Bashundhara

Exercise Book

Write Your Future

TAHSIN

2020-2-60-112

✓ Agent :- is everything that can be viewed as perceiving it's environment through 'Sensor' & 'Acting' upon that environment through 'Actuators'.

ex - Human agent - eyes, ears, ... for sensor.
- hand, legs, mouth ... for actuators.

Robotic agent - camera, infrared range finder.
- various motors.

✓ PEAS : Performance measure, Environment, Actuators, Sensors

✓ Agent types - simple reflex : Only current situation
- Goal based : Reaching until the goal.
- Utility based : same as goal based but Provides extra component like best way to reach the goal.
- Learning based : ✓ AI
✓ Past experience also have learning capability.

Searching

- tells agent what to do!
- to achieve goals or to maximize our utility.
- why not Dijkstra? ⇒ D's algo only works on finite space.

✓ Problem defined by 5 states -

(i) initial state.
(ii) Actions.
(iii) Transition model result
(iv) Goal test.
(v) Path cost.

✓ search strategies - by picking the order of node expansion.
- it follow's - Completeness (is the sol exist or not!), time complexity (no of nodes generated), optimality, space.
(least cost) (max numb of nodes in store.)

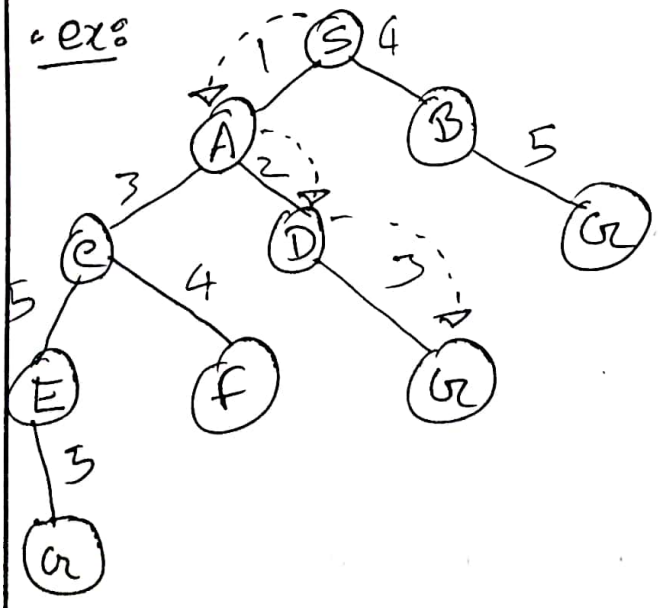
Types of search algo – (I) Blind (II) Heuristic Search.

- Blinds Strategies / Uninformed search – BFS, Uniform-Cost Search, DFS, Iterative deeping search, Bidirectional Search.

✓ Uniform Cost Search : • used for travelling a weighted tree or graph.

- Goal – find goal node which has lowest cost.
- (I) Priority Queue (II) lowest cost (III) BFS.
- Pros: in optimal

• ex:



Key → Value Index 0 1

LAB : DS: ✓Dictionary ✓List → S = ["x", "y"]

✓Tuple : • Value cannot be change as set.

• Faster

student = ("x", "y",)

✓Set : ① Curly braces ② set func

num1 = {1, 2, 3}

num2 = set([4,5,6])

num2.add(7)

func: ✓len()

✓append("x")

✓ insert(2, "OS")

index string

✓remove("OS")

✓sort()

✓reverse()

✓ pop() ← last item remove.

✓copy ✓count().

✓ index(2) → position

lab

OOP:

template – can use as,

✓ Class Student: ← blue print

roll = " "

gpa = " "

rahim = Student() ← Class Obj Create.

rahim.roll = 101 ← value

rahim.gpa = 3.5

Print (f"Roll: {rahim.roll}, GPA: {rahim.gpa}").

✓ function

Points to current obj

```
def display(self):
    Print {f"Roll: {self.roll}, GPA: {self.gp}"
```

```
rahim = Student()
rahim.roll = 101
rahim.gpa = 4.7
rahim.display()
```

or

```
def setvalue(self, roll, gpa):
    self.roll = roll
    self.gpa = gpa
rahim.setvalue(101, 3.7)
```

Construction: • __init__(self, roll, gpa)

↳ self. no need to func to set value
↳ obj তৈরি করার সময়ই value দিতে দিতে যাবে।

⇒ def __init__(self, roll, gpa):

self.roll = roll

self.gpa = gpa

rahim = student(101, 3.75).

exe (53)

Class - 6

The one Queue : • Priority queue

Inform search : we know the goal

• heuristic - a func that estimate how close a state in to a goal. (estimation)

✓ Greedy Search : • not optimal

✓ A* Search : • Combination of UCS & Greedy

• Uniform-cost orders by path cost.

• $f(n) = g(n) + h(n)$

• A* is optimal.

• Admissible Heuristics - • is admissible (optimistic)

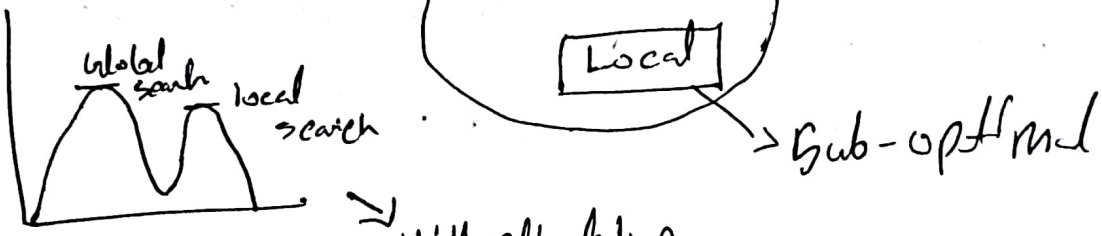
• estimate cost < true cost.

