

## Unit - I

### Introduction

- \* Intelligence means the capacity to learn and solve problems.
- \* Artificial Intelligence is the Science and engineering that is concerned with the theory and practice of developing systems that exhibit the characteristics we associate with the intelligence in human behaviour: solving, Learning and adaptation, etc.

→

#### Weak AI

→ weak AI refers to use of software to study or accomplish specific problem solving or reasoning tasks that do not encompass all human cognitive abilities.

→ Weak AI claims that machines can act intelligently or merely specific problem solver.

#### \* Four views of AI:

1.) Thinking humanly

2.) Acting humanly

3.) Thinking rationally

4.) Acting rationally

#### \* Applications of AI:

1.) Game playing

2.) Speech and character recognition

3.) Natural language Processing

4.) Expert Systems

5.) Robotics

6.) Information Predictors

7.) Computer vision and pattern recognition.

#### \* Challenges before AI:

1.) Natural Language processing

3.) Automated Reasoning

2.) Knowledge Representation

4.) Machine Learning

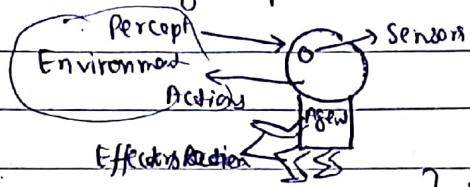
\* An intelligent agent is a software entity which senses its environment and carries out operations on behalf of user, with some degree of autonomy and employs some knowledge or representation of user's goals or desires.

#### Strong AI

→ Strong AI supposes, it is possible for machines to become self-aware but may or may not exhibit human thought process.

→ Strong AI claims that machines can act intelligently, has mind and understanding.

- \* An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors or actuators.



\* Agent = architecture + program } Structure of agent

\* An environment provides the conditions under which an entity exists (agent)

#### \* Properties of Environments:

(i) Accessible vs Inaccessible

(ii) Deterministic vs Nondeterministic

(iii) Episodic vs Nonepisodic (Note: experience divided into episodes)

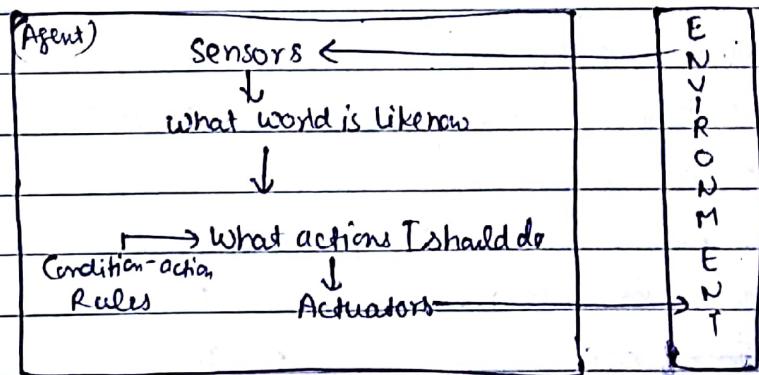
(iv) Static vs dynamic

(v) Discrete vs continuous

#### \* Types of Agents:

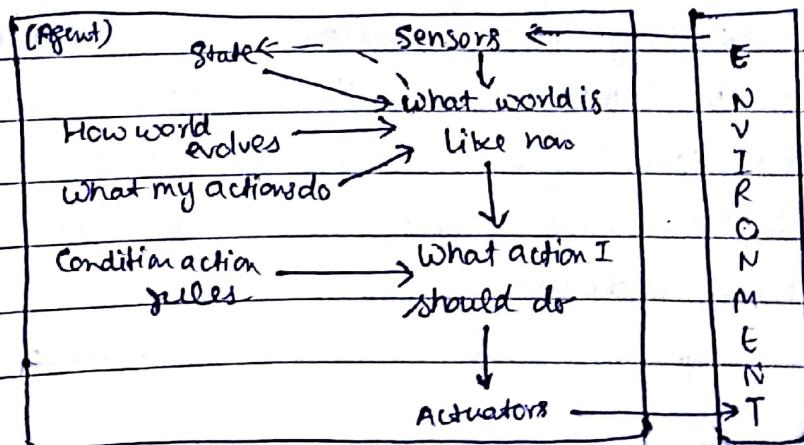
##### (i) Simple Reflex Agents:

- Select actions on basis of current percept ignoring rest of percept history.



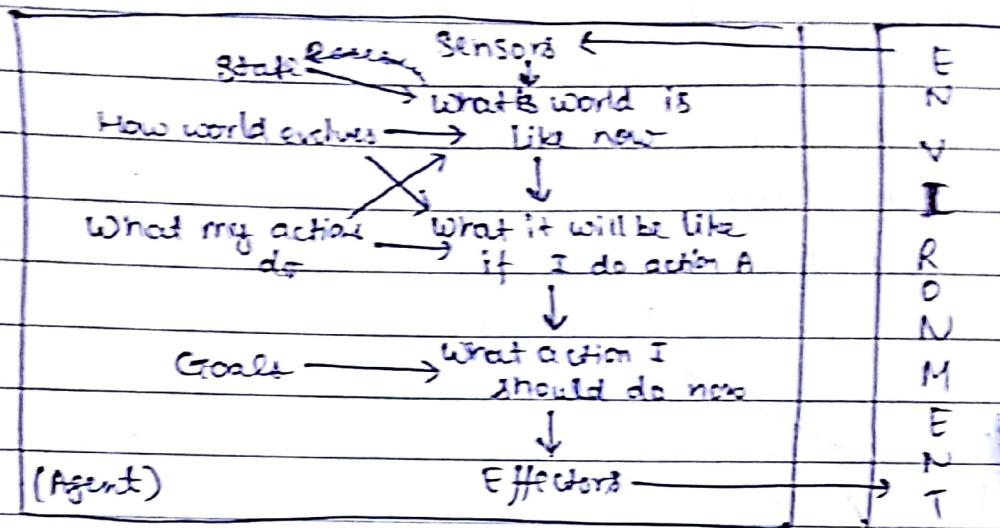
##### (ii) Model-Based Reflex Agents:

- Deal with partial accessibility; do this by keeping track of part of world it can see now.



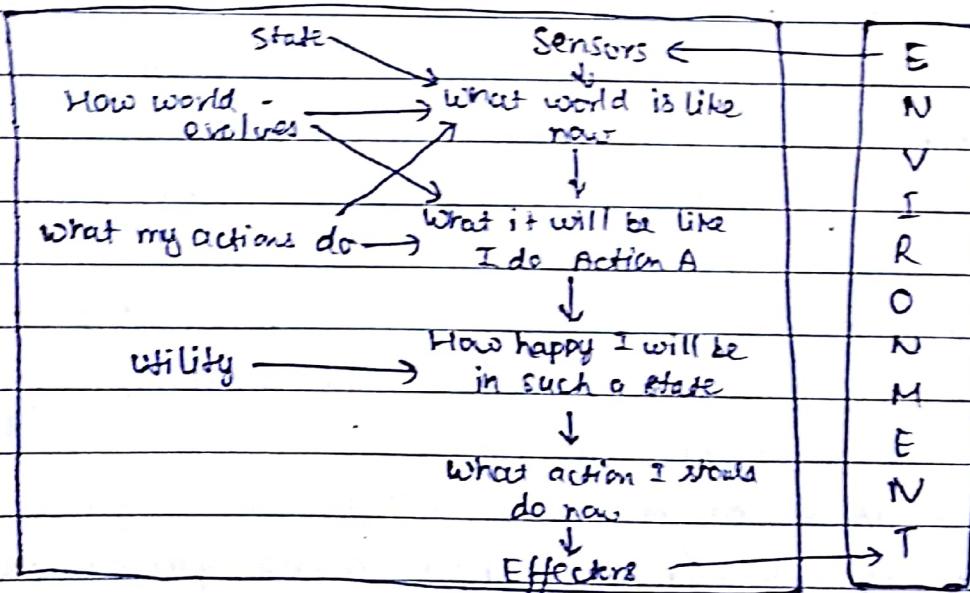
### (iii) Goal based Agents

- Set goals to achieve, this pushes right decisions when need to



### (iv) Utility based Agents

- A utility funcn to map each state after each action to a real no. representing how efficiently each action achieves the goal.



~~→ Designing an agent system:~~

→ PEAS of intelligent agent:

P → Performance

E → Environment

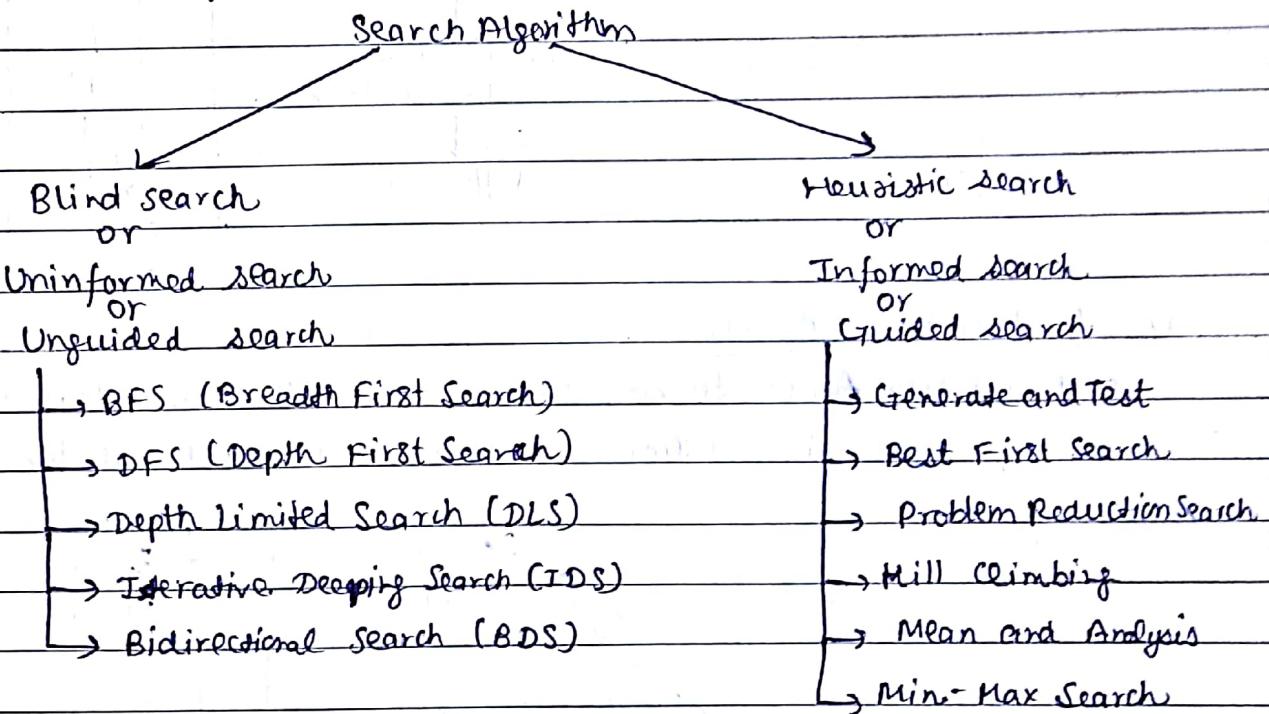
A → Actuators

S → Sensors

## \* Nature of AI problems:

- (i) path finding problem
- (ii) decomposable problem
- (iii) recoverable problem
- (iv) predictable problem
- (v) problem requiring Interaction

## \* Search Techniques in AI:



### \* Uninformed search

- (i) Also called Blind search, brute force search & exhaustive search
- (ii) Information is not given.
- (iii) Doesn't provide optimized sol<sup>n</sup>.
- (iv) Guaranteed sol<sup>n</sup> is found.

