Date : 15/9/22
Page No.

Principle of AI

- Definition of AI :-
AI is the part of CS concern with designing intelligent system i.e., system that exhibit the characteristic associate with intelligence in human behaviour.
- AI contains two terms :-
1.) Intelligence 2.) Artifical devices.
- A system with intelligence is expected to behave as intelligence as a human being. Secondly a system with intelligence is expected to behave in best possible manner
- In 1956 AI is invented by John Mccarthy.

Applications of AI

- Chemical Problem solving
- Expert System
- Natural language Processing
- Computer vision
- Robotics
- Comp. aided instruction
- Planning

20/9/22

(a) Symbolic Representation -
- AI programs mainly deals with non-numerical symbols & contrast strongly with commonly accepted view that computers can deal.
- AI programs perform numerical calculations also when necessary but their significance as symbolic value with generally enter into reasoning process performed by program.
→ The importance of physical symbolic hypothesis are as follows :-
- The is significant theory of human intelligence and as such is of great ~~interest~~ interest to the psychologist.

Date : / /
Page No.

- It forms basic defi. beliefs that it is possible to build a prog. which can perform intelligent task as performed by human.

→ Knowledge Representation :-

1. Knowledge can present as a modular (we can break into supports) and there is a correspond b/w external world & symbolic representation system.

- This knowledge can be studied & understood in human terms beoz symbols used for its representation are numerical.

2. Incomplete Data :- AI program has capability to provide some solution even if all data relevant to problem is not available at that time consequence of data being incomplete simply leads to conclusion that are less certain. IRL, our decision may be wrong in absence of relevant Data's. It can happen that absence of complete data is increasing in the problem as in game's bridge.

3- Conflicting Data :- Data items ~~are~~ can even be contradicting to each other, this type of Data is called conflicting Data or Data corrupted by error.

* Ability to learn :-

Intelligent computer have ability to learn from their mistake so that they can improve their performance by taking account of past errors. This is related to capacity

Date : / /
Page No.

for generalising to drawing analogy & discarding selective info.
The problem of given computers an learning ability similar to that of human during is that of stimulating the machine process the working of human minds never be programed in the machine so that no computer even will have as mind of its own.

History of AI

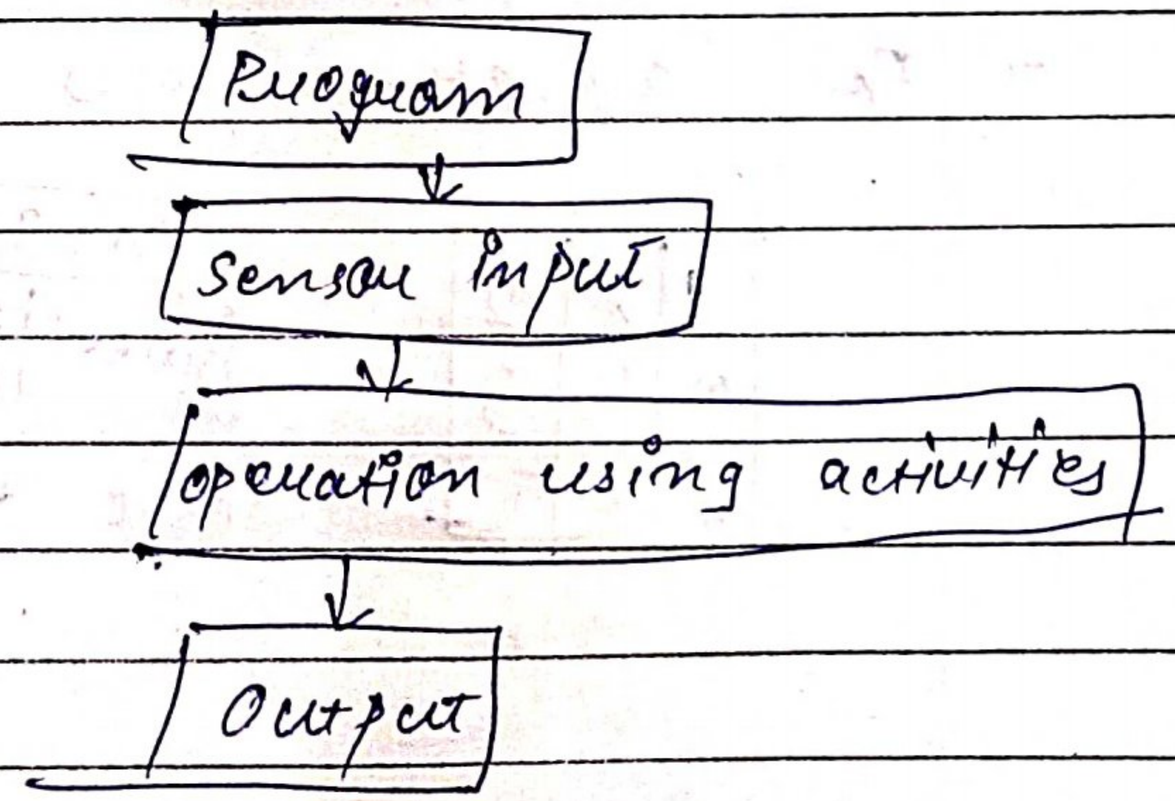
First Generation	Pre - 1950
second Generation	1950 - 1970
Third Generation	1970 - present.

FG - (First Generation)

1. Can a self-repairing machine be built which locate repair.
2. Can a digital computer be programmed to program itself.
3. using hierarchical concepts can the computer ~~prog~~ program.

22/9/22

Agent



Date : / /
Page No.

There are 3 parameters for an intelligent system

1.) knowledge base.
2.) operator.
3.) search technique.

State space problems :-

A combination of all possible states for a given problem is called SSP.

Initial

x	1	3	4
5	6	7	8
9	10	11	12
13	14	15	

Goal state

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

1.) If n is odd then puzzle is always solvable if no. of inversion is even in the input state.

2.) If n is even puzzle is solvable in two condition the blank is on an even row counting from bottom like second last and fourth last and no. of inversion is odd.
The Black is on an odd row from bottom counting and no. of inversion is even.

3.) For all other cases puzzles is not solveable.

1	8	2
x	4	3
7	6	5

$\boxed{n=3}$

inversion count = 10

→ solve this by Bubble sort.

Date : / /
Page No.

Que

3	9	1	15
14	11	4	6
13	x	10	12
2	7	8	5

Inversion count = 56

~~Insolve~~ not solveable. (By 2 (1) condition)

Ans

If Inversion count = 41 (solveable)

as n = even, inversion count = Odd, even number row from bottom.

☆ Architecture of AI Machine.

- Knowledge
- Inference process or control strategy
- Expert tools.
- Hardware & software.

26/09/22

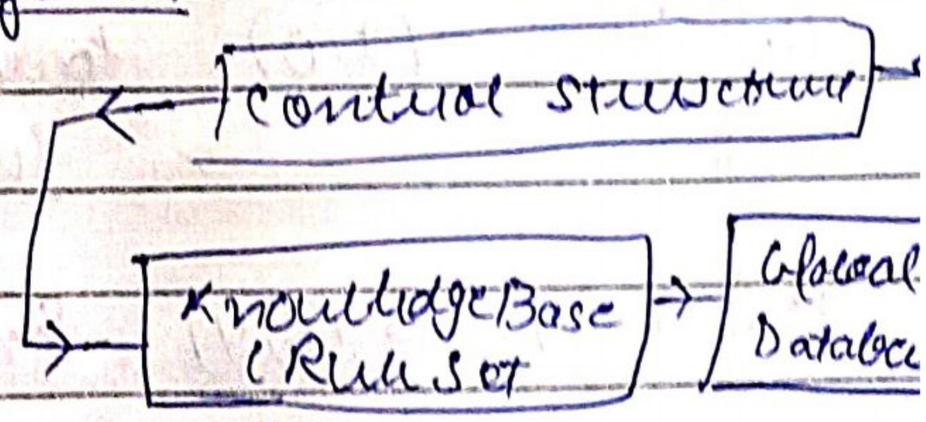
☆ Production system.

→ Roles of Production.

- Powerfull knowledge representation.
- Bridge b/w AI Research to Expert system.
- Heuristic Model for human behaviour
- Strong data-driven nature.
- New Rules can easily added to system.

→ Architecture of Production system

- Knowledge Base
- Global Data Base
- Control structure



Problems statement / representation

Transforming a problem into component of production system is called problem representation

27/9/22

1.) → (x,y) if $x<4 \to (4,y)$ → fill 4 gallon jug.

2.) → (x,y) if $y<3$ $(x,3)$ → fill 3 gallon jug.

3.) → (x,y) if $x=d$ & $d>0 \to (x-d,y)$ Pour some water out of 4 gallon jug

4.) → (x,y) if $y=d$ & $d>0$ → $(x,y-d)$ Pour some water out of 3 gallon jug

5.) (x,y) if $x>0 \to (0,y)$ → Empty the 4 gallon water on the ground.

6.) (x,y) if $y>0 \to (x,0)$ → Empty the 3 gallon water on the ground.

7.) (x,y) if $x+y \leq 4$ and pour water from 3 gallon $y>0 \to 4, (y-(4-x))$ jug into it ~~$x>0 \to 4, (y$~~ two jug is full.

8.) (x,y) if $x+y \geq 3$ Pour water from 4 gallon and $x>0 \to [x-(3-y)]$ into 3 gallon jug until it is full.