✓ Consistency of Heuristics : ✓

- admissible heuristic, Tree A* in optimal
- consistent " , Graph A* " "
- with $h=0$ (heuristic) UCS " "

} Tip

Local search



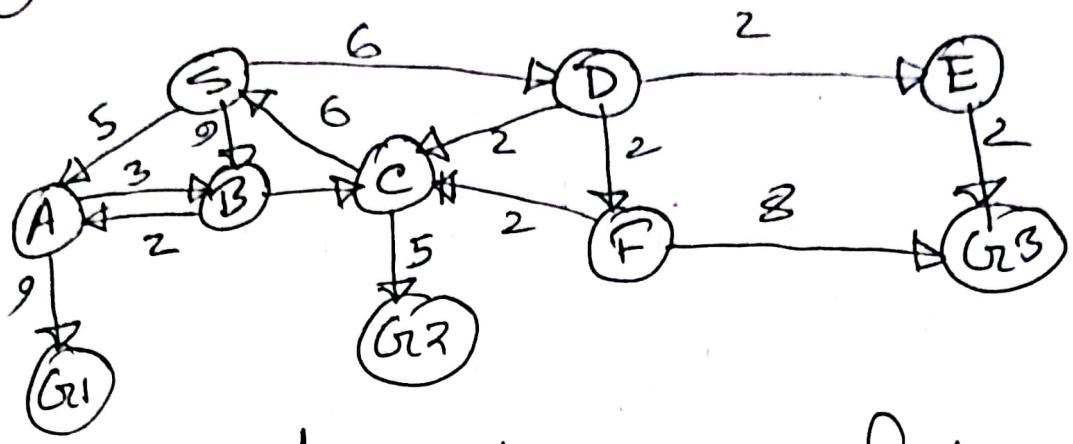
- Hill climbing
- simulated

✓ graph - pune de- obj

- BFS ಮತ್ತು UCS A* MAZE

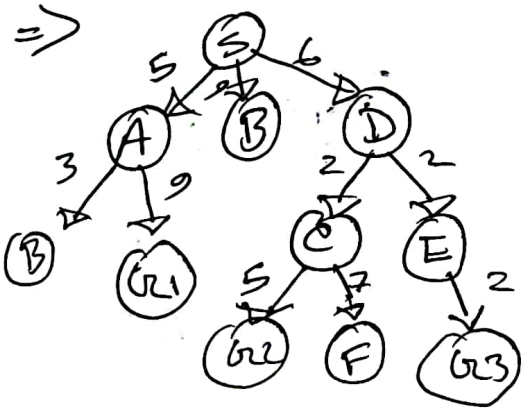
Uniform Cost Search:

(2)



We need to reach any one of the destination node (G1, G2, G3) starting from (S). Find the path from S to any of these destination node state with the least cumulative cost ?

=>



S → B , cost = 9

S → D , " = 6

S → A → B , " = 5 + 3 = 8

✓S → A → (G1) , = 5 + 9 = 14 .

S → D → E , = 6 + 2 = 8

S → D → C 6 + 2 = 8

✓S → D → C → (G2) = 13

S → D → C → F 15

S → D → E → (G3) 15

✓ DFS: LIFO

✓ Iterative deepening search:

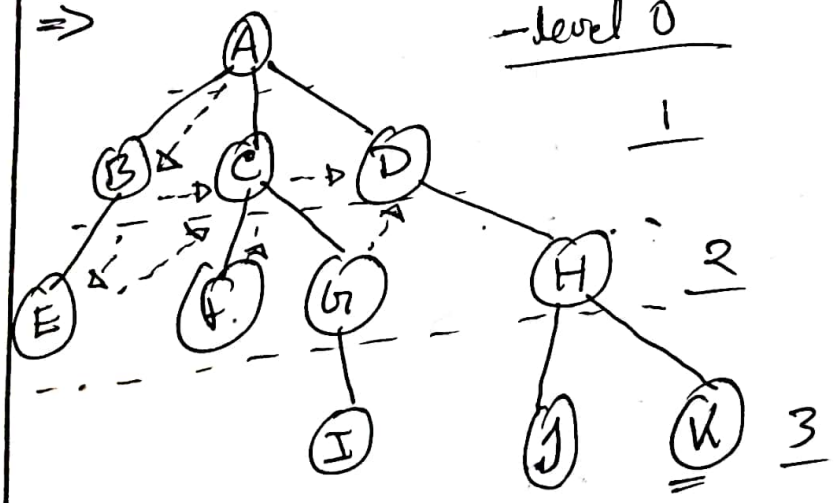
- To avoid the infinite dept problem of DFS
- Combination of BFS & DFS → Pro: less memory.

{ Pro - it will find goal node at any way
  cons - memory consuming.

so, Iterative finds goals at less memory.

- checks level-wise.

