Unit - I

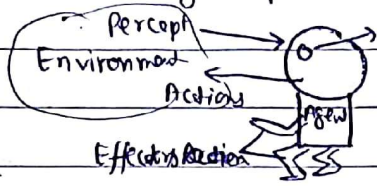
Introduction

* 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	Strong 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	→ Strong AI supposes, that it is possible for machines to become self-aware but may or may not exhibit human thought process.
→ weak AI claims that machines can act intelligently or merely specific problem solver	→ Strong AI claims that machines can act intelligently, has mind and understanding.

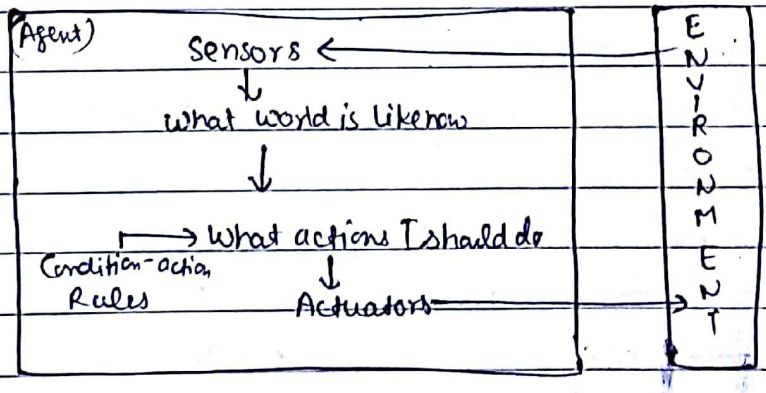
* Four views of AI:
  1.) Thinking humanly
  2.) Acting humanly
  3.) Thinking rationally
  4.) Acting rationally
* Applications of AI:
  1.) Game playing
  2.) Speech and character recognition
  3.) Natural language Processing
  4.) Expert Systems
  5.) ~~Robotics~~ Robotics
  6.) Information Predictors
  7.) Computer vision and pattern recognition
* Challenges before AI:
  1.) Natural language processing
  2.) knowledge Representation
  3.) Automated Reasoning
  4.) Machine learning
* An intelligent agent is a software entity which senses its environment and carries out operations on behalf of user, with some degrees of autonomy and employs some knowledge or representation of user's goals or desires.

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

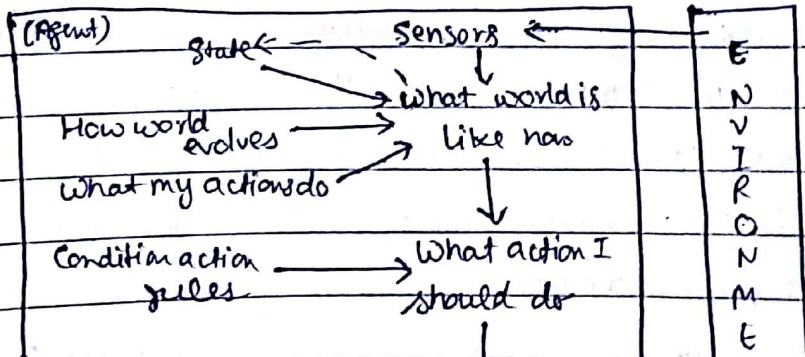


* Agent = architecture + program } Structure of agent
* An environment provides the conditions under which an entity exists (agent)
* Properties of Enviroments:
  (i) Accessible vs Inaccessible
  (ii) Deterministic vs Nondeterministic
  (iii) Episodic vs Nonepisodic (Note: Experience divided into episodes)
  (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.

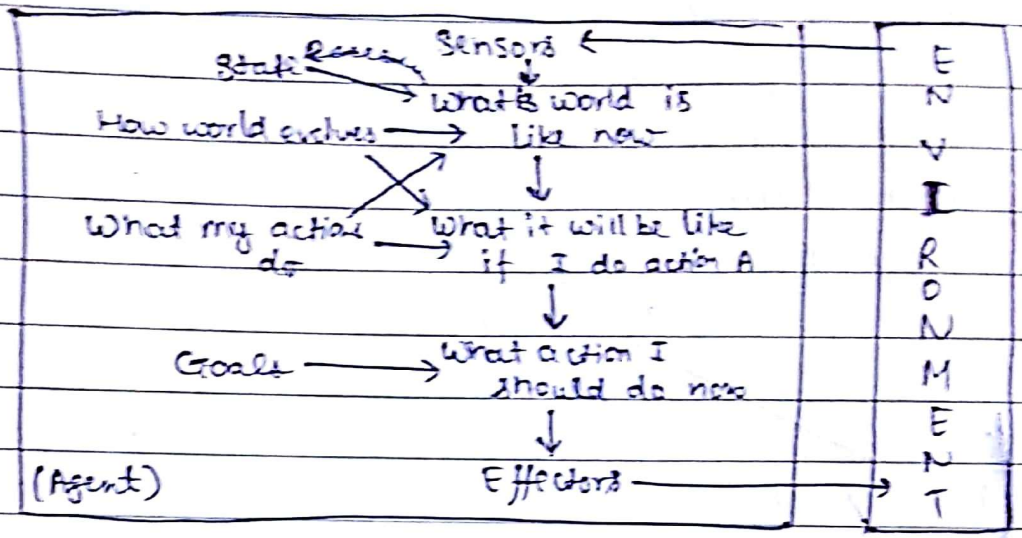


(ii) Model-Based Reflex Agents:
- Deal with partial accessibility; do this by keeping track of part of world it can see now.