1-(4-4) 9.) (x,y) if $x+y \leq 4$ and Pour all water of 3 gallon $y>0 \to [x+y, 0)$ into 4 gallon.

10.) (x,y) if $x+y \leq 3$ and Pour all water from 4 gallon $x>0$ $(0, x+y)$ to 3 gallon.

11.) $(0,2) \to (2,0)$ Pour 2 gallon water from 3 gallon jug to 4 gallon jug.

12.) $(2,4) \to (0,4)$ Empty the 2-gallon in the 4-gallon jug on the ground.

Date : / /
Page No.

x	y	Rule
0	0	
0	3 →	2
3	0 →	9
3	3 →	2
4	2 →	7
0	2 →	5 or 12
2	0 →	9 or 11

x	y	
0	0	
4	0 →	1
1	3 →	8.
1	0 →	6
0	1 →	10
4	1 →	8. 1
2	3 →	8..

★ Algorithm for Breadth first search. → Queue data structure.

1.) Create a variable called Node list and set it to initial state.

2.) Until a Goal state is found or NODE-list is empty do.

2.1) Remove first element from NODE-list and call it E. If NODE-list empty quit.

2.2) For each way that each rule can match the state discribed in E do.

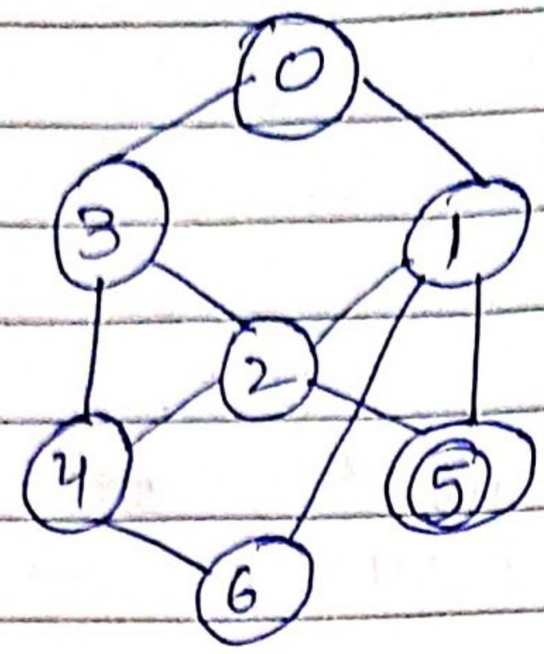
2.2.1) Apply the rule to generate a new state.

2.2.2) If the new state is a goal state quit and return this state.

2.2.3) otherwise add new list to end of NODE-list.

Date : / /
Page No.

Path is terminated.
→ Reaches dead end
→ Produces previous state
→ Becomes larger than limit.



~~Backtracking~~ :-

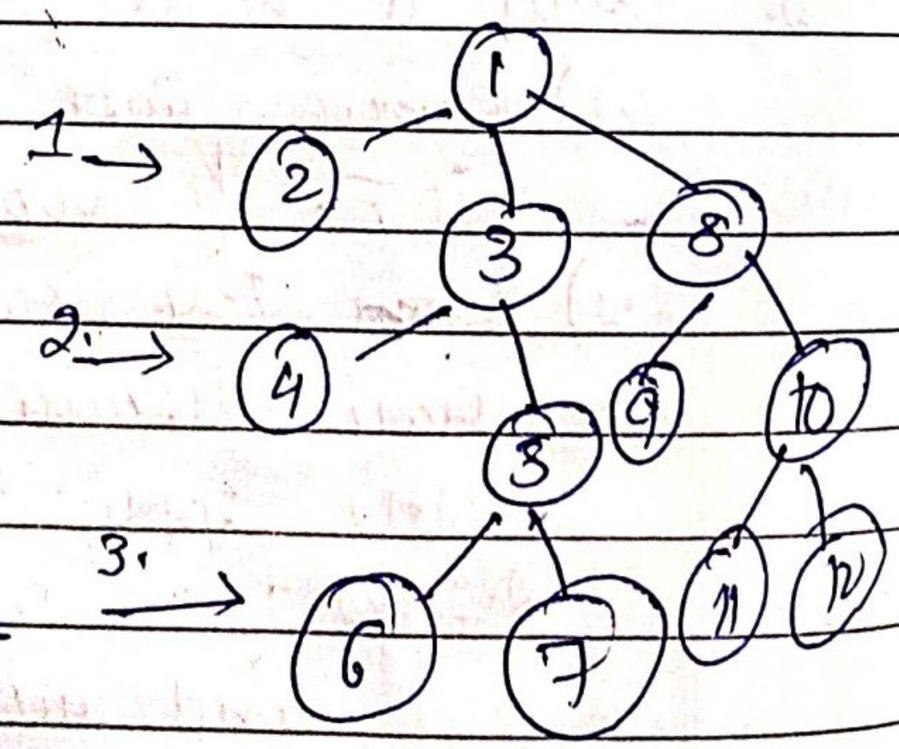
DFS :- Depth first search

Begin

1. PUSH the starting node at stack pointed to by the stack-top.
2. while stack is not empty do
   Begin
   pop stack to get a stack-top element.
   If stack top-element = goal
   Return sucess & Top.
   Else
   Push the children of stack-top element in any order.
   End while
   End.

Space complexity = $d(b-1)+1$
d = depth
b = branching factor

Time complexity = $\frac{(b^{d-1})}{b-1}$



Time complexity of BFS
$= \frac{b^d(b^d+1)}{2(b-1)}$

* Space complexity $= b^d$.

* HILL CLIMBING :-

Begin
1.) Identify possible starting states and measure the distance of their closeness with the goal node. PUSH them in a stack according to ascending order of their distance.
2.) Repeat
pop stack to get the stack-top element.
If the stack-top element is the goal announce it and exit.
Else
Push its children into the stack in the ascending order of distance.
until stack empty.
END.

→ It is simply a loop which continously moves in a direction of increasing value that is, up. It terminates when it reaches at the peak where no neighbour has higher value. The algo. does not maintain a search tree so the current node data structure need only record and its objective function. Hill climbing does not look a head beyond the immediate neighbours of the current state. Hill climbing is sometimes called greedy-local search bcoz it grabs the good neighbour states without thinking where to go next. Hill climbing often makes very rapid progress towards a solution because it is usefull quite easy to improve a bad state. There some difficulties in hill climbing algorithm

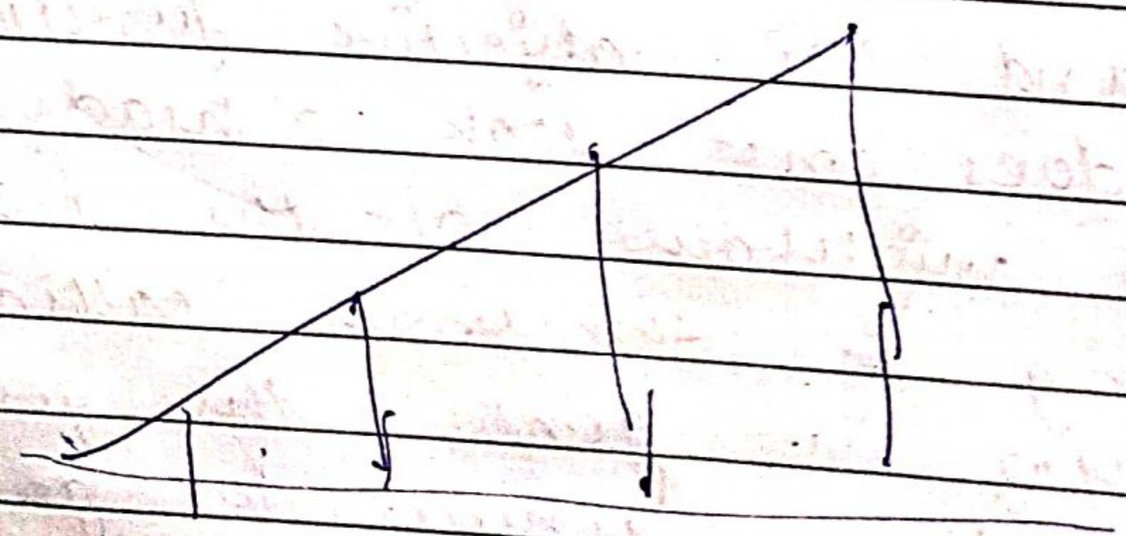
- Local Maxima
- Plateau.
- Ridge.

Date : / /
Page No.

→ a local maxima is a peak which is higher than its neighbour but lower than global maxima. so it is very difficult for greedy algorithm to navigate to all nodes.

→ Plateau - is a area of the state search where evaluation of the function is flat. It can be flat local maxima from which no uphill exist. A hill-climbing is not able to find exit from plateau.

→ Ridge - It is an area of the search space which is higher than corresponding area. and then itself have the slop but the orientation of high regions compares to the set of available nodes & direction in which they moves makes it impossible to overcome this moves applied two or more times before performing the test.



Informed search or Heuristic search

→ Hill climbing

→ Best first search

Date : / /
Page No.

~~Uninformed~~ Uninformed search

→ Optimal Solution.

★ Informed search :-

Information about a particular problem domain can be envold during the search.

The technique for doing required additional information about the property of specific domain which is built into the state & operator definition.

