S. No.	Date	Title	Page No.	Teacher's Sign / Remarks

Problem defination

1) State of the problem → state space

X
O
–

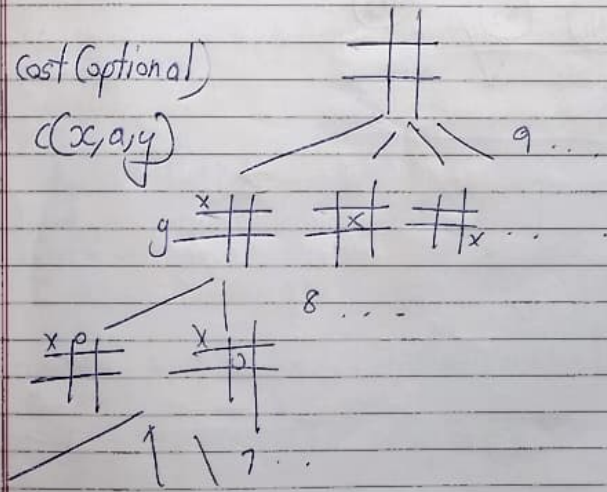
2) Initial state of the problem

3) Goal state

4) Action (valid moves)

5) Cost (optional)

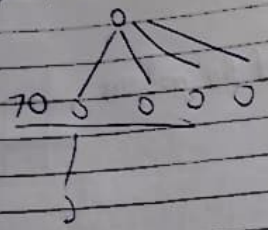
c(x, a, y)



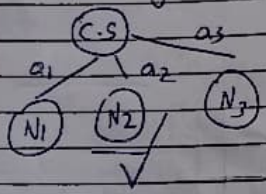
Searching algo

BFS
DFS
d = 10
l = 7

Depth limited search



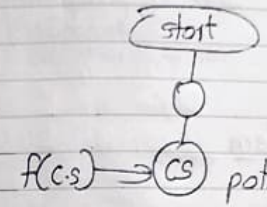
Practical
⇒ Heuristic Algo



Best first search

—A * Algo ⟶ $f(n) = g(n) + h(n)$

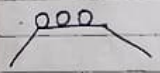
$g(n)$ ↓ Path cost
$h(n)$ ↓ heuristic



Simulated Annealing

It takes paras :-
① problem ② Schedule

inputs : c is a problem



② Schedue that takes care of mapping
local var current
next - next node
T ⇒ controls temp

current ← make_node (Initial State (problem))
for t = 1 to ∞
  T ← Schedule [t]
  if T = 0 then return current
  next ← randomly selected successor of current state

$\Delta E \leftarrow value[next] - value[current]$

If ΔE is greater than 0 current ← next

else current ← next with the probability $e^{\Delta E/T}$ acceptance

* Constraint satisfaction problem

) N variables $N_1 \ldots N_n$

$C_1 \ldots C_m$

Soln values to $N_1 \ldots N_n$

feasible Soln

optimal solution - needs to feasible & minimum no of colours

Cryptarithmatic problem

operand A to Z

assign number to each letters (0-9)

$C_5(1)\ C_4\ C_3\ C_2\ C_1$

C R O S S

R O A(6) D(1) S

D(1) A(6) N 4 E R

↓

assumed

$E = D + S + C_1$

C5 will be 1. So O = 1

> 9 ↑ $A = C + R + C_4$ (1/0)

A = 0, 2, 3, 4, 5

(13) (12) (13) (14) (15)

$C + R + C_4$ (0)

↓

$N < 9$

or

$C + R + C_4$ (1)

$n > 10$

$C_5(1)$	$C_4(0)$	$C_3(6)$	$C_2(0)$	$C_1(0)$
	C(8)	R(6)	O(2)	S(3) S(3)
	R(6)	O(2)	A(5)	D(1) S(3)
D(1)	A(5)	N(8)	4(7)	E(4) R(6)

A(5) C(8) D(1) E(4) 4(9) R(6) S(3)

N(0)

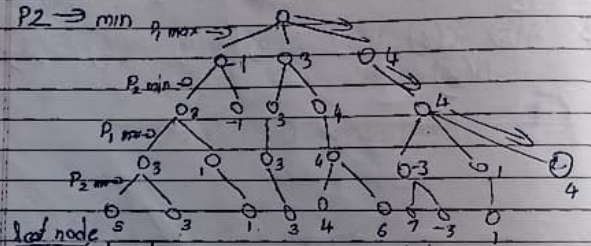
D O N A L D
G E R A L D

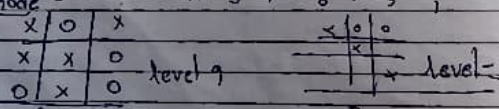
q > 0

Games max-min

2 players → objective function

P1 → max
P2 → min





utility function (h)
low

f(n)
higher on one side is worst

Alpha-Beta pruning

Prune a part of tree which gives better result

$-\infty = \alpha \rightarrow$ max player (high better for max player)
$+\infty \ \beta \rightarrow$ min (lower better for min player)