=>



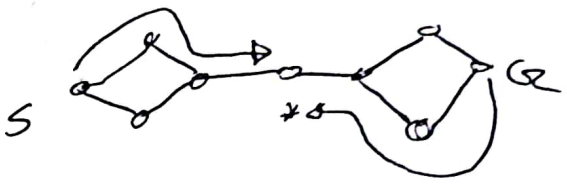
here, goal node = (K)

Depth or level	IDS
0	A
1	A B C D
2	A B E C F G
	D H
3	A B E C F G
	I D H J (K)

NOTE: * cons - Same node comes frequently.

Bi-Directional Search:

- Two simulation search from ~~one~~ an initial node to goal and backward from goal to initial, stopping when two meet.
- Time Complexity: $2\left(b^{d/2}\right)$ (d: depth, b: breadth)



- Complete in BFS
- Not in dfs.

Informed / heuristic search :

- Best-first , A^*
- $f(n) = g(n) + h(n)$
  - $f(n)$ → estimate of total cost
  - $g(n)$ → path cost for n nodes.
  - $h(n)$ → heuristic func. / estimate of cost to goal from n node.

Hurestics Search : note: a good heuristic can reduce the search process.

ex: 8 puzzle prob.—

1	2	3
	4	6
7	5	8

start state

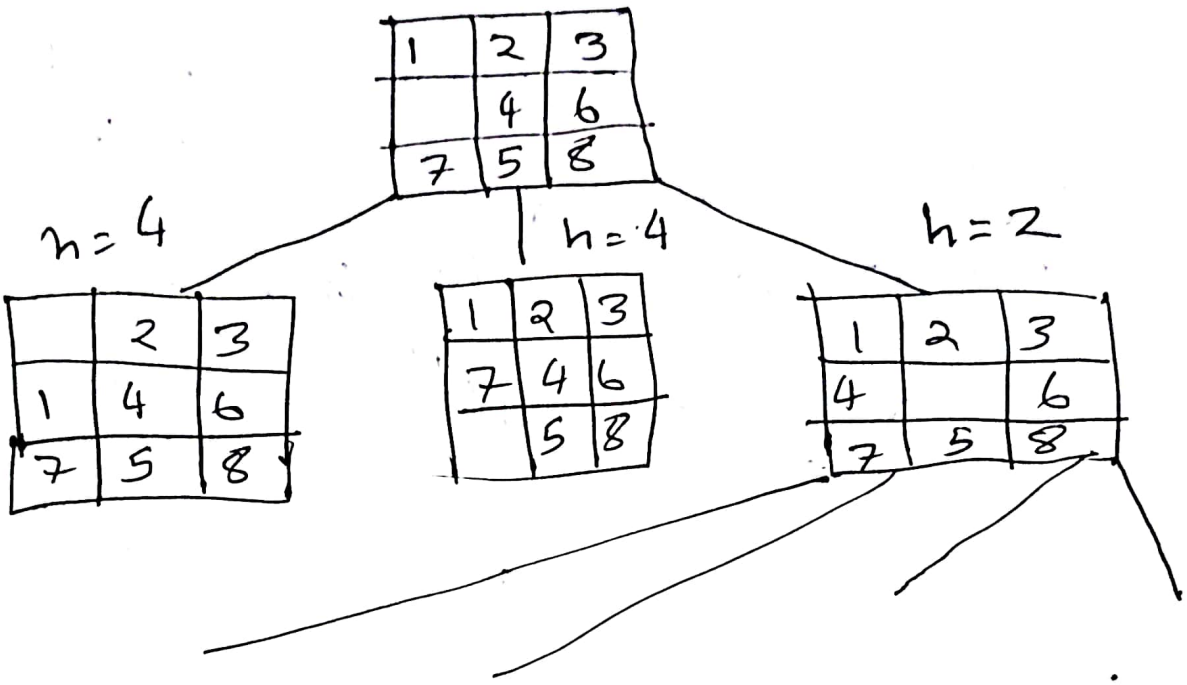
1	2	3
4	5	6
7	8	

goal state

here, h_1 = numb of misplaced tiles = 3

h_2 = Manhanettan distance.

$= 0 + 0 + 0 + 1 + 1 + 0 + 0 + 1 + 1$

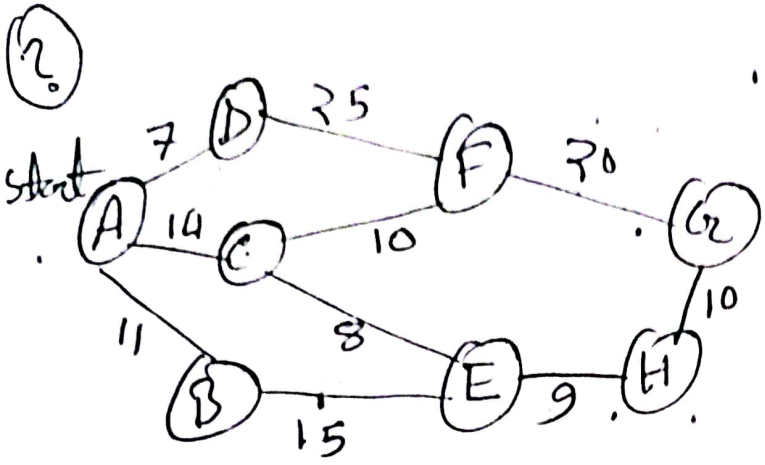


Nb:

* Greedy approach

  Best first Search.