### Informed search

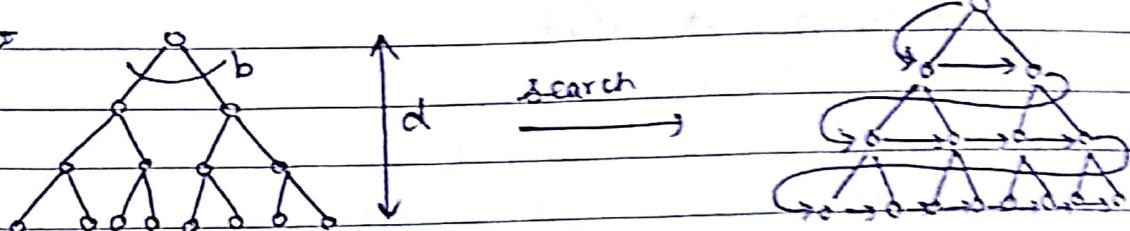
- (i) Also called intelligent search
- (ii) Information is given
- (iii) Provide optimized sol<sup>n</sup>.
- (iv) Sol<sup>n</sup> may not be guaranteed

## \* Breadth First Search:

- Expand root node
- Expand all children of root node
- Expand all grandchildren etc.
- Expand all nodes at depth d before expanding nodes at depth d+1

→ It searches breadth wise in problem space.  
In BFS, the shallowest unexpanded node is chosen for expansion.

→ ~~Right~~



say this is tree to be searched

- ~~After~~ Steps:
- (i) Start from root node
  - (ii) Expand its child & search among the children
  - (iii) If not expand, expand the children of the children & search among them
  - (iv) Repeat step (ii) & (iii) until  $sol^n$  is found or all the nodes are traversed.

→ Advantages: (i) Never get trapped exploring useless path forever  
(ii) If there is  $sol^n$ , BFS will definitely find it out.  
(iii) If more than one  $sol^n$ , BFS can find  $sol^n$  (with less no. of steps).

→ Disadvantages: (i) Memory requirements

(ii) consume lot of time if  $sol^n$  is farther from root

→ Time complexity =  $O(b^d)$        $b \rightarrow$  branching factor

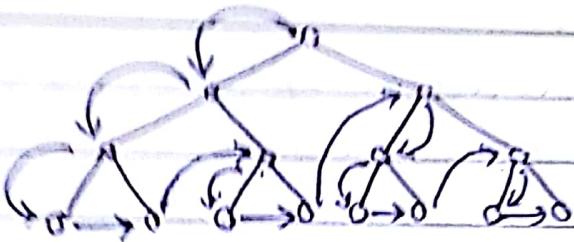
→ Space complexity =  $O(b^d)$        $d \rightarrow$  depth

### \* Depth-First Search (DFS)

- deepest node is expanded first

- Backtrack when path can't be further expanded

→ It progresses by expanding the first child node of tree that appears & thus going deeper & deeper until goal node is found or until it hits a node with no children, then backtracks & search unexplored node.



### Traversing the tree

- Advantages:-
  - Memory requirement is only linear w.r.t search graph
  - Time complexity is  $O(b^d)$
  - Take very less time & space
- Disadvantage:-
  - May go down leftmost path forever
  - No guarantee to find minimal sol<sup>n</sup>
- Complexity : Time =  $O(b^m) \rightarrow$  maximum depth  
Space =  $O(bm)$

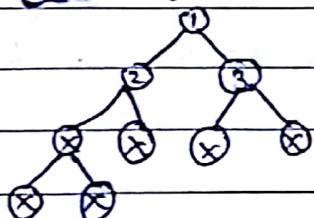
### \* Depth Limited Search

- A limit of depth is defined. Any nodes below that are omitted from search. (Similar to depth first search)
- It can terminate with two kinds of failure:
  - Standard failure value : No sol<sup>n</sup>
  - Cut-off value : No sol<sup>n</sup> within depth limit.
- Time complexity =  $O(b^d) \rightarrow$  limit of depth  
Space complexity =  $O(b^d)$

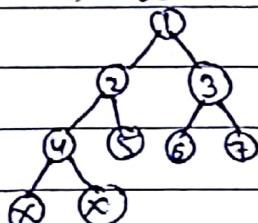
### \* Iterative Deepening Search

- Derivative of DLS & combines DFS & BFS
- Operates by performing DLS searches with increased depths until goal is found.
- The depth begins at one & increases until goal is found or no further nodes can be enumerated
- Always finds best sol<sup>n</sup> so complete & optimal search

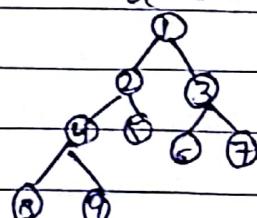
→ Ex:  $d=1$



$d=2$



$d=3$

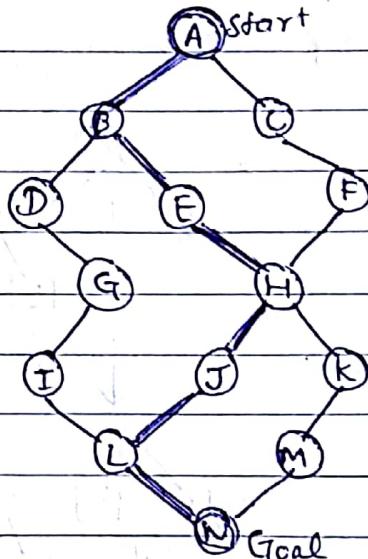


- Time complexity =  $O(b^d)$   
Space complexity =  $O(b^d)$

### \* Bi-directional Search

→ The idea is to reduce search time by searching forward from start and backward from goal simultaneously.

→



Bidirectional search meeting <sup>in</sup> middle at node H

→ Bidirectional search is complete & optimal

→ Complexity :  $b^{\frac{k}{2}} + b^{\frac{k}{2}} = 2b^{\frac{k}{2}} < b^k$

### \* Informed (Heuristic) Search

→ Informed search methods incorporate a heuristic, which is used to determine quality of any state in the search space.

→ A heuristic is a rule of thumb that <sup>may</sup> help solve a given problem. Heuristic take problem knowledge into consideration to help guide the search within the domain.

→ A heuristic funcn is a funcn that maps from problem state description to measures desirability, usually represented as number weights. The value of the heuristic funcn at a given node in the search process gives a good estimate of that node being on the desired path to solution.

### \* Generate & Test Algorithm

(i) Generate a possible soln

(ii) Test to see if <sup>this</sup> is a soln

(iii) If sol<sup>n</sup> found, quit. Otherwise, return to (i)

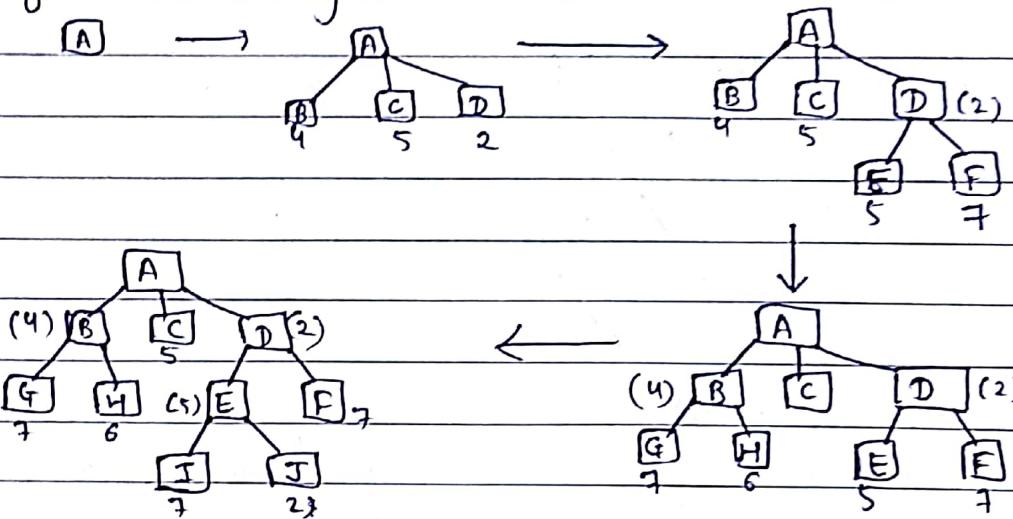
## \* Best First Search

→ Evaluation funcn with every node

→ Cost of evaluation funcn = cost / distance of current node from goal node

→ The decision of which node to be expanded next depends upon the value of evaluation funcn.

→ Eg:-



→ At any step, the most promising node having least value of utility function is chosen for expansion.

→ In best first search, two list of nodes are used to implement graph search: (i) open list : nodes not examined yet but have heuristic funcn applied to them.

(ii) Closed list: nodes already examined. When new node generated, check if generated earlier:

→ Advantages: Combines advantages of BFS & DFS

Disadvantages: Memory requirement

→ Best first search is implemented using priority queue.

→ Algo : (i) put initial node on START list

(ii) If START empty or equal to GOAL, quit search

(iii) Else remove first node from START, call it node 'a'

(iv) If ' $a$ ' =  $C_{T0AL}$ , terminate with success.

(v) Else if 'a' has successors, generate all child & apply heuristic func to it (distance from goal) & sort

(vi) Name this list as START1

(vii) Replace START with START1

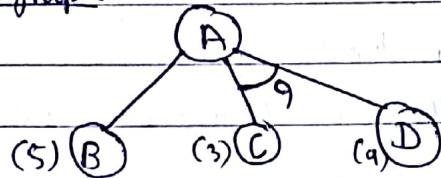
(viii) Goto step (ii)

## \* Optimal Best First Search (A\* algorithm)

- Evaluation funcn,  $f(n) = g(n) + h(n)$  → cost of current node from goal

## \* Problem Reduction Search

- AND-OR graph



$$f' = 5 + 1 = 6$$

$$= (3 + 1) + (4 + 1) = 4 + 5 = 9$$

## \* Means Ends Analysis (e.g. ex-CTPS)

- The search process over the problem space combines aspects of both forward & backward reasoning in that both the condition & action portions of rules are looked at when considering which rule to apply. Differences between current and goal states are used to propose operators which reduce the differences.

## \* Hill Climbing Search

- Search method for finding a maximum (or minimum) of an evaluation function. It considers the local neighbourhood of a node, evaluating each neighbour node and next examines those nodes with largest (or smallest) values.

- Unlike other search strategies, hill climbing search doesn't permit to shift attention back to previously suspended alternatives, even though they may have offered a better alternative than ones at hand.

- It is not guaranteed to lead to a sol'n, since it can get stuck on plateaus, or local optimum, or <sup>even</sup> wander on uncontrolled paths.

- It uses a simple heuristic funcn, i.e., amount of distance the node is from the goal.