$\alpha \geq \beta \rightarrow$ prune the tree

Algorithm

AlphaBetaSearch (state) → returns action

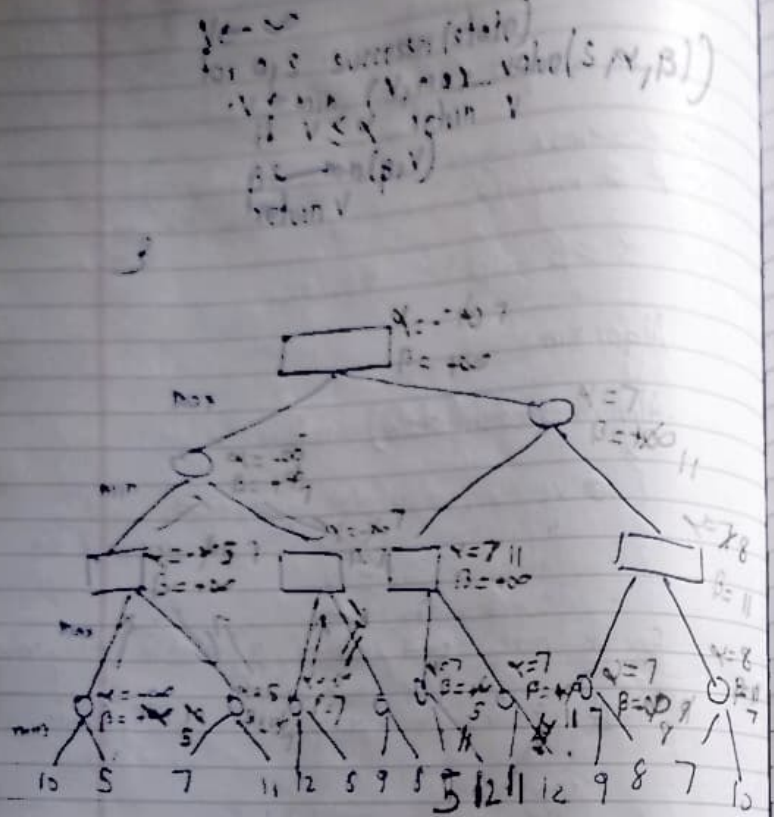
{
  v ← max_value (state, $-\infty$, $+\infty$)
  this returns the action in successor of the state with value v
  successor (state)
}

function max_value (state, α, β) → return utility value

{
  If terminal_test (state) return utility (state)
  else
  v ← $-\infty$
  for a, s successor (state)
    v ← max (v, min_value (s, α, β))
    if v ≥ β return v
    α ← max (α, v)
    return v
}

function min_value (state, α, β) → return utility value

{
  If terminal_test (state) return utility (state)
  else



Knowledge Representation

First order logic : Quantifiers
Predicate calculus : variables
: relations

$\forall x$
$\exists Y$

like (John, apple)
like (John, papaya) } facts
like (Joe, Mango)

like (x, y)
John
—
—

∴ Joe mango

All kings like crown

$\forall x \exists y$ King(x) $\wedge$ crown(y) $\wedge$ wears (x, y).

wear $(\forall x, y)$
↓
crown

stud (x) univesity (y) $\Rightarrow$ - mplication

all stud of gcc are affiliated to Gcu . all Gu entity not affiliated to GEC

$\forall x, y \exists z$ student (x) $\wedge$ college (y) $\wedge$ university (z) $\wedge$ affiliated (y, z) $\wedge$ study (x, y) $\Rightarrow$ associated (x, z)

↓
Implication

Premise $\Rightarrow$ conclusion

$A \Rightarrow B = \neg A \vee B$

$A \Leftrightarrow B = (A \Rightarrow B) \wedge (B \Rightarrow A)$
$(\neg A \vee B) \wedge (\neg B \vee A)$

$\neg(A \vee B) = \neg A \wedge \neg B$

facts
rules
connectives $\wedge \vee \neg$ facts { student(ravi, gec)
college(gec)
affiliated(gec, gu)

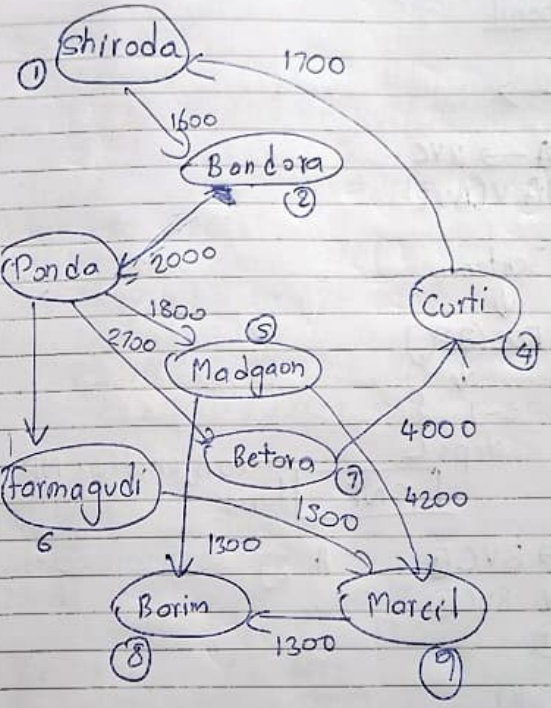
Unification
Resolution

student(x, y)
{x/ravi, y/gec}
fact student(ravi, gec)