Best First Search



heuristic val

A → G = 40

B → G = 32

C → G = 25

D → G = 35

E → G = 19

F → G = 17

H → G = 10

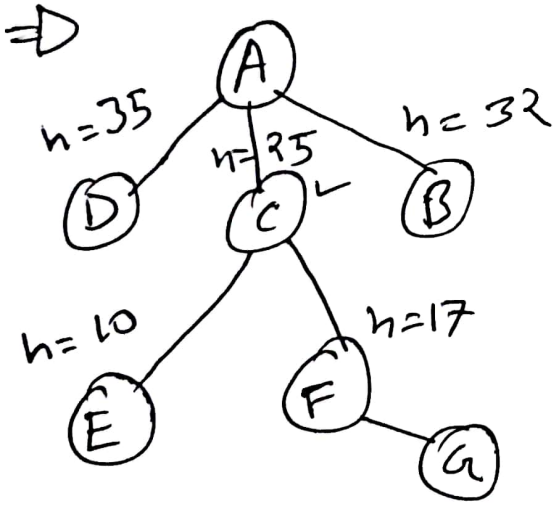
G → G = 0

NOTE: h(n) ⇒ Euclidean distance

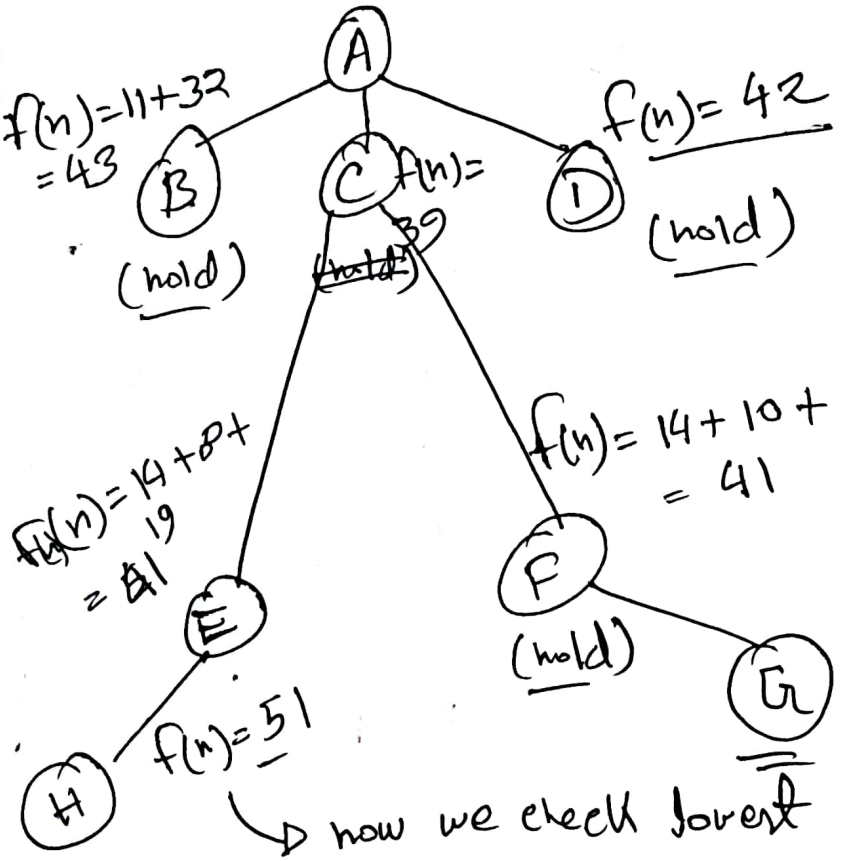
(x_1, y_1) & (x_2, y_2)

$\Rightarrow \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

⇒



A* Search :- (Same Graph) $f(n) = g(n) + h(n)$



↳ now we check lowest hold, so we go to f

it's an optimal solution.

A – C – F – G

(*** Class 5)

A* algo: (admissible → optimal)

✓ optimality of A* search:

(i) $h(n) >$ actual cost

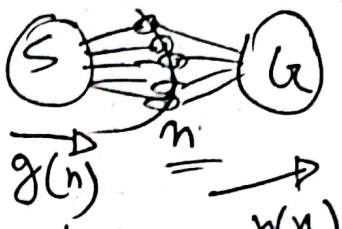
optimal না হবে এটা (ii) $h(n) =$ actual cost

(iii) $h(n) \leq$ actual cost

→ admissible heuristic

NOTE:

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



estimated — actual

$h(n) \leq h^*(n)$ cost

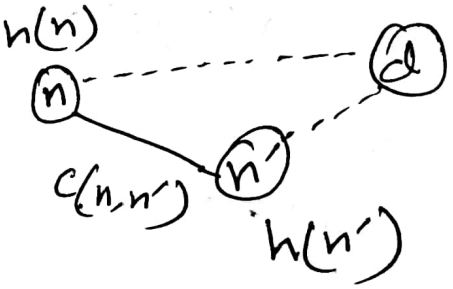
⇓

Underestimation

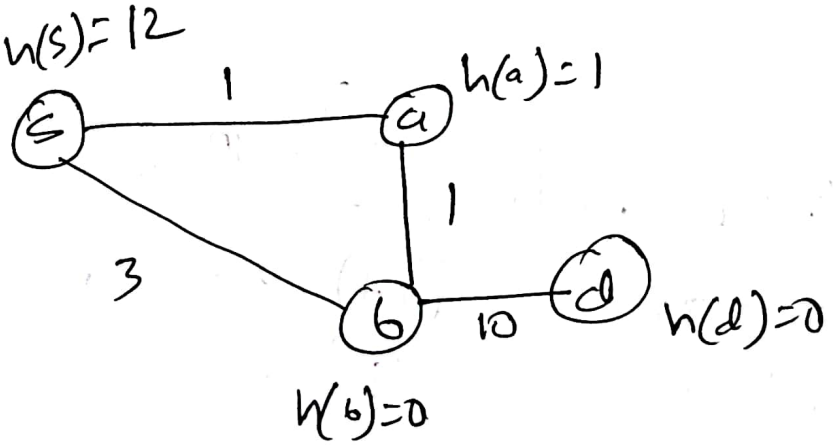
$h(n) \geq h^*(n)$ = overestimation.

✓ consistent heuristic

$h(n) \leq c(n, n') + h(n')$



ex:



∴ it's an admissible heuristic

cuz it follows $h(n) <$ actual cost.

∴ ~~it's an~~ check consistency ⇒

	$h(n)$	$c(n,n')$	$h(n')$
S–a	12 ≤	1	+ 11
S–b	12 ≥	3	+ 0 (X)

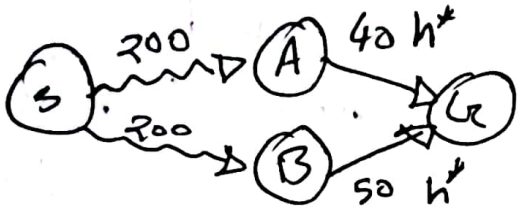
∴ so it's not consistent.

estimate val actual val → overestimate (↳ constraint)

* $h(n) \leq h^*(n) \Rightarrow$ Underestimation
  $h(n) \geq h^*(n) \Rightarrow$ Overestimation

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

How to make A* admissible:



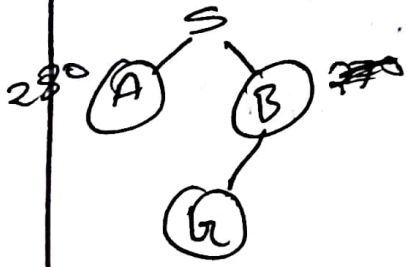
here, $g(A) = 200$, $g(B) = 200$ } actual value

case1: (Overestimation)

let, $h(A) = 80$, $h(B) = 70$ } $> h^*$

$\therefore f(A) = 200 + 80 = 280$

$f(B) = 200 + 70 = 270$



$\therefore f(G) = g(G) + h(G) = 250 + 0$

$= 250$

case2: (Underestimation) → [always get optimal sol.]

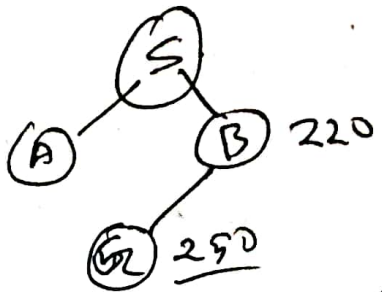
let, $h(A) = 30$

$h(B) = 20$

$\therefore f(A) = 200 + 30 = 230$

$f(B) = 200 + 20 = 220$

$f(G) = 200 + 50 = 250$



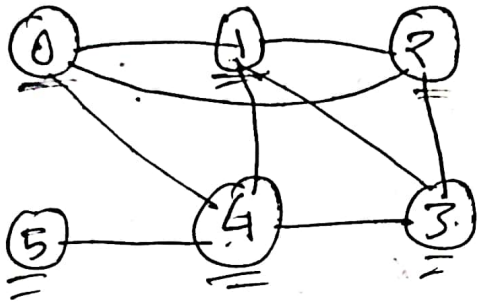
here, it will backtrack and go to (A),

$\therefore f(G) = g(G) + h(G)$

$= 200 + 40 = 240$ → optimal.

[N: so we get optimal even underestimate.]

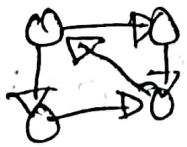
DFS: (i) Stack.
(ii) Back tracking.



⇒

Result: 0, 1, 4, 5, 3, 2

SCC: (i) Weakly (ii) Strongly
↳ undirected ↳ directed
edge ko replace.



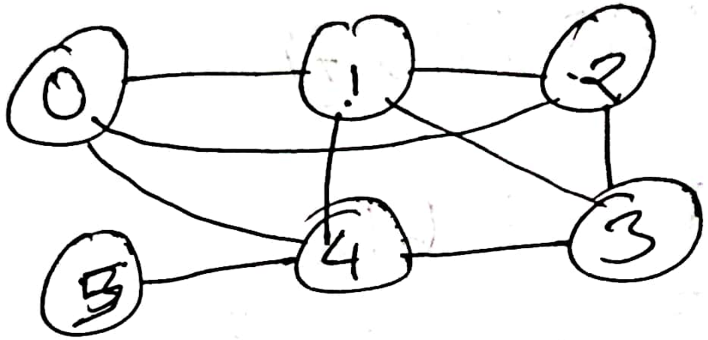
steps:

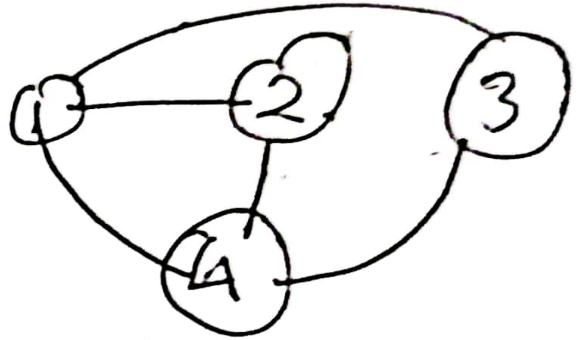
2 (i) Transpose (a)→(b) ⇒ (a)←(b) Gr^T

1 (ii) DFS or

3 (iii) component graph.

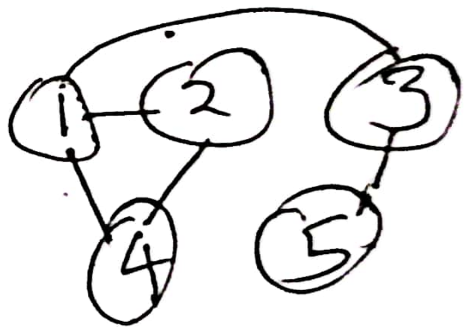
BFS: (i) Queue (ii) যে vertices নিয়ে কাজ করবো তার adj vertices queue তে insert (iii) Visited vertices গুলো আর insert হবে না। (iv) Inserted vertices repeat হবে না।





Queue: ~~0~~ ~~1~~ ~~2~~ ~~4~~ ~~3~~ 5

Result: 0 1 2 4 3 5



Chap-4

AI: Local Search → • Hill climb.
• Simulate annaling.
• Gradient Descent.
• Local beam
• Genetic algo.

Chap-5

Adversial Search →

• gameplan
• α-β pruning.
• MinMax

Chap-6

CSP →

• graph coloring.
• Crypto. Arithmatic.

(hypothesis → heuristic func.)

local search:

(*) 4 queens. (*) Hill climbing.

→ local search algo are designed to explore search space systematically.

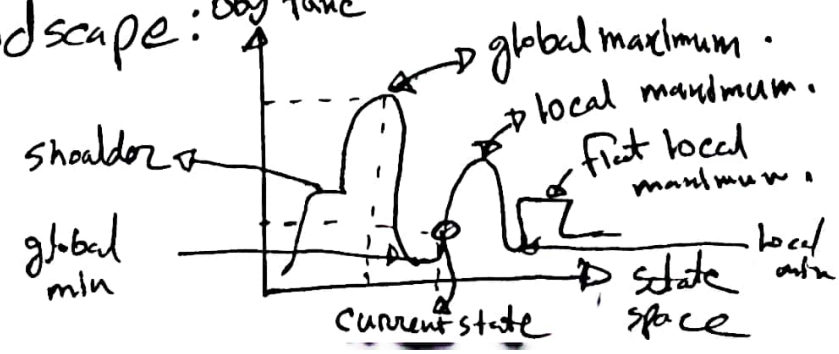
→ local search operate using a single current node (rather than multiple paths) and generally move only to neighbors of that node. The followed by the search are not retained Advantages -

(i) They are very little memory.

(ii) They can often find reasonable solution in large or infinite (continuous) state spaces for which systematic algo's are unsuitable

NOTE: ✓ goal state ~~है~~, object ~~होता~~, ✓ obj → optimize

✓ state-space => landscape:



→ • location - defined by state.

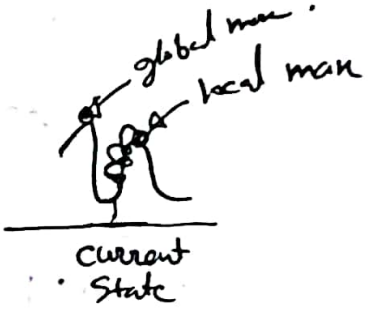
• elevation - defined by the value of the heuristic cost function or objective function.

• if elevation corresponds to cost, then the aim is to find the lowest valley - a global minimum.

• if elevation corresponds to an objective function, then the aim is to find the highest peak - global max.

Hill Climb Search:

function hill climbing (Problem) return a state that is a local maximum.



{ inputs: problem, a

local variable: current, a node
neighbor, a node

current ← MAKE-NODE (INITIAL-STATE [Problem])

loop do

neighbor ← a height-valued succesor of current

if value [neighbor] ≤ VALUE [current] then

return STATE [current]

current ← neighbor.

}

✓ Hill climbing algo problems: • local max • flat max
• Ridge max

↳ To solve this issue -
• simulated annealing algo

* Simulated annacaling search:

- Idea: escape local maximum by allowing some "bad" moves but gradually decrease their frequency.

