses atomic formulas are used. (Atomic formula is a formula that has no strict sub-formulas). Atomic logic formulas are called propositions.
- In case of artificial intelligence propositional logic is not categorized as the study of truth values, but it is based on relativity of truth values. (i.e. The relationship between the truth value of one statement to that of the truth value of other statement).

Syntax

Basic syntax followed by the propositional logic can be given as follows :

- Propositional symbols are denoted with capital letters like : A, B, C, etc.
- Propositional logic constants have a truth value generally truth values have a crisp nature (i.e. 0 (false) and 1 (true)). But for fuzzy logic truth values can vary in the range of 0 and 1.
- Propositional logic make use of wrapping parenthesis while writing atomic sentence. It is denoted as '(....)'.
- Literal is an atomic sentence or it can be negation of atomic sentence. (A, ¬A)
- If A is a sentence, then ¬A is a sentence.
- Propositional logic makes use of relationships between propositions and it is denoted by connectives, if A and B are propositions. Connectives used in proposition logic can be seen in the Table 4.1.

Table 4.1 : Connectives used in Propositional logic

Connective symbol	Name of the Connective symbol	Relationship between Propositional symbols	Name of the Relationship between Propositional symbols
^	And	A ^ B	Conjunction
v	Or	A v B	Disjunction
¬	Not	¬A	Negation
⇒	Implies	A ⇒ B	Implication / conditional
⇔	is equivalent/ if and only if	A ⇔ B	Biconditional

Semantics

- World is set of facts which we want to represent to form propositional logic. In order to represent these facts propositional symbols can be used where each propositional symbol's interpretation can be mapped to the real world feature.
- Semantics of a sentence is meaning of a sentence. Semantics determine the interpretation of a sentence.
- **For example :** You can define semantics of each propositional symbol in following manner :
  1. A means "It is hot"
  2. B means "It is humid", etc.

- Sentence is considered true when its interpretation in the real world is true. Every sentence results from a finite number of usages of the rules. For example, if A and B are sentences then (A ∧ B), (A ∨ B), (B → A) and (A ↔ B) are sentences.
- Thus real world is a model of the knowledge base when the knowledge base is true for that world. In other words a model can be thought of as a truth assignment to the symbols.
- If truth values of all symbols in a sentence are given then it can be evaluated for determining its truth value (i.e. we can say if it is true or false).

Q. 7 Explain the inference process in case of propositional logic with suitable examples. **(5 Marks)**

Ans. :

Inference Rules

- New sentences are formed with the logical inference. **For example :** If A = B and B = C then A = C. You must have come across this example many times it implies that if knowledge base has "A = B" and "B = C" then we can infer that "A = C".
- In short inference rule says that new sentence can be create by logically following the set of sentences of the knowledge base.

Table 4.2 : Inference Rules

Inference Rules	Premise (KB)	Conclusion
Modus Ponens	X, X→Y	Y
Substitution	X→Z &Y → Z	X = Y
Chain rule	X → Y, Y→Z	X →Z
AND introduction	X, Y	X ^ Y
Transposition	X →Y	~X →~Y

- Entailment is represented as : KB |= Q and Derivation is represented as : KB |- Q.
- **There are two types of inference rules :**

1. Sound inference
2. Complete inference

1. Sound inference

- Soundness property of inference says that, if "X is derived from the knowledge base" using given set of protocols of inference, then "X is entailed by knowledge base". Soundness property can be represented as : "If KB |- X then KB |= X".
- For **Modus Ponens (MP)** rule we assume that knowledge base has [A, A → B], from this we can conclude that knowledge base can have B. See following truth table :

A	B	A → B	Valid?
TRUE	TRUE	TRUE	Yes
TRUE	FALSE	FALSE	Yes
FALSE	TRUE	TRUE	Yes
FALSE	FALSE	TRUE	Yes

In general,

For atomic sentences p_i, p_i', and q, where there is a substitution θ such that

SUBST (θ, p_i) = SUBST (θ, p_i')

$$\text{for all i, } \frac{p_1', p_2', p_3', \ldots p_n', (p_1 \wedge p_2 \wedge p_3 \wedge \ldots \ldots p_n \Rightarrow q)}{\text{SUBST } (\theta, q)}$$

N + 1 premises = N atomic sentences + one implication.

Example :

A : It is rainy.

B : I will stay at home.

A→B : If it is rainy, I will stay at home.

Modus Tollens

When B is known to be false, and if there is a rule "if A, then B," it is valid to conclude that A is also false.

2. **Complete inference**

- Complete inference is converse of soundness. Completeness property of inference says that, if "X is entailed by knowledge base" then "X can be derived from the knowledge base" using the inference protocols.
- Completeness property can be represented as : " If KB |= Q then KB |- Q".

Q. 8 Explain Horn Clause with example. **(4 Marks)**

Ans. :

Horn Clause

- Clauses are generally written as sets of literals. Horn clause is also called as **horn sentence.** In a horn clause a conjunction of 0 or more symbols is to the left of "→" and 0 or 1 symbols to the right. See following formula :

  $A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_n \rightarrow B_m$ where n >= 0 and m is in range {0, 1}
- There can be following special cases in horn clause in the above mentioned formula :
  - For n = 0 and m=1 : A (This condition shows that assert A is true)
  - For n > 0 and m=0 : $A \wedge B \rightarrow$ (This constraint shows that both A and B cannot be true)
  - For n = 0 and m=0 : (This condition shows empty clause)
- Conjunctive normal form is a conjunction of clauses and by its set of clauses it is determined up to equivalence. For a horn clause conjunctive normal form can be used where, each sentence is a disjunction of literals with at most one non-negative literal as shown in the following formula : $\neg A_1 \vee \neg A_2 \vee \neg A_3 \ldots \vee \neg A_n \vee B$
- This can also be represented as : $(A \rightarrow B) = (\neg A \vee B)$

Significance of horn logic

- Horn sentences can be used in first order logic. Reasoning processes is simpler with horn clauses. Satisfiability of a propositional knowledge base is NP complete. (Satisfiability means the process of finding values for symbols which will make it true).
- For restricting knowledge base to horn sentences, satisfiability is in A. Due to this reason, first order logic horn sentences are the basis for prolog and datalog languages.
- Take one example which gives entailment for horn formulas.

- Find out if following horn formula is satisfiable?

  $(true \rightarrow X) \wedge (X \wedge Y \rightarrow Z) \wedge (Z \,!\, W) \wedge (Z \wedge W \rightarrow false) \wedge (true \rightarrow Y)$
- From the above equation, we entail if the query atom is false. Equation shows that there are clauses which state that true $\rightarrow$X and true $\rightarrow$Y, so we can assign X and Y to true value (i.e. true $\rightarrow X \wedge Y$).
- Then we can say that all premises of $X \wedge Y \rightarrow Z$ are true, based on this information we can assign Z to true. After that we can see all premises of $Z \rightarrow W$ are true, so we can assign W to true.
- As now all premises of $Z \wedge W \rightarrow$ false are true, from this we can entail that the query atom is false. Therefore, the horn formula is not satisfiable.

Q. 9 Write short note on predicate logic. **(5 Marks)**

Ans. : First Order Predicate Logic

- FOL is more expressive than PL, it can represent information using relations, variables and quantifiers, e.g., which was not possible with propositional logic.

  "Gorilla is Black" can be represented as :

  $$Gorilla(x) \rightarrow Black(x)$$

  "It is Sunday today" can be represented as :

  $$today(Sunday)$$
- First Order Logic (FOL) is also called as **First Order Predicate Logic (FOPL).** Since FOPL is much more expressive as a knowledge representation language than PL it is more commonly used in artificial intelligence.

Syntactic Elements, Semantic and Syntax

- FOL symbol can be a constant term, a variable term or a function.
- Assuming that "X" is a domain of values, we can define a term with following rules :
  1. **Constant term :** It is a term with fixed value which belongs to the domain.
  2. **Variable term :** It is a term which can be assigned values in the domain.
  3. **Function :** Say "f" is a function of "n" arguments. If we assume that $t_1, t_2, .., t_n$ are terms then $f(t_1, t_2, .., t_n)$ is also called as a term.
- All the terms are generated by applying the above three protocols.
- First order predicate logic makes use of propositional logic as a base logic, so the connectives used in PL and FOPL are common. Hence, it also supports $\wedge$ conjunction, $\vee$ disjunction, negation, $\Rightarrow$ implication and $\Leftrightarrow$ double implication.
- **Ground Term :** If a term does not have any variables it is called as a **ground term.** A sentence in which all the variables are quantified is called as a "well-formed formula".
  - Every ground term is mapped with an object.
  - Every condition (predicate) is mapped to a relation.
  - A ground atom is considered as true if the predicate's relation holds between the terms' objects.
- **Rules in FOL :** In predicate logic rule has two parts predecessor and successor. If the predecessor is evaluated to TRUE successor will be true. It uses the implication $\rightarrow$ symbol. Rule represents If-then types of sentences.
- **Example :** The sentence "If the bag is of blue colour, I will buy it." Will be represented as colour (bag, blue)$\rightarrow$buy(bag)

Q. 10 Write a short note on : Ontology. **(5 Marks)**

Ans.:

Ontology

- Ontology is study about what kind of things or entities exist in the universe. In AI, ontology is the specification of conceptualizations, used to help programs and humans to share knowledge about a particular domain. In turn, ontology is a set of concepts, like entity, relationships among entities, events that are expressed in a uniform way in order to create a vocabulary for information exchange. An ontology should also enable a person to verify what a symbol means. That is, given a concept, they want to be able to find the symbol, and, given the symbol, they want to be able to determine what it means. Typically, it specifies what types of individuals will be modelled, specifies what properties will be used, and gives some axioms that restrict the use of that vocabulary. Ontologies are usually written independently of a particular application and often involve a community to agree on the meanings of symbols
- **For example :** Consider a map showing hotels, railway station, buildings, schools, hospitals in a particular locality. In this map the symbols used to indicate these entities are enough to describe them. Hence the community who knows the meaning of these symbols can easily recognize it. Hence that becomes ontology of that map. In this ontology, it may define a building as human-constructed artifacts. It may give some restriction on the size of buildings so that shoeboxes cannot be buildings or that cities cannot be buildings. It may also state that a building cannot be at two geographically dispersed locations at the same time.

Q. 11 What are advantages and disadvantages of propositional logic. **(5 Marks)**

Ans.:

Advantages of Propositional Logic

- Propositional logic is a simple knowledge representation language.
- It is sufficient and efficient technique for solving some artificial intelligence based problems.
- Propositional logic forms the foundation for higher logics like **First Order Logic (FOL)**, etc.
- Propositional logic is NP complete and reasoning is decidable.
- The process of inference can be illustrated by PL.

Disadvantages of Propositional Logic

- Propositional logic is cannot express complex artificial intelligence problems.
- Propositional logic can be impractical for even small worlds, think about WUMPUS hunter problem.
- Even if we try to make use of propositional logic to express complex artificial intelligence problems, it can be very wordy and lengthy.
- PL is a weak knowledge representation language because :
  - With PL it is hard to identify if the used entity is "individual". For example : If there are entities like : Priya, Mumbai, 123, etc.
  - PL cannot directly represent properties of individual entities or relations between individual entities. For example, Pooja is tall.
  - PL cannot express specialization, generalizations, or patterns, etc. **For example :** All rectangles have 4 sides.

□□□

Unit V : Reasoning

Q. 1 Explain different Inference rule for FOL. SPPU : May 17, Dec. 17, May 18, Dec. 18, May 19, Dec. 19, 5 Marks

Ans. : Inference in First-Order Logic (FOL)

1. Forward Chaining

- For any type of inference there should be a path from start to goal. When based on the available data a decision is taken, then the process is called as the forward chaining. Forward chaining or data-driven inference works from an initial state, and by looking at the premises of the rules (IF-part), perform the actions (THEN-part), possibly updating the knowledge base or working memory. This continues until no more rules can be applied or some cycle limit is met.
- For example, "If it is raining then, we will take umbrella". Here, "it is raining" is the data and "we will take umbrella" is a decision. This means it was already known that it's raining that's why it was decided to take umbrella. This process is **forward chaining.**

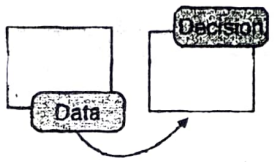


Fig. 5.1 : Forward Chaining

- "Forward chaining" is called as a data-driven inference technique.
- For Example :

  Given :

  - **Rule :** human (A) → mortal (A)
  - **Data :** human (Mandela)
- **To prove :** mortal (Mandela)

Forward Chaining Solution

- Human (Mandela) matches Left Hand Side of the Rule. So, we can get A = Mandela
- based on the rule statement we can get : mortal (Mandela)
- Forward chaining is used by the "design expert systems", as it performs operation in a forward direction (i.e. from start to the end).