$s(x, y) \wedge c(y) \wedge a(c, z) \Rightarrow associated(x, z)$

s(ravi, gec) x/ravi
c(gec) y/gec
a(gec, gu) z/gu

Exp 1
AIM - To implement BFS. DFS
Depth limited search.



x = joc

Resolution

↳

A → BVC
¬A V (BVC)

eg sentence
↓
FOL(BPL)
↓
6-7 steps { → conjunctive NF
CNF form

A → BVC (3) A (2) ∴
¬A V BVC (1)
¬C (3)

Pure B?
Assumption : ¬B → wrong ∴ B true

¬B ¬AVBVC
¬AVC A
C ¬C (3)
nil ← # assumption is wrong

steps
(1) Eliminate implecation

A → B
¬A V B

A ↔ B (A → B) ∧ (B → A)
(¬A V B) ∧ (¬B V A)

step 2
Reduce the scop of
¬ to single term
¬(a v b) = ¬a ∧ ¬b
¬(a ∧ b) = ¬a V ¬b
¬∀x : P(x) = ∃x : P(x)
¬∃x : P(x) = ∀x P(x)

step3
Standardize the variable such that each quantifier binds to a unique variable

∀x : P(x) ∀x : Q(x) ∀x father (x, X)
" ∀y : Q(y)

Step 4
Move all the quantifiers to the left of the formula without changing the relative orders
∀x P(x) ∧ ∀x : Q(x)
∀x P(x) ∧ ∀y : Q(y)

∀x ∀y P(x) ∧ Q(y)

5 Eliminate existential quantifier

Skolemisation → Skolem value
↘ " function

$\forall x \exists y$ President (S_1, x)

$\forall x \exists y$ father (y, x)

$\forall x \exists y$ father $(S_1(x), x)$

Step 6:
Eliminate universal quantifier
no extra rule as such

Step 7:
Convert the matrix to an conjunction of disjuncts

$(a \vee b) \wedge (c \vee d)$

Step 8:
Create a separate clause corresponding to each conjunct

$(a \vee c) \wedge (b \vee c)$

① Anyone passing a toc exam and winning a lottery is happy

② Anyone who studies or is lucky can pass the exam

③ John did not study but he was lucky

④ Anyone who is lucky wins a ~~bit~~ lottry

⇒ by using resolution prove weather john is happy

student(x) exam(y) passing(z)

① $\forall x$ pass(x, toc) $\wedge$ win(x, lottry) $\Rightarrow$ happy(x)

$\neg$(pass(x, toc) $\wedge$ win(x, lottry)) $\vee$ happy(x)

$\neg$(pass(x, toc)) $\vee \neg$(win(x, lottry)) $\vee$ happy(x)

① : i

② $\forall x \forall z$ studies(x) $\vee$ lucky(x) $\rightarrow$ pass(x, z)

$\neg$(study(x) $\vee$ lucky(x)) $\vee$ pass(x, z)

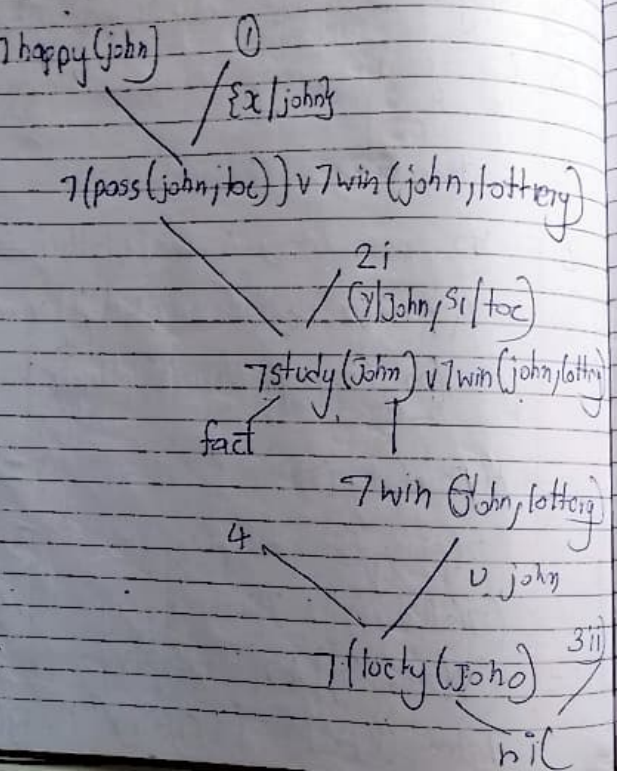
($\neg$(study(x)) $\wedge \neg$(lucky(x))) $\vee$ pass(x, z)