- function:
- search function SIMULATED-ANNEALING (Problem, schedule) returns a solution state
{ inputs: problem, a problem
schedule, a mapping from the time to 'temperature'

local variable: current, a node
next, a node
T, a "temperature" controlling Problem of downard steps

current ← MAKE NODE (INITIAL-STATE [Problem])
for t ← 1 to ∞ do
T ← schedule [t]

if $T=0$ then return current

next ← a randomly selected successor of current

$\Delta E \leftarrow$ VALUE [next] − VALUE [current]

if $\Delta E > 0$ then current ← next

else

current ← next only with probability $e^{\frac{\Delta E}{T}}$

}

Gradient Descent:

Assume we have cost-function $c(x_1, \ldots, x_n)$

Target:

We want to minimize over continuous variable $x_1, x_2, x_3, \ldots, x_n$

Algorithm:

Step1: Compute the gradient: $\partial / \partial x_i \; c(x_1, \ldots, x_n)$, $\forall i$

Step2: Take a small step downhill in the direction of the gradient:

$x_j \to x_i' = x_j - \eta \, \partial / \partial x_i \; c(x_1, \ldots x_n)$, $\forall i$

where, η is learning rate (hyper parameter)

Step3: check if $c(x_1, \ldots, x_i' \ldots x_n) < c(x_1, \ldots x_i, \ldots x_n)$

If true then accept, if not reject.

Step4: Repeat.

Local beam Search :

- Take care of space complexity.
- Beam width (β) is given.
  - how many choice / what / many path.
- It not complete

$\beta = 2$

Genetic algorithm:

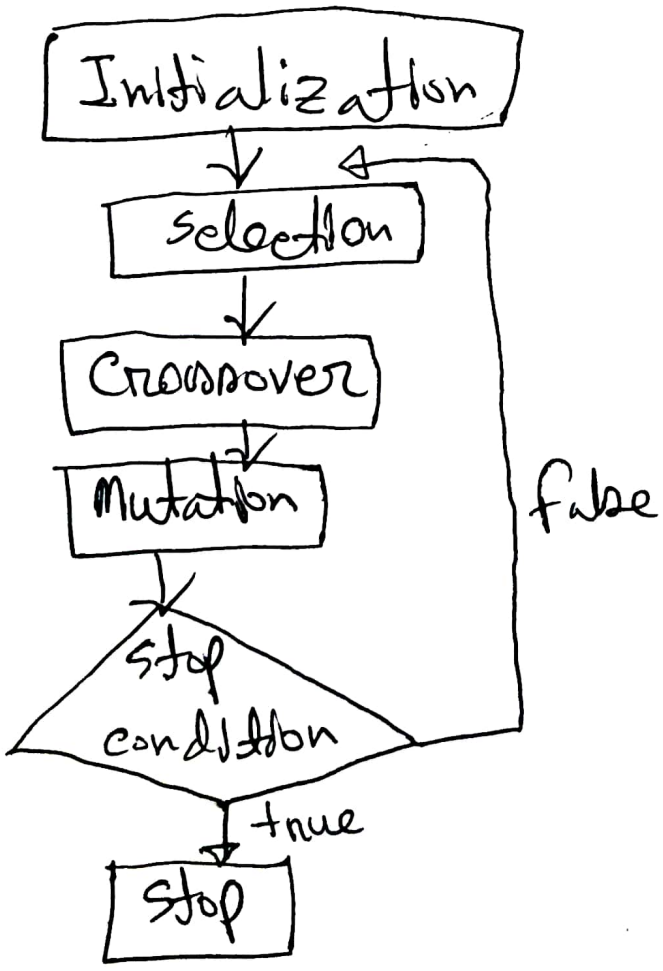
step1: Initialize population.

step2: Select according to fitness.

step3: Cross over between selected chromosom.

step4: Perform mutation.

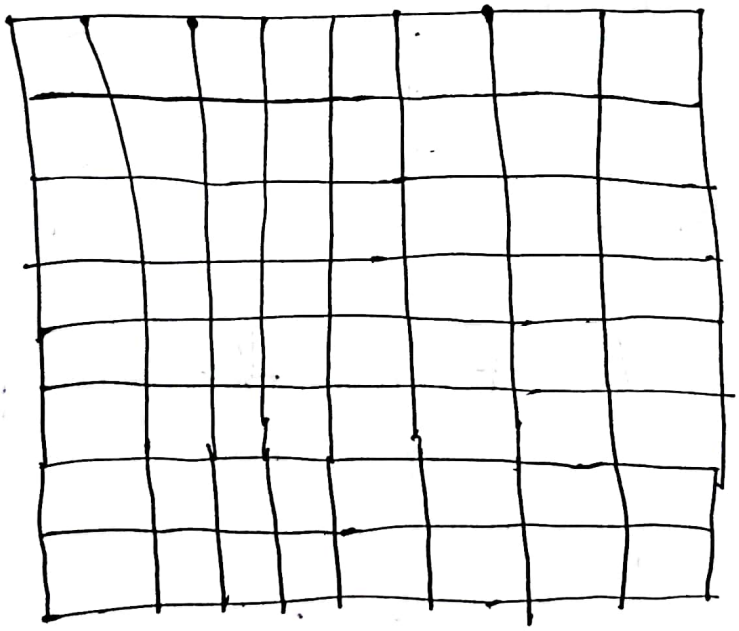
step5: Repeat output till the ~~column~~ condition of stop true.



fitness-function: evaluates how given solution is to the optimal solution of the problem.

A function should return higher values for better state.

exercise: 8 queens - solve with genetic algo!