Example :

- Given facts are as follows :
  1. It is a crime for an American to sell weapons to the enemy of America.
  2. Country Nono is an enemy of America.
  3. Nono has some missiles.
  4. All the missiles were sold to Nono by Colonel West.
  5. Missile is a weapon.
  6. Colonel West is American.
- To prove that West is a criminal.
- Represent these facts by FOL.

1. It is a crime for an American to sell weapons to the enemy nations.
   - American (x) ∧ Weapon (y) ∧ sell (x, y, z) ∧ enemy(z, America) => Criminal (x)
2. Country Nono is an enemy of America.Enemy (Nono, America)
3. Nono has some missiles.
   - Owns (Nono, x)
   - Missile (x)
4. All the missiles were sold to Nono by Colonel West.
   - Missile (x)∧ owns (Nono, x) => Sell (West, x, Nono)
5. Missile is a weapon.
   - Missile (x)=> weapon(x)
6. Colonel West is American.
   - American (West)

Proof by forward chaining

- The proof will start from the given facts. And as we can derive other facts from those, it will lead us to the solution. Please refer to Fig. 5.2. As observe from the given facts we can reach to the predicate Criminal (West).

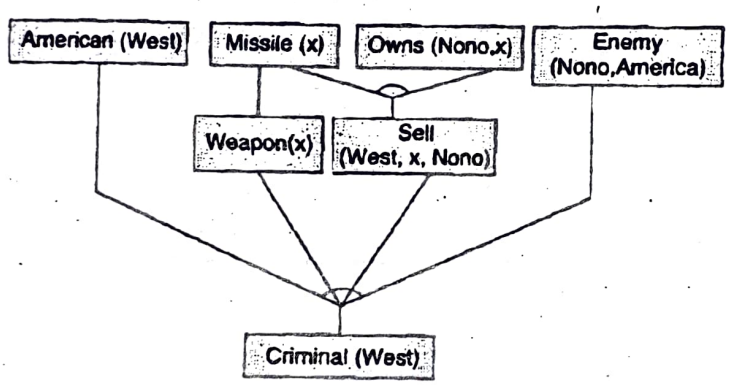


Fig. 5.2 : Proof by forward chaining

2. Backward Chaining

- If based on the decision the initial data is fetched, then it is called as **backward chaining**. Backward chaining or goal-driven inference works towards a final state, and by looking at the working memory to see if goal already there. If not look at the actions (THEN-parts) of rules that will establish goal, and set up sub-goals for achieving premises of the rules (IF-part). This continues until some rule can be applied, apply to achieve goal state.
- **For example,** If while going out one has taken umbrella. Then based on this decision it can be guessed that it is raining. Here, "taking umbrella" is a decision based on which the data is generated that "it's raining". This process is **backward chaining**. "Backward chaining" is called as a decision-driven or goal-driven inference technique.

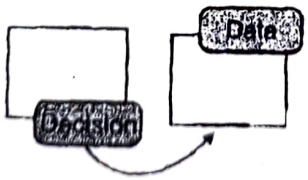


Fig. 5.3 : Backward Chaining

- **Given :**
  - Rule : human(A) → mortal(A)
  - Data : human(Mandela)
- **To prove :** mortal (Mandela)

Backward Chaining Solution

- Mortal (Mandela) will be matched with mortal (A) which gives human (A) i.e. human (Mandela) which is also a given fact. Hence proved.
- It makes use of right hand side matching. backward chaining is used by the "diagnostic expert systems", because it performs operations in a backward direction (i.e. from end to start).
- **Example :** Let us understand how the same example used in forward chaining can be solved using backward chaining.

Proof by backward Chaining

The proof will start from the fact to be proved. And as we can map it with given facts, it will lead us to the solution. Please refer to Fig. 5.4. As we observe, all leaf nodes of the proof are given facts that means "West is Criminal".

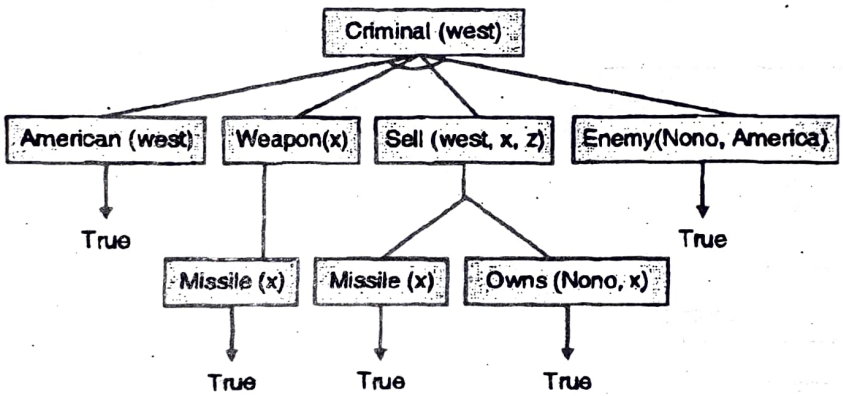


Fig. 5.4 : Proof by backward Chaining

Q. 2 Differentiate between propositional logic and predicate logic. **SPPU : May 19, 10 Marks**

Ans. :

Comparison between Propositional Logic and First Order Logic

Sr. No.	Propositional logic (PL)	Predicate logic (FOL)
1.	PLcan not represent small worlds like vacuum cleaner world.	FOL can very well represent small worlds' problems.
2.	PL is a weak knowledge representation language	FOL is a strong way of representing language.
3.	Propositional Language uses propositions in which the complete sentence is denoted by a symbol.	FOL uses predicated which involve constants, variables, functions, relations.
4.	PL cannot directly represent properties of individual entities or relations between individual entities. e.g. Meera is short.	FOL can directly represent properties of individual entities or relations between individual entities using individual predicates using functions. E.g. Short (Meera)

Sr. No.	Propositional logic (PL)	Predicate logic (FOL)
5.	PL cannot express specialization, generalizations, or patterns, etc. e.g. All rectangles have 4 sides.	FOL can express specialization, generalizations, or patterns, etc. Using relations. E.g. no_of_sides (rectangle, 4)
6.	PL is a foundation level logic.	FOL is a higher level logic.
7.	PL is not sufficiently expressive to represent complex statements.	FOL can represent complex statements.
8.	PL assumes the world contains facts.	FOL assumes the world contains objects, relations, functions like natural language.
9.	In PL Meaning of the facts is context-independent unlike natural language.	In FOL Meaning of the sentences is context dependent like natural language.
10.	PL is declarative in nature.	FOL is derivative in nature.

Q. 3 Write steps of knowledge engineering process. **(5 Marks)**

Ans. :

Knowledge Engineering Process

Knowledge engineering is a process of knowledgebase construction. It requires a knowledge engineer to investigate a particular domain, learn the important concepts in that domain, and create a formal representation and logical relations among objects in that domain.

Following is the general knowledge engineering process which can be applied to problem of any domain.

1. **Identify the task :** This step is analogues to PEAS process while designing an agent. While identifying task, the knowledge engineer must define the scope of knowledgebase and the range of questions that can be answered through the database. He also need to specify the type of facts that will be available for each specific problem instance.
2. **Assemble the relevant knowledge :** Assembling the relevant knowledge of that particular domain is called the process of knowledge acquisition. In this the knowledge engineer needs to extract the domain knowledge either by himself provided he is the domain expert or needs to work with the real experts of the domain. In this process knowledge engineer learns how the domain actually works and can determine the scope of the knowledgebase as per the identified tasks.
3. **Defining vocabulary :** Defining a complete vocabulary including predicates, functions and constants is a very important step of knowledge engineering. This process transforms the domain level concepts to logic level symbols. It should be exhaustive and precise. This vocabulary is called as ontology of the domain. And once the ontology is defined, it means, the existence of the domain is defined. That is, what kind of things exist in the domain is been decided.
4. **Encoding of general knowledge about the domain :** In this step the knowledge engineer defines axioms for all the vocabulary terms by define meaning of each term. This enables expert to cross check the vocabulary and the contents. If he finds any misinterpretations or gaps, it can be fixed at this point by redoing step 3.
5. **Encode the problem :** In this step, the specific problem instance is encoded using the defined ontology. This step will be very easy if the ontology is defined properly. Encoding means writing atomic sentences about problem instances which are already part of ontology. It can be analogues to input data for a computer program.

6. **Query the Knowledgebase :** Once all the above steps are done, all input for the system is set and now is a time to generate some output from the system. So, in order to get some interested facts inferred from the provided knowledge, we can query the knowledgebase. The inference procedure will operate on the axioms and facts to derive the new inferences. This lessens the task of a programmer to write application specific programs.
7. **Debug the knowledgebase :** This is the step in which one can prove or check the toughness of the knowledgebase. In the sense, if the inference procedure is able to give appropriate answers to all the queries asked or it stops in between because of the incomplete axioms; will be easily identified by debugging process. If one observes the reasoning chain stopping in between or some of the queries could not be answered then, it is an indication of a missing or a weak axioms. Then the corrective measures can be taken by repeating the required steps and system can be claimed to have a complete and precise knowledgebase.

Q. 4 Explain Modus Ponens with suitable example. **SPPU : May 16, Dec. 17, 4 Marks**

Ans. :

Generalized Modus Ponens

- The process of encapsulating inference rule is called as **Generalized Modus Ponens.**
- For atomic sentences p_i, p_i', and q, where there is a substitution θ such that

  $SUBST(\theta, p_i) = SUBST(\theta, p_i')$

  for all i, $$\frac{p_1', p_2', p_3', \ldots p_n', (p_1 \wedge p_2 \wedge p_3 \wedge \ldots \ldots p_n \Rightarrow q)}{SUBST(\theta, q)}$$

  N + 1 premises = N atomic sentences + one implication.
- Applying SUBST (θ , q) produces the conclusion we seek.

  $p_1' = King(Ram)$ $p_2' = Brave(y)$

  $p_1 = King(x)$, $p_2 = Brave(x)$

  $\theta = \{x / Ram, y / Ram\}$ $q = Noble(x)$

  $SUBST(\theta, q)$ is $Noble(Ram)$
- Generalized Modus Ponens is a **lifted** version of Modus Ponens. It raises Modus Ponens from ground (variable-free) propositional logic to first-order logic. Hence it is called as **lifting.**
- Here "lifted" indicates transformed from.
- The major advantage of lifted inference rules over propositional logic is that only those substitutions are made that are required so as particular inferences are allowed to proceed.

Q. 5 Explain steps to convert logical statements to clausal form. **(5 Marks)**

Ans. :

Conversion from FOL Clausal Normal Form (CNF)

1. Elimination of implication i.e. Eliminate all '→' : Replace $P \rightarrow Q$ with $\neg P \vee Q$
2. Distribute negations : Replace $\neg \neg P$ with P, $\neg(P \vee Q)$ with $\neg P \wedge \neg Q$ and so on.
3. Eliminate existential quantifiers by replacing with Skolem constants or Skolem functions :

   e.g. $\forall X \exists Y(P_1(X,Y) \vee (P_2(X,Y)) \equiv \forall X (P_1(X, f(X)) \vee (P_2(X, f(X)))$
4. Rename variables to avoid duplicate quantifiers.
5. Drop all universal quantifiers

---

6. Place expression into Cconjunctive Normal Form.
7. Convert to clauses i.e. separates all conjunctions as separate clause.
8. Rename variables to avoid duplicate clauses.

Q. 6 Consider following statements :

(a) Ravi Likes all kind of food.

(b) Apple and Chicken are food

(c) Anything anyone eats and is not killed is food.

(d) Ajay eats peanuts and still alive.

(e) Rita eats everything that Ajay eats.

Prove that Ravi Likes Peanuts using resolution. What food does Rita eat? **(10 Marks)**

Ans. :

(A) Proof by Resolution

Step 1 : Negate the statement to be proved.

~ likes (Ravi, Peanuts)

Step 2 : Convert given facts to FOL

(a) $\forall$ x, food (x) $\rightarrow$ likes (Ravi, x)

(b) food (Apple)

(c) food (Chicken)

(d) $\forall$ x $\forall$ y : eats (x, y) $\wedge$ ~ killed (x) $\rightarrow$ food (y)

(e) eats (Ajay, Peanuts) $\wedge$ alive (Ajay)

(f) $\forall$ x : eats (Ajay, x) $\rightarrow$ eats (Rita, x)

In this case we have to add few common sense predicate which are always true.