($\neg$(study(x)) $\wedge \neg$(lucky(x))) $\vee$ pass(x, S_1)

$\neg$study(x) $\vee$ pass(x, S_1)-2i) $\neg$lucky(x) $\vee$ pass(x, S_1)-2ii)

③ [i] ¬study(John) 3i)
[ii] lucky(John) 3ii)

∀v lucky(v) → win(v, lottery)
¬[lucky(v)] ∨ win(v, lottery) — 4)



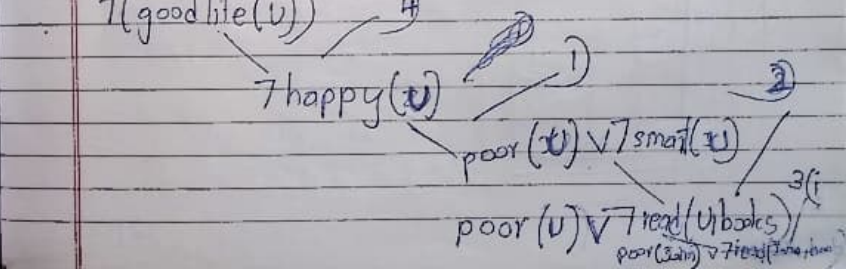
∴ john was happy

Q1) 1) All people who are not poor and are smart are happy
2) Those people who read books are intelligent
3) John reads the book and is wealthy
4) Happy people have a good life (rich)

By resolution prove if anyone be found with a good life

1) ∀x ¬poor(x) ∧ smart(x) ⇒ happy(x)
2) ∀x ¬(¬poor(x) ∧ smart(x)) ∨ happy(x)
¬poor(x) ∨ ¬smart(x) ∨ happy(x)

3) ∀y read(y, books) ⇒ intelligent(y)
¬read(y, books) ∨ intelligent(y)

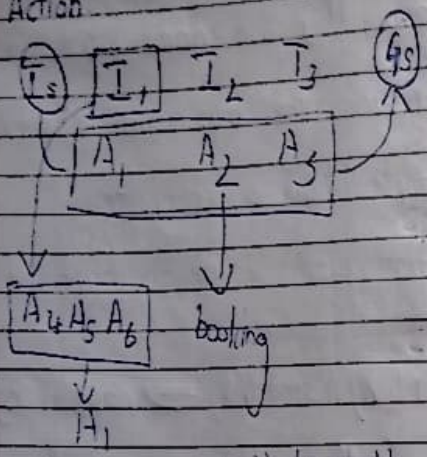


Planning

→ sequence of action to achive sequence of tasks

PDDL

Action



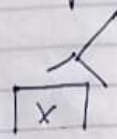
action: Generic form (should be reusable)

Precondition | effect

At

Action for robotic line

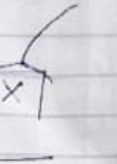
① Pickup(X)



PC
OT(X)
Armempty()
At(X)
clear(X)

Effect
~~OT(X)~~
~~Armepty()~~
holding(X)

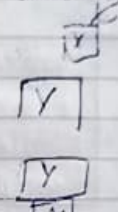
putdown(X)



PC
Tableempty()
holding(X)

effect
~~holding(X)~~
armempty(X)
OT(X)

stack(Y)



stack(X,Y)
PC holding(X)
clear(Y)

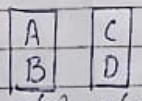
effects
~~holding(X)~~
armempty()
on(X,Y)

unstack(X,Y)

unstack(X,Y)
PC
armempty()
on(X,Y)
clear(X)

effect
holding(X)
~~armempty~~
~~on(X,Y)~~
clear(Y)

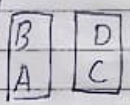
Initial



OT(B), OT(D)
on(A,B), on(C,D)
clear(A), clear(C)