- Using heuristics, it will find direction which takes closest to goal. It doesn't maintain search tree. It stores current node structure.
- It becomes inefficient in large problem spaces & when combinatorial explosion occurs.
- Advantages: (i) Optimization technique for solving computationally hard problems.  
 (ii) Memory efficient & always attempts to improve current state.
- Difficulties: (i) Local maximum  
 (State better than all neighbours but not better than farther states)
- (ii) Plateau (flat area of search space in which neighbours have same value)
- (iii) Ridge (area is higher than surrounding areas but can't be searched in a simple move)
- Three variations:
  - (i) Steepest Ascent Hill Climbing
    - Climbing all successors are compared & closest to sol<sup>n</sup> is chosen (Similar to best first search)
  - (ii) Stochastic Hill Climbing
    - Select neighbour at random & decides whether to move or examine other.
  - (iii) Random-restart Hill Climbing / Shotgun Hill Climbing
    - Does Hill Climbing each time with a random initial condition  $x_0$ .
    - The best  $x_m$  is kept: if a new run of hill climbing produces a better  $x_m$  than stored space, it replaces stored state.

## \* Constraint Satisfaction Problem

- CSP consists of triplets  $\{X, D, C\}$  where  $X = \{x_1, \dots, x_n\}$  (set of variables),  $D = \{D_1, \dots, D_n\}$  (set of domain values),  $C = \{C_1, \dots, C_n\}$  (set of constraints).

## → Algorithm / Pseudocode

Until a complete sol<sup>n</sup> is found or until all paths have lead to dead ends, do the following:

- (i) select an unexpanded node of the search graph
- (ii) apply the constraint inference rules to the selected node to generate all possible new constraints
- (iii) if set of constraints contain a contradiction, then report that this path is a dead end.
- (iv) if set of constraints describe a complete sol<sup>n</sup> then report success.
- (v) if neither a constraint nor a complete sol<sup>n</sup> is found then apply rules to generate a new partial sol<sup>n</sup> & insert these partial sol<sup>n</sup> into search graph.

→ Example of problems: CryptArithmetic Problem, Water Jug problem etc.

## \* Means Ends Analysis Problem

Ex:- Moving a desk from one room to other with two things:

	PUSH	CARRY	WALK	PICKUP	PUTDOWN	PLACE
MOVE MOVE object object	*	*				
MOVE ROBOT	*	*	*			
MOVE CLEAR OBJECT				*		
GET OBJECT ON OBJECT					*	*
MOVE ARM EMPTY						*
BE HOLDING OBJECT				*		*

Operator	Preconditions	Results
PUSH(obj, loc)	at(robot, obj) & large(obj) & clear(obj) & arm empty	at(obj, loc) & at(robot, loc)
CARRY(obj; loc)	at(robot, obj) & small(obj)	at(obj, loc) & at(robot, loc)
WALK(loc)	none	at(robot, loc)
PICKUP(obj)	at(robot, obj)	Holding(obj)
PUTDOWN(obj)	Holding(obj)	not holding(obj)
PLACE(obj1, obj2)	at(robot, obj2) & holding(obj1)	on(obj1, obj2)

## \* ~~Graph~~ Water Jug Problem

ex:- 1<sup>st</sup> jug - 4 l + 2<sup>nd</sup> jug - 3 l

Fill 4 l jug with 2 l

State space  $(x, y)$

<u>Production rules</u> :	1 : $(x:y) \rightarrow (4,y)$	(fill 4 l jug; $x \leq 4$ )
rule 2 :	$(x:y) \rightarrow (x,3)$	(fill 3 l jug; $y \leq 3$ )
rule 3 :	$(x:y) \rightarrow (x-x, y)$	(pour water out of 4 l jug)
rule 4 :	$(x:y) \rightarrow (x, y-y)$	(pour " " " 3 l " )
rule 5 :	$(x:y) \rightarrow (0,y)$	(Empty 4 l jug)
rule 6 :	$(x:y) \rightarrow (x,0)$	(Empty 3 l " )
rule 7 :	$(x:y) \rightarrow (4, y-(4-x))$	(Fill 4 l by pouring <sup>some</sup> water from 3 l jug)
rule 8 :	$(x:y) \rightarrow (x-(x-y), 3)$	(" 3 l " " " " 4 l jug)
rule 9 :	$(x:y) \rightarrow (x+y, 0)$	(Pour all water from 3 l jug to 4 l jug)
rule 10 :	$(x:y) \rightarrow (0, x+y)$	(" " " " 4 l " " 3 l " )
rule 11 :	$(0,2) \rightarrow (2,0)$	( <sup>or</sup> 2 l from 3 l jug to 4 l jug)
rule 12 :	$(2,y) \rightarrow (0,y)$	(Empty " " 4 l jug on ground)

Output: Rule No. applied

Initial state.

Jug 1 (4 l)

Jug (3 l)

1	0	0
8	1	3
6	1	0
10	0	1
1	4	1
8	2	3

(or)

Initial state

2	0	0
9	3	0
2	3	3
7	4	2
5	0	2
9	2	0

## \* Cryptarithmic Problem

Eg:- SEND

+ MORE  
MONEY

$$\begin{array}{cccc} c_1 & c_2 & c_3 & c_4 \\ & S & E & N & D \\ + & M & O & R & E \\ \hline & M & O & N & E Y \end{array}$$

(i) M=1 & single digit no sum can't < 19

so M=1,  $c_1=1$

$$\begin{array}{l} \text{(ii)} \quad \cancel{\text{SEND}} = c_1 \overset{10}{\cancel{+}} c_0 + c_2 + c_3 + c_4 \\ \quad \quad \quad S + M + E + N + D = c_1 \times 10 + 0 \\ \quad \quad \quad c_1 + S + M + E + N = 10 + 0 \\ \quad \quad \quad c_2 + S = 9 + 0 \end{array} \quad \left| \begin{array}{l} c_2 + S + M > 9 \\ c_2 + S + 1 > 9 \end{array} \right.$$

$$\text{If } c_2 = 0$$

$$S + M + c_3 \cancel{+} 10 > 9$$

$$8 + 1 + 0 = 9 \times$$

$$8 + 1 + 1 = 10 \times \text{ as } M \neq 0$$

$$8 + 1 + 1 = 10 \times$$

$$9 + 1 + 0 = 10 \times$$

but S=9 so  $c_2 \neq 0$  let  $c_2 = 0$  so S=9

$$\text{If } c_2 = 1$$

$$8 + 1 + 1 > 9$$

$$8 + 1 + 1 > 9$$

$$\begin{array}{l} \text{(iii)} \quad E + 0 \cancel{+} c_3 < 10 \\ \quad \quad \quad E + 0 + c_3 < 10 \quad \text{if } N < 10 \quad N \neq 9 \text{ (as } S=9) \\ \quad \quad \quad E + c_3 < 10 \quad \text{so } N < 9 \end{array}$$

$$E + c_3 < 9$$

so  $c_3 \neq 0$  as  $E=N$

so  $c_3 = 1$  &  $E + 1 = N$

$$\text{(iv)} \quad \cancel{\text{MORE}} - \cancel{\text{SEND}} = \cancel{\text{C10}} \text{ i.e. } \cancel{\text{D+E}} \quad \cancel{\text{C4+N+R}} = 10 + E$$

• Let

$$c_1 + E + R = 10 + E$$

$$c_1 + R = 9$$

Let  $c_1 = 0$  so  $R = 9$  but  $S=9$  so X

Let  $c_1 = 1$  so  $R = 8$  &

if R other

$$\text{(v)} \quad D + E = 10 + Y$$

$$D + E = 10 + Y \quad \begin{array}{l} \cancel{10} \times \\ \cancel{1} \end{array}$$

$$\cancel{10} + \cancel{E} \cancel{= 1} \quad \begin{array}{l} \cancel{12} \times \\ \cancel{1} \end{array}$$

$$\cancel{10} + \cancel{E} \cancel{= 1} \quad \begin{array}{l} \cancel{13} \times \\ \cancel{1} \end{array} \text{ as } 7+6, 6+7$$

~~7+6=13~~

or

~~6+7=13~~

$$12 = 4+8, 8+4, 5+7$$

so

$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline \text{MONEY} \end{array}$$

$$\Rightarrow \begin{array}{r} 9567 \\ + 1085 \\ \hline 10652 \end{array}$$

Ex:- C<sub>4</sub> (some)  
SOME  
+ TIME  
SPENT

(i)  $S = 1$  &  $C_4 = 1$  as sum of digits  
 sum of two digit no plus carry. can't be greater than 19

$$(ii) C_3 + S + T = P + 10 C_4$$

$$C_3 + S + T = P + 10 \quad \text{if } E + E = 10 C_4 + T$$

sum of two even or odd nos is always even so  $T$  is 0, 2, 4, 6, 8

(iii)  $C_3$  can be either 1 or 0

Let  $C_3 = 0$  so

$$S + T = P + 10 \quad [\because S = 1]$$

$$T = P + 9$$

If  $P = 0$ ,  $T = 9$  [Not possible as  $T$  is 0, 2, 4, 6, 8]

If  $P = 1$ ,  $T = 10$  [Not possible as  $T < 9$ ]

so  $C_3 \neq 0$

(iv) Let  $C_3 = 1$  so

$$1 + S + T = P + 10 \quad [\because S = 1]$$

$$T = P + 8$$

If  $P = 0$ ,  $T = 8$

$P = 1$ ,  $T = 9$  [Not possible as  $T$  is even]

$P = 2$ ,  $T = 10$  [Not possible as  $T < 9$ ]

so  $P = 0$  if  $T = 8$  &  $C_3 = 1$

(v) If  $T = 8$  then  $E$  can be 4 or 9

$$E + E = 10 C_1 + 8 = 18 \text{ or } 8$$

(vi) Let  $E = 4$  so  $C_1 = 0$

$$C_1 + M + M = N + 10 C_2$$

$M + M = N + 10 C_2$  so  $N$  is even no [2, 4, 6, 8]

$$N = 2, 4, 6, 8$$

$N \neq 8$  as  $N \neq T$  [ $\because T = 8$ ]

$$C_2 + 0 + I = 10 C_3 + E$$

$$\therefore E = 4 \text{ so } C_2 + 0 + I = 14$$

$$C_3 \neq 1 \quad M + M = N + 10 C_2$$

(vii) Let  $C_2 = 0$

$$0 + I = 14, M + M = N \text{ so } M + M < 10 \\ N \neq 8$$

Note For  $N=6, M=3$  [only possible]

$N=4, M=2$  [Not possible as  $E=4 \neq E+N$ ]

$N=2, M=1$  [Not possible as  $S=1$ ]

So  $N=6, M=3$

(viii)  $O+J=14$

$$7+7=14 \quad P \quad [\text{as } O \neq J]$$

$$8+6=14 \quad P \quad [\text{as } T=8 \neq N=6]$$

$$9+5=14 \quad V \quad P$$

So either  $O=9$  &  $I=5$  or  $J=5$  &  $O=9$

SOME	1934	1534
<u>+ TIME</u>	<u>or</u>	<u>+ 8934</u>
<u>SPENT</u>	<u>10468</u>	<u>10468</u>

(ii) CROSS                    9 6 2 3 3

$$\begin{array}{r} + ROADS \\ \hline DANGER \end{array} \Rightarrow \begin{array}{r} 6 2 5 1 3 \\ 1 5 8 7 4 6 \end{array}$$

(iii) GIVE                    1407  
+ THEM                      2379  
HELP                      3786

## Means End Analysis Algorithm

① Compare current to goal, if there are no differences then return

② Otherwise select the most important difference of day to reduce it by doing the following until success or failure is signalled.