(g) $\forall$ x : ~ killed (x) $\rightarrow$ alive (x)

(h) $\forall$ x : alive (x) $\rightarrow$ ~ killed (x)

Step 3 : Converting FOLs to CNF

(a) ~ food (x) $\vee$ likes (Ravi, x)

(b) food (Apple)

(c) food (Chicken)

(d) ~ eats (x, y) $\vee$ killed (x) $\vee$ food (y)

(e) eats (Ajay, Peanuts)

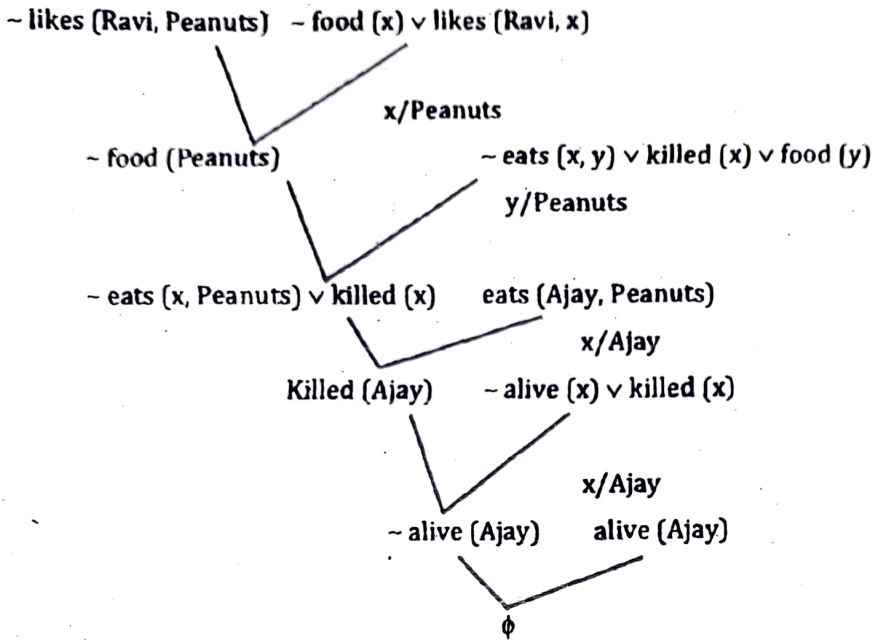
(f) alive (Ajay)

(g) ~ eats (Ajay, x) $\vee$ eats (Rita, x)

(h) killed (x) $\vee$ alive (x)

(i) ~ alive (x) $\vee$ killed (x)

Step 4 : Proof by Resolution

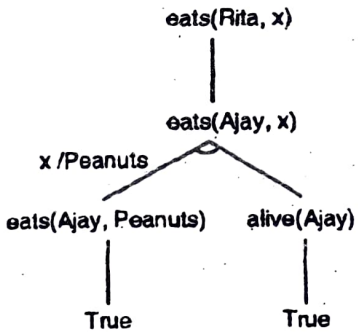


As the result of this resolution is NIL, it means out assumption is wrong. Hence proved that "Ravi likes Peanuts".

To answer : What food Rita eats ?

(B) Proof by backward chaining :

(Referring to FOLs of Step 2)



Hence the answer is Rita eats peanuts.

Q. 7 Write a short note on : Semantic Networks. **(5 Marks)**

Ans. :

Semantic Networks

- A semantic net or semantic network is a knowledge representation technique used for propositional information, so it is also called a propositional net. In semantic networks the knowledge is represented as objects and relationships between objects.
- They are two dimensional representations of knowledge. It conveys meaning. Relationships provide the basic structure for organizing knowledge.
- It uses graphical notations to draw the networks. Mathematically a semantic net can be defined as a labeled directed graph. As nodes are associated with other nodes semantic nets are also referred to as associative nets.

- Semantic nets consist of nodes, links and link labels. Nodes of the graph denote objects while the links indicate relations among the objects.
- Nodes can appear as circles, ellipses, or rectangles to represent objects such as physical objects, concepts or situations. Links are drawn as arrows to express the relationships between objects, and link labels specify specifications of relationships.
- The two nodes connected to each other via a link are related to each other. The relationships can be of two types : "IS-A" relationship or "HAS" relationship. IS-A relationship stands for one object being "part of" the other related object. And "HAS" relationship indicates one object "consists of" the other related object.
- These relationships are nothing but the superclass subclass relationships. It is assumed that all members of a subclass will inherit all the properties of their superclasses. That's how semantic network allows efficient representation of inheritance reasoning.
- For example Fig. 5.5 is showing an instance of a semantic net. In the Fig. 5.5 all the objects are within ovals and connected using labeled arcs. The links are given labels according to the relationships.

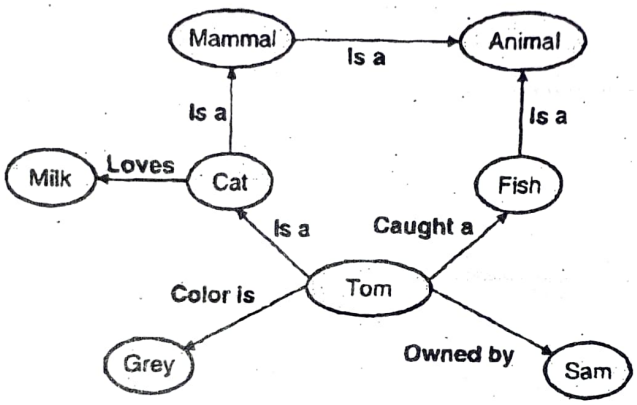


Fig. 5.5 : Semantic net example

- This makes the network more readable and conveys more information about all the related objects. For example, the "Member of" link between Jill and female persons indicates that Jill belongs to the category of female persons. It does also indicate the inheritance among the related objects. Like, Jill inherits the property of having two legs as she belongs to the category of Female persons which in turn belongs to the category of Persons which has a boxed Legs link with value 2.

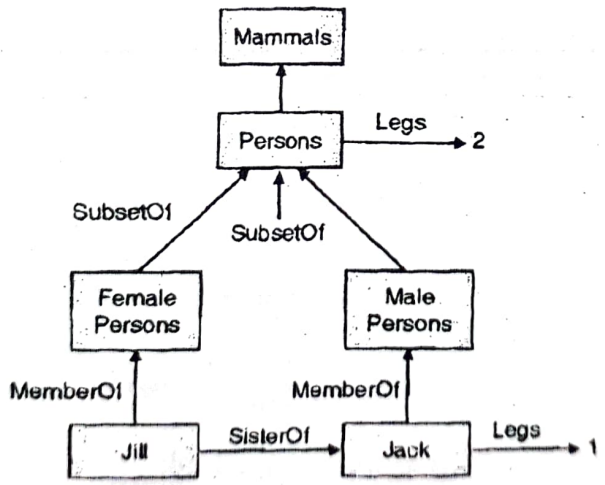


Fig. 5.6 : Semantic net example

- Semantic nets also can represent multiple inheritance through which, an object can belong to more than one objects and an object can be a subset of more than one another objects. It does also allow allows a common form of inference known as inverse links.
- The inverse links make the job of inference algorithms much easier to answer reverse queries. For example, in Fig. 5.6 there is a has Sister link which is the inverse of Sister of link.
- If there is a query "such as who the sister of Jack ?" The inference system will discover that Has Sister is the inverse of Sister of, to make the inference algorithm follow the link Has Sister from Jack to Jill and answer the query.

Advantages of Semantic Nets

1. Semantic nets can represent default values for categories which also can be overridden for a particular instance. For example, considering Fig. 5.6; all persons have two legs, and Jack has one leg while he is a person. So persons have two legs is the default status which can be overridden by a specific value.
2. Semantic nets represent meaning of relationships in a transparent manner.
3. Semantic nets are simple and easy to understand.
4. Semantic nets are easy to implement using PROLOG.

Disadvantage of Semantic Nets

1. The links between the objects represent only binary relations. In relations where more than two objects are involved cannot be represented using semantic nets.For example, the sentence Run (Chennai Express, Chennai, Bangalore, Today) cannot be represented directly.
2. There is no standard definition of link names.

Q. 8 Differentiate between Forward Chaining and Backward Chaining. **(6 Marks)**

Ans. :

Difference between Forward Chaining and Backward Chaining

Attribute	Backward Chaining	Forward Chaining
Also known as	Goal-driven	Data-driven
Starts from	Possible conclusion	New data
Processing	Efficient	Somewhat wasteful
Aims for	Necessary data	Any Conclusion(s)
Approach	Conservative/Cautious	Opportunistic
Practical if	Number of possible final answers is reasonable or a set of known alternatives is available.	Combinatorial explosion creates an infinite number of possible right answers.
Appropriate for	Diagnostic, prescription and debugging application	Planning, monitoring, control and interpretation application
Reasoning	Top-down reasoning	Bottom-up reasoning

Attribute	Backward Chaining	Forward Chaining
Type of Search	Depth-first search	Breadth-first search
Who determine search	Consequents determine search	Antecedents determine search
Flow	Consequent to antecedent	Antecedent to consequent

Q. 9 Using predicate logic find the course of Anish's liking for the following :

(i) Anish only likes easy courses.

(ii) Computer courses are hard.

(iii) All electronics courses are easy

(iv) DSP is an electronics course. **(5 Marks)**

Ans. :

Step 1 : Converting given facts to FOL

(i) $\forall x$: course (x) $\wedge$ easy (x) $\rightarrow$ likes (Anish, x)

(ii) $\forall x$: course (x) $\wedge$ computes (x) $\rightarrow$ hard (x)

(iii) $\forall x$: course (x) $\wedge$ electronics (x) $\rightarrow$ easy (x)

(iv) Electronics (DSP)

(v) course (DSP)

Step 2 : Proof by backward chaining :

As we have to find out which course Anish likes. So we will start the proof from the same fact.

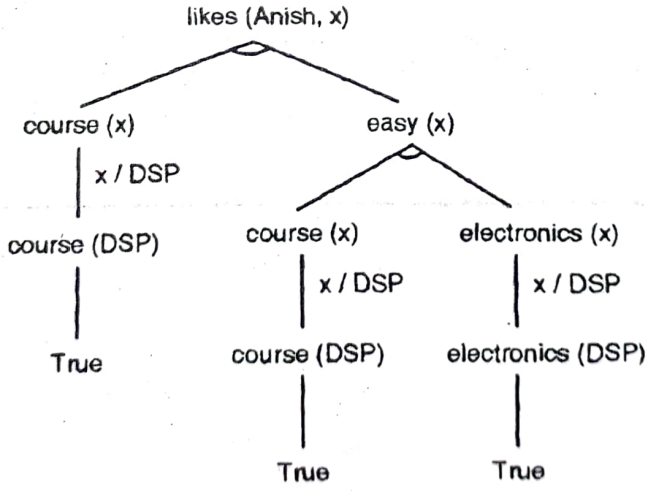


Fig. 5.7

Hence proved that Anish likes DSP course.

Q. 10 Write a short note on : Resolution. **(5 Marks)**

Ans. :

Resolution

- Resolution is a valid inference rule. Resolution produces a new clause which is implied by two clauses containing complementary literals. This resolution rule was discovered by Alan Robinson in the mid 1960's.
- A literal is an atomic symbol or a negation of the atomic symbol (i.e. A, ¬A).
- Resolution is the only interference rule you need, in order to build a sound (soundness means that every sentence produced by a procedure will be "true") and complete (completeness means every "true" sentence can be produced by a procedure) theorem proof maker.
- Take an example where we are given that :
  - A clause X containing the literal : Z
  - A clause Y containing the literal : ¬ Z
- Based on resolution and the information given above we can conclude : (X – {Z}) U (Y – {¬Z})
- Take a generalized version of the above problem :

  Given :
  - A clause X containing the literal : Z
  - A clause Y containing the literal : ¬Y
  - A most general unifier G of Z and ¬Y

  ∴ {(X – {Z}) U (Y – {¬Y})) | G

Resolution Procedure

- Knowledge base be a set of true sentences which do not have any contradictions, and Z be a sentence that we want to prove.
- The Idea is based on the proof by negation. So, we should assume ¬Z and then try to find a contradiction (You must have followed such methods while solving geometry proofs). Then based on the Intuition that, if all the knowledge base sentences are true, and assuming ¬Z creates a contradiction then Z must be inferred from knowledge base. Then we need to convert knowledge base U {¬Z} to clause form.
- If there is a contradiction in knowledge base, that means Z is proved. Terminate the process after that.
- Otherwise select two clauses and add their resolvents to the current knowledge base. If we do not find any resolvable clauses then the procedure fails and then we terminate. Else, we have to start finding if there is a contradiction in knowledge base, and so on.