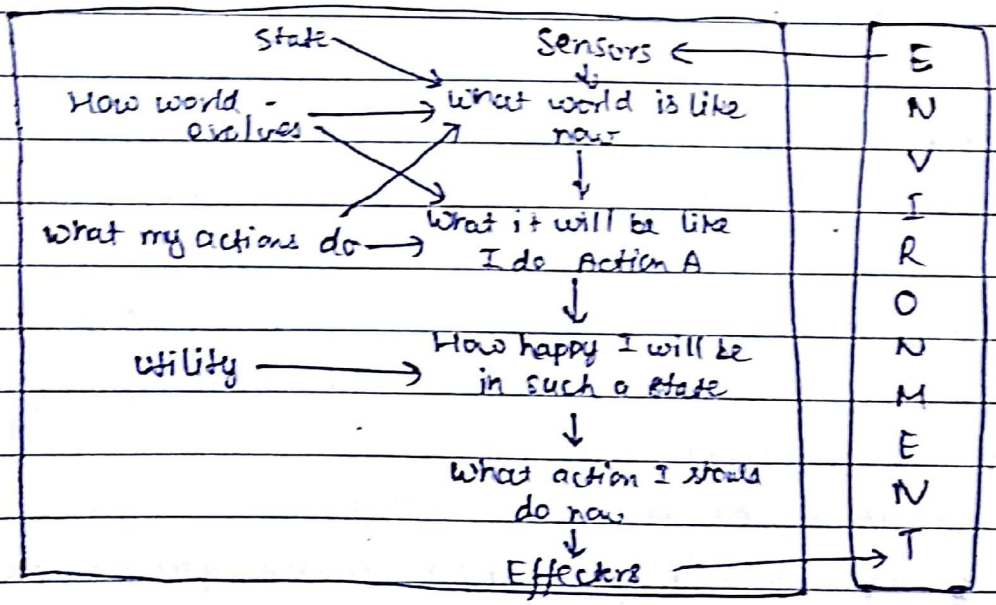
(iii) Goal based Agents

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



(iv) Utility based Agents

- A utility func^n 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

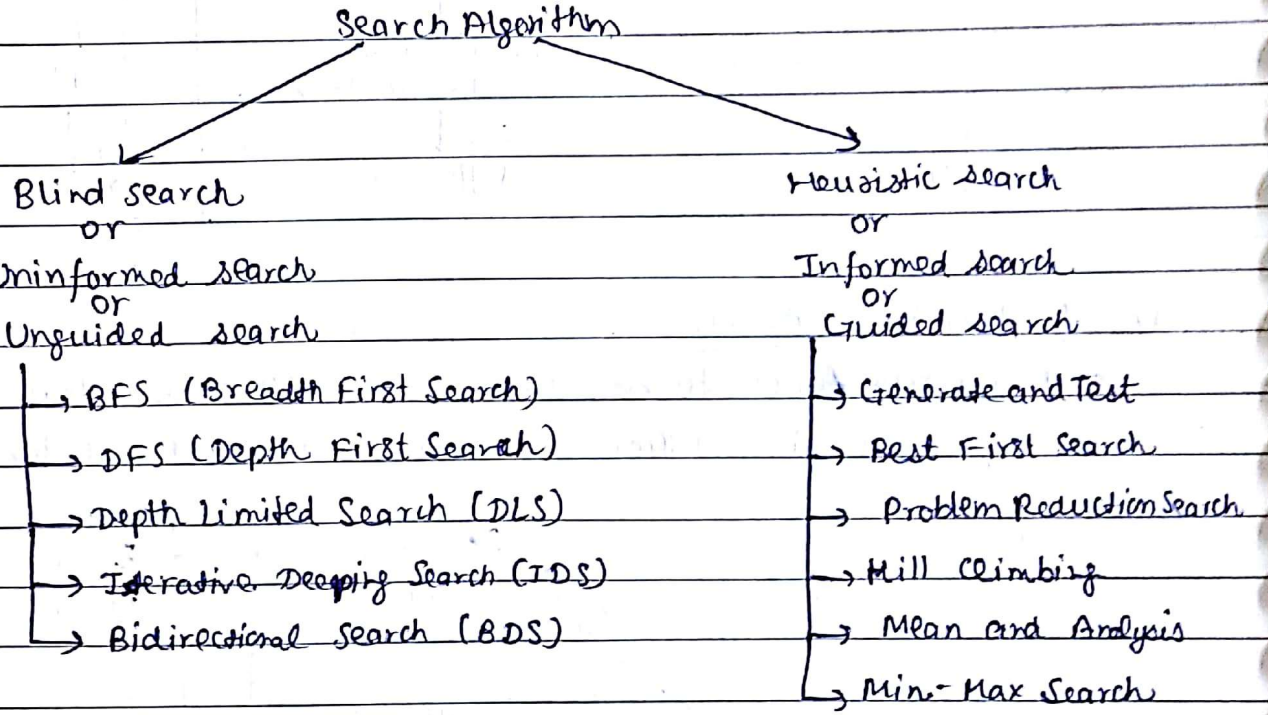
E → Environment

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:

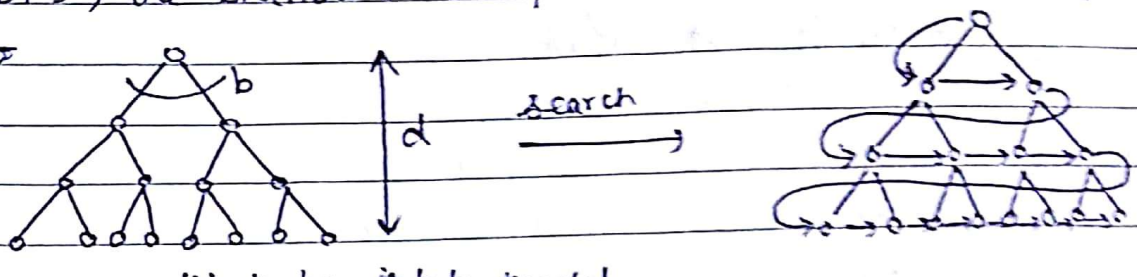


* Uniformed search	Informed Search
(i) Also called Blind search, bruteforce search & exhaustive search	(i) Also called intelligent search
(ii) Information is not given.	(ii) Information is given
(iii) Doesn't provide optimized soln.	(iii) Provide optimized soln
(iv) Guaranteed soln is found	(iv) Soln may not be guaranteed

* Breadth First Search:
  - Expand root node
  - Expand all children of root node
  - Expand all grandchildren etc.

→ It searches breadth wise in problem space.

In BFS, the shallowest unexpanded node is chosen for expansion.

→ 

say this is tree to be searched

→ Steps: (i) Start from root node

(ii) Expand its child & search among the children

(iii) If not expand, expand the childer 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 minimal sol^n (with least no. of steps).

→ Disadvantages: (i) Memory requirements

(ii) Consume lot of time if sol^n farther from root

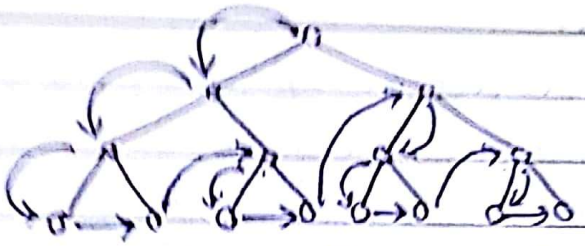
→ Time complexity = $O(b^d)$ b → branching factor

→ Space complexity = $O(b^d)$ d → 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:- (i) Memory requirement is only linear wrt search graph
(ii) Time complexity is $O(b^d)$
(iii) Take very less time & space

→ Disadvantage:- (i) May go down leftmost path forever
(ii) No guarantee to find minimal sol^n

→ Complexity : Time = $O(b^m)$ → 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:
(i) Standard failure value : No sol^n
(ii) Cut-off value : No sol^n within depth limit.

→ Time complexity = $O(b^d)$ → limit of depth
Space complexity = $O(b^d)$

* Iterative Deepening Search.