(i) Select an ~~wanted~~<sup>un-needed</sup> operator that is applicable to current diff. if there are no operator then signal failure

(ii) Attempt to apply operator  $O$  to current. Generate description of two states  $O$ -start (a state in which  $O$ 's precondition are satisfied) &  $O$ -result (the state that will result if  $O$  were applied in  $O$ )

(iii) If (First part  $\leftarrow$  MEA (current,  $O$ -start) & (Last part  $\leftarrow$  MEA ( $O$ -result, goal)) are successful then signal success & return result of corresponding converting (first part,  $O$ , last-part)

## Unit - II

### \* Knowledge Based Agents

→ A knowledge based agent can be described at three levels:

(i) Knowledge level

(ii) Logical level (knowledge encoded into sentences)

(iii) Implementation Level (Level that runs on agent architecture,

- physical representation of sentences at logical  
level is contained in this level )

### \* Logic

(i) Syntax : Rules & regulations of guiding a formal way to write a language

(ii) Semantics : Meaning of the formal language

④ → Two types of logic:

(i) Propositional Logic

(ii) Predicate Calculus

### \* Propositional Calculus logic

→ It is two valued logic (True/False, 0/1, Yes/No)

→ Tautology  $\Rightarrow$  when all conditions are true in truth table

→ Contradiction  $\Rightarrow$  when all conditions are false in truth table

or  
Absurdity

→ Contingency  $\Rightarrow$  when both true & false conditions in truth table.

→ Propositional variables :- p, q, etc.

p . q	p $\wedge$ q
T T	T
T F	F
F T	F
F F	F

→ Two propositional constant :- T/F or 0/1

→ Proposition two types :- (i) Atomic : Ex:- p  $\wedge$  q

(ii) Composite : Ex :-  $(p \wedge q) \Rightarrow (q \wedge r)$

- Connectives: (i) AND ( $\wedge$ ) / conjunction  
(ii) OR ( $\vee$ ) / disjunction  
(iii) NOT ( $\neg$  or  $\sim$ ) / negation  
(iv) If... Then ( $\rightarrow$  or  $\Rightarrow$ ) / Implication  
(v) If & Only If (iff) ( $\Leftrightarrow$  or  $\leftrightarrow$ )

$$\begin{aligned}
&= p \wedge q = \neg(\neg p \vee \neg q) \\
&= \text{NAND}(\wedge) \\
&= \text{NOR}(\downarrow) \\
&= p \vee \neg q = \neg(p \wedge q)
\end{aligned}$$

$\Leftrightarrow$	p	q	$p \rightarrow q$	$p \leftrightarrow q$
	0	0	1	1
	0	1	1	0
	1	0	0	0
	1	1	1	1

Ex:- Let p: Earth is flat

NOTE:

$$p \rightarrow q \equiv \neg p \vee q$$

q: All birds sing

$$p \leftrightarrow q \equiv p \odot q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$$

r: Manhattan is an island

(i)  $\neg q \wedge r$ : Some birds don't sing and Manhattan is an island

(ii)  $\neg(q \wedge p)$ : It is not the case that all birds sing & that earth is flat

(iii)  $r \vee (p \wedge q)$ : Either Manhattan is an island or it is the case that earth is flat & all birds sing

(iv)  $p \wedge \neg(q \vee r)$ : Earth is flat and it is not the case that all birds sing <sup>or that</sup> Manhattan is an island.

→ Principle of duality:

Two formulas  $A_1$  &  $A_2$  are said to be duals of each other if either one can be obtained from other by replacing  $\wedge$  by  $\vee$  &  $\vee$  by  $\wedge$ .

Ex:- If any formula is valid then dual is also valid

Ex:-  $(\wedge, \vee)$  or  $(\uparrow, \downarrow)$  are duals to each other

→ Logical Equivalence:

If two proposition  $P(p, q, \dots)$  &  $Q(p, q, \dots)$  where  $p, q$  are propositional variables have same truth table in every possible case then the propositions are logically equivalent

$$P \equiv Q$$

→ Converse, Contrapositive & Inverse

If  $P$  &  $Q$  are propositions then, the

Converse of  $P \Rightarrow Q$  is  $Q \Rightarrow P$

Contrapositive of  $P \Rightarrow Q$  is  $\sim Q \Rightarrow \sim P$

Inverse of  $P \Rightarrow Q$  is  $\sim P \Rightarrow \sim Q$

Ex:- Indian team wins whenever match is played in Kolkata, home town of Ganguly

Let  $P$ : Indian team wins

$Q$ : Match in Kolkata, home town of Ganguly

Given:  $\therefore Q \Rightarrow P$

Converse:  $P \Rightarrow Q$

If Indian team wins then match is in Kolkata, home town of Ganguly

Contrapositive:  $\sim P \Rightarrow \sim Q$

If Indian team doesn't win then match is not in Kolkata, home town of Ganguly

Inverse:  $\sim Q \Rightarrow \sim P$       doesn't win

Indian team loses whenever match is not played in Kolkata, home town of Ganguly

→ Laws of Algebra of Propositions

(i) Idempotent Law :      (a)  $P \vee P \equiv P$       (b)  $Q \wedge Q \equiv Q$

(ii) Commutative Law :      (a)  $P \vee Q \equiv Q \vee P$       (b)  $P \wedge Q \equiv Q \wedge P$

(iii) Associative Law :      (a)  $P \vee (Q \vee R) = (P \vee Q) \vee R$

(b)  $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

(iv) Distributive Law :      (a)  $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$

(b)  $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$

(v) Identity Law :      (a)  $P \vee \text{True} = \text{True}$

(b)  $P \wedge \text{False} = P$

(c)  $P \wedge \text{True} = P$

(d)  $P \wedge \text{False} = \text{False}$

(vi) Complement Law :      (a)  $P \vee \sim P = \text{True}$       (b)  $P \wedge \sim P = \text{False}$

(vii) Involution law :  $\textcircled{a} \sim \sim p = p$

(viii) De Morgan's law :  $\textcircled{a} \sim (p \vee q) = \sim p \wedge \sim q$

$$\textcircled{b} \sim (p \wedge q) = \sim p \vee \sim q$$

(ix) Absorption law :  $\textcircled{a} p \vee (p \wedge q) = p$      $\textcircled{b} p \wedge (p \vee q) = p$

(x) Contrapositive law :  $\textcircled{a} p \rightarrow q = \sim q \rightarrow \sim p$

$$\textcircled{b} p \Rightarrow q \equiv (\sim p \vee q)$$

$$\textcircled{c} (p \Rightarrow q) \wedge (p \Rightarrow \sim q) \Rightarrow \sim p$$

$$\textcircled{d} p \Leftrightarrow q = (p \wedge q) \vee (\sim p \wedge \sim q)$$

### → Well-Formed Formula (WFF)

⇒ If is defined as follows:

(i) If P is propositional variable then it is wff

(ii) If  $\alpha$  is a wff e, then  $\sim \alpha$  is wff

(iii) If  $x$  &  $y$  are wff, then  $x \vee y$ ,  $x \wedge y$ ,  $x \Rightarrow y$ ,  $x \Leftrightarrow y$  are wff

(iv) ~~and~~ If any of the above three rules result in any string of symbols then it is a wff.

### → Normal Forms :

(i) DNF (Disjunctive Normal form) (Sum of products) (~~Disjunction of min terms~~)

ex:-	p	q	r	f(p,q,r)	
	0	0	0	1	$\sim p \wedge \sim q \wedge \sim r$
	0	0	1	0	
	0	1	0	1	$\sim p \wedge q \wedge \sim r$
	0	1	1	0	
	1	0	0	1	$p \wedge \sim q \wedge \sim r$
	1	0	1	0	
	1	1	0	0	
	1	1	1	1	$p \wedge q \wedge r$

$$\text{So DNF} = (\sim p \wedge \sim q \wedge \sim r) \vee (\sim p \wedge q \wedge \sim r) \vee (p \wedge \sim q \wedge \sim r) \vee (p \wedge q \wedge r)$$

(ii) CNF (Conjunctive Normal form) (Product of sums)

ex:-	(i)	P	q	r	f(p, q, r)
		0	0	0	1 <del>(p, q, r)</del>
		0	0	1	0 <del>(p, q, r)</del> (n p v n q, v r)
		0	1	0	1 <del>(p, q)</del>
		0	1	1	0 <del>(p, q, r)</del> (n p v q, v r)
		1	0	0	1      (p v n q, v r)
		1	0	1	0' <del>(p, q, r)</del>
		1	1	0	0      (p v q, v n r)
		1	1	1	1

$CNF = \neg(p \wedge q \wedge r) \wedge (\neg p \vee \neg q \vee r) \wedge (p \vee \neg q \vee r) \wedge (p \vee q \vee \neg r)$

(ii)  $p \wedge (p \Rightarrow q)$

$$\Rightarrow p \wedge (\neg p \vee q) \quad (CNF)$$

~~$\Rightarrow (p \wedge \neg p) \vee (p \wedge q)$~~

~~$\Rightarrow p \wedge q$~~

(iii)  $(q \vee (p \wedge r)) \wedge \neg((p \vee r) \wedge q)$

$$= (q \vee (p \wedge r)) \wedge (\neg(p \vee r) \wedge \neg q)$$

$$= (q \vee (p \wedge r)) \wedge ((\neg p \wedge \neg r) \vee \neg q)$$

$$= (q \vee p) \wedge (q \vee r) \wedge (\neg p \wedge \neg q) \wedge (\neg q \wedge \neg r)$$

(iii) Principal Disjunctive Normal Form (PDNF)

- Sum of product Canonical form (say three variables are present then all products should contain all three)

Truth table: for every truth table value 1, select minterm

min terms are:  $p \wedge q \wedge r, p \wedge \neg q \wedge r, \neg p \wedge q \wedge r$  of  $p \wedge q$

ex:-  $q \vee (p \vee \neg q)$

p	q	$\neg q$	$p \vee \neg q$	$q \vee (p \vee \neg q)$	
0	0	1	1	1	$\neg p \wedge \neg q$
0	1	0	0	1	$\neg p \wedge q$
1	0	1	1	1	$p \wedge \neg q$
1	1	0	1	1	$p \wedge q$

so PDNF:  $(\neg p \wedge \neg q) \vee (\neg p \wedge q) \vee (p \wedge \neg q) \vee (p \wedge q)$

#### (iv) Principal Conjunctive Normal Form

- Product of sum canonical form (max terms only)

Ex:-  $p \wedge q$

$P$	$q$	$p \wedge q$	
0	0	0	
0	1	0	
1	0	0	
1	1	1	

max terms

$\neg p \wedge \neg q$

$\neg p \wedge q$

$p \wedge \neg q$

So PCNF :  $(\neg p \wedge \neg q) \wedge (\neg p \wedge q) \wedge (p \wedge \neg q)$

#### → Rules of Inference

Rule	Tautological Form	Name
① $\frac{}{p}$	$p \Rightarrow (p \vee q)$	Addition
② $\frac{p \wedge q}{\therefore p}$	$(p \wedge q) \Rightarrow p$	Simplification
③ $\frac{\frac{p}{q}}{p \wedge q}$	$((p) \wedge (q)) \Rightarrow (p \wedge q)$	Conjunction
④ $\frac{\frac{p \Rightarrow q}{p}}{\therefore q}$	$[(p \Rightarrow q) \wedge \neg p] \Rightarrow (q)$	Modus ponens
⑤ $\frac{\frac{p \Rightarrow q}{\neg q}}{\therefore \neg p}$	$[(p \Rightarrow q) \wedge \neg q] \Rightarrow (\neg p)$	Modus tollens
⑥ $\frac{\frac{p \Rightarrow q}{q \Rightarrow r}}{\therefore p \Rightarrow r}$	$[(p \Rightarrow q) \wedge (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$	Hypothetical Syllogism
⑦ $\frac{\frac{p \vee q}{\neg p}}{\therefore q}$	$[(p \vee q) \wedge \neg p] \Rightarrow q$	Disjunction syllogism
⑧ $\frac{\frac{p \Rightarrow q \wedge (r \Rightarrow s)}{\frac{p \vee r}{\therefore q \vee s}}}{\therefore (q \vee s)}$	$[(p \Rightarrow q) \wedge (r \Rightarrow s) \wedge (p \vee r)] \Rightarrow (q \vee s)$	Constructive Dilemma
⑨ $\frac{\frac{p \Rightarrow q \wedge (r \Rightarrow s)}{\frac{\frac{p \vee r}{\frac{p \vee r \wedge (q \vee s)}{\therefore (q \vee r \wedge s)}}}{\therefore (p \vee r \wedge s)}}}{\therefore (p \vee r \wedge s)}$	$[(p \Rightarrow q) \wedge (r \Rightarrow s) \wedge (p \vee r \wedge s)] \Rightarrow (p \vee r \wedge s)$	Destructive Dilemma