Q.11 Using a predicate logic convert the following sentences to predicates and prove that the statement "Ram did not jump" is false.

(a) Ram went to temple.

(b) The way to temple is, walk till post box and take left or right road.

(c) The left road has a ditch.

(d) Way to cross the ditch is to jump

(e) A log is across the right road.

(f) One needs to jump across the log to go ahead. **(10 Marks)**

Ans. :

Step 1 : Negate the statement to be proved.

$\sim$ jump (Ram)

Step 2 : Converting given statement to FOL

(a) At (Ram, temple)

(b_1) $\forall$ x : At (x, temple) $\longrightarrow$ At (x, PostBox) $\wedge$ take left (x)

(b_2) $\forall$ x : At (x, temple) $\longrightarrow$ At (x, PostBox) $\wedge$ take right (x)

(c) $\forall$ x: take left (x) $\longrightarrow$ cross (x, ditch)

(d) $\forall$ x cross (x, ditch) $\longrightarrow$ jump (x)

(e) $\forall$ x : take right (x) $\longrightarrow$ at (x, log)

(f) $\forall$ x : at (x, log) $\longrightarrow$ jump (x)

Step 3 : Converting FOLs to CNF

(a) At (Ram, temple)

(b_{11}) $\sim$ At (x, temple) $\vee$ At (x, PostBox)

(b_{12}) $\sim$ At (x, temple) $\vee$ take left (x)

(b_{21}) $\sim$ At (x, temple) $\vee$ At (x, PostBox)

(b_{22}) $\sim$ At (x, temple) $\vee$ take right (x)

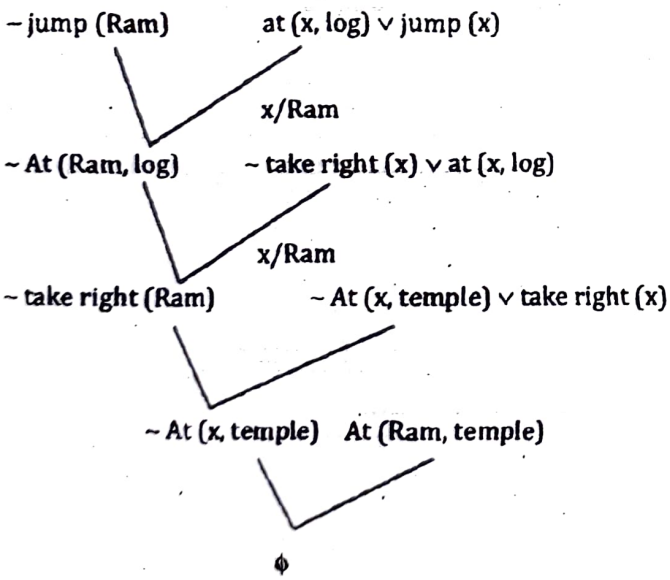
(c) $\sim$ take left (x) $\vee$ cross (x, ditch)

(d) $\sim$ cross (x, ditch) $\vee$ jump (x)

(e) $\sim$ take right (x) $\vee$ at (x, log)

(f) $\sim$ at (x, log) $\vee$ jump (x)

Step 4 : Proof by Resolution



Hence proved.

Q. 12 Explain unification in detail. **(6 Marks)**

Ans.:

Unification

The processes of finding legal substitutions that make different logical expressions look identical. The unification algorithm is a recursive algorithm; the problem of unification is : given two atoms, to find if they unify, and, if they do, return an MGU (Most General Unifier) of them.

Procedure Unify(t_1,t_2)

Inputs

t_1,t_2: atoms **Output**

most general unifier of t_1 and t_2 if it exists or $\perp$ otherwise

Local

E: a set of equality statements

S: substitution

$E \leftarrow \{t_1 = t_2\}$

$S = \{\}$

while ($E \neq \{\}$)

select and remove $x=y$ from E

if (y is not identical to x) then

if (x is a variable) then

replace x with y everywhere in E and S

$S \leftarrow \{x/y\} \cup S$

else if (y is a variable) then

replace y with x everywhere in E and S

$S \leftarrow \{y/x\} \cup S$

else if (x is $f(x_1,\ldots,x_n)$ and y is $f(y_1,\ldots,y_n)$ then

$E \leftarrow E \cup \{x_1=y_1,\ldots,x_n=y_n\}$

Else

return $\perp$

return S

- Unification algorithm for Datalog

  Example : "x King(x) $\wedge$ Brave(x) $\Rightarrow$ Noble(x)

  King(Ram)

  Brave(Ram)

- An 'x' where, 'x' is a king and 'x' is brave (Then x is noble) then ideally what we want is θ= {substitution set}

  i.e. θ= {x/ Ram}

  Hence, Ram Unifies x.

Q. 13 Write a short note on : Events and Mental Objects. **(5 Marks)**

Ans. :

Events and Mental Objects

- Knowledge is the most important aspect of knowledge base agent. They use the available knowledge and deduce new set of knowledge based on that. As in case of human, if we need some information, we ask a question to the authorized person, who knows the matter we need.
- Similarly, the knowledge base agent also needs to know what information it has in order to deduce new one. It needs to know about the inference process, so that it gets the answer it wants to have.
- This initiates the need to know the objects in someone's head called as **mental objects** and the process to manipulate those objects is called as **mental events**. Using mental objects and events, agent can reason about beliefs of agents. Mental objects for agents can show their attitudes such as knows, wants, believes, intends, informs. These attitudes cannot be used as a normal predicates. Hence we need to reify them.

For example :

- "John knows that spiderman can catch."

  It can be represented as , Knows (John, can_catch(spiderman))
- Usually can_catch(spiderman) is a sentence. But here it is appearing as a term. We can solve this problem by making it as a fluent. It is true that spidermanis , then we can write, can_catch(....) and the above sentence can be written as :

  (Spiderman = ...) /\ Knows (John, can_catch(Spiderman)) → Knows(John, can_catch(...))

Q.14 Represent following sentences in FOL using a consistence vocabulary.

(i) Every person who buys a policy is smart.

(ii) No person buys an expensive policy.

(iii) There is an agent who sells policies only to people who are not insured.

(iv) There is a barber who shaves all men in town who do not save themselves. **(5 Marks)**

Ans. :

(i) $\forall x \forall y$: person (x) $\wedge$ policy (y) $\wedge$ buys (x, y) $\rightarrow$ smart (x)

(ii) $\forall x, \forall y$: person (x) $\wedge$ policy (y) $\wedge$ expensive (y) $\rightarrow \sim$ buys (x, y)

(iii) $\forall x$: person (x) $\wedge \sim$ insured (x)

$\forall y : \exists x \wedge$ policy (y) – agent (x) $\rightarrow$ sells (x, y, x)

(iv) $\exists x \ \forall y$: barber (x) $\wedge$ person (y) $\wedge \sim$ shaves (y, y) $\rightarrow$ shaves (x, y)

Q. 15 Write first order logic statements for following statements :

(i) If a perfect square is divisible by a prime p then it is also divisible by square of p.

(ii) Every perfect square is divisible by some prime.

(iii) Alice does not like chemistry and history.

(iv) If it is Saturday and warm, then sam is in the park.

(v) Anything anyone eats and is not killed by is food. **(5 Marks)**

Ans. :

(i) $\forall x$: square(x) $\wedge$ prime (y) $\wedge$ divides (p, x) $\rightarrow$ [$\exists z$: square_of(z, p) $\wedge$ divides(z, x)]

(ii) $\forall x \exists y$: square(x) $\wedge$ divides (p, x)

(iii) ~ likes(Alice, History) $\wedge$ ~ likes (Alice, Chemistry)

(iv) day(Saturday) $\wedge$ weather(warm) $\rightarrow$ in_park(Sam)

(v) $\forall x$: $\forall y$: person(x) $\wedge$ eats(x, y) $\wedge$ ~killed (x) $\rightarrow$ food(y)

Q.16 Convert following propositional logic statement into CNF.

$A \rightarrow (B \rightarrow C)$

(5 Marks)

Ans. :

FOL : $A \rightarrow (B \leftrightarrow C)$

Normalizing the given statement.

(i) $A \rightarrow (B \rightarrow C \wedge C \rightarrow B)$

(ii) $(A \rightarrow (B \rightarrow C)) \wedge (A \rightarrow (C \rightarrow B))$

Converting to CNF.

Applying Rule, $\alpha \rightarrow \beta \equiv -\alpha \vee \beta$

$\sim A \vee (\sim B \vee C) \wedge \sim A \vee (\sim C \vee B)$

i.e. $\sim A \vee ((\sim B \vee C) \wedge (\sim C \vee B))$

Q.17 Consider following axioms.

All people who are graduating are happy.

All happy people smile.

Someone is graduating.

(i) Represent these axioms in FOL.

(ii) Convert each formula to CNF.

(iii) Prove that someone is smiling using resolution technique. Draw the resolution tree. **(5 Marks)**

Ans. :

Step 1 : Converting axioms to FOL.

(i) $\forall x$: graduating (x) $\rightarrow$ happy (x).

(ii) $\forall x$: happy (x) $\rightarrow$ smile (x)

(iii) $\exists x$: graduating (x)

Step 2 : Converting FOL to CNF.

(i) ~ graduating (x) $\vee$ happy (x)

(ii) ~ happy (x_1) $\vee$ smile (x_1)

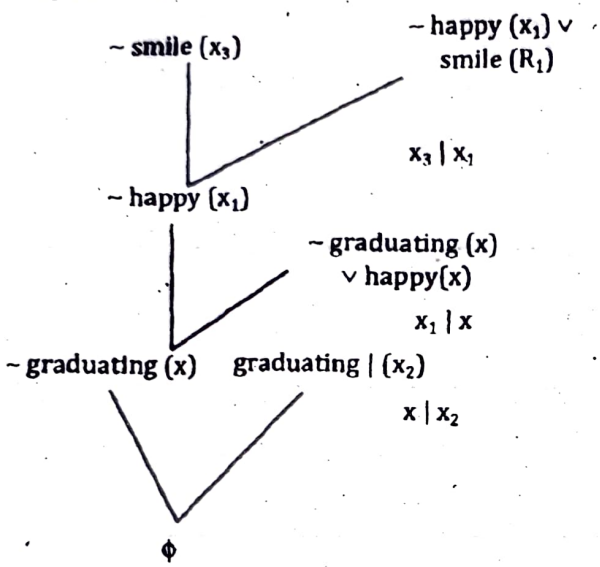
(iii) graduating (x_2)

Step 3 : *T.P.T. x_3 smile (x_3)*

Negating the stmt

~ smile (x_3)

Proof by resolution



Hence our assumption is wrong.

Hence proved.

□□□

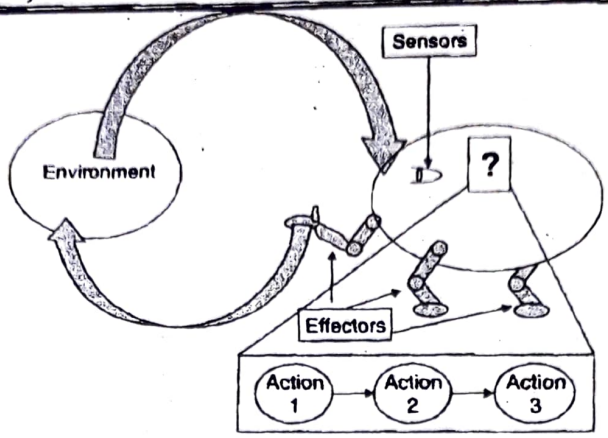


Fig. 6.2 : Planning agent

Q. 2 How planning problem differs from search problem ? **(5 Marks)**

Ans. :

Problem Solving and Planning

- Generally problem solving and planning methodologies can solve similar type of problems. Main difference between problem solving and planning is that planning is a more open process and agents follow logic-based representation. Planning is supposed to be more powerful than problem solving because of these two reasons.
- Planning agent has situations (i.e. states), goals (i.e. target end conditions) and operations (i.e. actions performed). All these parameters are decomposed into sets of sentences and further in sets words depending on the need of the system.
- Planning agents can deal with situations/states more efficiently because of its explicit reasoning capability also it can communicate with the world. Agents can reflect on their targets and we can minimize the complexity of the planning problem by independently planning for sub-goals of an agent. Agents have information about past actions, presents actions and the important point is that it can predict the effect of actions by inspecting the operations.
- Planning is a logical representation, based on situation, goals and operations, of problem solving.