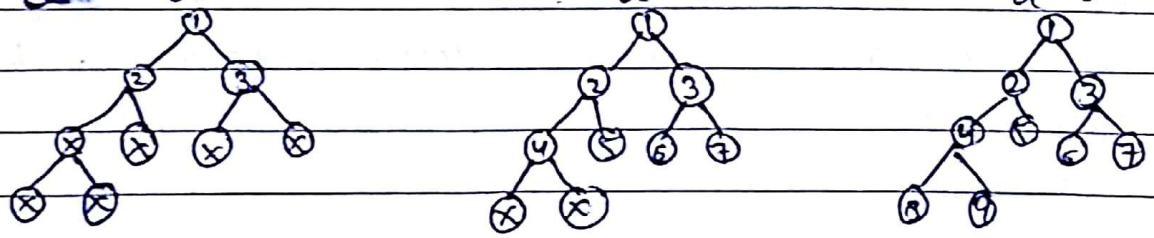
→ Derivative of DLS & combines DFS & BFS

→ 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 find best sol^n so complete & optimal search

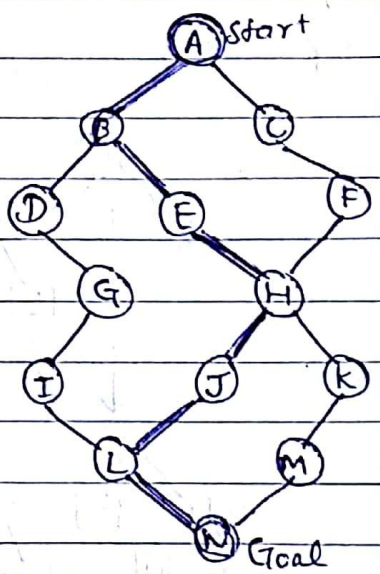
→ d=1 d=2 d=3



* Bi-directional Search

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

→ e.g.:



Bidirectional search meeting in middle at node H

→ Bidirectional search is complete & optimal

→ Time Complexity : $b^{k/2} + b^{k/2} = 2b^{k/2} < b^k$

* 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 may 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 this is a soln

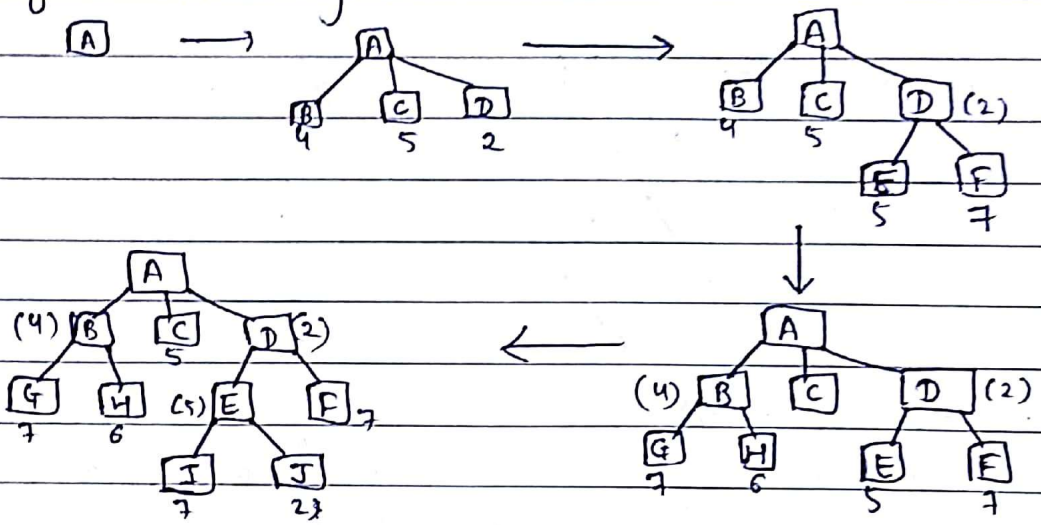
* Best First Search

→ Evaluation func^n with every node

→ Cost of evaluation func^n = cost / distance of current node from goal node

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

→ Eg:-



→ At any step, the most promising node having least value of utility func^n 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 func^n 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' = GOAL, terminate with success

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

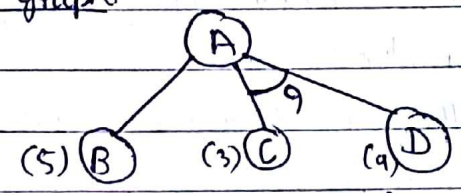
(vi) Name this list as START1

* Optimal Best First Search (A* algorithm)

- Evaluation funcn, $f(n) = g(n) + h(n)$ → cost of current node from start
  - $g(n)$: 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 (eg ex- GPS)

→ 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 offerred a better alternative than ones at hand.

→ It is not guaranteed to lead to a soln, since it can get stuck on plateaus, or local optimum, or even wander on ∞ uncontrolled paths.

→ 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 that is higher than surrounded 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^n 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 Satisfactory Problem

→ CSP consists of triplets $\{X, D, C\}$ where $X = \{X_1, \ldots X_n\}$ (set of variables), $D = \{D_1, \ldots D_n\}$ (set of domain

→ Algorithm / Pseudocode

Until a complete soln 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 soln then report success.

(v) if neither a constraint nor a complete soln is found then apply rules to generate a new partial soln & insert these partial soln 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 object	*	*	*			
Move robot			*			
Clear object				*		
Get object on object					*	*
Get 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)

* Water Jug Problem

ex:- I^st jug - 4 l & II^nd jug - 3 l

Fill 4 l jug with 2 l

State space (x, y)

Production rules: Rule 1 : (x:y) → (4,y) (fill 4 l jug ; x ≤ 4)

Rule 2 : (x:y) → (x,3) (fill 3 l jug ; y ≤ 3)

Rule 3: (x:y) → (x-x1, y) (pour water out of 4 l jug)

Rule 4 : (x:y) → (x+y-y1) (pour " " " 3 l ")

Rule 5 : (x:y) → (0, y) (Empty 4 l jug)

Rule 6 : (x:y) → (x, 0) (Empty 3 l ")

Rule 7 : (x:y) → (4, y-(4-x)) (Fill 4 l by pouring some water from 3 l jug)

Rule 8 : (x:y) → (x-(3-y), 3) (" 3 l " " " " " 4 l jug)

Rule 9 : (x:y) → (x+y, 0) (Pour all water from 3 l jug to 4 l jug)

Rule 10 : (x:y) → (0, x+y) (" " " " 4 l " " 3 l ")

Rule 11 : (2, 2) → (2, 0) (4 2 l from 3 l jug to 4 L jug)

Rule 12 : (2,y) → (0,y) (Empty " " 4 l jug on ground)

Output:

Rule No. applied	Jug 1 (4 l)	Jug (3 l)
Initial state	0	0
1	4	0
8	1	3
6	1	0
10	0	1
1	4	1
8	2	3

(or)

Initial state	0	0
2	0	3
9	3	0
2	3	3
7	4	2
5	0	2

* Cryptarithmetic Problem

eg:- SEND + MORE = MONEY

C_1 C_2 C_3 C_4

S E N D

+ M O R E

M O N E Y

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

so M=1, C_1=1

(ii) S+M = C_1×10+O | C_2+S+M>9, C_2+S+1=9

C_2+S+1 = 10+O

C_2+S = 9+O

If C_2=0 | If C_2=1

S+M+C_2 | C_2+S+M>9

8+1+0 = 9 ✗ | 8+1+1>9 as M≠O

8+1+1 = 10 ✓

9+1+0 = 10 ✓

Let S=9 so C_2=0 Let C_2=0 so S=9

(iii) E+O = N+C_3×10 | N≠9 (as S=9)