Ex:- If there was a ball game, then travelling was difficult.  
 If they arrived on time, then travelling was not difficult.  
 They arrived on time. Therefore there was no ball game

p: There was a ball game

q: Travelling was difficult

r: They arrived on time

Steps

Reason

$$1 \rightarrow \text{Given} \quad (i) \quad p \rightarrow q$$

$$(ii) \quad r \rightarrow \neg q$$

$$(iii) \quad r$$

$$(iv) \quad \therefore \neg p$$

Proof:

Steps

Reason

$$1. \quad r \rightarrow \neg q \quad (ii)$$

$$2. \quad q \rightarrow \neg r \quad \text{Q} \text{ Step 1 \& contrapositive}$$

$$3. \quad p \rightarrow q$$

(i)

$$4. \quad p \rightarrow \neg r$$

Step 2, 3 \& hypothetical syllogism  
law of

$$5. \quad r$$

(iii)

$$6. \quad \neg(\neg r)$$

Step 5 \& double negation

$$7. \quad \therefore \neg p$$

Step 4 \& 6 \& law of non-contradiction

## \* Predicate Calculus

### → Quantifier

→ Two types : (i) Universal quantifier ( $\forall$ )

means "for all values" ex:-  $\forall x \in A : P(x)$

(ii) Existential quantifier ( $\exists$ )

means "there exists" or "for some" ex:-  $\exists x \in A : P(x)$

Ex:-  $p(x)$ :  $x$  is a student

$q(x)$ :  $x$  is clever

$r(x)$ :  $x$  is successful

(i) There exists a student

$\Rightarrow (\exists x) \in S(p(x))$

(ii) Some students are clever

$$\therefore \exists x (P(x) \wedge Q(x))$$

(iii) Some students are not successful

$$\Rightarrow \exists x (P(x) \wedge \neg R(x))$$

ex :- (i) Some people have six fingers on one hand

let  $P(x)$ :  $x$  is people

$Q(x)$ :  $x$  is people having six fingers on one hand

$$\exists x (P(x) \rightarrow Q(x))$$

(ii) all cows are not white

let  $P(x)$ :  $x$  is cow

$Q(x)$ :  $x$  is white

$$\forall x (P(x) \rightarrow \neg Q(x))$$

### → Rules of Inference for Quantified Statements

#### (i) Universal Instantiation

$$\forall x P(x) \quad \text{ex:- Every man is mortal, Ram is a man}$$

$P(c)$  for all  $c$

Ram is mortal

#### (ii) Universal Generalization

$P(c)$  for all  $c$

$$\forall x P(x)$$

#### (iii) Existential Instantiation

$$\exists x P(x)$$

$P(c)$  for some  $c$

#### (iv) Existential Generalization

$\exists P(c)$  for some  $c$

$$\exists x P(x)$$

### → Universal Modus Ponens

$\forall x, \text{ if } P(x) \text{ then } Q(x)$

$P(a)$  for a particular  $a$

$\therefore Q(a)$

concept of lifting

$\rightarrow$  Universal Modus Tollens

$\forall x \text{ if } P(x) \text{ then } Q(x)$

$\sim Q(a) \text{ for a particular } a$   
 $\therefore \sim P(a)$

Concept of Lifting

Ex:- Check validity:

All graduates are educated

Ram is a graduate,

Therefore, Ram is educated.

Let  $A(x)$ :  $x$  is a graduate

$B(x)$ :  $x$  is educated

Let  $\neg R$  denote  $R$

(i)  $\forall x (A(x) \Rightarrow B(x))$

(ii)  $A(R)$

$\therefore B(R)$

By universal instantiation  $\forall x (A(x) \Rightarrow B(x))$  can be written

as  $A(R) \Rightarrow B(R)$  &  $A(R)$  gives  $B(R)$  by modus ponens

∴ Conclusion is valid.

\* Skolemization Process (Part of FOL)

Rule 1: If left most quantifier is an existential quantifier, replace all occurrences of the variable it quantifies with an arbitrary constant & delete the quantifier.

Rule 2: For each existential quantifier, which is preceded by one or more universal quantifiers, replace all occurrences of quantified variable by a func<sup>n</sup> not appearing anywhere else.

Ex:- Given  $\exists u \forall v \forall n \exists y P(f(u), v, n, y) \vee Q(y, v, y)$

Applying Skolemization Process,

( $\because$  Rule 1)  $\forall v \forall n \exists y P(f(a), v, n, y) \vee Q(a, v, y)$

( $\because$  Rule 2)  $\forall v \forall n P(f(a), v, n, g(v, n)) \vee Q(a, v, g(v, n))$

Result

## \* First Order Predicate Logic (FOL)

- Predicate is defined as reln that binds two atoms together
- Quantifiers is a symbol that permits one to declare or identify range or scope of variables in a logical expression.

Two types :  $\forall$  ,  $\exists$

- Constants & variable <sup>functional</sup> expressions are referred to as term.
- A term is a logical expression which refers to an object.
- First order logic is a expressive language which has well-defined syntax & semantics. It use three things to represent the world model: (i) Objects (ii) Relations (iii) Functions

ex:- Four plus Five equals Nine

Object - Four, Five, Nine

Relation - Equals

Function - Plus

ex:- Translate sentences into FOL

(i) Lipton is a tea

∴  $\exists x : \text{tea}(Lipton)$

(ii) Ruma dislikes children who drinks tea

∴  $\forall x : \text{child}(x) \wedge \text{drank\_tea}(x) \Rightarrow \text{dislikes}(\text{Ruma}, x)$

(iii) All purple mushrooms are poisonous

$\forall x : \text{Mushroom}(x) \wedge \text{purple}(x) \Rightarrow \text{poisonous}(x)$

→ Semantics Inference Rules

(i) Universal Instantiation

$\forall v P$

$\text{SUBST}([v/g], P)$

ex:-  $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

$x \rightarrow \text{Aman, Raman, Queen}$

or <sup>two</sup> sentences obtained  
 $\text{King}(\text{Aman}) \wedge \text{Greedy}(\text{Aman}) \Rightarrow \text{Evil}(\text{Aman})$   
 $\text{King}(\text{Raman}) \wedge \text{Greedy}(\text{Raman}) \Rightarrow \text{Evil}(\text{Raman})$

(ii) Existential Instantiation

$\exists v P$

$\text{SUBST}([v/k], P)$

ex:-  $\exists x \text{ Crown}(x) \wedge \text{OnHead}(x, \text{Rom})$

∴  $\text{Crown}(\text{H}) \wedge \text{OnHead}(\text{H}, \text{Rom})$

## → Concept of Lifting

### Generalized Modus Ponens

$P(a)$	Assertion
$\forall x P(x) \rightarrow Q(x)$	Implication
$Q(a)$	Conclusion

## → Unification

Process of finding substitutions for lifted inference rules, which can make different logical expressions to look identical

$$\text{Unify } (P, Q) = \theta \quad \xrightarrow{\text{Identifier}}$$

$$\text{SUBST}(\theta, P) = \text{SUBST}(\theta, Q)$$

ex:— we have a query:

whom does Ram know? , i.e., knows(Ram, u)

knowledge base : knows(Ram, Sita)

knows(y, Shyam)

knows(y, Father(y))

knows(x, Radha)

$$\text{Unify}(\text{knows}(\text{Ram}, x), \text{knows}(\text{Ram}, \text{Sita})) = \{x / \text{Sita}\}$$

$$\text{Unify}(\text{knows}(\text{Ram}, x), \text{knows}(y, \text{Shyam})) = \{x / \text{Shyam}, y / \text{Ram}\}$$

$$\text{Unify}(\text{knows}(\text{Ram}, x), \text{knows}(y, \text{Father}(y))) = \{y / \text{Ram}, x / \text{Father}(\text{Ram})\}$$

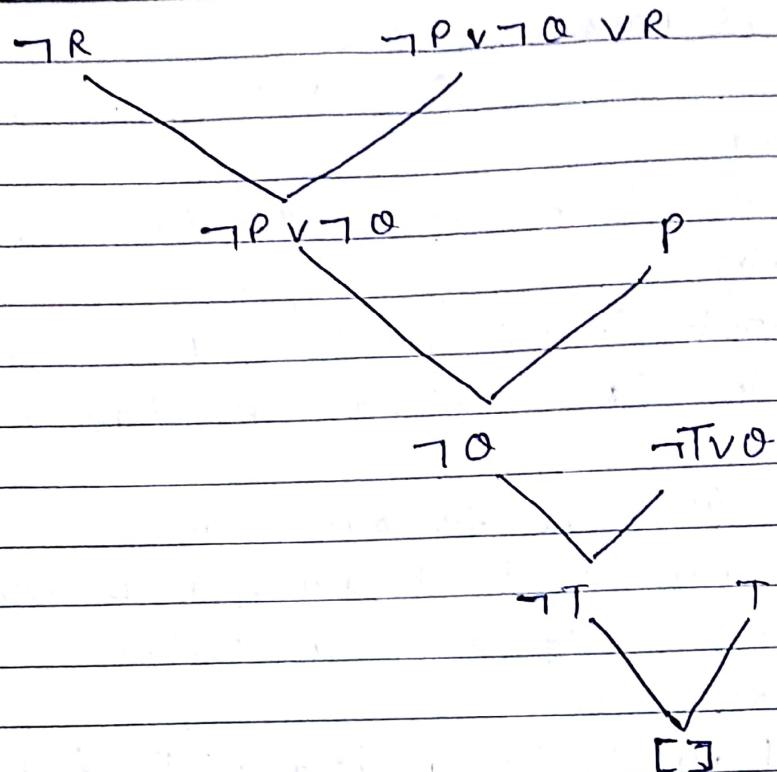
$$\text{Unify}(\text{knows}(\text{Ram}, u), \text{knows}(x, \text{Radha})) = \text{Fail}$$

## → Resolution.

Any two clauses A & B if there is a literal  $P_1$  in A which has a complementary  $P_2$  in B, delete  $P_1$  &  $P_2$  from A & B and construct a disjunction of the remaining clauses. The clause so constructed is called resolvent of A & B.

ex:— prove R is true by resolution

(i)	$P$	$\vdash$ (causes form)	
(ii)	$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q \vee R$	$P_1$
(iii)	$(S \vee T) \rightarrow Q$	$\neg S \vee \neg T \vee Q$	$P_2$
(iv)	T	T	Either one of them will work



∴ we found contradiction  $\therefore \neg R$  is true

### \* Forward and Backward Chaining

- There are two kinds of rule system: forward chaining system & backward chaining systems
- In forward chaining systems, we start with initial facts, & keep using the rules to draw new intermediate conclusions (or take certain actions) given those facts. The process terminates when the final conclusion is established. (hypothesis)
- In backward chaining system, we start with some goal statements, which are intended to be established and keep looking for rules that would allow us to conclude, setting new sub-goals in the process of reaching the ultimate goal. In the next round, the sub goals become the new goals to be established. The process terminates when in this process all the sub goals are given facts.
- Forward chaining systems are primarily data driven while backward chaining systems are goal driven.