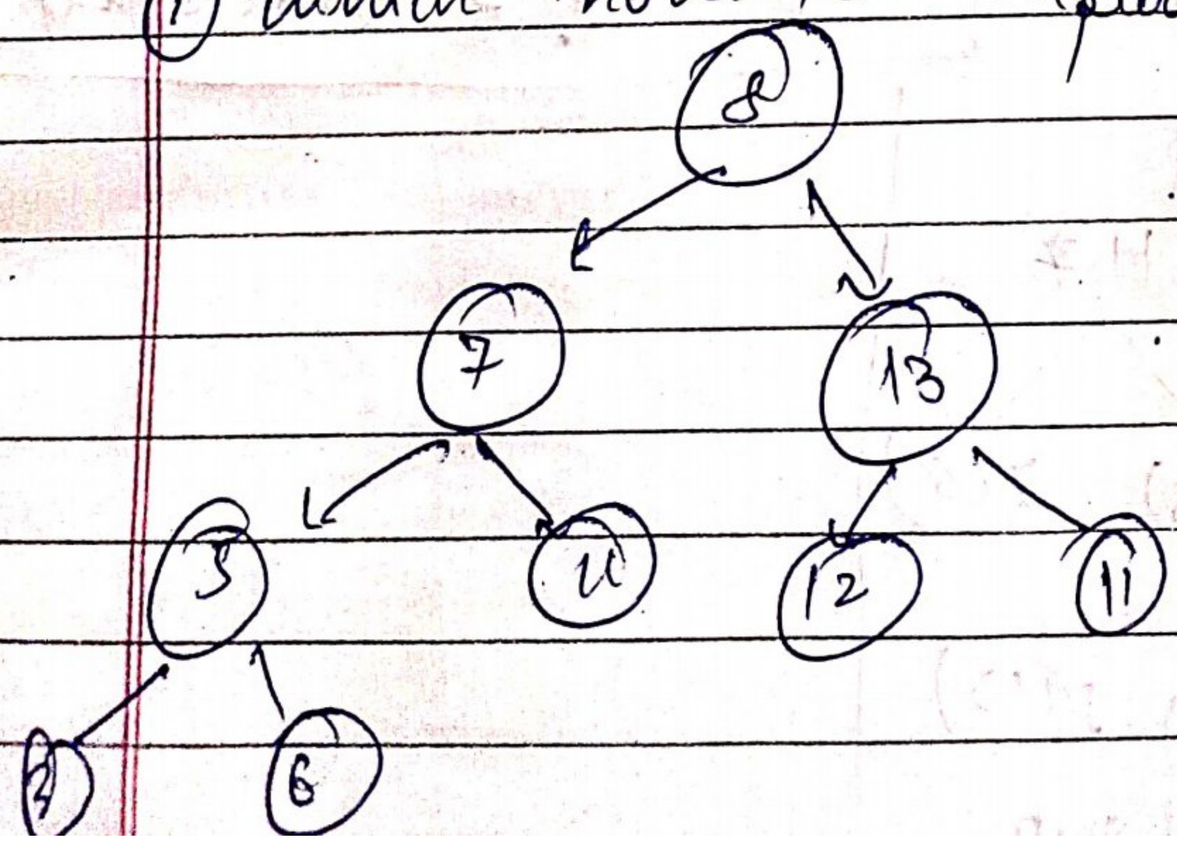
Informed search is the 1 where we are how information about state space can be proven search firm clark.

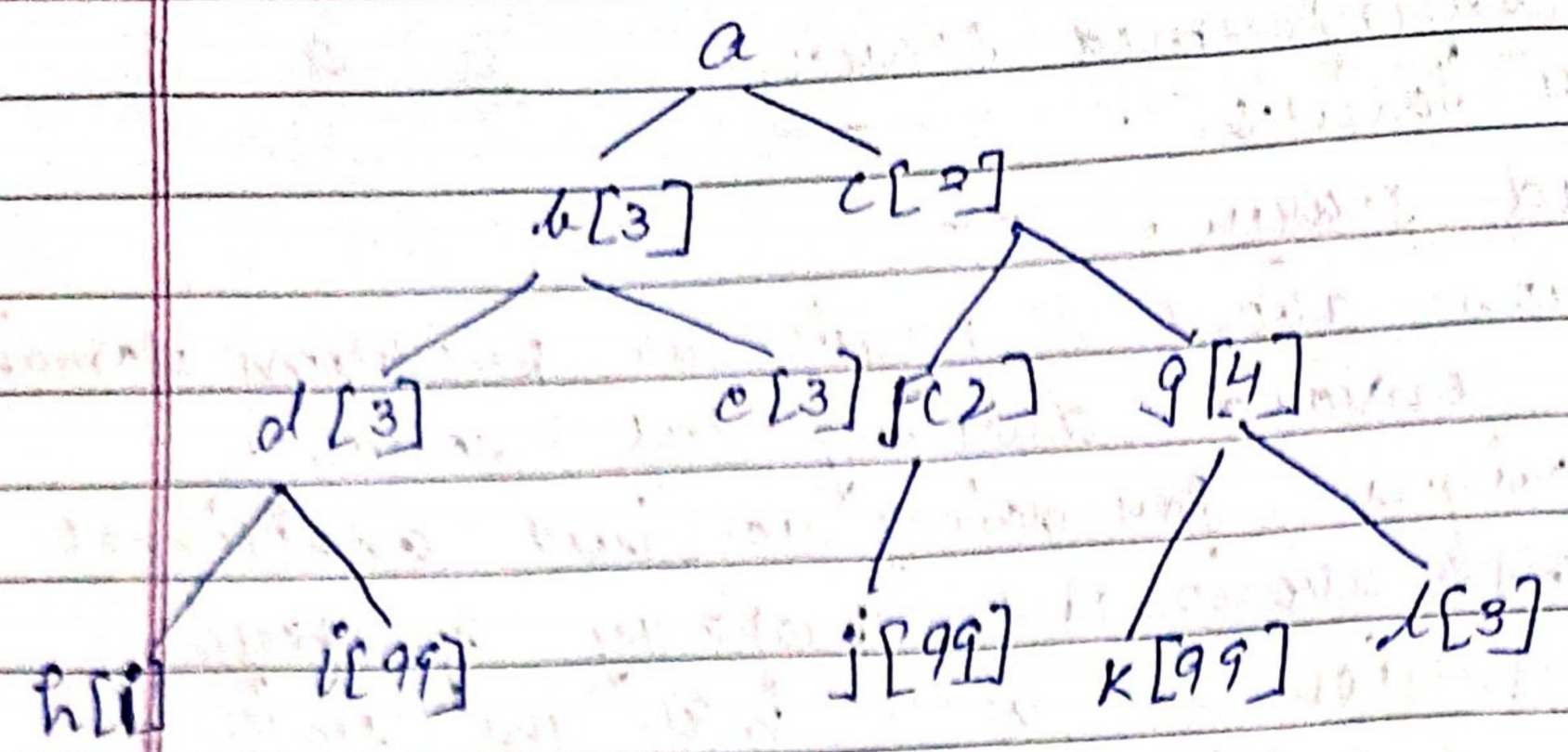
Information of a crant type is called Heuristic and the search method using this is also called heuristic search.

In heuristic search we are using heuristic function for evaluating good or bad node for the must the value of the search find out whether the given sol^n is best sol^n or not.

In heuristic search method first we have to define:

(1) union node to expanded next





Hill climbing

→ start at a

→ children of a [b[3], c[2]]

→ Best child c(2)

→ children of c (f(2), g(4))

→ children of [j(99)] → not goal

→ ~~Best~~ Back track to g through c.

→ children of g - [k(99), l(3)]

→ Best child l → Goal state

Best first search

start at a.

children of a a[b[3], c[2]]

Best first = c

children of c = [f(2), g(4)]

consider [b(3), f(2), g(4)]

Best first → f(2)

children of f → j(99)

consider [b[3], j(99), g(4)]

Best b(3)