E+C_3 < 10 & N<10 so N<9

E+C_3 < 9

so C_3≠0 as E=N

so C_3=1 & E+1=N

(iv) N+R+C_4 = C_3×10+E | E+C_4+N+R = 10+E

Let C_4+E+N+R = 10+E

C_4+R = 9

Let C_4=0 so R=9 but S=9 so ✗

Let C_4=1 so R=8

(v) D+E = 10+Y

D+E = 10+Y = 10 ✗

= 11 ✗

= 12 ✓

= 13 ✗ as 7+6, 6+7

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

Ex:- $C_4\ C_3\ 0\ C_1$
SOME
+ TIME
SPENT

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

(ii) $C_3 + S + T = P + 10C_4$
$C_3 + S + T = P + 10$ & $E + E = 10C_1 + T$
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
$S + T = P + 10$ [$\because S = 1$]
$T = P + 9$
If $P = 0$, $T = 9$ [No 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
$1 + S + T = P + 10$ [$\because S = 1$]
$T + 2 = P + 8$
If $P = 0$, $T = 8$
$P = 1$, $T = 9$ [Not possible T is even]
$P = 2$, $T = 10$ [Not possible T is < 9]
so $P = 0$ & $T = 8$ & $C_3 = 1$

(v) If $T = 8$ then E can be 4 or 9
$E + E = 10C_1 + 8 = 18$ or 8

(vi) Let $E = 4$ so $C_1 = 0$
$C_1 + M + M = N + 10C_3$
$M + M = N + 10C_2$ so N is even no [2, 4, 6, 8]
$N = 2, 4, 6, 8$
$N \neq 8$ as $N \neq T$ [$\because T = 8$]
$C_2 + 0 + I = 10C_3 + E$
$\because E = 4$ so $C_2 + 0 + I = 14$
$C_3 = 1$ $M + M = N + 10C_2$

(vii) Let $C_2 = 0$
$0 + I = 14$ $M + M = N$ & $M + M < 10$

Note For N=6, M=3 [only possible]
N=4, M=2 [Not possible as E=4 & E≠N]
N=2, M=1 [Not possible as S=1]
So N=6, M=3

(iii) O + I = 14
7 + 7 = 14 ✗ [as O ≠ I]
8 + 6 = 14 ✗ [as T=8 & N=6]
9 + 5 = 14 ✓ ✓
so either O=9 & I=5 or I=5 & O=9

SOME		1934		1534
+ TIME	or	+ 8534	or	+ 8934
SPENT		10468		10468

(iv)
CROSS		96233
+ ROADS	=>	62513
DANGER		158746

(v)
GIVE		1407
+ THEM		
HELP	=>	2379
		3786

Means End Analysis Algorithm

(1) Compare current to goal, if there are no differences then return
(2) Otherwise select the most important difference & try to reduce it by doing the following until success or failure is signalled.
(i) Select an un-tried 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 precaditions are satisfied) & O-result (the state that will result if O were applied in O)
(iii) If (First part ← MEA (current, O-start) & (last part ← MEA (O-result, goal)) are successful then signal success & return result of 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
→ Two types of logic:
(i) Propositional Logic (ii) Predicate Calculas

* Propositional logic

→ It is two valued logic (True/False, 0/1, Yes/No)
→ Tautology ⇒ when all conditions are true in truth table
→ Contradiction ⇒ when all conditions are false in truth table
or
Absurdity
→ Contingency ⇒ when both true & false conditions in truth table.
→ Propositional variables :- p, q, etc.

p	q	p ∧ 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 ∧ q
(ii) Composite : Ex :- (p ∧ q) ⇔ ∨ ~ (q ∧ ~ ∨ r)

→ Connectives: (i) AND (∧) / Conjunction
(ii) OR (∨) / disjunction
(iii) NOT (¬ or ~) / negation
(iv) If... Then (→ or ⇒) / Implication
(v) If & Only If (Iff) (↔ or ⇔)
(vi) NAND (↑) = p↑q = ~(p∧q)
(vii) NOR (↓) = p↓q = ~(p∨q)

p	q	p → q	p ↔ q
0	0	1	1
0	1	1	0
1	0	0	0
1	1	1	1

ex:- Let p: Earth is flat
q: All birds sing
r: Manhattan is an island

NOTE:
$p \to q \equiv \sim p \vee q$
$p \leftrightarrow q \equiv p \odot q \equiv (p \wedge q) \vee (\sim p \wedge \sim q)$

(i) $\sim q \wedge r$: Some birds don't sing and Manhattan is an island
(ii) $\sim(q \wedge p)$: 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) $p \wedge \sim q \vee r)$: Earth is flat and it is not the case that all birds sing or that manhattan is 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 ∧ by ∨ & ∨ by ∧.
If any formula is is valid then dual is also valid
Ex:- (∧, ∨) & (↑, ↓) are duals to each other

→ Logical Equivalence:
If two proposition $P(p,q....)$ & $Q(p,q,....)$ 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 $P \Rightarrow Q$ $\sim P \Rightarrow \sim Q$

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: $Q \Rightarrow P$

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: $\sim Q \Rightarrow \sim P$

Indian team doesn't win 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) $p \wedge \text{False} = \text{False}$

(vii) Involution Law : (a) $\sim\sim p = p$

(viii) De Morgan's Law : (a) $\sim(p \vee q) = \sim p \wedge \sim q$

(b) $\sim(p \wedge q) = \sim p \vee \sim q$

(ix) Absorption law : (a) $p \vee (p \wedge Q) = p$ (b) $p \wedge (p \vee Q) = p$

(x) Contrapositive law : (a) $p \to q = \sim q \to \sim p$

(xi) (a) $p \Rightarrow q \equiv (\sim p \vee q)$

(xii) (a) $(p \Rightarrow q) \wedge (p \Rightarrow \sim q) \Rightarrow p$

(xiii) (a) $p \Leftrightarrow q = (p \wedge q) \vee (\sim p \wedge \sim q)$

→ Well-Formed Formula (WFF)

⇒ It is defined as follows:

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

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

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

(iv) 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	$p \wedge q \wedge r$

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
0	0	1	0 $(p \vee q \vee \sim r)$
0	1	0	1
0	1	1	0 $(\sim p \vee q \vee r)$
1	0	0	1
1	0	1	0 $(p \vee \sim q \vee r)$
1	1	0	0 $(p \vee q \vee \sim r)$
1	1	1	1

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

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

$\Rightarrow p \wedge (\sim p \vee q)$ (CNF)

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

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

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

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

(iii) Principal Disjunctive Normal Form (PDNF)

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

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

min terms are: $p \wedge q$, $p \wedge \sim q$, $\sim p \wedge q$, $\sim p \wedge \sim q$ of $p \& q$

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

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

PDNF: $(\sim p \wedge \sim q) \vee (\sim p \wedge q) \vee (p \wedge \sim q) \vee (p \wedge q)$

(iv) Principal Conjuctive Normal Form

- Product of sum canonical form (max terms only)