## → Forward vs Backward Chaining

\* (i) If there is more fan-out than fan-in, use backward chaining.

(ii) If there is more fan-in than fan-out, use forward chaining.

[ Fan-out (Rules such that typical set of facts leads to many conclusions)

    Fan-in (Rules such that typical hypothesis leads to many questions)

- If neither fan-out nor fan-in dominates then :

(i) If have not yet gathered any facts & interested in only whether one of many possible conclusions is true, use backward chaining.

(ii) If B already have in hand all the facts going to get & wants to know everything that can be concluded from facts, use forward chaining.

## Unit - III

### Game Playing

#### \* Introduction

John von Neumann (Father of Game Theory) → Game Theory → 1928 & 1937

#### \* Adversarial Search

- In such applications where multiple persons are searching for sol<sup>n</sup> in the same sol<sup>n</sup> space & their goals are contrary to each other. Usually in competitive environments, in which the goals are in conflict, give rise to adversarial search problems.

- A mixture of reasoning & creativity, the player has to keep in mind the impact of his next move on the overall configuration of the game

#### \* Definition of a Game

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

- (i) Initial State      (ii) A set of operators (Legal moves)
- (iii) A Terminal Test      (iv) Payoff function (Utility Function)

\* One player's turn is called ply. A round of all players' turns is called move. A list of optimal choices is called strategy.

#### \* Types of Games:

① Perfect vs Imperfect Information

② Deterministic vs (Change in state fully determined by player move vs

Stochastic (Change in state partially determined by player move)

③ General Zero Sum Game (Net change in wealth or benefit is zero)

④ Constant sum Game (Algebraic sum of outcomes is always constant)

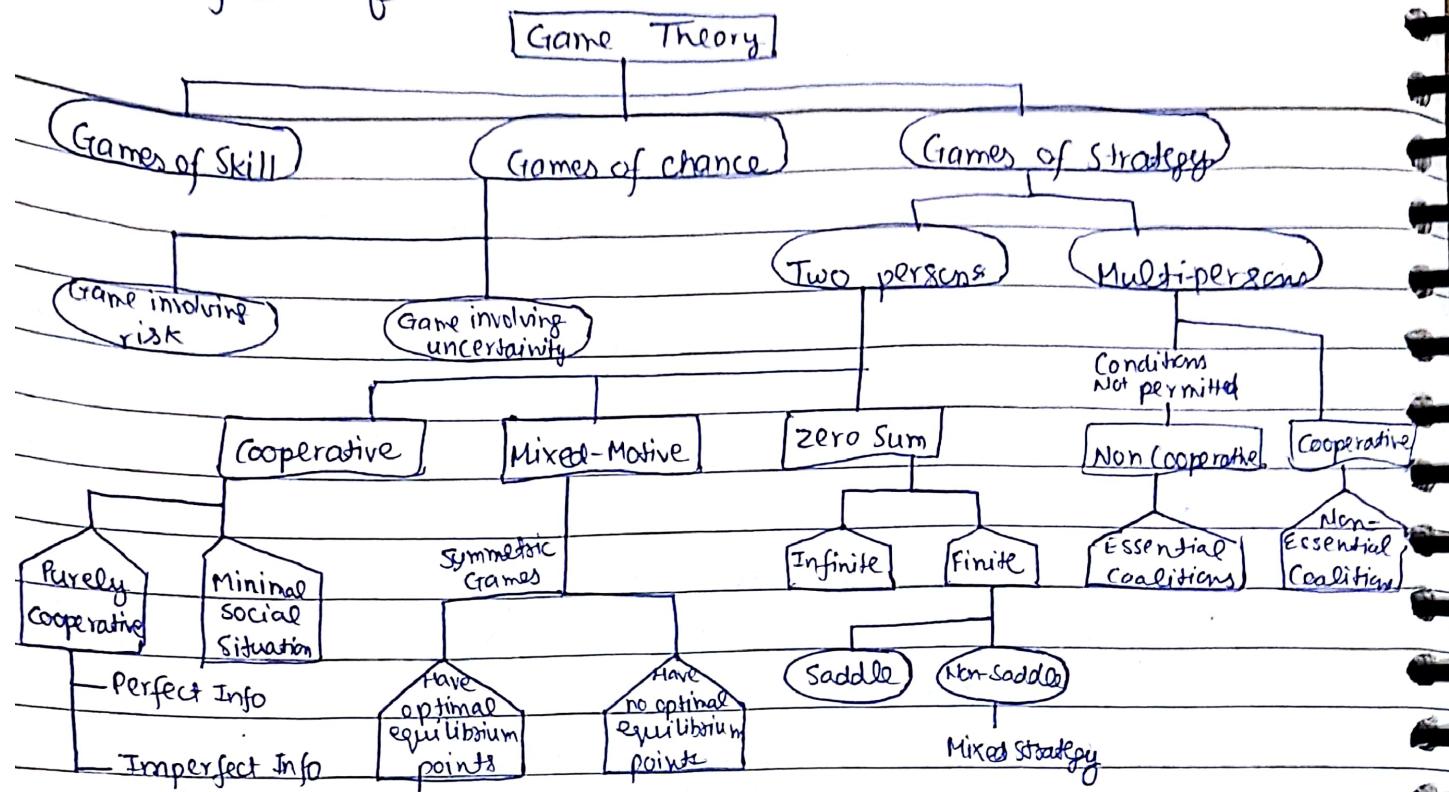
⑤ Non-zero sum Game (Some combinations of Actions provide a net gain (treasure) or loss (-ve sum), to two of them)

⑥ N-person game (Involves more than two players in which coalitions may form & change a game radically)

#### \* Characteristics of Game Playing:

- 1) There is always "Unpredictable" opponent
- 2) Time Limited

\* Classification of Games:



\* In game theory, a game tree is a directed graph whose nodes are positions in a game & whose edges are moves. The complete game tree for a game is the game tree starting containing all possible moves from each position & starting at initial position.

\* Three basic operations for game playing :

(i) Generation : Breadth First & Depth First Searches

(ii) Evaluation : - ~~Legal move~~ Legal move & plausible move  
(Superset of all possible moves) (Result in good score for player)

- SEF (Static Evaluation Function)

(iii) Pruning

\* Min max Algorithm

- A way of finding an optimal move in a two player game

- Player moves : Max nodes represented as  $\square$  or  $\Delta$

Opponent moves : Min nodes represented as  $\circ$  or  $\nabla$

- Assume max player makes first move & then proceed.

- Ex:- Tic tac toe game

Initial State :

X	0	0
0		X
X		

} After making game tree

assign values 0 & 1 at leafs  
for loss & gain

} for max mode select

max of  
for min mode select min

1.) Max move

X	O	O
O	X	
X		

best way

X → Player move  
O → Opponent move

2.) Min move

Min value: 0

(1)

Min value: 1

3.) Max move

(1)

Max value: -1

- Limitations:

(i) Effectiveness limited by depth of game tree

(SEF)

(ii) A rapid tie b/w assigned & actual value to static evaluation func' fails to decide expansion of game tree

(iii) Horizon effect: No guarantee, score is optimal

### \* Alpha-Beta Pruning

- Problem with MinMax Algorithm is that no. of game states it has to examine is exponential in no. of moves.

-  $\alpha\beta$  pruning is a search algorithm that seeks to decrease no. of nodes evaluated by minmax algorithm in its search tree.

- It is an adversarial search algorithm.

-  $\beta$ : upper minimum upper bound of possible sol'n

$\alpha$ : maximum lower bound of possible sol'n

-  $\alpha$  or  $\beta$  pruning if  $\alpha \geq \beta$  or new node considered if  $\alpha \leq N \leq \beta$

current value ↑

-  $\alpha, \beta$  values can go from parent to child but not vice versa

- Rules for updating  $\alpha, \beta$  values

→ if min node: if current  $\beta >$  current value node  
then  $\beta = \text{current value node}$

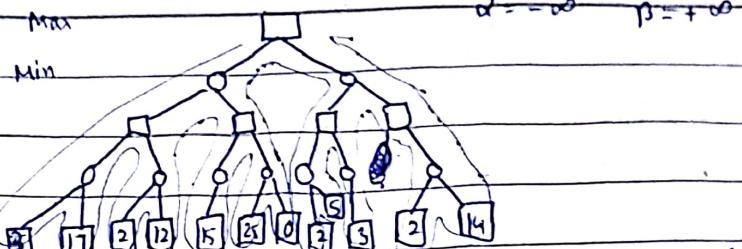
→ if max node: if current  $\alpha <$  current value node  
then  $\alpha = \text{current value node}$

- Initially assume for all nodes  $\alpha = -\infty$  &  $\beta = \infty$

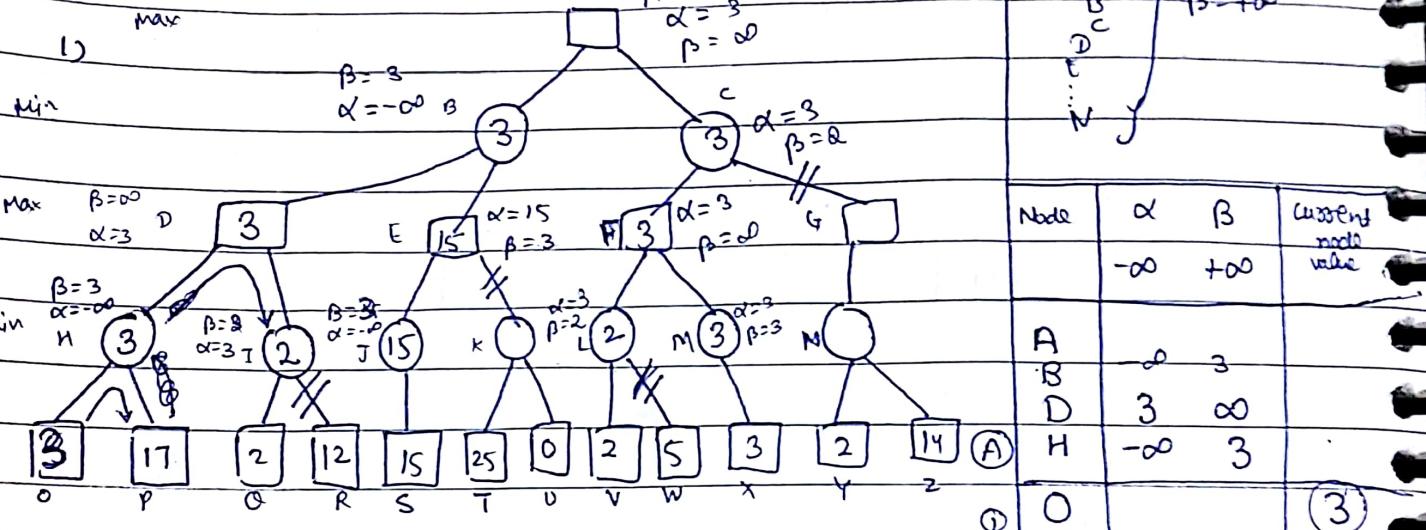
In this method for node value use minmax

i.e., min algo  
then min value  
from child  
& vice-versa

Ex:-



$$\alpha = -\infty \quad \beta = +\infty$$



A, B, C, D, E, F, G, H, I, J, K, L, M, N

Node	$\alpha$	$\beta$	Current node value
A	-infinity	+infinity	
B	-infinity	3	
D	3	+infinity	
H	-infinity	3	
I	0	3	(3)
P	0	17	
J	3	+infinity	
E	-infinity	3	
K	0	2	(2)
L	2	2	
M	3	3	(3)
N	3	3	
O	0	15	
R	0	12	
S	0	17	
T	0	25	
V	0	0	
W	0	2	
X	0	5	
Y	0	3	
Z	0	2	
A	0	14	