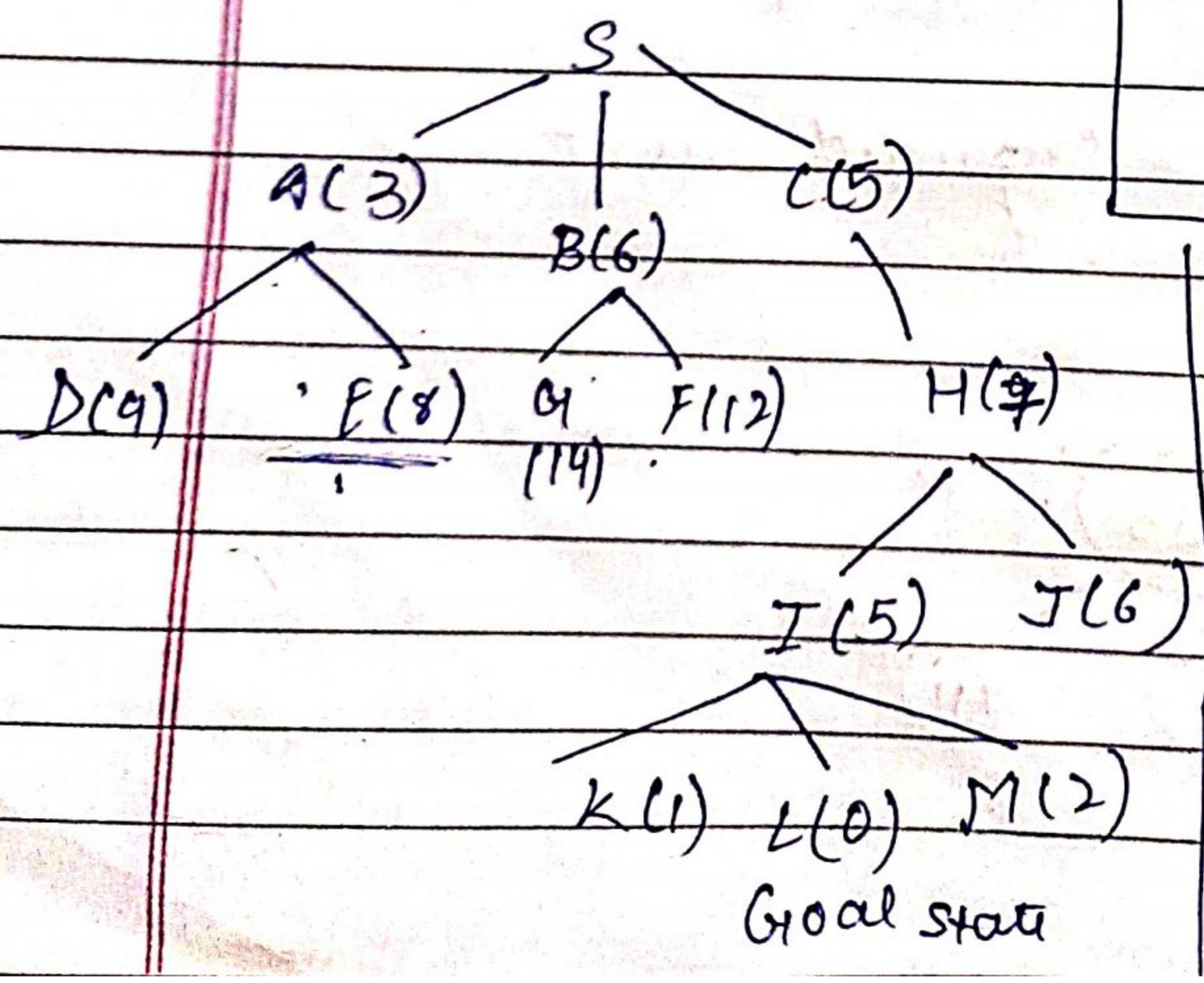
consider of b (d[3] e(3) g(4)

Best → d(3)

children of d → [h[1] i[99]

consider {h(1) i(99) e(3), j(99) g(4)]

Best → h → (goal state)



Date : / /
Page No.

Hill climbing

Start at a.

children of a [A(3), B(6) / (5)]

Best child A(3)

children of A [D(9), E(8)]

~~chil~~ Best child E(8) - (Not goal)

Back track to S through A.

Best child C(5)

children of C (H(7))

children of ~~C~~ H (I(5), J(6))

~~children of I~~ Best child I(5)

children of I (K(1), L(0), M(2))

Best child - L(0) goal state

* Time & space complexity of Best first search

$= O(b^d)$

d is maximum depth of state space.

Scanned with CamScanner

Date : / /
Page No.

Algorithm of Best first search.

1) Put the initial node on a list.
2) OPEN.
2) If (open is empty) or (OPEN = GOAL) terminate
3) Remove first node from OPEN and call this node a.
4) If (a = GOAL) then terminate and success.
5) Else if node has successors generate all of them find out how far they are from goal node Sort all the children generate so far by the remaining distance from goal
6) Name this list as closed
7) Replace it OPEN with closed.
8) GO to step 2.

6/10/22

Example of Hill climbing.

(4)

A
H
G
F
E
D
C
B

Initial state

(8)

H
G
F
E
D
C
B
A

Goal state

H	
G	
F	
E	
D	
C	
B	A

4+1 = 5

Intermediate step.

There possible moves.

(4)

A
H
G
F
E
D
C
B

(6)

	G
	F
	E
	D
	C
	B
0	H
+1	A

		G
		F
		E
		D
		C
		B
+1	+1	H
		A

6+2 = 8

we are using two operators in the given examples R_1, R_2

1.) (R_1) operator pick up one block and put up on table. (+1)

2.) (R_2) pick up one block & put it on another (-1)

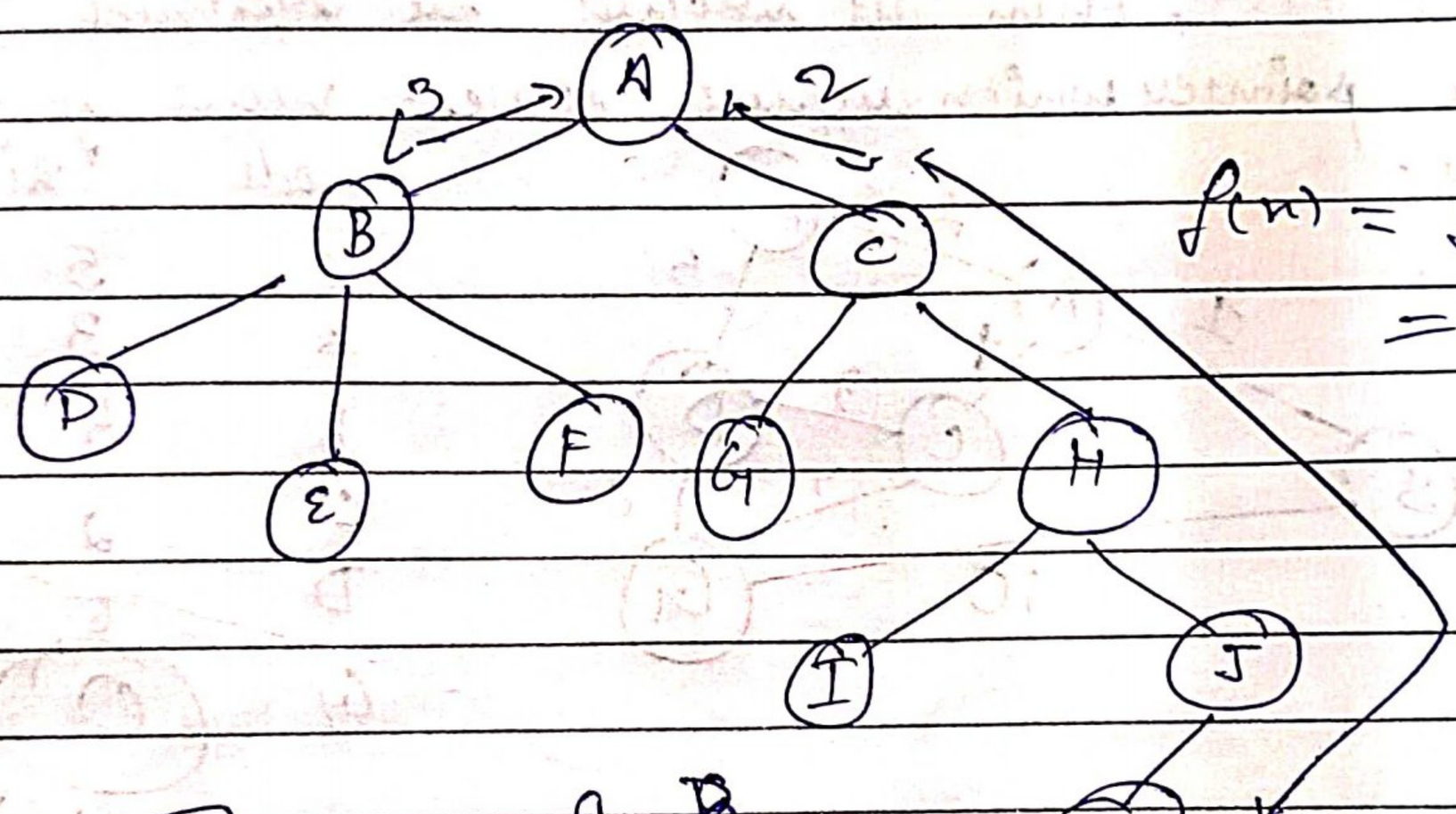
For the heuristic function calculation and add one point for every block which presented at proper position. subtract one point for every block which is sitting on wrong position. Initiave - state have score of 4 and goal state has score of 4.

A* algorithm.

$$f(n) = g(n) + h(n)$$

g(n) is actual lowest cost of the path from start point to given node.

h(n) is the estimated cost of cheapest path from given node to goal state.