Ex:- $p \wedge q$

p	q	$p \wedge q$	
0	0	0	$\sim p \vee \sim q$
0	1	0	$\sim p \vee q$
1	0	0	$p \vee \sim q$
1	1	1	

max terms

So PCNF : $(\sim p \vee \sim q) \wedge (\sim p \vee q) \wedge (p \vee \sim q)$

→ Rules of Inference

	Rule	Tautological Form	Name
①	$\frac{p}{\therefore p \vee q}$	$p \Rightarrow (p \vee q)$	Addition
②	$\frac{p \wedge q}{\therefore p}$	$(p \wedge q) \Rightarrow p$	Simplification
③	$\frac{p, q}{\therefore p \wedge q}$	$((p) \wedge (q)) \Rightarrow (p \wedge q)$	Conjuction
④	$\frac{p \Rightarrow q, p}{\therefore q}$	$[(p \Rightarrow q) \wedge p] \Rightarrow (q)$	Modus ponens
⑤	$\frac{p \Rightarrow q, \sim q}{\therefore \sim p}$	$[(p \Rightarrow q) \wedge \sim q] \Rightarrow (\sim p)$	Modus tolens
⑥	$\frac{p \Rightarrow q, q \Rightarrow r}{\therefore p \Rightarrow r}$	$[(p \Rightarrow q) \wedge (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$	Hypothetical syllogism
⑦	$\frac{p \vee q, \sim p}{\therefore q}$	$[(p \vee q) \wedge \sim p] \Rightarrow q$	Disjunction syllogism
⑧	$\frac{p \Rightarrow q \wedge (r \Rightarrow s), p \vee r}{\therefore q \vee s}$	$[(p \Rightarrow q) \wedge (r \Rightarrow s) \wedge (p \vee r)] \Rightarrow (q \vee s)$	Constructive 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
(i)	$p \rightarrow q$	
(ii)	$r \rightarrow \sim q$	
(iii)	r	
(iv)	$\therefore \sim p$	

Proof:

Steps		Reason
1.	$r \rightarrow \sim q$	(ii)
2.	$q \rightarrow \sim r$	Step 1 & Contrapositive
3.	$p \rightarrow q$	(i)
4.	$p \rightarrow \sim r$	Step 2, 3 & law of hypothetical syllogism
5.	r	(iii)
6.	$\sim(\sim r)$	Step 5 & double negation
7.	$\therefore \sim p$	Step 4 & 6 & modus tollens

* Predicate Calculus

→ Quantifier

⇒ Two types: (i) Universal Quantifier ($\forall$)
means "for all" ex:- $\forall x \in A : P(x)$

(ii) Existantial 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

(ii) Some students are clever
$\Rightarrow (\exists x)(P(x) \wedge q(x))$

(iii) Some students are not successful
$\Rightarrow (\exists x)(p(x) \wedge \sim r(x))$

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

(ii) all cows are not white
let $p(x)$: x is cow
$q(x)$: x is white
$\forall x (p(x) \rightarrow \sim q(x))$

→ Rules of Inference for Quantified Statements

(i) Universal Instantiation

$\forall x\, P(x)$ / $P(c)$ for all c

ex:- Every man is mortal, Ram is a man / Ram is mortal

(ii) Universal Generalization

$P(c)$ for all c / $\forall x\, P(x)$

(iii) Existantial Instantiation

$\exists x\, P(x)$ / $P(c)$ for some c

(iv) Existantial Generalization

$P(c)$ for some c / $\exists x\, P(x)$

→ Universal Modus Ponens

$\forall x$, if $P(x)$ then $Q(x)$
$P(a)$ for a particular a
$\therefore Q(a)$

Concept of lifting

→ Universal Modus Tollens

$\forall x$ if $P(x)$ then $Q(x)$
$\sim Q(a)$ 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 R denote R

(i) $\forall x\ (A(x) \Rightarrow B(x))$
(ii) $A(R)$
$\therefore B(R)$

By universal ~~instant~~ 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
$\therefore$ Conclusion is valid.

* Skolemization Process (Part of FOPL)

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

Rule 2: For each existential quantifier, which is preceeded by one or more universal quantifier, replace all occurances of quantified variable by a func^n not appearing anywhere else.

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

Applying Skolemization Process,

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

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

Result

* First Order Predicate Logic (FOPL)

→ 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 : ∀ , ∃

→ Constants & variable functional 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 FOPL
(i) Lipton is a tea
∴ tea(Lipton)

(ii) Ruma dislikes children who drinks tea
∴ ∀x : child(x) ∧ drink tea(x) ⇒ dislikes(Ruma, x)

(iii) All purple mushrooms are poisonous
∀x : Mushroom(x) ∧ purple(x) ⇒ poisonous(x)

→ Inference Rules

(i) Universal Instantiation

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

ex :- ∀x King(x) ∧ Greedy(x) ⇒ Evil(x)
x → Aman, Raman
∴ two sentences obtained
King(Aman) ∧ Greedy(Aman) ⇒ Evil(Aman)
King(Raman) ∧ Greedy(Raman) ⇒ Evil(Raman)

(ii) Existential Instantiation

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

ex :- ∃x Crown(x) ∧ On Head(x, Ram)
∴ Crown(H) ∧ On Head(H, Ram)

→ 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

Unify $(P, Q) = \theta$ → Identifier

$SUBST(\theta, P) = SUBST(\theta, Q)$

ex:— We have a query:

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

knowledge bas : knows (Ram, Sita)

knows (y, Shyam)

knows (y, Father(y))

knows (x, Radha)

Unify (knows (Ram, x), knows (Ram, Sita)) = (x / Sita)

Unify (knows (Ram, x), knows (y, Shyam)) = (x / Shyam, y / Ram)

Unify (knows (Ram, x), knows (y, Father(y)) = (y / Ram, x / Father (Ram))

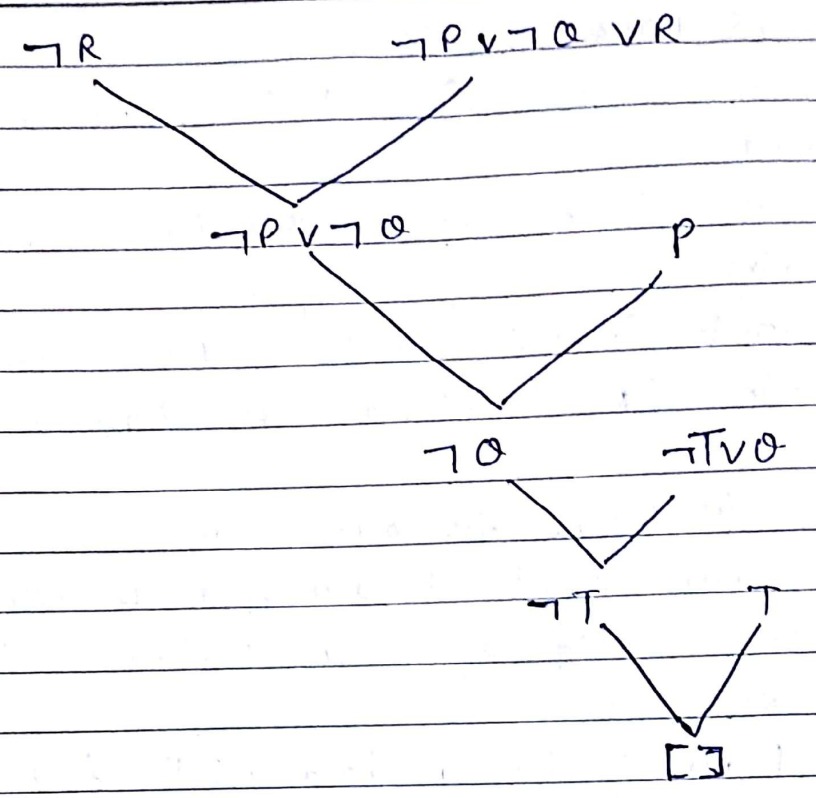
Unify (knows (Ram, x), knows (x, Radha) = Fail

→ Resolution

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

ex:— Prove R is true by resolution

	(Clause Form)	
(i) P	P	
(ii) $(Q \wedge S) \rightarrow R$	$\neg P \vee \neg Q \vee R$	P_1
(iii) $P \rightarrow T$	$\neg P \vee T$	P_2
(iv) $(T \wedge S) \rightarrow Q$	$\neg T \vee \neg S \vee Q$	P_3
(v) T	T	P_4
(vi) S	S	P_5
	$\neg R$	P_6
	$\neg P \vee \neg Q$	P_1, P_6
	$\neg P \vee \neg T \vee \neg S$	P_3, P_6
	$\neg T \vee \neg S$	P_1, P_4
	$\neg S$	P_4, P_5
	NIL	P_5, P_6



∴ we found contradiction, So, R is true

* Forward and Backward Chaining

→ There are two kinds of rule system: forward chaining systems & 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.