Ignored as  $17 > 3$

Updated D (Maxnode)

parent value to  
child value

Now branching to I

~~00~~  $\alpha = -\infty$   $\beta = +\infty$

A  $\alpha = -\infty$   $\beta = +\infty$

B  $\alpha = -\infty$   $\beta = 3$

C  $\alpha = -\infty$   $\beta = +\infty$

D  $\alpha = 3$   $\beta = +\infty$

E  $\alpha = -\infty$   $\beta = +\infty$

F  $\alpha = -\infty$   $\beta = +\infty$

G  $\alpha = -\infty$   $\beta = +\infty$

H  $\alpha = -\infty$   $\beta = 3$

I  $\alpha = 3$   $\beta = +\infty$

J  $\alpha = -\infty$   $\beta = 15$

K  $\alpha = -\infty$   $\beta = +\infty$

L  $\alpha = -\infty$   $\beta = +\infty$

M  $\alpha = -\infty$   $\beta = +\infty$

N  $\alpha = -\infty$   $\beta = +\infty$

Updated as  $\alpha > 2$

as  $\alpha > \beta$   
so  $\beta$  pruning &  
updating B

Now branching to E

E  $\alpha = -\infty$   $\beta = 3$

J  $\alpha = -\infty$   $\beta = 15$

I  $\alpha = 15$   $\beta = +\infty$

E  $\alpha = 15$   $\beta = 3$

Ignored  $15 > 3$  so  
no  $\alpha, \beta$  updation

no evaluating G  
 $\alpha = 15$

as  $\alpha > \beta$

so  $\alpha$  pruning & updating  $\beta = 3$

now A  $\alpha = 3$  &  $\beta = +\infty$

now traversing C & its child

	$\alpha$	$\beta$	
A	3	$\infty$	
C	3	$\infty$	
F	3	$\infty$	
L	3	$\delta$	
V			(2) $y & < \infty$ so L updated
$\rightarrow$ L	3	2	$\alpha > \beta$ at L
$\Rightarrow$			3 as <del>500</del> so ignored $\beta$ paunning
M	3	$\infty$	
X			3 so updating M
M	3	$\infty$	
B			

to update F choose max among L & M

so f then update C,  $\infty$

	$\alpha$	$\beta$	value
C	$\infty$	3	$\infty$
F			3

& C is min node &  $3 < \infty$  then  $\beta$  updated

	$\alpha$	$\beta$
C	3	$\infty$

as  $3 > \infty$  & C is min node so  $\beta$  paunning  
then updating C = 3

### \* The Horizon Effect

- The major weakness of computer players for turn based games is horizon effect. The horizon effect occurs when a fixed sequence of moves ends up with what appears to be an excellent position but one additional move will show that that position is, in fact, very bad.
- A negative horizon is where state seen by evaluating func<sup>n</sup> is evaluated as better than it really is because an undesirable effect is just beyond this node (i.e., search horizon)
- A positive horizon is where evaluation function wrongly underestimates the value of a state where positive actions just over the search horizon indicate the otherwise.

### NLP (Natural Language Processing).

\* AI method of communicating with an intelligent system using a natural language such as English.

\* Processing of natural language is required when you want an intelligent system like robot to perform as per your instruction or when you want to hear decisions from a dialogue based clinical expert systems etc.

\* Field of NLP involves making computers to perform useful tasks with the natural languages human use. I/P & O/P of an NLP system can be speech or written text.

### \* Component of NLP

#### NL Understanding (NLU)

- mapping the given input in natural language into useful representations
- Analyzing different aspects of the language

#### NL Generation (NLG)

- Process of producing some meaningful phrases & sentences in the form of natural language from some internal representation

- It involves:
- ① Text planning (Retaining relevant content from knowledge base)
  - ② Sentence planning (Choosing required words, forming meaningful phrases, setting tone of sentence)
  - ③ Text realization (mapping sentence plan into sentence structure)

- NLU is harder than NLG
- Difficulties in NLU :  $\rightarrow$  NL has extremely rich form & structure  
 $\rightarrow$  NL is very ambiguous
  - syntax level ambiguity
  - lexical ambiguity
  - referential ambiguity

### \* NLP Terminology

- ① phonology - study of organizing sound systematically
- ② morphology - study of constructing words from primitive meaningful units
- ③ syntax ④ semantics ⑤ pragmatics ⑥ Discourse ⑦ word knowledge

### \* Steps in NLP

- Lexical analysis:  $\rightarrow$  Identifying & analyzing structure of word  
 $\rightarrow$  divide whole chunk of text into paragraph, sentences of words

#### - Syntactic Analysis:

- Semantic Analysis: Text is checked for meaningfulness

#### - Discourse Integration

#### - Pragmatic

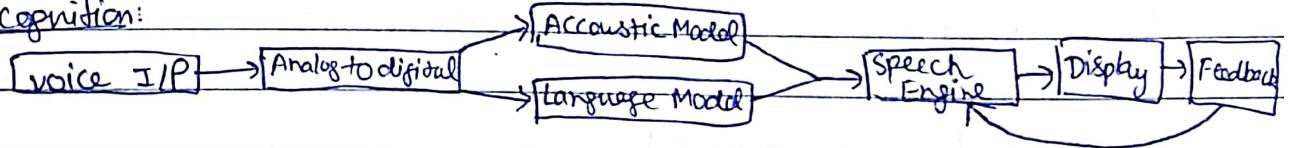
- Implementation of Semantic Analysis  $\leftarrow$  Topdown parse  $\nearrow$  CFG

## Speech Recognition

\* Automatic speech recognition or computer speech  $\Rightarrow$  means understanding voice by the computer & performing any required task

\* Uses: Dictation, system control / navigation, industrial application, voice dialing etc.

### \* Recognition:



\* Acoustic model takes audio recordings & their text transcription & using software to create statistical representation of sound that make each word. It is used by speech recognition system to recognise speech

### \* Language model

- Many NLP applications such as SR (Speech Recognition) tries to capture properties of a language & predict the next word in speech sequence.

- Language model is of two types:
  - (i) Speech Dependent. (Needs a dictation software)
    - Works by learning ~~independent~~ unique characteristics of single person, recognition of his voice. Ex:- Siri, Bixby, Google Assistant
    - Training is required from the specified person to identify the behaviour of his speech
  - (ii) Speech Independent. (For telephonic application)
    - Recognise anyone voice Ex:- AIVRS system
    - Less accurate

### \* Speech Engine

- It uses the ~~output~~ output acoustic of language model to convert speech into text.
- It processes this text to be converted to a computer language.
- Text is now displayed on display screen & feedback is send back to speech engine based on speech to text converter.

## Robotics

\* Robots are artificial agents acting in real world environment. Robots are aimed at manipulating the objects by pertaining, picking, moving the physical properties of object, destroying it or have an effect thereby freeing manpower from being repetitive func<sup>n</sup> without getting ~~bored~~, bored, distracted or exhausted.

### \* Robot Locomotion.

- Locomotion is the mechanism that makes a robot capable of moving in its environment.

- Types of locomotion
  - Legged
  - Wheeled
  - Combination of Legged & Wheeled
  - Tracked slip/skid

### - Legged Locomotion

- Consumes more power while demonstrating walk, jump, trot, hop etc.
- Requires more no. of motors to accomplish a movement.
- Suited for rough as well as smooth terrain
- Little difficult to implement because of stability issues.

- comes with variety of one, two, four or six legs
- with multiple legs of a robot, leg-coordination is necessary for locomotion
- The total no. of possible gates a periodic sequence of lift & release events for each of the total legs a robot can travel depends upon no. of its legs.
- If robot has  $k$  legs, no. of possible events is  $N = (2k-1)!$
- Ex:- for  $k=3$ ,  $N = 2(4-1)! = 3! = 6$

→ Complexity of robots  $\propto$  no. of legs

#### - Wheeled locomotion

→ requires power no. of motors to accomplish a movement.

→ little easy to implement as there are less stability issue.

→ power efficient as compared to legged locomotion.

- standard wheel - castor wheel (rotate around axle) - Balls (spherical wheels)

- swedish  $45^\circ$  &  $90^\circ$  wheels (rotate around axle, contact point & around rollers)

#### - Slip/Skid locomotion → offer stability

→ In this type, the ~~wheels~~ vehicles use tracks as in a tank. The robot is stored by mounting the tracks with different speeds in same or opposite direction.

#### \* Component of a Robot:

- 1) Power supply
- 2) Actuators (convert energy into movement)
- 3) Electronics motors AC/DC
- 4) Pneumatic Air Muscles
- 5) Muscle wires
- 6) Piezo motors & ultrasonic motors

#### \* Computer vision

- Technology of AI with which the robot can see.
- Computer vision automatically extracts, analyse & comprehends useful information from a single image or array of images.
- This process involves development of algorithms to accomplish automatic visual comprehension.

#### - Hardware of a computer vision

→ Requires a device such as camera, a processor

→ accessories such as camera, stand, cables & connectors

→ display device for monitoring the system

#### - Tasks of Computer-vision:

→ OCR: Optical Character Reader, a software to convert scanned document into editable text, which accomplishes a scanner

→ Face-Detection: Read the face & take picture of that person's expression.  
Used to detect a user across the software on correct match.

→ Object Recognition: Preinstalled in supermarkets, cameras, cars (BMW, Volvo etc.)

→ Estimating Position

- Application Domains of Computer Vision

Agriculture, autonomous vehicles, biometrics, character recognition, face recognition, process control etc.

\* Application of Robotics:

① Industries: Robots used for handling materials, cutting, welding etc

② Military: Autonomous robot can reach inaccessible & hazardous areas during war

Ex: Robot named 'dakship' developed by DRDO function to destroy life-threatening objects safely.

③ Medicine: Robots are capable of carrying out hundreds of clinical test  
: Perform complex surgeries

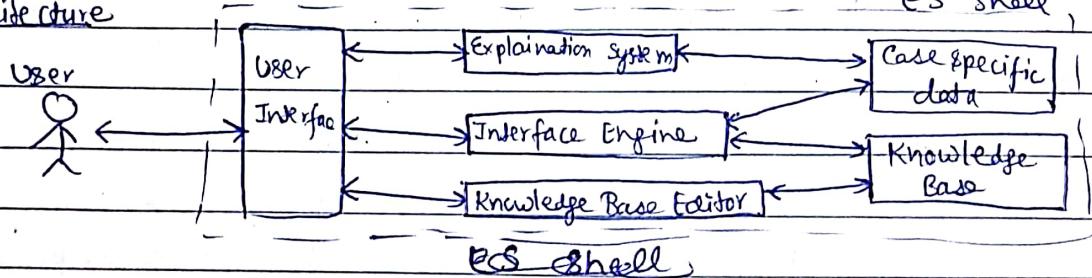
: Rehabilitating permanent disabled people

④ Exploration: Robot rock climbers used for space exploration, underwater drones for ocean exploration.