Goal

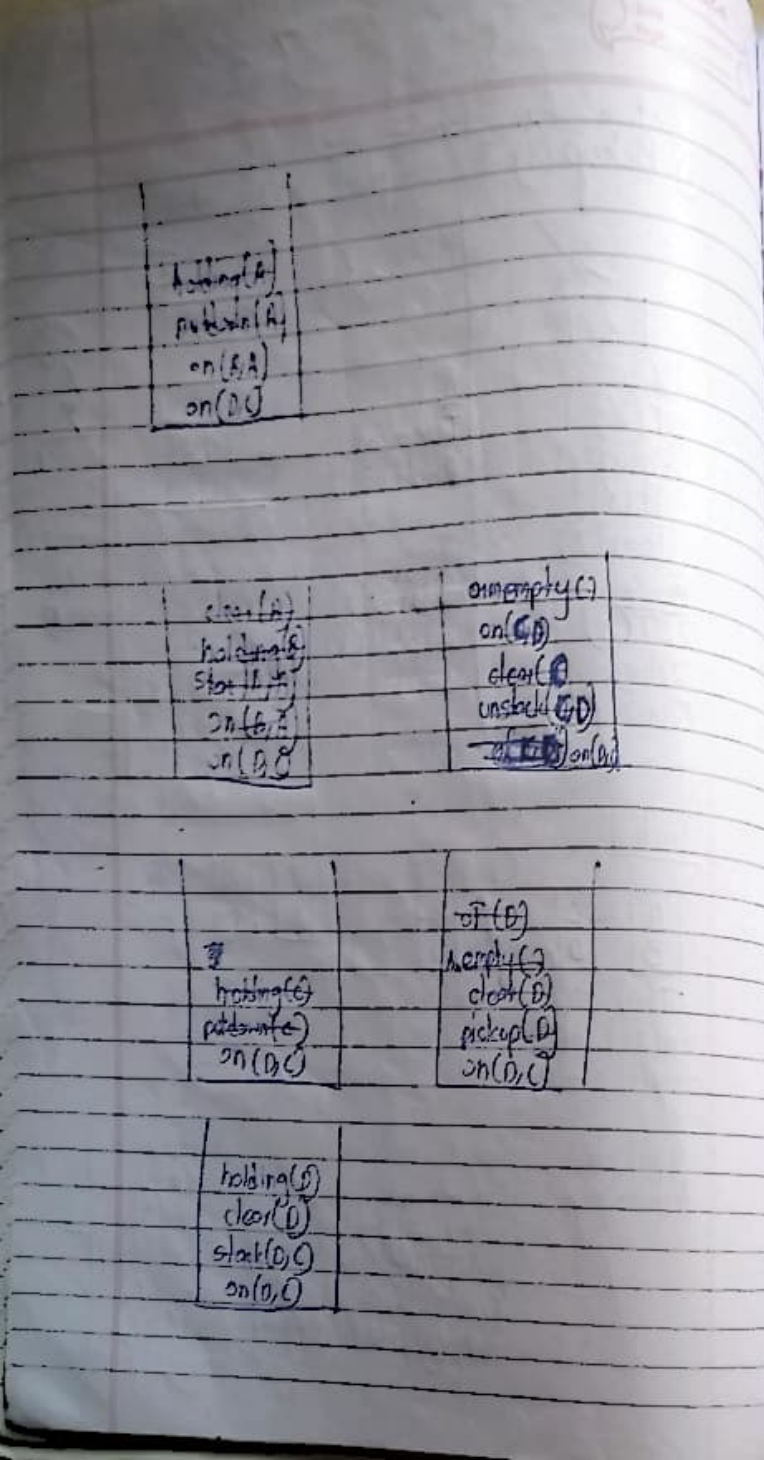
on(B,A)
on(D,C)

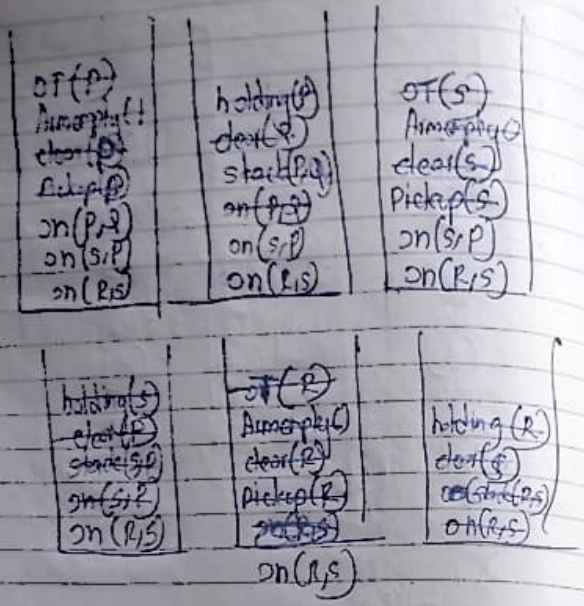


Goal stack
Armempty()

planning problem
~~unstack(A,B)~~
unstack(A,B), putdown(A)
pickup(B), stack(B,A)
(unstack(C,D), putdown(C)
pickup(D), stack(D,C)

Armempty ⇒
on(A,B)
clear(A)
unstack(A,B)
on(B,A)
on(D,C)


P
Q
R
S
on(P,Q)
on(Q,R)
OT(S)
OT(R)
C(P)
C(S)
armempty()
unstack(P,Q)
Putdown(P)
unstac(Q,R)
Putdown(Q)
pickup(R)
stack(R,S)
pickup(P)
stack(P,Q)
pickup(S)
stack(S,P)
pickup(R)
stack(R,S)
R
S
P
Q
on(P,Q)
on(S,P)
on(R,S)
OT(Q)
C(R)
holding(P)
Putdown(P)
unstack(P,Q)
OT(Q)
on(R,S)
on(S,P)
on(P,Q)
Armempty()
Pickup(R)
on(R,S)
on(S,P)
on(P,Q)
stack(P,Q)
on(R,S)
on(S,P)
on(P,Q)