→ In backward chaining system, we start with some goal (hypothesis) 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 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 soln in the same soln 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

* Defination 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~~

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

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

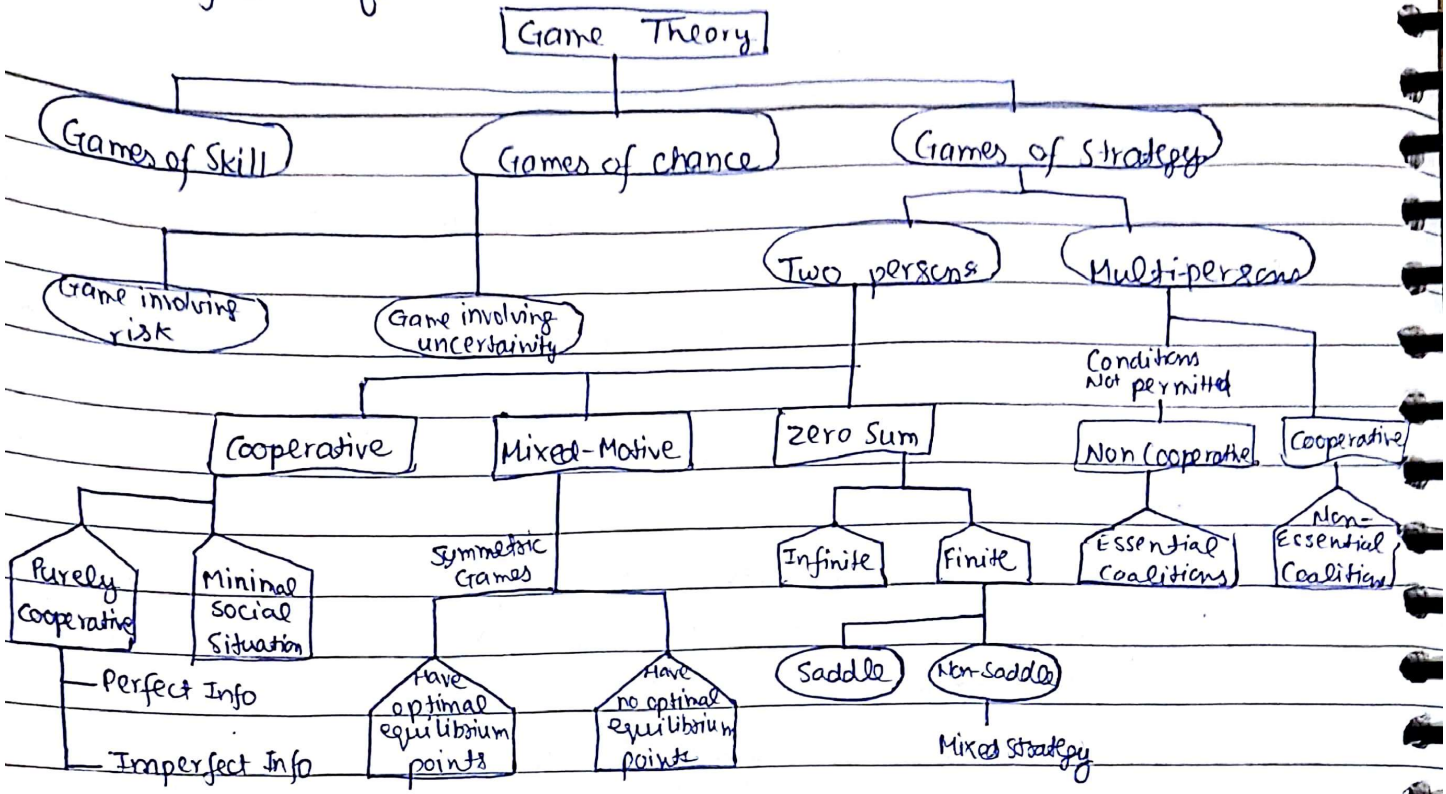
⑤ Non-zero sum Game (Some combinations of Actions provide a net gain (+ve sum) or loss (-ve sum) to two of them)

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 Limits

* 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 & plausible move
(Superset of all possible moves) (Result in good score for player)

- SEF (Static Evaluation Function)