Planning = Problem solving + Logical representation

Q. 3 Explain goal of planning with supermarket example. **(4 Marks)**

Ans. :

Goal of Planning

- Plan activities in order to achieve some goals. Main goal can be divided into sub-goals to make planning more efficient.
- Take example of a grocery shopping at supermarket, suppose you want to buy milk, bread and egg from supermarket, then your initial state will be - "at home" and goal state will be - "get milk, bread and egg".
- Now if you look at the Fig. 6.3, branching factor can be enormous depending upon the set of actions, for e.g. Watch TV, read book, etc., available at that point of time.
- **Thus branching factor** can be defined as a set of all probable actions at any state, set can be very large, such as in the supermarket example or block problem. If the domain of probable actions increases, the branching factor will also increase as they are directly proportional to each other, this will result in the increase of search space.

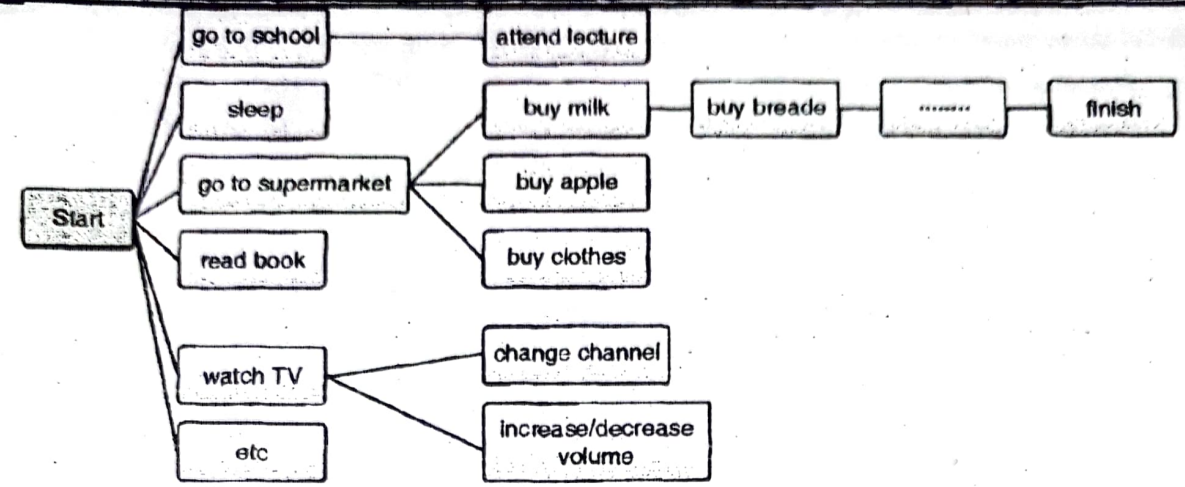


Fig. 6.3 : Supermarket examples to understand need of planning

- To reach the goal state you have to follow many steps, if you consider using heuristic functions, then you have to remember that it will not be able to eliminate states; these functions will be helpful only for guiding the search of states.
- So it becomes difficult to choose best actions. (i.e. even if we go to supermarket we need to make sure that all three listed items are picked, only then goal state can be achieved).
- As there are many possible actions and it is difficult to describe every state, and there can be combined goals (as seen in supermarket example) searching is inadequate to achieve goals efficiently. In order to be more efficient planning is required.
- Planning requires explicit knowledge that means in case of planning we need to know the exact sequence of actions which will be useful in order to achieve the goal.
- Advantage of planning is that, the order of planning and the order of execution need not be same. For example, you can plan how to pay bills for grocery before planning to go to supermarket.
- Another advantage of planning is that you can make use of divide and conquer policy by dividing /decomposing the goal into sub goals.

Q. 4 What are the major approaches /algorithms of planning ? **(5 Marks)**

Ans. :

Algorithms for Classical Planning

- There are many approaches to solve planning problems. Following are few major approaches or algorithms used for planning :
  - Planning with state space search.
  - Partial ordered planning.
  - Hierarchical planning / hierarchical decomposition (HTN planning).
  - Planning situation calculus/ planning with operators.
  - Conditional planning.
  - Planning with operators.
  - Planning with graphs.
  - Planning with propositional logic.
  - Planning reactive.

- Out of these major approaches we will be learning about following algorithms in detail :
  - Planning with state space search
  - Partial ordered planning and
  - Hierarchical planning / Hierarchical decomposition (HTN planning).
  - Conditional planning.
  - Planning with operators.

Q. 5 Explain Water Jug problem with State Space Search Method. **(5 Marks)**

Ans. :

Example of State Space Search

An example of a water jug problem are as follows :

- Two jugs, a 4 - gallon one and a 3- gallon one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water how can you get exact 2 gallons of water into the 4 - gallon jug?
- The state space for this problem can be described as of ordered pairs of integers (x, y), such that x = 0, 1, 2, 3 or 4, representing the number of gallons of water in the 4- gallon jug and y = 0, 1, 2 or 3, representing the quantity of water in the 3- gallon jug. The start state is (0, 0). The goal state is (2, n) for any value of n, since the problem does not specify how many gallons need to be in the 3- gallon jug.
- The operators to be used to solve the problem can be described as shown bellow. They are represented as rules whose left side are matched against the current state and whose right sides describe the new state that results from applying the rule.

Rule set

1. (x,y) ⟶ (4,y) fill the 4- gallon jug
   If x < 4
2. (x,y) ⟶ (x,3) fill the 3-gallon jug
   If x < 3
3. (x,y) ⟶ (x-d,y) pour some water out of the 4- gallon jug
   If x > 0
4. (x,y) ⟶ (x-d,y) pour some water out of the 3- gallon jug
   If y > 0
5. (x,y) ⟶ (0,y) empty the 4- gallon jug on the ground
   If x > 0
6. (x,y) ⟶ (x,0) empty the 3- gallon jug on the ground
   If y > 0
7. (x,y) ⟶ (4,y-(4-x)) pour water from the 3- gallon jug into the 4- gallon
   If x + y >= 4 and y > 0 jug until the 4-galoon jug is full
8. (x,y) ⟶ (x-(3-y),3)) pour water from the 4- gallon jug into the 3-gallon
   If x + y >= 3 and x > 0 jug until the 3-gallon jug is full
9. (x,y) ⟶ (x+y,0) pour all the water from the 3 -gallon jug into
   If x + y <= 4 and y > 0 the 3-gallon jug

10. (x,y) ⟶ (0,x+y) pour all the water from the 4 -gallon jug into
    If x + y <= 3 and x > 0 the 3-gallon jug
11. (0,2) ⟶ (2,0) pour the 2-gallon from the 3 -gallon jug into the 4-gallon jug
12. (2,y) ⟶ (0,x) empty the 2 gallon in the 4 gallon on the ground

Production for the water jug problem

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

One solution to the water jug problem.

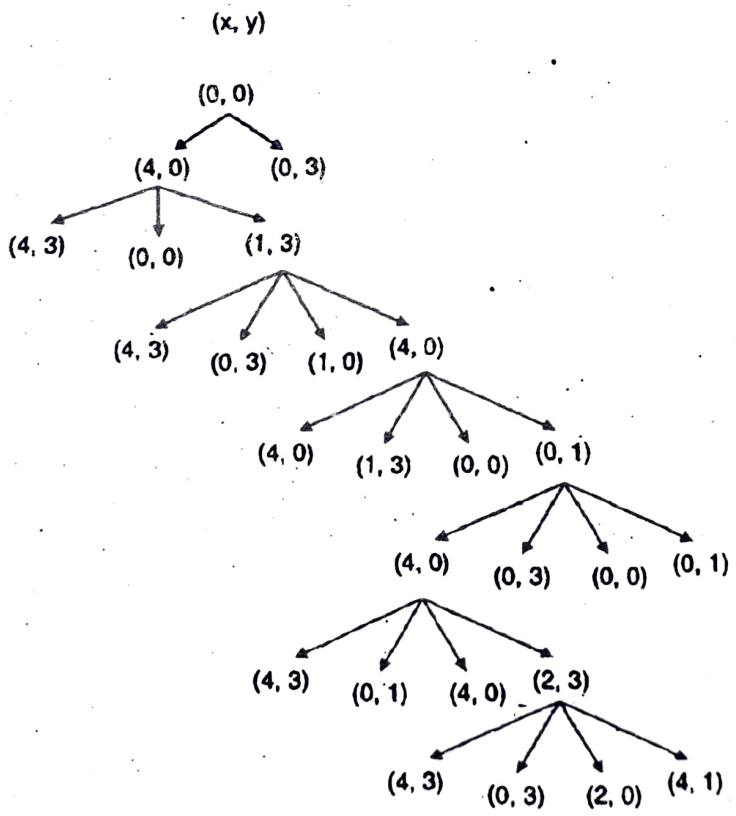


Fig. 6.4

Q. 6 Explain progression planners with example. **(5 Marks)**

Ans. :

Progression Planners

- "Forward state-space search" is also called as **"progression planner"**. It is a deterministic planning technique, as we plan sequence of actions starting from the initial state in order to attain the goal.

- With forward state space searching method we start with the initial state and go to final goal state. While doing this we need to consider the probable effects of the actions taken at every state.
- Thus the prerequisite for this type of planning is to have initial world state information, details of the available actions of the agent, and description of the goal state.
- Remember that details of the available actions include preconditions and effects of that action.
- See Fig. 6.5 it gives a state-space graph of progression planner for a simple example where flight 1 is at location A and flight 2 is also at location A. these flights are moving from location A to location B. In 1st case only flight 1 moves from location A to location B, so the resulting state shows that after performing that action flight 1 is at location B whereas flight 2 is at its original location – A. similarly In 2nd case only flight 2 moves from location A to location B and the resulting state shows that after performing that action flight 2 is at location B while flight 1 is at its original location – A.

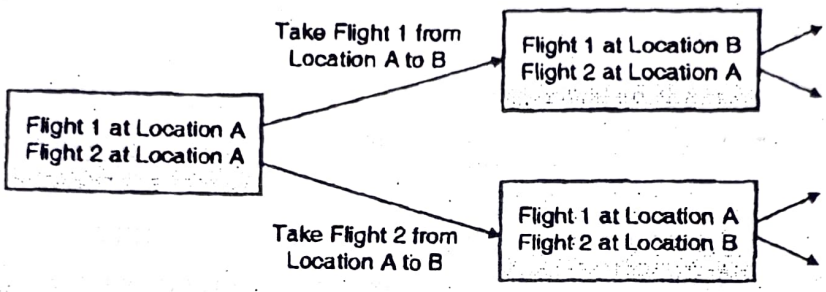


Fig. 6.5 : State-space graph of a Progression planners

- It can be observed from the Fig. 6.5 that, rectangles show state of the flights (i.e. their current location), and lines give the corresponding actions from one state to another (i.e. move from one location to other location).
- Note that the lines coming out of every state matches to all of the permissible actions which can be accepted if the agent is in that state.

Progression planner algorithm

1. Formulize the state space search problem :
   - Initial state is the first state of the planning problem which has a set of positive, the literals which don't appear are considered as false.
   - If preconditions are satisfied then the actions are favoured i.e. if the preconditions are satisfied then positive effect literals are added for that action else the negative effect literals are deleted for that action.
   - Perform goal testing by checking if the state will satisfy the goal.
   - Lastly keep the step cost for each action as 1.
2. Consider example of A* algorithm. A complete graph search is considered as a complete planning algorithm. Functions are not used.
3. Progression planner algorithm is supposed to be inefficient because of the irrelevant action problem and requirement of good heuristics for efficient search.

Q. 7 Explain regression planners with example. **(5 Marks)**

Ans. :

Regression Planners

- "Backward state-space search" is also called as **"regression planner"** from the name of this method you can make out that the processing will start from the finishing state and then you will go backwards to the initial state.

- For example, we cannot wear left shoe without wearing the left sock and we cannot wear the right shoe without wearing the right sock. So while creating the sequence of actions in total ordered planning, wearing left sock action should be executed before wearing the left shoe and wearing the right sock action should be executed before wearing the right shoe. As shown in Fig. 6.7.