Q

on(S)
on(R,S)
on(Q,R)
on(P,Q)
Aempty()

on(P)
on(Q,P)
on(R,Q)
on(R,S)

Forword & backward search

fact → goal

fact

run(x) → sweat(x)

run(John)

sweat(John)

planning problem

Cargo

C_1

C_2

Airport

A_1

A_2

goal

$A_1 \rightarrow C_2$

$A_2 - C_1$

Action — Pre

effect

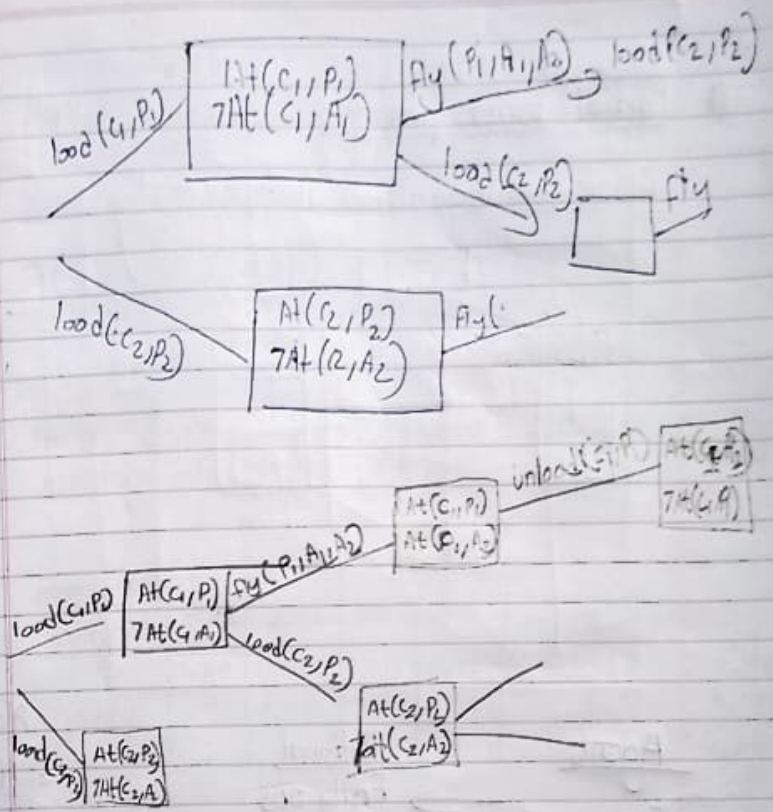
Load(C,P)
Pre: cargo(C) ∧ plane(P) ∧ At(P,A) ∧ Airport(A) ∧ At(C,A)

effect
In(C,P) ∧ ¬At(C,A)

Unload(C,P)
Pre: cargo(C) ∧ plane(P) ∧ In(C,P) ∧ At(P,A) ∧ airport(A)
effect: ¬In(C,P) ∧ at(C,A)

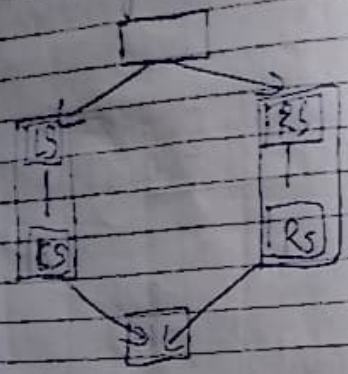
Fly(P,S,D):
Precondition: at(P,S) ∧ airport(S) ∧ airport(D) ∧ In(C,P) ∧ Cargo(C)
effect ¬at(P,S) ∧ at(P,D)

Initial
C(C1)
C(C2)
A(A1)
A(A2)
At(C1,A1)
At(C2,A2)
At(P1,A1)
At(P2,A2)
P(P1)
P(P2)



* Partial order planning

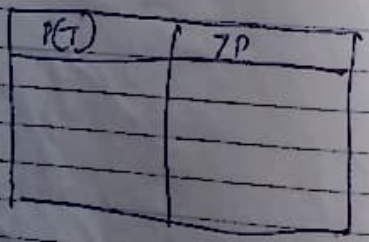
Goals → subgoals → independently



Agent → Rational → Fully obs / Partly → likelyhood → Probability

Environment

$P(a/b) = \frac{P(a \wedge b)}{P(b)}$



Partial order planning

$P(a/b) = \frac{P(a \wedge b)}{P(b)}$

In a cenario two things can occure cavity or gum problem

	TA		TTA		
	gum	~gum	gum	~gum	
cavity	0.108	0.012	0.072	0.008	0.2
~cavity	0.016	0.064	0.144	0.576	