$f(n) = g(n) + h(n)$

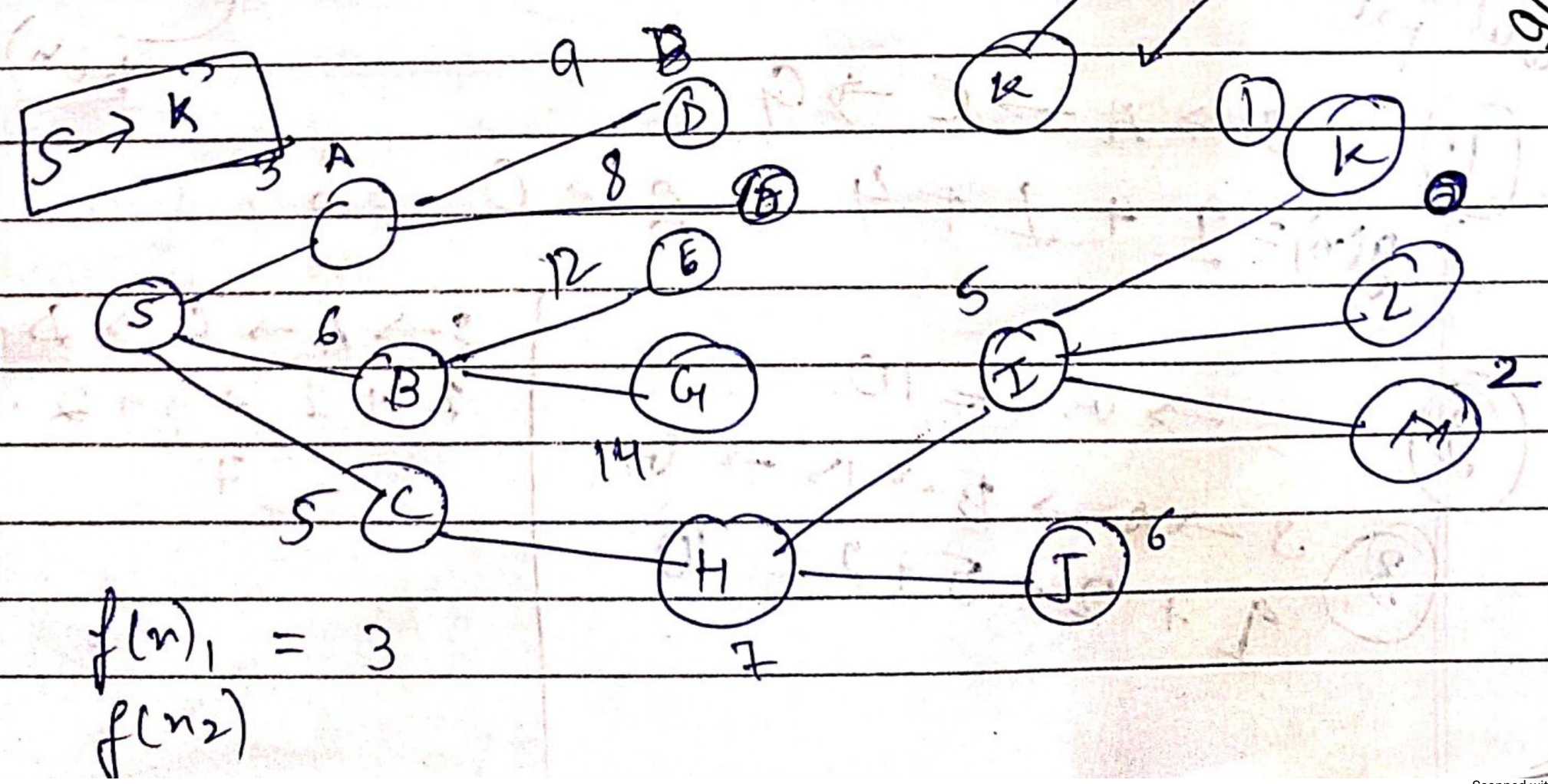
$= 2 + 10K$ distance

18

$f(n)_1 = 2/7$

$g(n) = 1$

$h(n) = 1$



$f(n)_1 = 3$

$f(n_2)$

Date : / /
Page No.

A* Algorithm

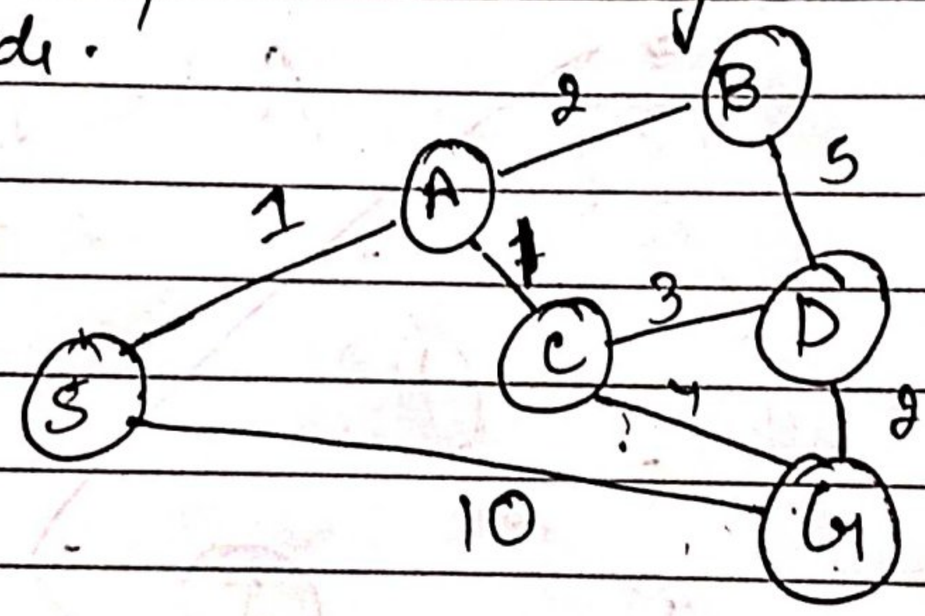
1. Place starting node in OPEN LIST

2.) Check if the OPEN LIST is empty or not.
If list is empty then return failure & stop.

3.) If ~~open~~ OPEN LIST is not empty then select node from the OPEN LIST which has smallest value of heuristic function ($g(n) + h(n)$) if node n is goal state then return success & stop.

4.) Otherwise Expand node n and generate all of it successor and put n into CLOSED LIST for each successor of n check whether n is already in OPEN or CLOSED LIST. If not then compute function for n and place into OPEN LIST

5.) Else if node n is already in OPEN or CLOSED LIST, then it should be attached to back pointer which reflect lowest value of g(n) node.



Node	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

h(n)

shortest path

1) $S \to A \to C \to G$
$g(n) = 1 + 1 + 4 = 6 + 0 = 6$

2) $S \to G = 10$

3) $S \to A \to B \to D \to G$
$1 + 2 + 5 + 2 = 10$

$S \to A \to C \to D \to G$
$1 + 1 + 3 + 2$.
7

Advantages / disadvantages

- This algorithm will help to solve the complex algorithm.

Date : / /
Page No.

disadvantages

- It does not introduce always shortest path sometimes due to heuristic function evaluation.

Space & time complexity = b^d.

17/10/22

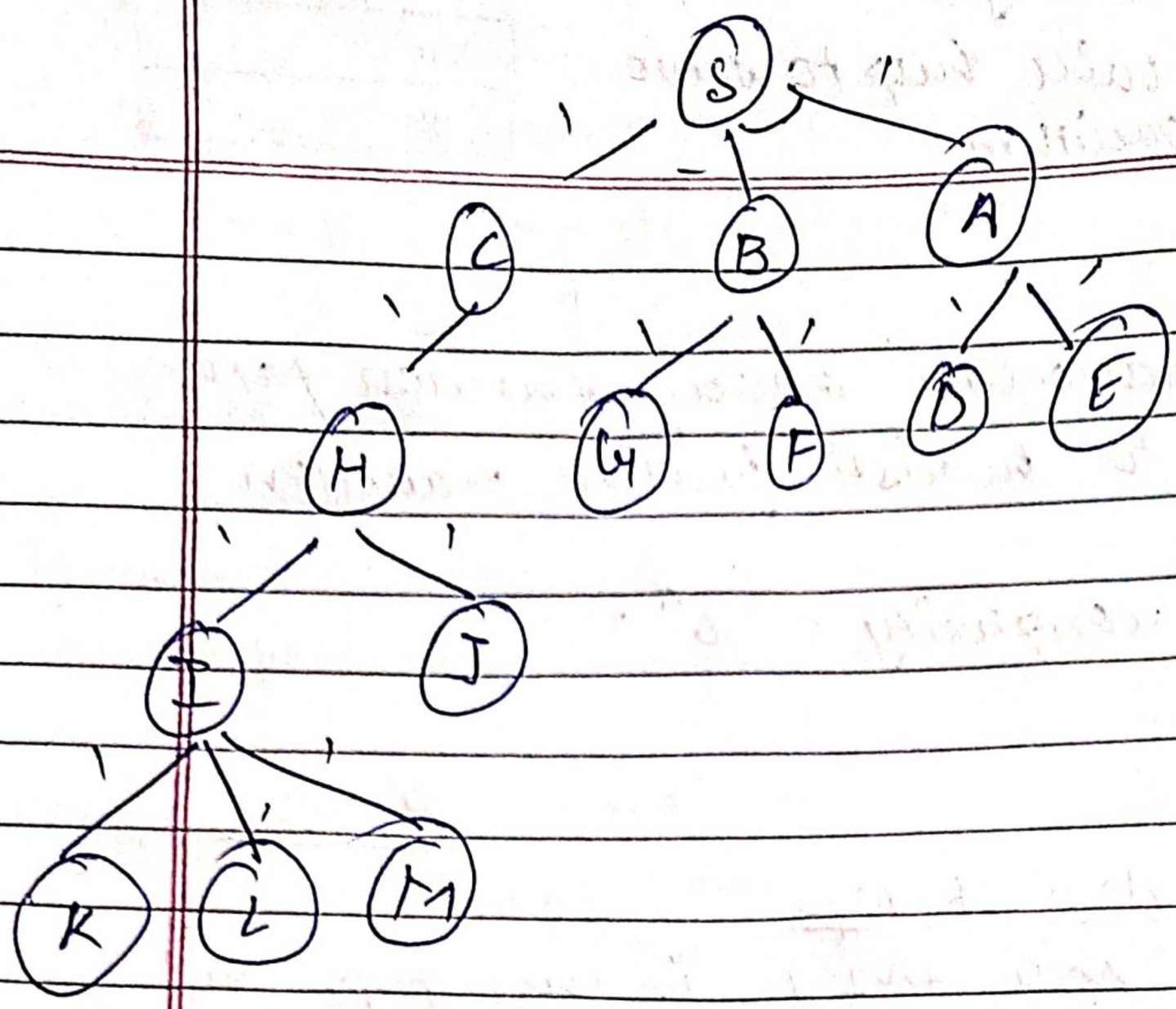
AO* AND OR Algo

1.) Given the goal node INIT in the graph G
Evaluate ~~A*~~ 'h' at INIT.