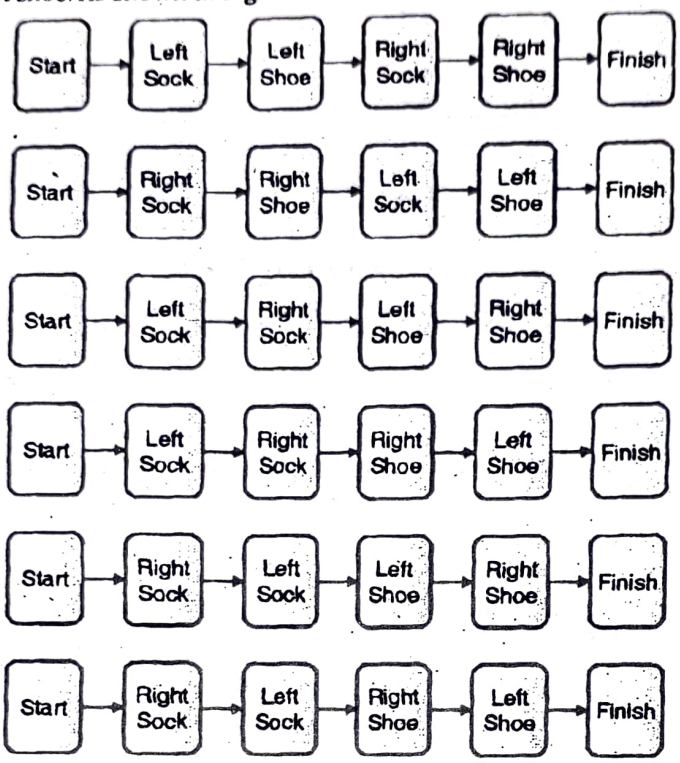


Fig. 6.7 : Total order planning of Wearing Shoe

If there is a cycle of constraints then, total ordered planning cannot give good results. TOP can fail in non-cooperative environments. So we have Partial Ordered Planning method.

Q. 9 Discuss partial order planning giving suitable example. **(5 Marks)**

Ans. :

Partial Order Planning

- In case of **Partial Ordered Planning (POP)**, ordering of the actions is partial. Also partial ordered planning does not specify which action will come first out of the two actions which are placed in plan.
- With partial ordered planning, problem can be decomposed, so it can work well in case the environment is non-cooperative.
- Take same example of wearing shoe to understand partial ordered planning.
- **A partial order planning combines two action sequences**
  - First branch covers left-sock and left-shoe.
  - In this case to wear a left shoe, wearing left sock is the precondition, similarly.
  - Second branch covers right-sock and right-shoe.
  - Here, wearing a right sock is the precondition for wearing the right shoe.
- Once these actions are taken we achieve our goal and reach the finish state.

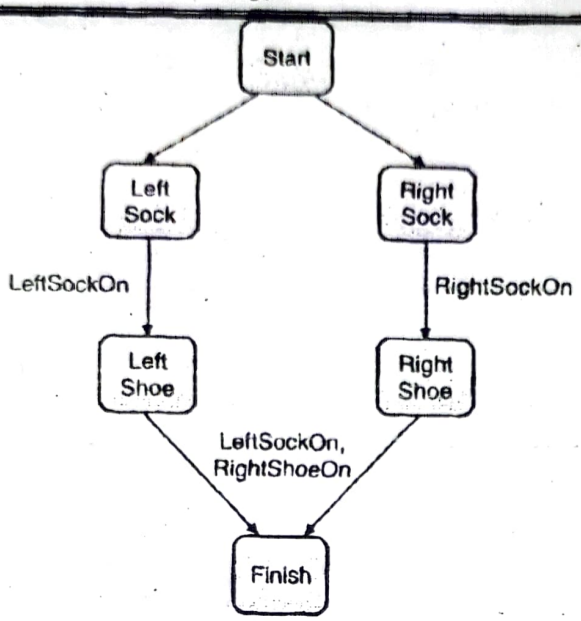


Fig. 6.8 : Partial order planning of Wearing Shoe

Q. 10 Explain process of generating solution to partial order planning problem. **(5 Marks)**

Ans. :

POP as a Search Problem

- If we considered POP as a search problem, then we say that states are small plans.
- States are generally unfinished actions. If we take an empty plan then, it will consist of only starting and finishing actions.
- Every plan has four main components, which can be given as follows :

1. **Set of actions**
   - These are the steps of a plan. Actions which can be performed in order to achieve goal are stored in set of actions component.
   - **For example :** *Set of Actions* = { Start, Rightsock, Rightshoe, Leftsock, Leftshoe, Finish}
   - Here, wearing left sock, wearing left shoe, wearing right sock, wearing right shoe are set of actions.
2. **Set of ordering constraints/ preconditions**
   - Preconditions are considered as ordering constraints. (i.e. without performing action "x" we cannot perform action "y")
   - **For example :** Set of Ordering ={Right-sock < Right-shoe; Left-sock < Left-shoe} that is In order to wear shoe, first we should wear a sock.
   - So the ordering constraint can be Wear Left-sock < wear Left-shoe (Wearing Left-sock action should be taken before wearing Left-shoe) Or Wear right-sock < wear right-shoe (Wearing right-sock action should be taken before wearing right-shoe).
   - If constraints are cyclic then it represents inconsistency.
   - If we want to have a consistent plan then there should not be any cycle of preconditions.

3. Set of causal links

- Action A achieves effect "E" for action B

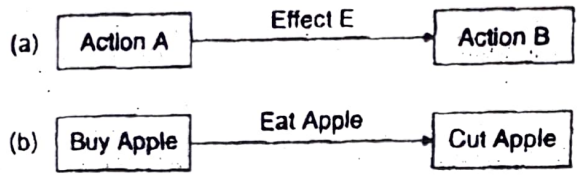


Fig. 6.9 : (a) Causal Link Partial Order Planning (b) Causal Link Example

- From Fig. 6.9(b) understand that if you buy an apple it's effect can be eating an apple and the precondition of eating an apple is cutting apple.
- There can be conflict if there is an action C that has an effect ¬ E and, according to the ordering constraints it comes after action A and before action B.
- Say we don't want to eat an apple instead of that we want to make a decorative apple swan. This action can be between A and B and It does not have effect "E".
- **For example :** Set of Causal Links = {Right-sock->Right-sock-on → Right-shoe, Leftsock → Leftsockon → Leftshoe, Rightshoe → Rightshoeon → Finish, leftshoe → leftshoeon → Finish }.
- To have a consistent plan there should not be any conflicts with the causal links.

4. Set of open preconditions

- Preconditions are called open if it cannot be achieved by some actions in the plan. Least commitment strategy can be used by delaying the choice during search.
- To have a consistent plan there should not be any open precondition

---

Q. 11 Write short note hierarchical planning. **(5 Marks)**

Ans. : Hierarchical Planning

- Hierarchical planning is also called as plan decomposition. Generally plans are organized in a hierarchical format.
- Complex actions can be decomposed into more primitive actions and it can be denoted with the help of links between various states at different levels of the hierarchy. This is called as operator expansion.

For example

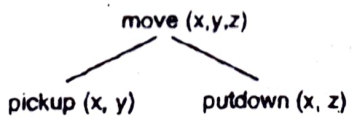


Fig. 6.10 : Operator expansion

- Fig. 6.11 shows, how to create a hierarchical plan to travel from some source to a destination. Also you can observe, at every level how we follow some sequence of actions.

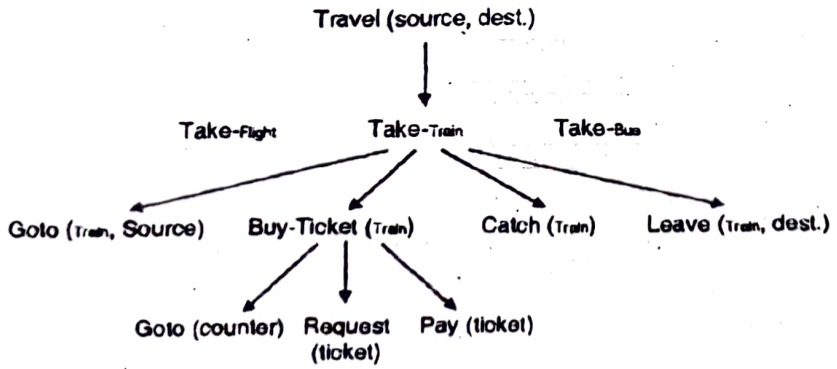


Fig. 6.11 : Hierarchical planning example

Q. 12 Compare STRIPS and ADL **(8 Marks)**

Ans. :

Comparison between STRIPS and ADL

Sr. No.	STRIPS language	ADL
1.	Only allows positive literals in the states, **For example :** A valid sentence in STRIPS is expressed as ⇒ Intelligent^ Beautiful.	Can support both positive and negative literals. **For example :** Same sentence is expressed as ⇒ ¬Stupid ∧ ¬Ugly
2.	Makes use of closed-world assumption (i.e. Unmentioned literals are false)	Makes use of Open World Assumption (i.e. unmentioned literals are unknown)
3.	We only can find ground literals in goals. **For example :** Intelligent ∧ Beautiful.	We can find quantified variables in goals. **For example :** ∃x At (P1, x) ∧ At(P2, x) is the goal of having P1 and P2 in the same place in the example of the blocks
4.	Goals are conjunctions **For example :** (Intelligent ∧ Beautiful).	Goals may involve conjunctions and disjunctions **For example :** (Intelligent ∧ (Beautiful ∨ Rich)).
5.	Effects are conjunctions	Conditional effects are allowed: when P:E means E is an effect only if P is satisfied
6.	Does not support equality.	Equality predicate (x = y) is built in.
7.	Does not have support for types	Supported for types **For example :** The variable p : Person

Q. 13 Design planning agent to solve block world problem. Assume suitable initial state and final state for the problem. **(8 Marks)**

Ans. :

Example of Block World Puzzle Standard sequence of actions is

1. Grab Z and Pickup Z
2. Then Place Z on the table
3. Grab Y and Pickup Y
4. Then Stack Y on Z
5. Grab X and Pickup X
6. Stack X on Y

Fig. 6.12

- Elementary problem is that framing problem in AI is concerned with the question of what piece of knowledge or information is pertinent to the situation.

- To solve this problem we have tom make an Elementary Assumption which is a Closed world assumption. (i.e. If something is not asserted in the knowledge base then it is assumed to be false, this is also called as "Negation by failure")
- Standard sequence of actions can be given as for the block world problem :

on(Y, table)	on(Z, table)
on(X, table)	on(Y, Z)
on(Z, X)	on(X, Y)
hand empty	hand empty
clear(Z)	clear(X)
clear(Y)	

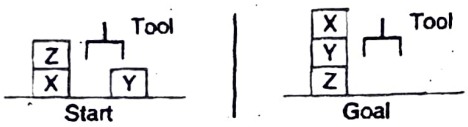


Fig. 6.13

- Write 4 main rules for the block world problem as follows :

	Rule	Precondition and Deletion List	Add List
Rule 1	pickup(X)	hand empty, on(X,table), clear(X)	holding(X)
Rule 2	putdown(X)	holding(X)	hand empty, on(X,table), clear(X)
Rule 3	stack(X,Y)	holding(X), clear(Y)	on(X,Y), clear(X)
Rule 4	unstack(X,Y)	on(X,Y), clear(X)	holding(X), clear(Y)

- Based on the above rules, plan for the block world problem : Start → goal can be specified as follows :

  1. unstack(Z,X) 2. putdown(Z) 3. pickup(Y) 4. stack(Y,Z) 5. pickup(X) 6. stack(X,Y)

- Execution of this plan can be done by making use of a data structure called "Triangular Table".

1	on (C, A) clear (C) hand empty	unstuck (C, A)					
2		holding (C)	putdown (C)				
3	on (B, table)		hand empty	pickup (B)			
4			clear (C)	Holding (B)	stack (B,C)		
5	on (A, table)	clear (A)			Hand empty	pickup (A)	
6					clear (B)	holding (A)	stack (A, B)
7			on (C, table)		on (B, C)		on (A, B) clear (A)
	0	1	2	3	4	5	6

Fig. 6.14

- In a triangular table there are N + 1 rows and columns. It can be seen from the Fig. 6.15 that rows have 1 → n + 1 condition and for columns 0 → n condition is followed. The first column of the triangular table indicates the starting state and the last row of the triangular table indicates the goal state.
- With the help of triangular table a tree is formed as shown below to achieve the goal state :

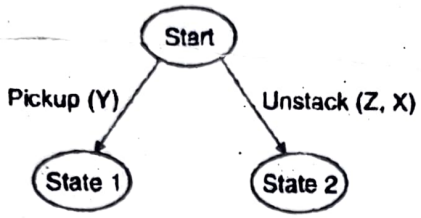


Fig. 6.15

- An agent (in this case robotic arm) can have some amount of fault tolerance. Fig. 6.16 shows one such example.