0.108
0.092
0.200

$P(T/E) = \frac{P(T \wedge E)}{P(C)} = \frac{0.120}{0.2} = 0.6$

$P(T/C,4)$

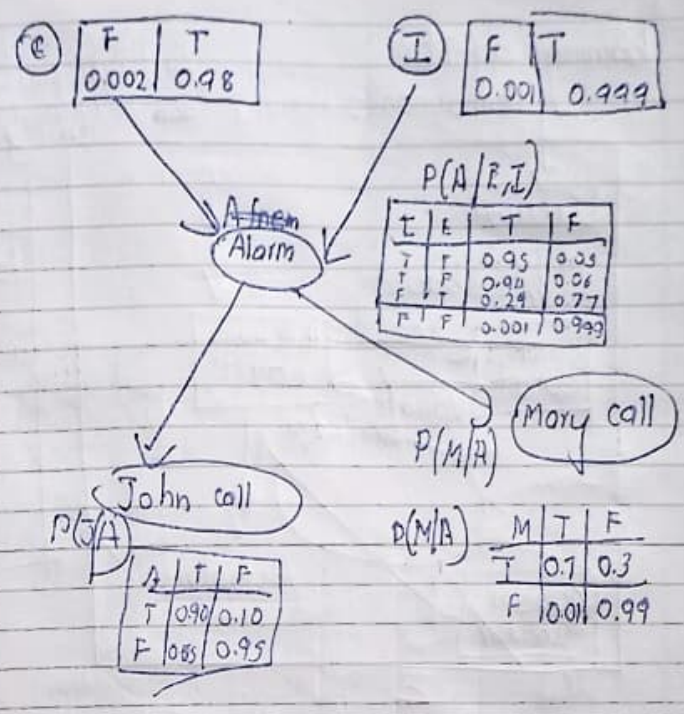
$P(C/T) = \frac{P(C \wedge T)}{P(T)} = \frac{P(Y/X)P(X)}{P(Y)}$

$= \frac{P(T/C)P(C)}{P(T)} \quad \frac{0.6 \times 0.2}{0.2} = 0.6$

Used

Consider an event of ringing of alarm because of an earthquake or because of an intrusion & in response either john calls of marry calls

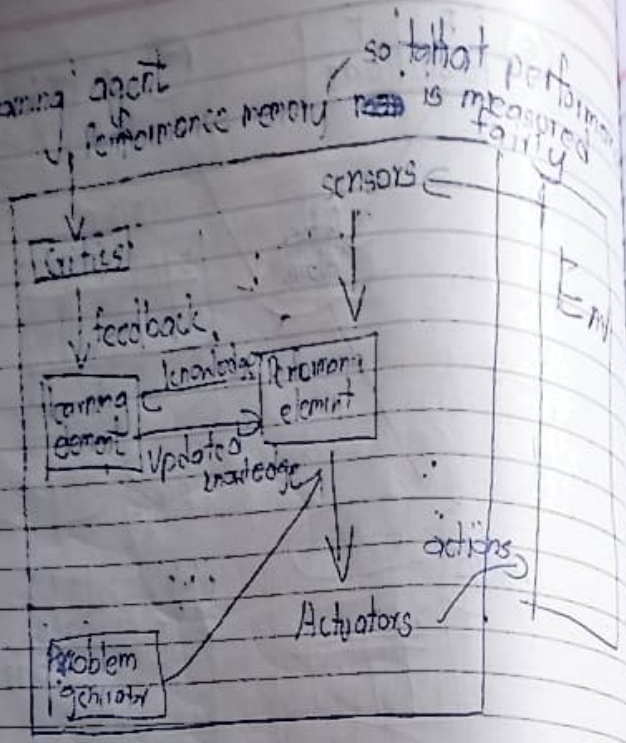
	John		Marry	
	Intru	Intrusion	Intru	7I
E				
7E				

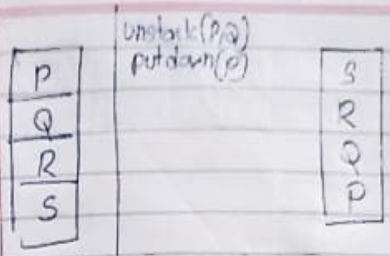


Learning → Updating KB

New modifing deleting

1) observation
2) practise → Rote learning
3) Experiments
4) Implementation





OT(S) Clear(P)
on(R,S)
on(Q,R)
on(P,Q)
Amempty()

OT(P)
on(Q,P)
on(Q,R)
on(R,S)