2.) Repeat
a.) Trace the arc from node INIT If any such exists and select one of the unexpanded nodes, named node the occur on the path for expansion.
b.) If node cannot expanded then assign FULILITY as 'h' value of NODE indicating not solvable.
Else go to successor.

Begin
i) Append successor to graph G.
ii) If successor is a terminal node Then label it solved and set it is 'h' value 0.
iii) If successor is not terminal node, Then estimate 'h'.

Date : / /
Page No.



18/10/2022

Constraint satisfaction -

- Constraint can be used to check the correctness of partial solution.
- To calculate some parameters using given constraint
- To choose some parameters and fix the value of that for further evaluation.
- The first two constraints can be envaluted using general software method i.e., called constraint proposition.
- ~~The~~ If we evaluate first two constraints, third constraint is automatically powerful.
- The constraints are may be of two types -

1.) Those given as a part of problem domain.
2.) Those which are assumed and may be changed later using backtracking.

m_{11}	m_{12}	m_{13}
m_{21}	m_{22}	m_{23}
m_{31}	m_{32}	m_{33}

2	7	6
9	5	1
4	3	8

magic square

$m_{11} + m_{12} + m_{13} = 15$

$m_{21} + m_{22} + m_{23} = 15$

$m_{31} + m_{32} + m_{33} = 15$

$m_{11} + m_{21} + m_{31} = 15$

$m_{12} + m_{22} + m_{32} = 15$

$m_{13} + m_{23} + m_{33} = 15$

$m_{11} + m_{22} + m_{33} = 15$

$m_{13} + m_{22} + m_{31} = 15$

Scanned with CamScanner

Date: / /
Page No.

1) Propagate available constraints
first set OPEN is set as all object which must have value assigned to them
In a complete solution, then do until an inconsistency is detected or until OPEN is empty.

1.1 Select an object from OPEN.
Strengthen as much as possible with the set of constraints which apply on object.

1.2 If this set is different from the set which was assigned the last time to object was examined, then add to OPEN all object which ~~is~~ have any constraints with object.

2.) If the union of the constraints discovered by above solⁿ is defined solⁿ, then quit.

3.) If the union of constraint discovered is contradiction then return failure.

4.) If none of above action occur then it is necessary to make guess at something in order to proceed to do this loop until a solⁿ is found or all possible solⁿ.

4.1. Select an object whose value is not yet determined and select a way strengthening the constraints on that object.

4.2. Recursively invoke constraints satisfaction with the current set of constraints augmented by strengthening constraint just selected.

This algo is generated to apply it in a particular domain requires the use two kinds of rules.

1.) which defines the way constraints may be propagated.

2.) which suggest guesses are means.

$$\begin{array}{r} C_4\ C_3\ C_2\ C_1 \\ S\ E\ N\ D \\ M\ O\ R\ E \\ \hline M\ O\ N\ E\ Y \end{array}$$

$M = 1$ $\quad O = 0$ or $1 \Rightarrow O = 0$

$S = 3$ or 9 $\quad N = E$ or $E+1 \rightarrow N = E$

$C_2 = 1$ $\quad N + R > 8$ $\quad E <> 9$

Statement of constraints.

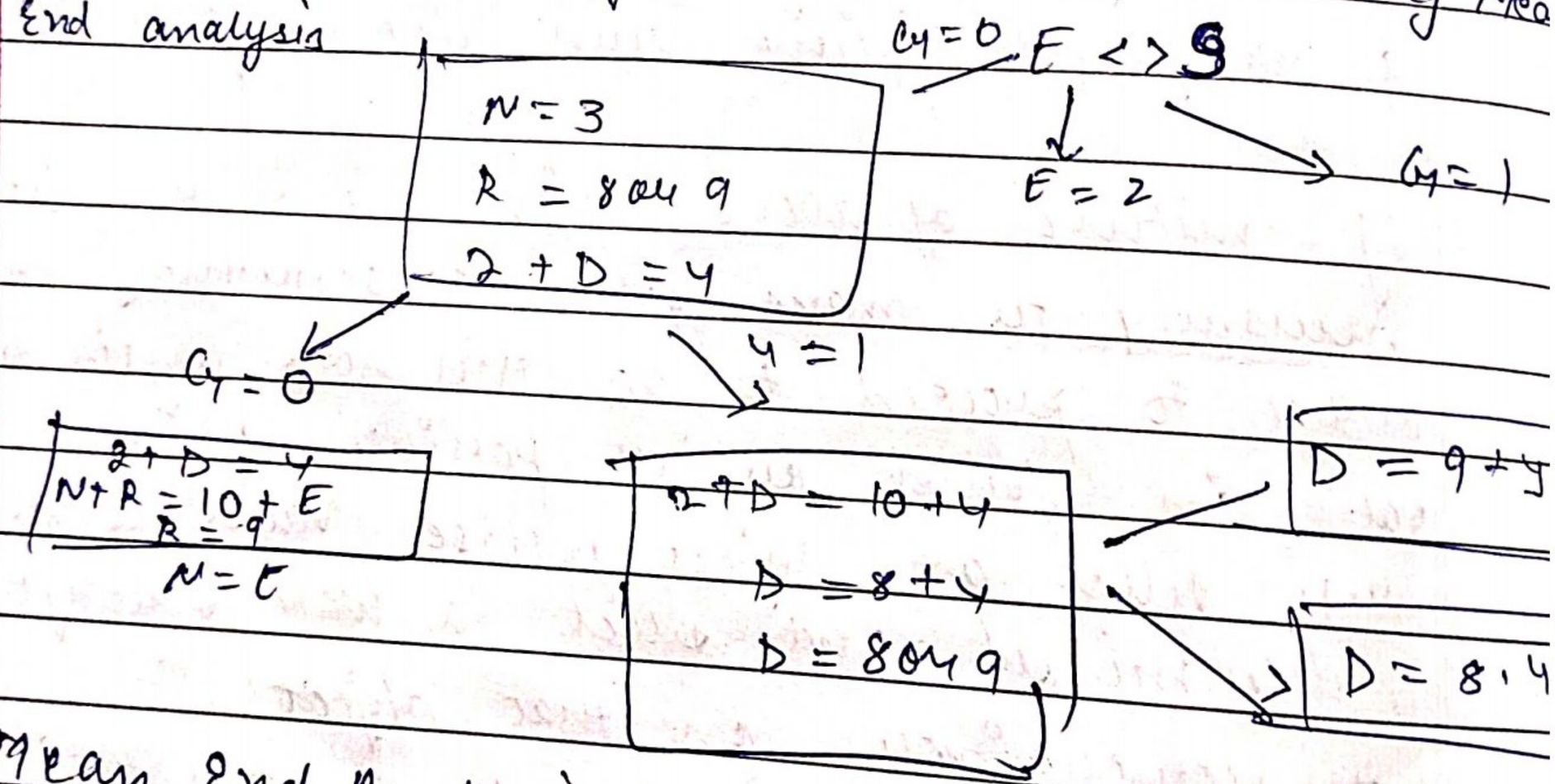
1.) values are to be assigned from 0 to 9.

2.) No two letters should assign same value.

3.) If same letters occurs more than one it must be assigned same value.

4.) The sum of digits sum are to given problem

In this the search searching technologies, searching direction either forward or backward using Mea End analysis



Mean End Analysis.

Till here the searching technology only same is an direction either forward or Backwa Mean end analysis using bidirectional search means in this technique part of problems a independently solved using forward & Backward search simultaneously then the sub problems are joined to obtain the compute sol^n.

Date : / /
Page No.

- It evaluates the difference b/w current & goal state and one ~~[illegible]~~ such a difference is identified an operator that can reduce the difference must be found. Be applied backward chaining in which operators are selected then subgoals are setup to establish pre condition of the operator this is called operator sub goals.

→ There are certain different initial condition on which that rules & operator can be applied. These are called precondition.

- The resulting generated state after rules are applied are called post condition.
- A table that keeps track of diff. b/w current state & goal state is called difference table.

Ex. of mean analysis :-

GPS = MEA

STRIPS = Global planner - MEA.

★ steps in MEA :

1.) Involves detection of difference b/w current & goal state.
2.) Once difference identified then found an operator for reducing difference.
3.) Some time operator cannot applied directly to current state.
4.) sub-problem of getting to state where operator applied.
5.) Operator may not give result in goal state
6.) 2nd ~~stat~~ sub-problem of getting new state from goal state.
7.) MEA process applied recursively.