(iii) Pruning

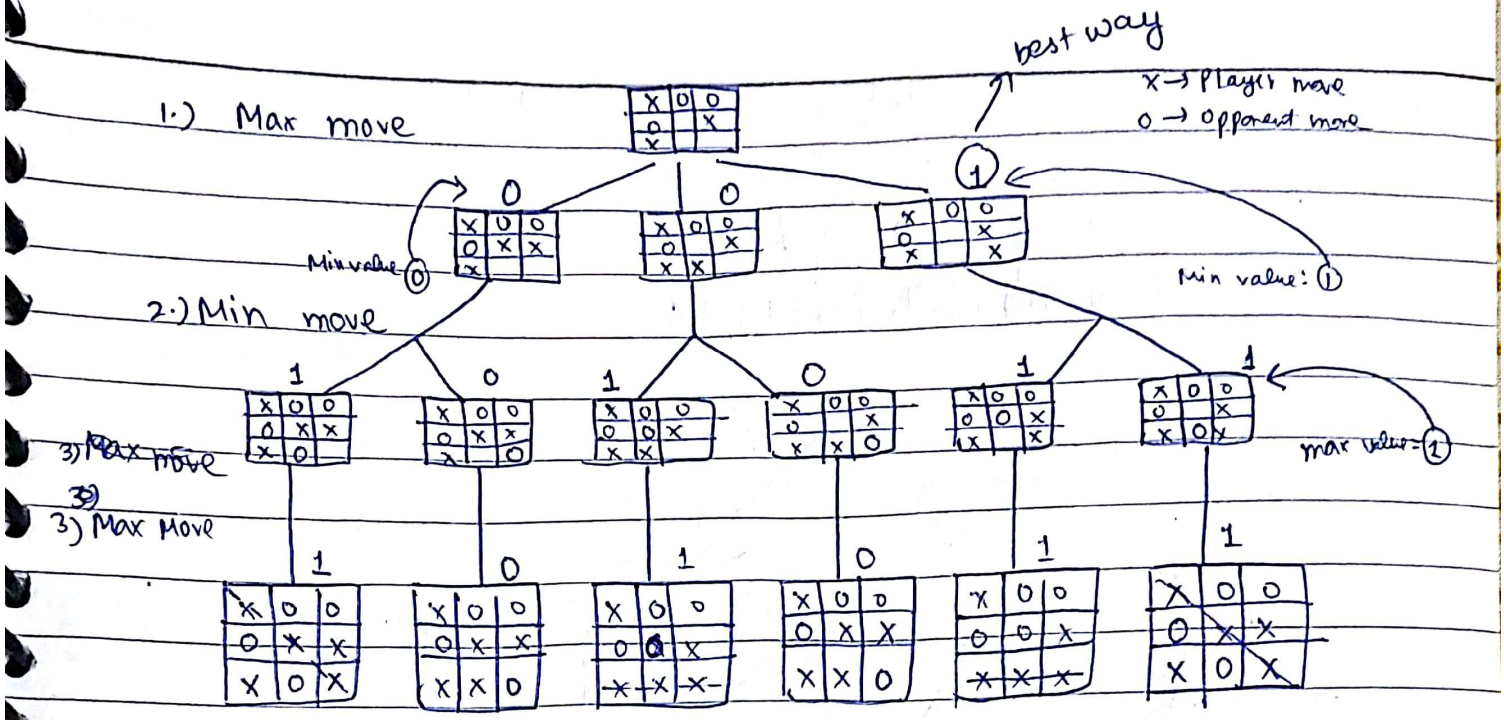
* Min Max Algorithm

- A way of finding an optimal move in a two player game
- Player moves : Max nodes represented as □ or △
  Opponent moves : Min nodes represented as ○ or ▽
- Assume max player makes first move & then proceeds.
- Ex:- Tic tac toe game

Initial State:

X	O	O
O		X
X		

After making game tree assign values 0 & 1 at leafs for loss & gain & for max node



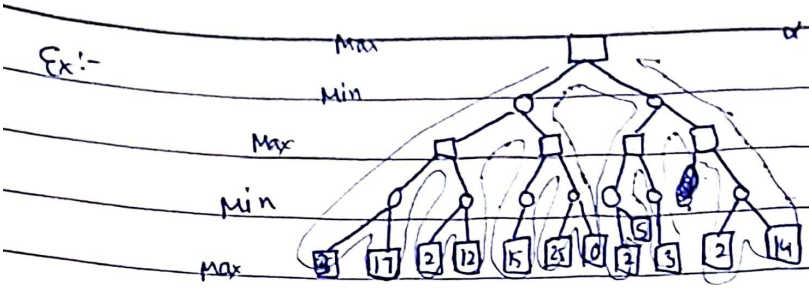
- Limitations:
  (i) Effectiveness limited by depth of game tree
  (ii) A rapid tie b/w assigned & actual value to static evaluation (SEF) func^n fails to decide expansion of game tree
  (iii) Horizon Effect : No guarantee that score is optimal

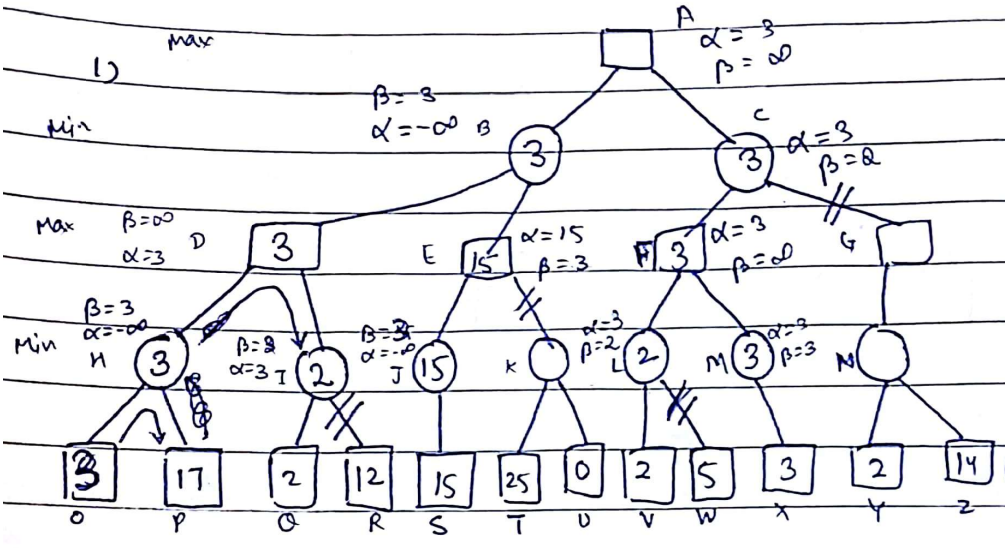
* Alpha - Beta Pruning

- 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 adversial search Algorithm.
- β : minimum upper bound of possible sol^n
  α : maximum lower bound of possible sol^n
- α (of max node) or β (of min node) pruning if $\alpha \geq \beta$ or new node considered if $\alpha \leq N \leq \beta$ (current value)
- α, β values can go from parent to child but not vice versa
- Rules for updating α, β values
  → if min node : if current β > current value node
  then β = current value node
  → if max node : if current α < current value node
  then α = current value node
- Initially assume for all nodes $\alpha = -\infty$ & $\beta = \infty$

In this method for node value use minmax algo i.e., min node then min value from child & vice-versa

Ex:-





A, B, C, D, ..., N } α = -∞, β = +∞

Node	α	β	current node value
	-∞	+∞	
A			
B	-∞	3	
D	3	∞	
H	-∞	3	
① O			3
② P			17
③ H			3
Now branching to I			
(B) I	3	∞	
① Q			②
I	3	2	
Now branching to E			
E	-∞	3	
J	-∞	3	
S			15
E	15	3	

Ignored as 17 > 3

Updated D (Max node) { parent value to child value

Updated as 2 > 2 {

as α > β so β pruning & updating β

15 > 3 so no α, β updation {

no evaluating E

α = 15

as α ≥ β

so α pruning & updating β = 3

now A α = ~~∞~~ 3 & β = ∞

	α	β
A		
B	-∞	3
C		
D	3	∞
E		
F		
G		
H	-∞	3
I	3	~~∞~~ 2
J		
K		
L		
M		
N		

now traversing C & its child

	α	β
A	3	∞
C	3	∞
F	3	∞
L	3	∞
V		
→ L	3	2
M	3	∞
X		
M	3	3
B		

② } 2 < ∞ so L updated

α > β at L

} as β pruning

3 so updating M

to update f choose max among L & M so f then update C.

	α	β	value
C	3	∞	
F			3

& C is min node & 3 < ∞ then β updated

now

	α	β
C	3	3

as 3 ≥ 3 & C is min node so β pruning then updating $C_{value} = 3$

* 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^n 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 ~~an intelligent syst~~ 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.