1

~~armempty()~~
~~clear(P)~~
~~on(P,Q)~~
~~unstack(P,Q)~~

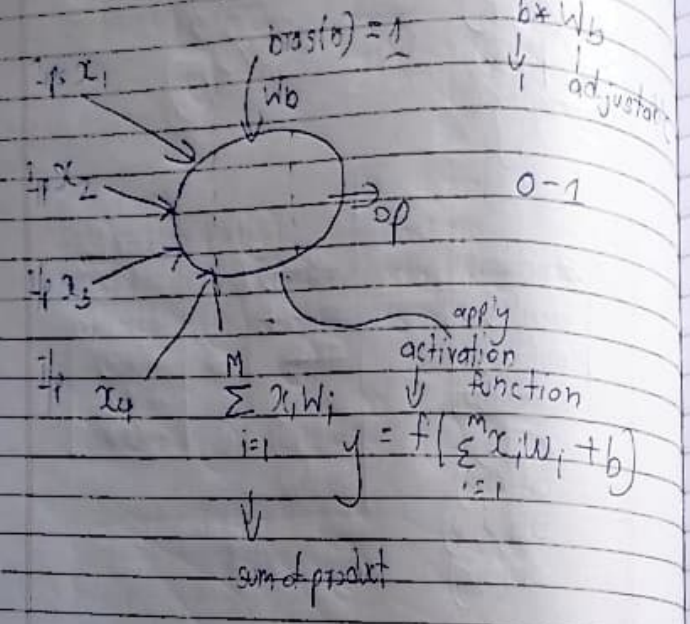
→

~~holding(P)~~
~~putdown(P)~~
~~OT(P)~~
on(Q,R)
on(R,S)
on(S,R)

→

armempty()
clear(Q)
on(Q,R)
unstack(Q,R)
on(Q,R)
on(R,S)
on(S,R)

OT(P)
on(Q,P)
on(R,Q)
on(S,R)



$b * w_b$

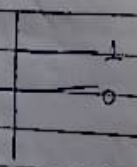
↓ adjustable

0-1

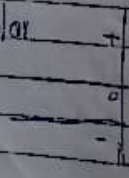
$\sum_{i=1}^{n} x_i w_i$

$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right)$

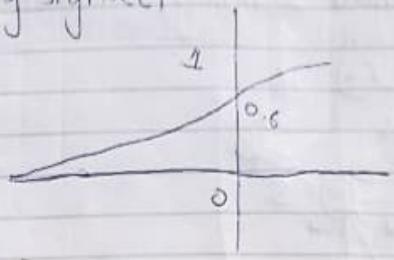
1) step



bipolar



log sigmoid

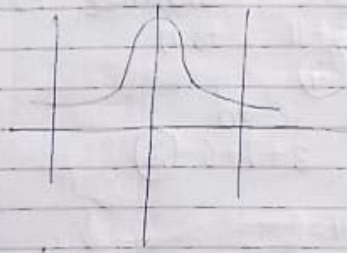


$y = \frac{1}{1+e^{-\lambda x}}$ $\lambda = 1$

$\sum x_i w_i + b$

tan sigmoid

Radial Basis func

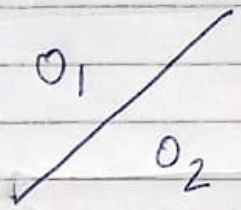


N/W

Unit 1
Planing problem
first half of unit 2

Neural Network

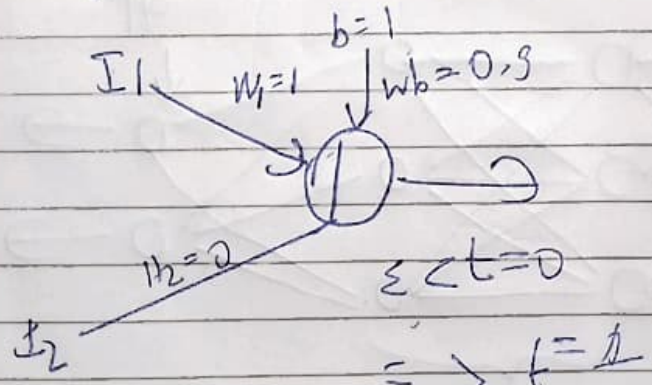
learn



Linearly seprable

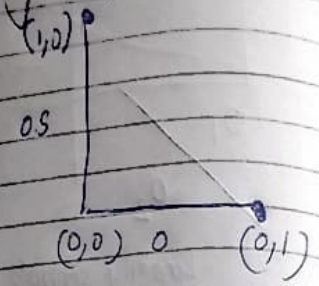
OR

I_1	I_2	o/p	Cond o/p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1



XOR is not linearly seprable

I_1	I_2	
0	0	0
0	1	1
1	1	1
1	0	0



Input

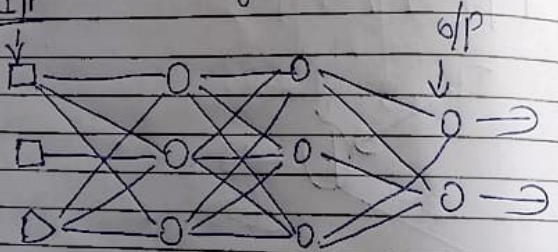
hidden ?? HW

Each layer

o/p

how many neurons

I/P



Hidden layers

By open close High low value