Transform A to B

Reduce diff. b/w A & B leaving out A'

Translate A' to B

★ Three primary goal of MEA

1.) Transform. 2.) Reduce 3.) Apply

31/10/22

(PAI)

unit-2

Knowledge

Date: / /
Page No.

→ facts
→ concepts
→ Theories
→ Procedures
→ Relationship b/w them.

Knowledge based limited to specific domain which i) can reason
ii) Make decision
(iii) Make jugement.

Knowledge is a collection of facts, organized in some manner so that the information is meaningful.

Knowledge is central to intelligence we need it to
1.) understand the language.
2.) Make decision
3.) Reagination of object.
4.) Interpret the situation
5.) Plan strategies.

→ Types of Knowledge
1.) Declarative Knowledge
2.) procedural knowledge.

→ Declarative knowledge - representation of facts that tell us about a situation.

→ procedural knowledge - represent action according to the situation.

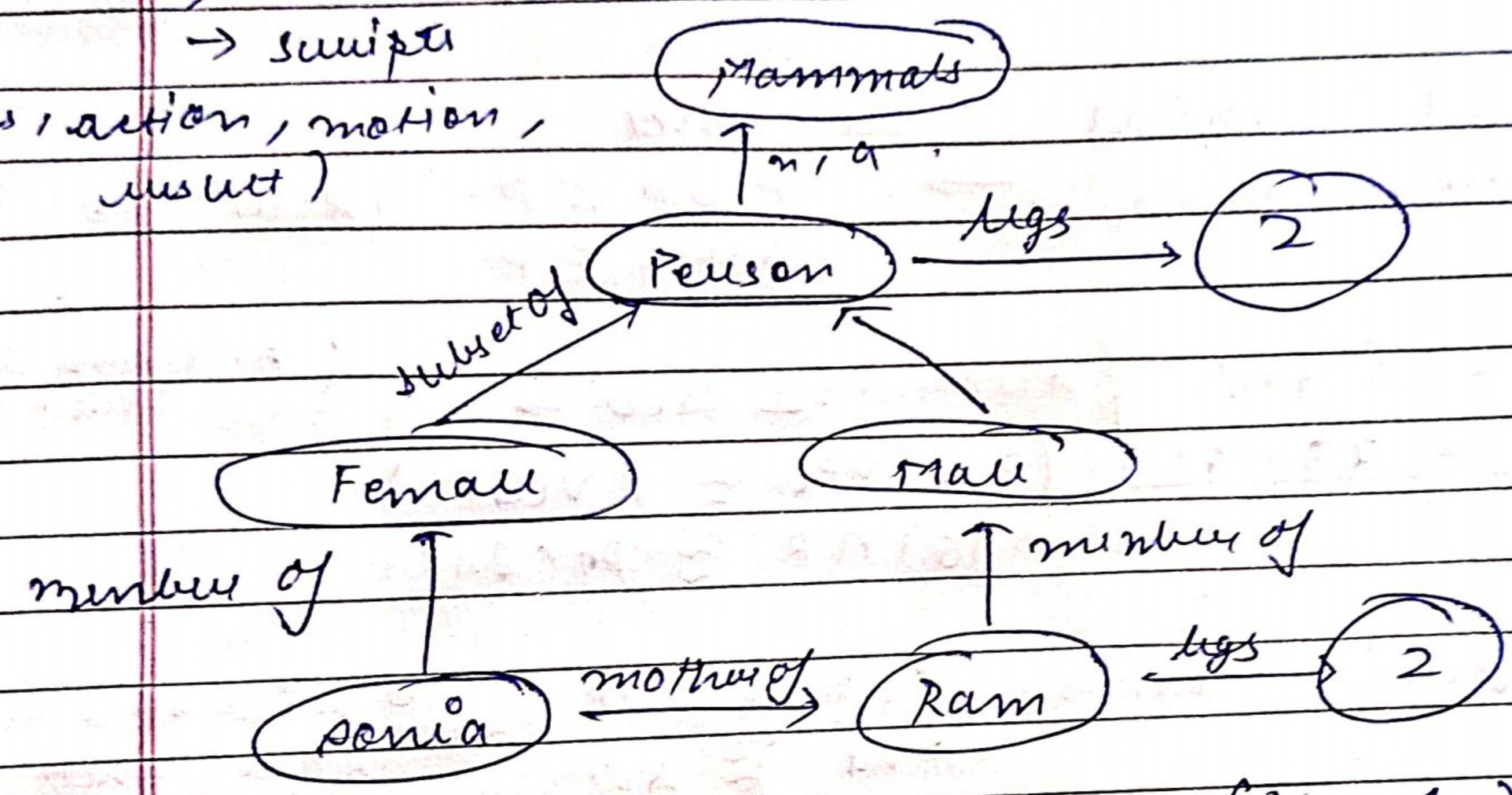
01/11/22

Date : / /
Page No.

Logic → Syntax
→ Semantic

Propositional logic (T, F) Predicate logic (V, F)

→ Rules
→ Semantic Net
→ Frame (slots, fillers)
→ Scripts
(Roles, action, motion, result)



Logic ∧ (AND), ∨ (OR), → (implies)

P	Q	~P	~Q	P∧Q	P∨Q	P→Q	P↔Q
T	T	F	F	T	T	T	T
T	F	F	T	F	T	F	F
F	T	T	F	F	T	T	F
F	F	T	T	F	F	T	T

P → Q ⇒ (~P ∨ Q)

P ↔ Q → (~P ∨ Q) ∧ (~Q ∨ P)

P	(~P ∨ Q)	(~Q ∨ P)
T	T	T
F	F	T
F	T	F
T	T	T

03/11/22 Properties of propositional statement

Date: / /
Page No.

Predefined gates

1.) satisfiable

2.) Tautology (T T T T) $(p \vee \sim p)$

3.) contradiction F F F F $(p \wedge \sim p)$

4.) Contigent: $nT = nF$ $T = F$ mean $(p(T) \sim\sim p)$

p	$\sim p$	$p \vee \sim p$		
T	F	T	T	T
F	T	T	F	F

Standard logical equivalence

Idempotent laws $\rightarrow$ $p \vee p \approx p$ (similarly)

$p \wedge p \approx p$

$A + (B + C) = (A + B) + C$

$A(B + C) = AB + AC$

Association law $\rightarrow$ (Addition is valid)

$(p \vee Q) \vee R \approx p \vee (Q \vee R)$

$(p \wedge Q) \wedge R \approx p \wedge (Q \wedge R)$

Commutative law $\rightarrow$ $p \wedge Q \approx Q \wedge p$ (Not bi conditional stat only CL applied)

$p \vee Q \approx Q \vee p$

$p \rightarrow Q \approx Q \rightarrow p$

distributive $\rightarrow$ $p \wedge (Q \vee R) \approx (p \wedge Q) \vee (p \wedge R)$

$p \vee (Q \wedge R) \approx (p \vee Q) \wedge (p \vee R)$

De Morgan law

$\sim(p \vee Q) \approx \sim p \wedge \sim Q$

$\sim(p \wedge Q) \approx \sim p \vee \sim Q$

Conditional or Implication Law -

$p \rightarrow Q \approx (p \rightarrow Q) \wedge (Q \leftrightarrow p)$

contraposition $\rightarrow$

$p \rightarrow Q \approx \sim Q \rightarrow \sim p.$

Biconditional

$p \leftrightarrow Q \approx (p \rightarrow Q) \wedge (Q \rightarrow p)$

$\sim Q \rightarrow \sim p$

$\sim(\sim Q) \rightarrow \sim p$

$Q \rightarrow \sim p$

Date : / /
Page No.

distributive law.

P	Q	R	Q∪R	P∩(Q∪R)	P∩Q	P∩R
T	T	T	T	T	T	T
T	F	T	T	T	F	T
F	T	T	T	F	F	F
F	F	F	F	F	F	F

P	Q	R	Q∪R	P∩(Q∪R)	P∩Q	P∩R	(P∩Q)∪P∩R
T	T	T	T	T	T	T	T
T	T	F	T	T	T	F	T
T	F	T	T	T	F	T	T
T	F	F	F	F	F	F	F
F	T	T	T	F	F	F	F
F	T	F	T	F	F	F	F
F	F	T	T	F	F	F	F
F	F	F	F	F	F	F	F

De Morgan Law

P	Q	P∪Q	~(P∪Q)	~P	~Q	~P∩~Q
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

Contraposition

$P \to Q$ $\qquad$ $Q \to \sim P$

P	Q	~P	~P∪Q	~Q	Q∪~P
T	T	F	T	F	T
T	F	F	F	T	F
F	T	T	T	F	T
F	F	T	T	T	T

Theorem ~~Rules~~ 7 meaning in propositional
→ semantic (mean of sentence) logic.