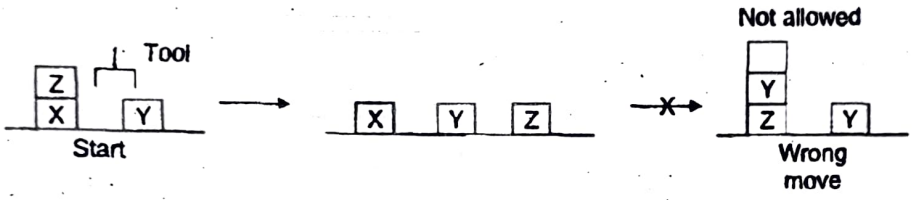


Fig. 6.16

Q. 15 Explain various types of planning methods for handling indeterminacy. **(5 Marks)**

Ans. :

Types of planning methods

- Real world problems however are not predictable and completely unobservable. The planning and acting on real world problems require more sophisticated approach.
- One of the most important characteristic of real world is uncertainty. In an uncertain environment, it is very important for agent to rely upon its percepts (series of past experience)
- Whenever some unexpected condition encountered, agent should refer its percept and should change course of action accordingly. In other words agent should be able to cancel or replace the currently executing plan with some other more suitable and reliable plan if something unexpected happens.
- Real world itself is not uncertain but human perception related to world is uncertain. In artificial intelligence, we try to give human perception ability to machine, and hence machine also receives the perception of uncertainty about the real world. So machine has to deal with incomplete and incorrect information like human does.
- Determining the condition of state depends on available knowledge. In real world, knowledge availability is always limited, so most of the time, conditions are non deterministic.
- The amount or degree of indeterminacy depends upon the knowledge available. The inter determinacy is called "bounded indeterminacy" when actions can have unpredictable effects.
- Four planning strategies are there for handling indeterminacy :

(i) Sensorless planning
(ii) Conditional planning
(iii) Execution monitoring and replanning
(iv) Continuous planning

(i) **Sensorless planning :** Sensorless planning is also known as conformant planning. These kinds of planning are not based on any perception. The algorithm ensures that plan should reach its goal at any cost.

(ii) **Conditional planning :** Conditional plannings are sometimes termed as contingency planning and deals with bounded indeterminacy. Agent makes a plan, evaluate the plan and then execute it fully or partly depending on the condition.

(iii) **Execution monitoring and replanning :** In this kind of planning agent can employ any of the strategy of planning. Additionally it observes the plan execution and if needed, replan it and again executes and observes.

(iv) **Continuous planning :** Continuous planning does not stop after performing action. It persist over time and keeps on planning on some predefined events.

These events include any type unexpected circumstance in environment.

Q. 16 Write a short note on multi-agent planning. **(5 Marks)**

Ans. : Multi-Agent Planning

- Agent acts alone in a single user environment.
- When the environment consists of multiple agent to, then the way a single agent plan its action get changed.
- A glimpse of environment where multiple agents have to take actions based on current state. The environment could be co-operative or competitive. In both the cases agent's action influences each other.

Few of the multi agent planning strategies are listed as follows :

(i) Co-operation
(ii) Multi body planning
(iii) Co-ordination mechanisms
(iv) Competition

(i) **Co-operation :** In co-operation strategy agents have joint goals and plans. Goals can be divided into sub goals but ultimately combined to achieve ultimate goal.

(ii) **Multibody planning :** Multi body planning is the strategy of implementing correct joint plan.

(iii) **Co-ordination mechanisms :** These strategies specify the co-ordination between co-operating agents. Co-ordination mechanism is used in several co-operating plannings.

(iv) **Competition :** Competition strategies are used when agents are not co-operating but competing with each other. Every agent wants to achieve the goal first.

Q. 17 Explain limits and ethics of AI. **(8 Marks)**

Ans. :

Limits of AI

1) **High Costs of Creation :** As AI is updating every day the hardware and software need to get updated with time to meet the latest requirements. Machines need repairing and maintenance which need plenty of costs. It' s creation requires huge costs as they are very complex machines.

2) **Making Humans Lazy :** AI is making humans lazy with its applications automating the majority of the work. Humans tend to get addicted to these inventions which can cause a problem to future generations.

3) **Unemployment :** As AI is replacing the majority of the repetitive tasks and other works with robots,human interference is becoming less which will cause a major problem in the employment standards. Every organization is looking to replace the minimum qualified individuals with AI robots which can do similar work with more efficiency.

4) **No Emotions :** There is no doubt that machines are much better when it comes to working efficiently but they cannot replace the human connection that makes the team. Machines cannot develop a bond with humans which is an essential attribute when comes to Team Management.

5) **Lacking Out of Box Thinking :** Machines can perform only those tasks which they are designed or programmed to do, anything out of that they tend to crash or give irrelevant outputs which could be a major backdrop.

Ethics of AI

1. **Unemployment : What happens after the end of jobs ?**

   The hierarchy of labour is concerned primarily with automation. As we've invented ways to automate jobs, we could create room for people to assume more complex roles, moving from the physical work that dominated the pre-industrial globe to the cognitive labour that characterizes strategic and administrative work in our globalized society.

2. **Inequality : How do we distribute the wealth created by machines ?**

   The majority of companies are still dependent on hourly work when it comes to products and services. But by using artificial intelligence, a company can drastically cut down on relying on the human workforce, and this means that revenues will go to fewer people. Consequently, individuals who have ownership in AI-driven companies will make all the money.

3. **Humanity : How do machines affect our behaviour and interaction ?**

   Artificially intelligent bots are becoming better and better at modelling human conversation and relationships. A bot named Eugene Goostman won the Turing Challenge for the first time. In this challenge, human raters used text input to chat with an unknown entity, then guessed whether they had been chatting with a human or a machine. Eugene Goostman fooled more than half of the human raters into thinking they had been talking to a human being.

4. **Artificial stupidity : How can we guard against mistakes?**

   Intelligence comes from learning, whether you're human or machine. Systems usually have a training phase in which they "learn" to detect the right patterns and act according to their input. Once a system is fully trained, it can then go into test phase, where it is hit with more examples and we see how it performs.

5. **Racist robots : How do we eliminate AI bias?**

   Though artificial intelligence is capable of a speed and capacity of processing that's far beyond that of humans, it cannot always be trusted to be fair and neutral. Google and its parent company Alphabet are one of the leaders when it comes to artificial intelligence, as seen in Google's Photos service, where AI is used to identify people, objects and scenes. But it can go wrong, such as when a camera missed the mark on racial sensitivity, or when a software used to predict future criminals showed bias against black people.

6. **Security : How do we keep AI safe from adversaries?**

   The more powerful a technology becomes, the more can it be used for nefarious reasons as well as good. This applies not only to robots produced to replace human soldiers, or autonomous weapons, but to AI systems that can cause damage if used maliciously. Because these fights won't be fought on the battleground only, cybersecurity will become even more important. After all, we're dealing with a system that is faster and more capable than us by orders of magnitude.

7. **Robot rights : How do we define the humane treatment of AI?**

   While neuroscientists are still working on unlocking the secrets of conscious experience, we understand more about the basic mechanisms of reward and aversion. We share these mechanisms with even simple animals. In a way, we are building similar mechanisms of reward and aversion in systems of artificial intelligence. For example, reinforcement learning is similar to training a dog: improved performance is reinforced with a virtual reward.

Q. 18 What are the different Components of AI ? **(5 marks)**

Ans. :

AI Components

AI is a vast field for research and it has got applications in almost all possible domains. By keeping this in mind, components of AI can be identified as follows : (refer Fig. 6.17)

1.	Perception	2.	Knowledge representation
3.	Learning	4.	Reasoning
5.	Problem Solving	6.	Natural Language Processing (language-understanding).

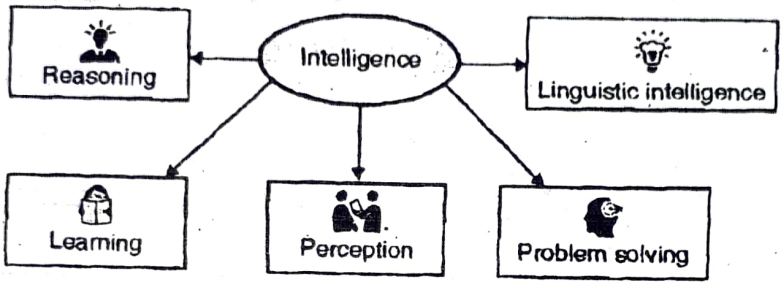


Fig. 6.17 : Components of AI

1. **Perception :** In order to work in the environment, intelligent agents need to scan the environment and the various objects in it. Agent scans the environment using various sense organs like camera, temperature sensor, etc. This is called as perception. After capturing various scenes, perceiver analyses the different objects in it and extracts their features and relationships among them.
2. **Knowledge representation :** The information obtained from environment through sensors may not be in the format required by the system. Hence, it need to be represented in standard formats for further processing like learning various patterns, deducing inference, comparing with past objects, etc. There are various knowledge representation techniques like Prepositional logic and first order logic.
3. **Learning :** Learning is a very essential part of AI and it happens in various forms. The simplest form of learning is by trial and error. In this form the program remembers the action that has given desired output and discards the other trial actions and learns by itself. It is also called as unsupervised learning. In case of rote learning, the program simply remembers the problem solution pairs or individual items. In other case, solution to few of the problems is given as input to the system, basis on which the system or program needs to generate solutions for new problems. This is known as supervised learning.
4. **Reasoning :** Reasoning is also called as logic or generating inferences form the given set of facts. Reasoning is carried out based on strict rule of validity to perform a specified task. Reasoning can be of two types, deductive or inductive. The deductive reasoning is in which the truth of the premises guarantees the truth of the conclusion while, in case of inductive reasoning, the truth of the premises supports the conclusion, but it cannot be fully dependent on the premises. In programming logic generally deductive inferences are used. Reasoning involves drawing inferences that are relevant to the given problem or situation.
5. **Problem-solving :** AI addresses huge variety of problems. For example, finding out winning moves on the board games, planning actions in order to achieve the defined task, identifying various objects from given images, etc. As per the types of problem, there is variety of problem solving strategies in AI. Problem solving methods are mainly divided into general purpose methods and special purpose methods. General purpose methods are applicable to wide range of problems while, special purpose methods are customized to solve particular type of problems.

6. **Natural Language Processing :** Natural Language Processing, involves machines or robots to understand and process the language that human speak, and infer knowledge from the speech input. It also involves the active participation from machine in the form of dialog i.e. NLP aims at the text or verbal output from the machine or robot. The input and output of an NLP system can be speech and written text respectively.

Q. 19 Write a short note on : Planning as State-Space Search. **(5 Marks)**

Ans. :

- Fig. 6.18 shows that the office agent is at location 250 on the state space grid. When he gets a task he has to decide which task can be performed more efficiently in lesser time.

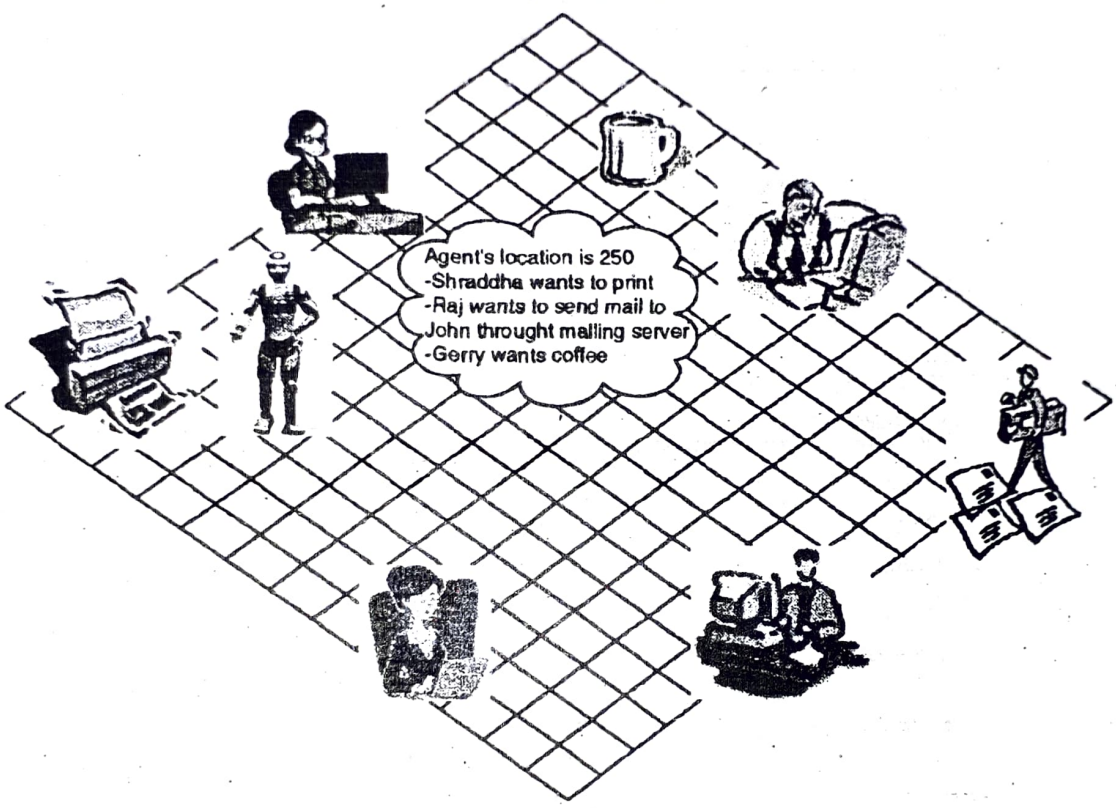


Fig. 6.18 : Office agent example with finite state space

- If it finds some input and output locations nearer on state space grid for example in case of printing task then the probability of performing that task will increase.
- But to do this it should be aware of it's own current location, the locations of people who are assigning tasks and the locations of the required devices.
- State space search is unfavourable for solving real-world problems because, it requires complete description of every searched state, also search should be carried out locally.
- There can be two ways of representations for a state :

1. Complete world description
2. Path from an initial state

1. Complete world description

- Description is available in terms of an assignment of a value to each previous suggestion.
- Or we can say that description is available as a suggestion that defines the state.
- Drawback of this types is that it requires a large amount of space.