⑤ Entertainment: Disney engineers have created robots for movie making.

## Expert System (ES) ex- DENDRAL, MYCIN

\* Architecture



- Computer application developed to solve complex programs in a particular domain at the level of extraordinary human <sup>intelligence</sup> expertise

\* Characteristics:

- High performance - Understandable - Reliable - Highly Responsive

\* Capabilities:

(1) Advising (2) Instructing & assisting humans in decision making

(3) Interpreting Inputs (4) Predicting Results (5) Deriving a soln

(6) Demonstrating (7) Explaining (8) Justifying evaluation (9) Suggesting options to a problem

## \* Incapabilities:

- ① Substituting human decision makers
- ② Possessing human capabilities
- ③ Producing accurate ~~optimal~~ output for inadequate knowledge base
- ④ Refining their own knowledge

## \* Components:

### ① Knowledge Base (KB)

- Software that represents knowledge. It contains ~~other~~ specific & high-quality knowledge. Knowledge is required to exhibit intelligence

- What is knowledge:

④ → Data is collection of facts

→ Info is organised as data & facts about the task domain

→ Data, info & past-experience combined together are termed knowledge

- Components of Knowledge Base:

(a) Factual knowledge: Info widely accepted by knowledge engineer & scholars in task domain

(b) Heuristic knowledge: About prediction, accurate judgement & one's ability of evaluation & guessing

- Knowledge representation: organizing & formulating the knowledge

- Knowledge acquisition:

→ automated acquire of knowledge from different experts

→ success of any ES majorly depends on quality, accuracy & completeness of the info stored in knowledge base

### ② Interface Engine (IE)

- Deduct a correct flawless soln by applying efficient procedures & rules

- In case of knowledge base ES, IE acquires & manipulate the knowledge from knowledge base to arrive correct soln

- In case of rule based ES, apply rules repeatedly to the facts. Add new knowledge to <sup>KB</sup> base if required & resolve rule conflicts when multiple rules are applicable to particular case.

- To recommend a soln, interface IE uses following strategies:

① Forward chaining

② Backward chaining

### (3) User Interface

#### \* Limitations:

- Limitations of the technology
- Difficult to maintain

- Different knowledge acquisition
- High development costs

#### \* Applications:

- Design domain: Camera lens design, automobile design
  - Medical domain: diagnosis system
  - Monitoring systems: comparing data, continuously with observed system
  - Process-control system
  - Knowledge domain: finding out faults in vehicles & computers
- \* ES Shell :- shell is ES without knowledge base.
- :- shell provides developers with knowledge base acquisition, interface engine, user-interface, explanation system.

Ex:- Java ES shell

#### \* Development of ES.

##### ① Identify problem domain

- problem must be suitable for an ES to solve it
- find experts in task domain
- Establish cost effectiveness

##### ② Design the system

- Identify ES technology
- know & establish the degree of integration with other systems
- Realize how the concepts can represent domain knowledge best

##### ③ Develop prototype

- knowledge acquisition of these rule

##### ④ Test & refine prototype

##### ⑤ Develop & complete the ES

##### ⑥ Maintain the ES

#### \* Benefits of ES:

##### ① Availability

##### ② less production cost

##### ③ Speed

##### ④ less error rate

##### ⑤ Reducing Risk

##### ⑥ Steady Response

## Theorem Proving

- \* Reasoning by theorem proving is a weak method, compared to expert systems because it does not make use of domain knowledge.
  - \* It requires: a logic (syntax), a set of axioms & inference rules, a strategy on when to search through possible applications of axioms & rules.
  - \* It involves resolution, skolemization, unification.
- AI Techniques - Abstraction & Search Knowledge
- \* Abstraction involves induction of ideas or the synthesis of particular facts into one general theory about something.
  - \* Search knowledge involves search the knowledge using various search algorithm like BFS, DFS etc.

## Unit-IV

### \* Types of knowledge

#### (1) Declarative knowledge

It's about statements of facts that describes a particular object & its attributes including some behaviour in reln with it.

#### (2) Procedural Knowledge

Gives information / knowledge about how to achieve something

#### (3) Inheritable knowledge

#### (4) Inferential knowledge : ~~deduced~~ from information provided

#### (5) Relational knowledge : Describes what relnship exists b/w concepts & objects

#### (6) Heuristic knowledge : Representing knowledge of some expert in a field or subject

#### (7) Common sense

#### (8) Implicit/Tacit Knowledge : Achieved through experience

#### (9) Explicit Knowledge : Knowledge through mediums like organized records etc

#### (10) Uncertain Knowledge

### \* Types of Learning

#### (1) ROTE Learning / Learning By Memorizing

#### (2) Direct Instruction / Learning By Taking Advice

#### (3) Learning By Induction

#### (4) Learning By Deduction

#### (5) Learning By Analog

#### (6) Explanation Based Learning

#### (7) Society Based Learning

### Inductive Learning

\* This type of AI learning model is based on inferring a general rule from datasets of input-output pairs.

\* It involves the process of learning by example & involves classification.

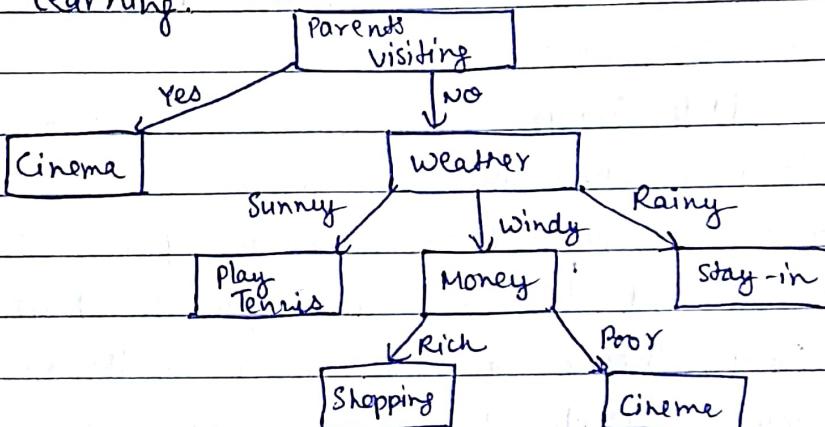
\* A learning system has to be capable of evolving its own class descriptions.

\* Algorithms such as Knowledge Based Inductive Learning (KBIL) are example of this AI learning technique.

## Learning Decision Trees

- \* Decision trees could be used for inductive learning <sup>inference</sup>
- \* Decision tree learning is a method that uses inductive learning to approximate a target funcn, which will produce discrete values.
- \* The learned funcn is represented by a decision tree. A learned decision tree can also be re-represented by if-then rules
- \* Decision tree is one of the most widely used f practical method for inductive inference
- \* Decision trees are robust to noisy data & capable of learning disjunctive expressions.
- \* Decision tree learning method searches a completely expressive hypothesis
- \* Inductive learning involves the process of learning by example - where system tries to induce a general rule from set of observed instances
- \* Inductive learning also involves classification - assigning to a particular input, a class to which it belongs
- \* The task of constructing classes definitions is called induction or concept learning.

\* Ex:-



## Computational Learning Theory

- \* Theoretical results in machine learning mainly deal with type of inductive learning called supervised learning.
- \* In supervised learning, an algorithm is given samples that are labeled in some useful way. The algorithm takes previously labeled samples & uses them to induce a classifier.
- \* This classifier is a funcn that assigns labels to samples including samples that have never been previously seen by the algorithm.

- \* The goal of supervised learning algorithm is to optimize some measure of performance such as minimizing no. of mistakes made on new samples.
- \* In addition to performance bounds, computational learning theory studies time complexity & feasibility of learning. A computation is considered feasible if it can be done in polynomial time.
- \* There are two kinds of time complexity results:
  - (i) Positive Results: Showing that a certain class of funcn is learnable in polynomial time.
  - (ii) Negative Results: Showing that certain classes can't be learned in polynomial time.
- \* Negative results often rely on commonly believed but yet unproven assumptions such as : computational complexity -  $P \neq NP$  (PvNP problem), cryptographic - one way functions exist

### Explanation Based Learning (EBL)

- \* It is a form of machine learning that exploits a very strong, or even perfect, domain theory in order to make generalization or form concepts from training examples.
- \* An example of EBL ~~is~~ using a perfect domain theory is a program to that learns to play chess through example.
- \* A domain theory is perfect or complete if it contains, in principle, all information needed to decide any question about the domain. For example, domain theory for chess is simply the rules of chess.
- \* EBL uses training examples to make searching for deductive consequences of a domain theory efficient in practice.
- \* In essence, an EBL system works by finding a way to deduce each training example from the system's existing database of domain theory. Having a short proof of the training example extends the domain theory database, enabling the ER system to find & classify future examples that are similar to the training example very quickly.
- \* The main drawback of the method - the cost of applying learned macros, as ~~these~~ these become numerous.

- \* EBL software takes four inputs:
  - a hypothesis space
  - training examples
  - a domain theory
  - operability criteria
- \* EBL uses deductive learning technique which starts with the series of rules & infers new rules that are more efficient in context of a specific AI algorithm.

## Applications of AI

### \* Environmental Science

- It is possible to use satellites to detect phytoplankton presence in the ocean due to concentration in the surface. However even using satellites is not an easy task as aerosol concentrations, suspended sediments give rise to complications.
- To overcome those problems, machine learning algorithms such as artificial neural networks & support vector machines are used to automatically classify & detect the presence of phytoplankton in the ocean.
- It is also possible to use remote sensing to classify & detect land cover applying the same algorithms to identify & classify different types of vegetation & including forests, dead trees etc. These are some of samples of AI in environmental science.

### \* Robotics

- The basic contributions of AI in robotics are:
  - Perception: Not only taking data from environment, but transforming it into knowledge to be able to interpret & modify its behaviour according to result of this perception.
  - Reasoning: Drawing conclusion from data / knowledge taken from perception
  - Learning: With new experiences, robot needs to perceive & reasons to obtain conclusions but when experiences are repeated, a learning process is required to store knowledge & speed up process of intelligent response.
  - Decision Making: The ability to prioritize actions, is necessary to be able to be safe & effective in the sol<sup>n</sup> of different autonomous applications.
  - Human-Robot Interaction: At many level necessary. For ex, NLP

## \* Medical Science

- Medical AI mainly uses computer techniques to perform clinical diagnosis & suggest treatments
- AI has the ability capability of detecting meaningful relationships in a data set & has been widely used in many clinical situations to diagnose, treat, & predict the results.
- Some ways in which AI is changing healthcare:
  - i Managing medical records & other data
  - ii Doing repetitive jobs & treatment design
  - iii Digital consultation & virtual nurses
  - iv Medication management & drug creation
  - v Precision medicine & health monitoring
  - vi Healthcare system analysis

## \* Aerospace

- AI powered autopilots are taking increasingly creative forms as they are maturing. AI innovation in aviation has focused on control of airplane & its systems.
- AI & MAS (Multi-Agent Systems) solutions have repeatedly demonstrated more robustness, flexibility & scalability than more traditional topdown approaches.
- various applications in aerospace can be:
  - i knowledge representation, reasoning & logic in aerospace applications
  - ii Agent-based modeling & simulation of socio-technical systems in aerospace
  - iii Planning & scheduling in air transport
  - iv Airport operations, maintenance, swarming of satellites etc.