→ syntactic

Semantic

Forward chaining | Backward chaining

p	q	r	$p \to q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

then

1.) $p, q \Rightarrow p \cap q$

2.) $p, p \to q \Rightarrow q$ (Modus Ponens Rule)

3.) $\sim p, p \vee q \Rightarrow q$ (rule of disjunctive inference)

4.) $\sim p, p \to q \Rightarrow \sim p$ (Modus Tollens Rule)

5.) $p \vee q, p \to r, q \to r \Rightarrow r$

6.) $p \to r$
$q \to r \Rightarrow p \to r$ (chaining Rule)

7.) $p \to q$
~~$q \to r$ $p \to r$ (chain~~ $q \to r \Rightarrow r$ (Modus Ponens & chaining

8.) $p \vee (q \cap \sim q) \Leftrightarrow p$

9.) $p \cap (q \vee \sim q) \Leftrightarrow p$

10.) $p \to q \Leftrightarrow \sim p \vee q$

11.) $\sim (p \to q) \Leftrightarrow p \cap \sim q$

12.) $p \leftrightarrow q \Leftrightarrow (p \to q) \cap (q \to p)$

13.) $p \to q \Leftrightarrow \sim q \to \sim p$

→ $p \to q \Leftrightarrow \sim p \vee q$

→ $\sim(p \to q) \Leftrightarrow p \wedge \sim q$

→ $p \leftrightarrow q \Leftrightarrow (p \to q) \wedge (q \to p)$

→ $p \to q \Leftrightarrow \sim q \to \sim p$

7 NOV

FIRST ORDER LOGIC ⇒ When ans are not True or false.

- OBJECTS
- RELATION
- FUNCTION
- relation

dog ( )

cat (,) relation

- functions animal :-

⇒ $\wedge, \vee, \sim, \to$

OR

$\forall x \; man(x) \to drink(x, coffee)$

Some boys are intelligent

$\exists x \; boys(x) \to intelligent(x)$

→ The main connections from universal quantifier $\forall$ is a implication

→ The main connect for existial qualifier ($\exists$) and ($\wedge$)

Not all students like both Mathematics and Science.

$\forall(\sim X) \; student(X) \to likes(X, maths) \wedge likes(X, science)$

Free & Bound variable

→ $(\forall x)\ \exists (y)\ [P(x,y,z)]$
for all x there exists

y → Bounded variable
It comes under the scope of quantifier

→ $\forall x\ [A(x)\ B(y)]$

UNIFICATION → variable ki value kaha change →

man(Gaurav)
~ man(Gaurav)

man(Gaurav)
~man(Saurabh)

10/11/22

Resolution →

Resolution is a procedure which gain its efficiency from the fact that its operates ~~from~~ on the statements that have been converted to standard form

To prove a statement, resolution attempts to show that the negation of statement produce a contradiction which is know as unsatisfiable statement

Procedural

~~Proposition~~ Knowledge → A representation in which the controlled information that is necessary to control ~~to~~ knowledge is embedded in knowledge itself.

~~Decl~~ Procedural

In Proposor knowledge we will get more than one value that ~~satisf~~ satisfy the predicate. In Procedural Proposor knowledge if only one value needed then the ans of question will depend upon order in which assertion examine during the search or implementation for a response.

Procedural knowledge uses for to find how to will complete the work

2) Declarative knowledge is a statement in which knowledge specified but the use to which the knowledge is to be even or ~~not~~ not.

So in the declarative representation we must have the program that explain what is to do with knowledge and how. So in the declarative knowledge implication statement define reasoning path and assertion provide starting point of that path, so they can work as if then else, like in the traditional programming

HORN CLAUSES →

The facts are represented in horn clauses

(i) Because horn clauses have a uniform representation which helps to the interpreter to implement the program

(ii) The logic of horn clauses is decided by

(iii) Before going to prolog programming we have to set horn clauses.

We will follow the following steps –

① If horn clauses contain NON NEGATIVE literals then it leave it as it

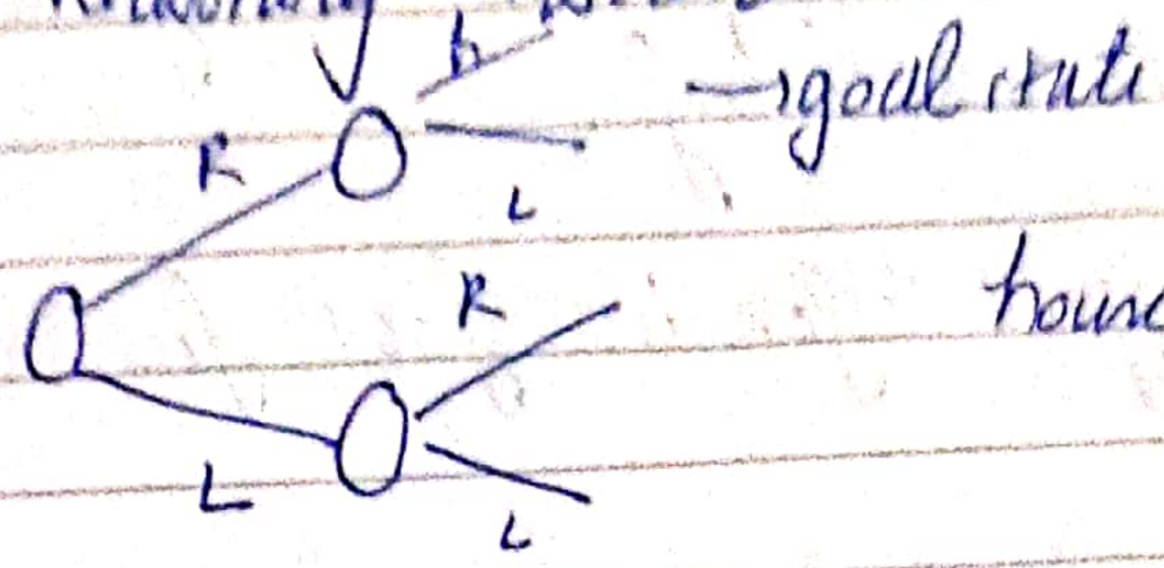
② Otherwise rewrite the horn clauses as the implication combining all negative literals into top to bottom approach and create a single +ve literal.

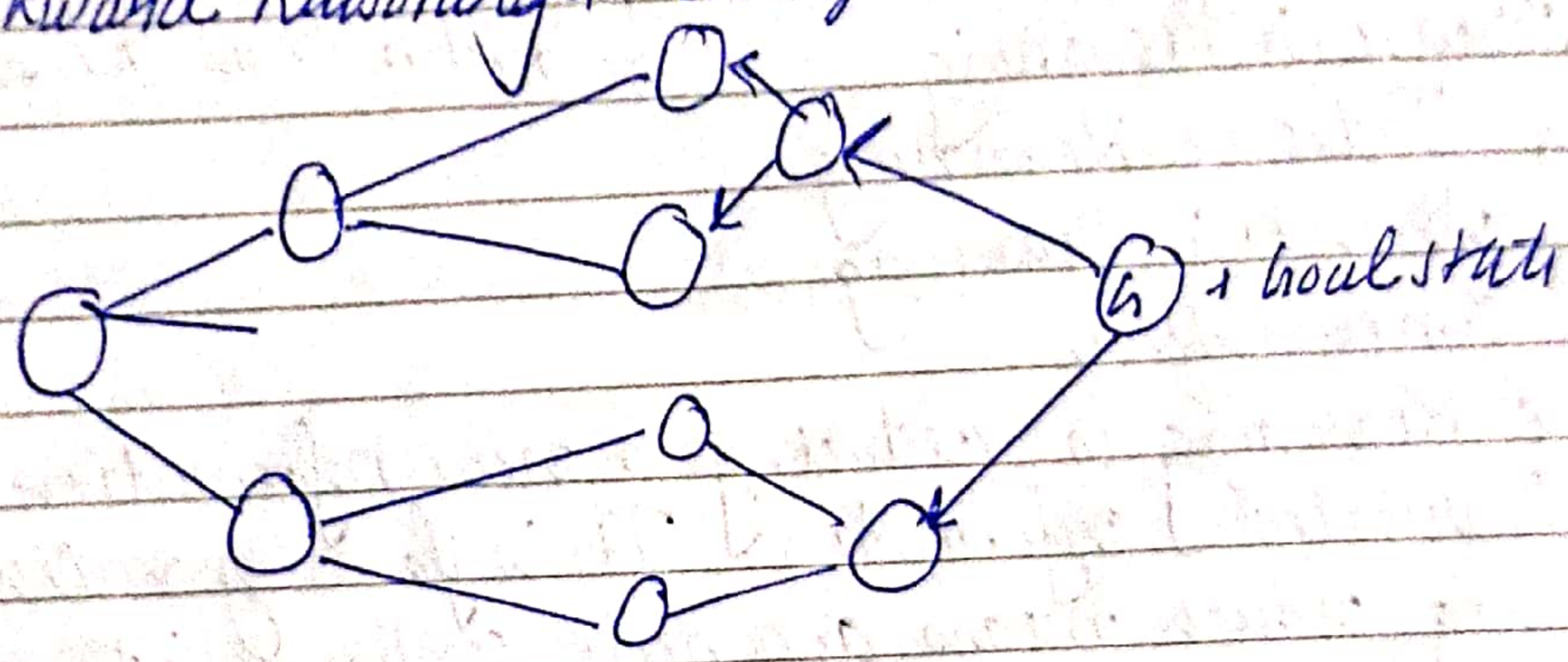
$\forall x \cdot cat(x) \wedge dog(x) \rightarrow pet(x)$

$pet(x) : cat(x)$

$pet(x) : dog(x)$ //

Forward Reasoning → search



home to college

Backward Reasoning : college to home