2. Path from an initial state

- As per the name, path from an initial state gives the sequence of actions which are used to reach a state from an initial state.
- In this case, what holds in a state can be deduced from the axiom which specifies the effects of an actions.
- Drawback of these types is that it does not explicitly specify "What holds in every state". Because of this it can be difficult to determine whether two states are same.

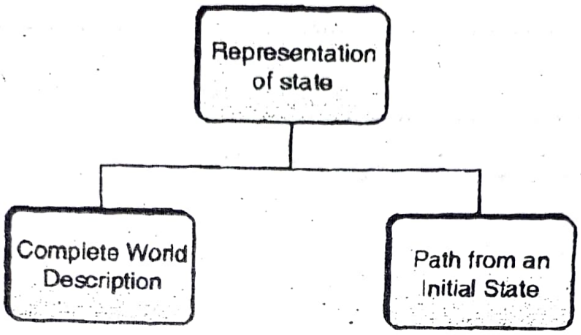


Fig. 6.19 : Representation of states

Q. 20 Explain planning problem for spare tire problem. **(5 Marks)**

Ans. : Example of the Spare Tire Problem

- Consider the problem of changing a flat tire. More precisely, the goal is to have a good spare tire properly mounted onto the car's axle, where the initial state has a flat tire on the axle and a good spare tire in the trunk. To keep it simple, our version of the problem is a very abstract one, with no sticky lug nuts or other complications.
- There are just four actions: removing the spare from the trunk, removing the flat tire from the axle, putting the spare on the axle, and leaving the car unattended overnight. We assume that the car is in a particularly bad neighborhood, so that the effect of leaving it overnight is that the tires disappear.
- The ADL description of the problem is shown: It is purely propositional. It goes beyond STRIPS in that it uses a negated precondition, ¬At(Flat, Axle), for the PutOn(Spare, Axle) action. This could be avoided by using Clear (Axle) instead, as shown as follows.

Solution using STRIPS

- Init (At (Flat, Axle) ∧ At(Spare, Trunk))
- Goal (At (Spare, Axle)) Action(Remove(Spare, Trunk),
- PRECOND : At (Spare, Trunk)
- EFFECT : ¬ At (Spare, Trunk) ∧ At (Spare, Ground))
- Action(Remove(Flat, Axle),
- PRECOND : At (Flat, Axle)
- EFFECT : ¬ At (Flat, Axle) ∧ At (Flat, Ground))
- Action(PutOn (Spare, Axle),
- PRECOND : At (Spare, Ground) ∧ ¬ At(Flat, Axle)
- EFFECT : ¬ At (Spare, Ground) ∧ At(Spare, Axle))
- Action(LeaveOvernight)
- PRECOND
- EFFECT : ¬ At(Spare, Ground) ∧ ¬ At(Spare, Axle) ∧ ¬ At(Spare, Trunk) ∧ ¬ At(Flat, Ground) ∧ ¬ At(Flat, Axle))

Q. 21 Write a short note on : Job Shop Scheduling Problem. **(5 Marks)**

Ans. :

Job Shop Scheduling Problem

- Job shop scheduling problem, also known as "job-shop problem" is an optimization problem in AI, in which ideal jobs are assigned to resources at particular time schedules.
- Let us consider a shop floor where jobs are processed by machines. Every job consists of a certain number of operations. Every operation needs to be performed by a dedicated machine and requires a fixed amount of time for processing. Each job has to follow a predefined sequence of operations and all the jobs are independent of each other and are independent of machines.
- **In its most basic version, the problem appears as follows :**
  - In the job shop scheduling problem there are n jobs of varying sizes say J_1, J_2, J_3,...J_n which are to be processed on m number of machines say M_1, M_2, M_3, ... M_m. Every job J_i consists of k number of operations say, Oi_1, Oi_2, Oi_3, ... ,O_{ik} and each operation has a processing time of P_{ik}. The aim is to find the sequence of operation for each machine such that the total time for any machine is minimum and at a time only one operation is processed on a machine.
  - This problem has number of variations in it, considering the three variables namely job, machines and the objective function itself. Few of them are as follows.
- **Variations based on machines :**
  - Machines can be related, independent.
  - Machines may have sequence dependent setups.
  - Machines may require a certain gap between jobs or no idle-time.
  - Machines having any other limitations of time, resources, etc.
- **Variations based on different objective Function :**
  - To minimize the total time each machine needs,
  - To minimize tardiness,
  - To maximize lateness etc.
  - It can also be combination of multiple objectives leading to multi-objective optimization problem
- **Variations based on Jobs :**
  - Jobs may have constraints or dependencies within themselves.
  - Jobs and machines have mutual constraints, for example, certain jobs can be scheduled on some machines only.
  - Jobs may have fixed or probabilistic processing time.
- Now a days, the problem is presented as an online problem (dynamic scheduling), that is, each job is presented, and the online algorithm needs to make a decision about that job before the next job is presented.

Q. 22 Write short note on planning graphs. **(5 Marks)**

Ans.:

Planning graphs

- Planning graph is a special data structure which is used to get better accuracy. It is a directed graph and is useful to accomplish improved heuristic estimates. Any of the search technique can make use of planning graphs. Also GRAPHPLAN can be used to extract a solution directly.

- Planning graphs work only for propositional problems without variables. Similarly, in case of planning graphs there are series of levels which match to time ladder in the plan. Every level has set of literals and a set of actions. Level 0 is the initial state of planning graph.

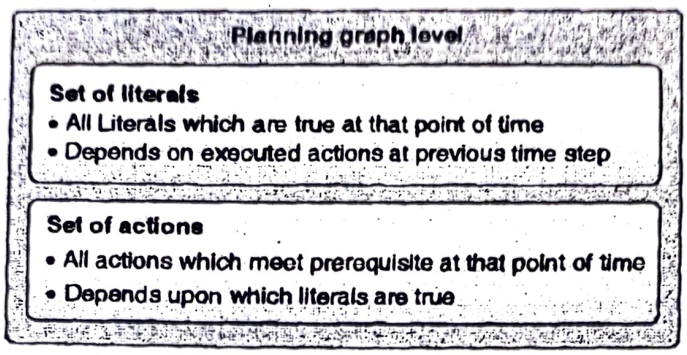


Fig. 6.20 : Planning graph level

Example

- Init (Have(Apple))
- Goal(Have(Apple) ∧ Ate(Apple))
- Action(Eat(Apple), PRECOND : Have(Apple)
- EFFECT : ¬Have(Apple) ∧ Ate(Apple))
- Action(Cut(Apple), PRECOND : ¬ Have(Apple)
- EFFECT : Have(Apple))

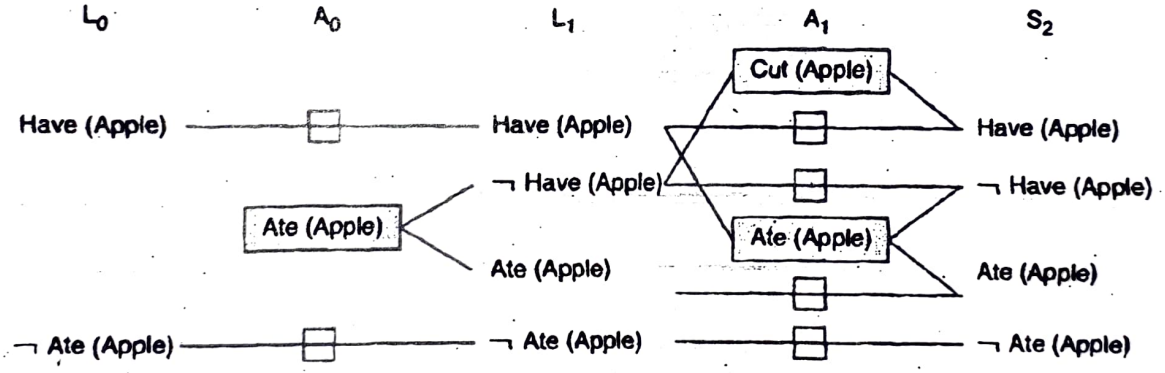


Fig. 6.21

- Start at level L_0 and determine action level A0 and next level L_1.
  - A_0 >> all actions whose prerequisite is satisfied at previous level.
  - Connect precondition and effect of actions L_0 – > L_1.
  - Inaction is represented by persistence actions.
  - Level A_0 contains the possible actions.
  - Conflicts between actions are shown by mutual exclusion links.
  - Level L_1 contains all literals that could result from picking any subset of actions in Level A_0.
  - Conflicts between literals which cannot occur together (as a effect of selection action) are represented by mutual exclusion links.
  - L_1 defines multiple states and the mutual exclusion links are the constraints that define this set of states.
- Continue until two consecutive levels are the same or contain the same amount of literals.

GRAPH PLAN algorithm

GRAPH PLAN can directly extract solution from planning graph with the help of following algorithm :

```
function GRAPHPLAN(problem) return solution or failure
    graph ← INITIAL-PLANNING-GRAPH(problem)
    goals ← GOALS[problem]
    loop do
      if goals all non-mutex in last level of graph then do
        solution ← EXTRACT-SOLUTION(graph, goals,
LENGTH(graph))
        if solution ≠ failure then return solution
        else if NO-SOLUTION-POSSIBLE(graph) then return failure
        graph ← EXPAND-GRAPH(graph, problem)
```

Q. 23 What are the Future of AI **(5 Marks)**

Ans. :

Future of AI

These scenarios represent possible trajectories for humanity. None of them, though, is unambiguously achievable or desirable. And while there are elements of important agreement and consensus among the visions, there are often revealing clashes, too.

1. **Shared economic prosperity**

The economic benefits of technological progress are widely shared around the world. The global economy is 10 times larger because AI has massively boosted productivity. Humans can do more and achieve more by sharing this prosperity. This vision could be pursued by adopting various interventions, from introducing a global tax regime to improving insurance against unemployment.

2. **Realigned companies**

Large companies focus on developing AI that benefits humanity, and they do so without holding excessive economic or political power. This could be pursued by changing corporate ownership structures and updating antitrust policies.

3. **Flexible labour markets**

Human creativity and hands-on support give people time to find new roles. People adapt to technological change and find work in newly created professions. Policies would focus on improving educational and retraining opportunities, as well as strengthening social safety nets for those who would otherwise be worse off due to automation.

4. **Human-centric AI**

Society decides against excessive automation. Business leaders, computer scientists, and policymakers choose to develop technologies that increase rather than decrease the demand for workers. Incentives to develop human-centric AI would be strengthened and automation taxed where necessary.

5. **Fulfilling jobs**

New jobs are more fulfilling than those that came before. Machines handle unsafe and boring tasks, while humans move into more productive, fulfilling, and flexible jobs with greater human interaction. Policies to achieve this include strengthening labour unions and increasing worker involvement on corporate boards.

6. **Civic empowerment and human flourishing**

In a world with less need to work and basic needs met by UBI, well-being increasingly comes from meaningful unpaid activities. People can engage in exploration, self-improvement, volunteering or whatever else they find satisfying. Greater social engagement would be supported.

□□□

Note