* Components of NLP

NL Understanding (NLU)
- mapping the given input in natural language into useful representations
- Analyzing different aspects ~~of~~ of the language

NL Generation (NLG)
- Process of producing some meaningful phrases & sentences in the form of natural language from some internal representation

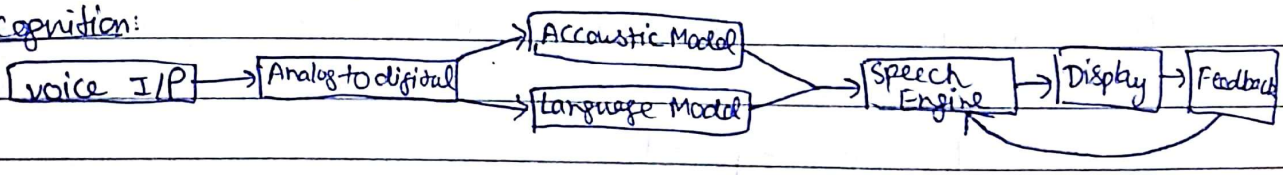
It involves: ① Text planning (Retrieving relevant content from knowledge base)
② Sentence planning (Choosing required words, forming meaningful phrases, setting)

- NLU is harder than NLG
- Difficulties in NLU : → NL has extremely rich form & structure
  → 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 premitive meaningful units
  ③ Syntax ④ Semantics ⑤ pragmatics ⑥ Discourse ⑦ word knowledge
* Steps in NLP
  - Lexical analysis: → Identifying & analyzing structure of word
    → Divide whole chunk of text into paragraph, sentences & words
  - Syntactic Analysis :
  - Semantic Analysis : Text is checked for meaningfulness
  - Discourse Integration
  - Pragmatic
  = Implementation of Semantic Analysis < CFG, Topdown parse

Speech Recognition

* Automatic speech recognition or computer speech → means understanding voice by the computer & performing any required task
* Uses : Dictation, system control/navigation, industrial application, voice dialing etc.
* Recognition:



* Accoustic model takes audio recordings & speech & 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

- 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 ~~acpt~~ output accoustic & 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 pertening, picking, moving the physical properties of object, destroying it or have an effect thereby freeing manpower from being repetitive func^n without getting ~~bored~~, bored, distracted or exhausted.

* ~~Robotics~~ 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

→ 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 $k=2$, $N = 2(4-1)! = 3! = 6$
→ Complexity of robots $\propto$ no. of legs

- Wheeled locomotion
→ requires power no. of motors to accompolish a movement.
→ little easy to implement as there as less stability issue.
→ power efficient as compared to legged locomotion.
- standard wheel - castor wheel (rotate around axle) - Balls (Spherical wheel)
- swedish 45° & 90° 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 image.
- 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:

→ Face-Detection: Read the face & take picture of that perfect expression. Used to detect a user access the software on correct match.
→ Object Recognition: Preinstalled in super markets, cameras, cars (BMW, Volvo etc)
→ Estimating Position

- Application Domains of Computer Vision

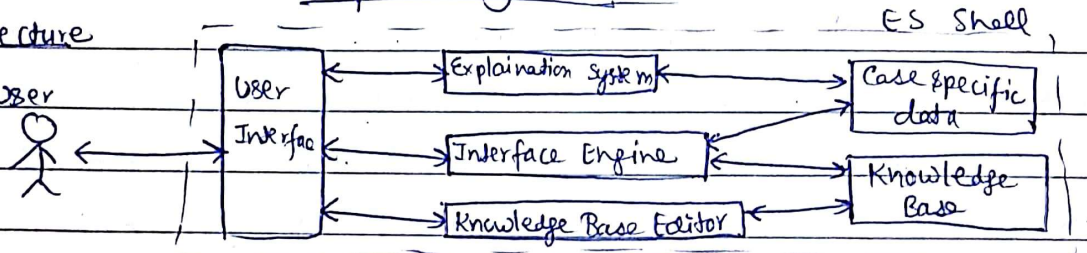
Agriculture, autonomous vehicles, biometrics, character recognition, face recognition, process control etc.

* Application of Robotics:

① Industries: Robots used for handling materials, cutting, welding etc
② Military: Autonous robot can reach inaccessible & hazardous areas during war
Ex:- Robot named 'daksip' developed by DRDC 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 develop to solve complex programs in a particular domain at the level of extraordinary human intelligence & 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 Justifying

* Incapabilities:
① Substituting human decision makers
② Possessing human capabilities
③ Producing accurate ~~opinion~~ output for inadequate knowledge base
④ Refining their own knowledge

* Components:
① Knowledge Base (KB)
- Software that represents knowledge. It contains ~~data~~ domain specific & high-quality knowledge. Knowledge is required to exhibit intelligence
- What is knowledge:
  - 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 representations: organizing & formalizing the knowledge
- Knowledge acquisation:
  - → 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 KB if required & resolve rule conflicts when multiple rules are applicable to particular case.
- To recommend a soln, ~~interface~~ IE uses following strategies:

③ User Interface

* Limitations:
  - Limitations of the technology
  - Difficult to maintain
  - Different knowledge acquisation
  - 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 acquisation, interface engine, user-interface, Explaination 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
    - Realise how the concepts can represent domain knowledge best
  ③ Develop prototype
    - knowledge acquisation & 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

Theoram Proving

* Reasoning by theoram proving is a weak method, compared to expert system 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

Its about statements of facts that describes a particular object & its attributes including some behaviour in rel^n 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 rel^{n}ship 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 : Acheived 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) Explaination Based Learning

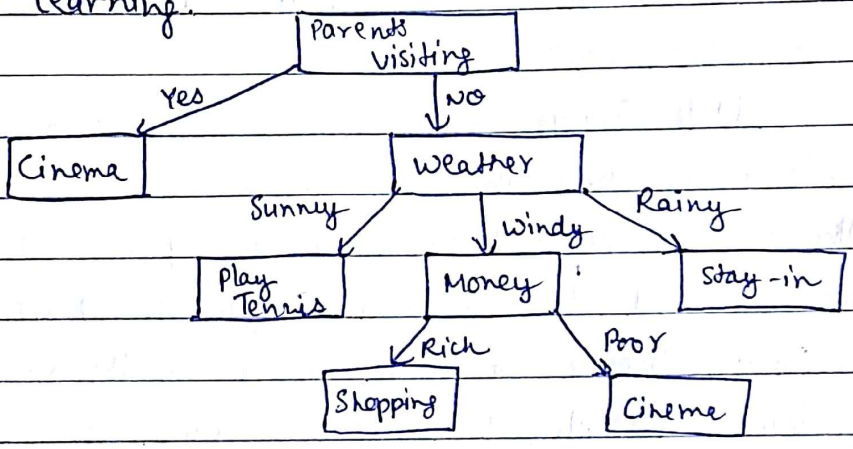
(7) Society Based Learning

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 (inference)
* Decision tree learning is a method that uses inductive learning to approximate a target func^n, which will produce discrete values.
* The learned func^n 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 & 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 definations is called induction or concept learning.
* Ex:-



Computational Learning Theory

* Theoritical 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 func^n that assigns labels to samples including

* 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 funch 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 - $P \neq NP$ (P vs NP problem), cryptographic - one way functions exist

Explaination 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 using a perfect domain theory is a program 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.

* EBL software takes four inputs:
- a hypothesis space — a domain theory
- training examples — operationality 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 priortize actions, is necessary to be able to be safe & effective in the sol^n of different autonomous applications

* Medical Science

- Medical AI mainly uses computer techniques to perform clinical diagnosis & suggest treatments
- AI has the ~~ability~~ capability of detecting meaningful rel^{n}ships 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 top down 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, maintainance, swarming of satellites etc.