Consider the following individual states for 8 queen problem. Perform genetic algorithm with the threshold of selection 16.

game plan Chap-5

Adversarial Search:

Types of games:

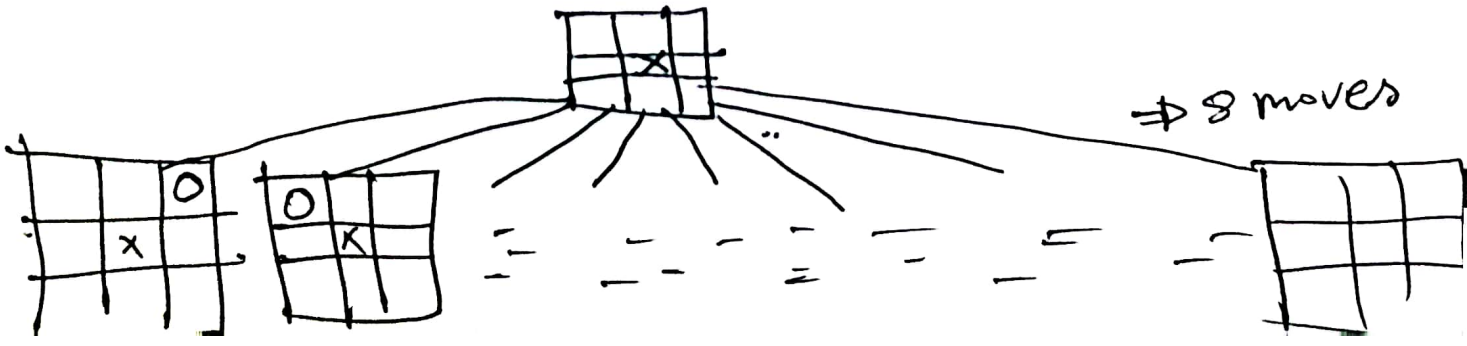
Perfect information	deterministic: chess, checkers	chance: monopoly, backgammon
Imperfect information	Battleship	bridge poker

Typical assumption: ☐ Two agents whose actions alternate. ☐ Utility values for each agent are opposite of the other.

☐ fully observable environment.

☐ Generalize to stochastic games, multiple players, non-zero-sum etc.

Game tree (2 player deterministic turns)



here, how we search this tree to find the optimal move?

Games - adversary

- Solution in strategy
  - * strategy specify move for every possible opponent reply.
- Time limit force an approx. solution.
- Evaluation function: evaluate "goodness" of game position.

<u>example</u>: chess, checkers, go etc.

Games as Search:

Two players: Max and MIN

* Max moves first and they take turns

Untill game is over.

- winner gets reward and looser gets penalty.
- "Zero-Sum": sum of reward and penalty is constant.
- Formal definition as a search Problem:
  - Initial state: setup specified by the Rules
    eg. initial board configuration of chess.
  - Player(s): Defines which player has the move in a state.
  - Action(s): Returns the set of legal moves in a state
  - Result: Transition model defines the result of a move
  - Terminal test: Is the game finished?

True if finished.

- Utility function: Gives numerical values of terminal state s for player p.

For example: win(+1), Lose(-1) and draw(0) in the tic toe.

(D) {•Backtracking • Best Move Strategy • Max Min}
• Max => Utility will be max (Best Move) • Min => Worse Move C-3

Min-Max procedure:

Designed to find the optimal strategy for max and find best move:

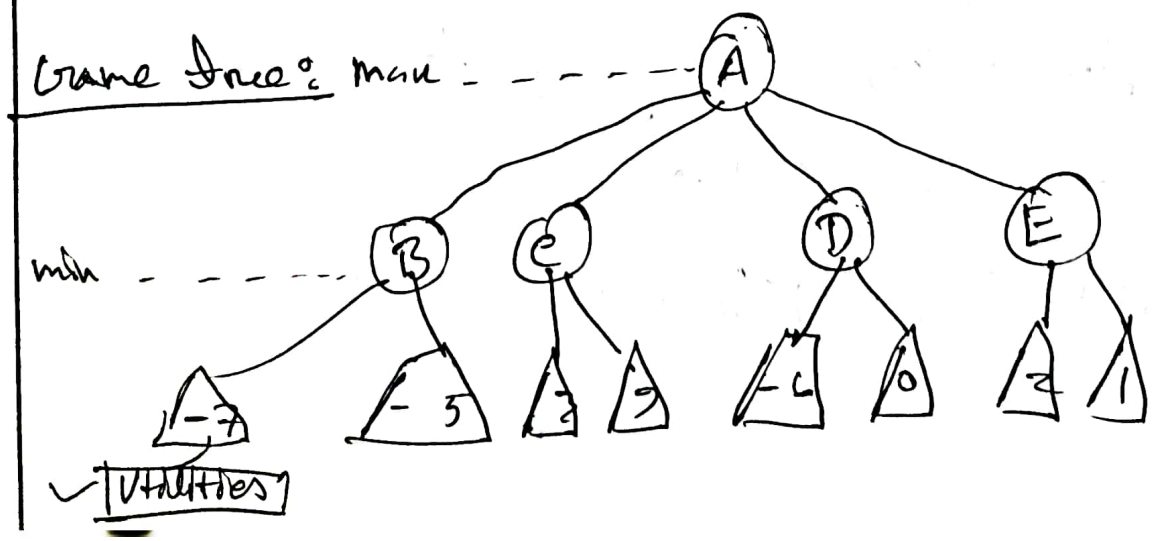
(i) Generate the whole game tree, down to the leaves.

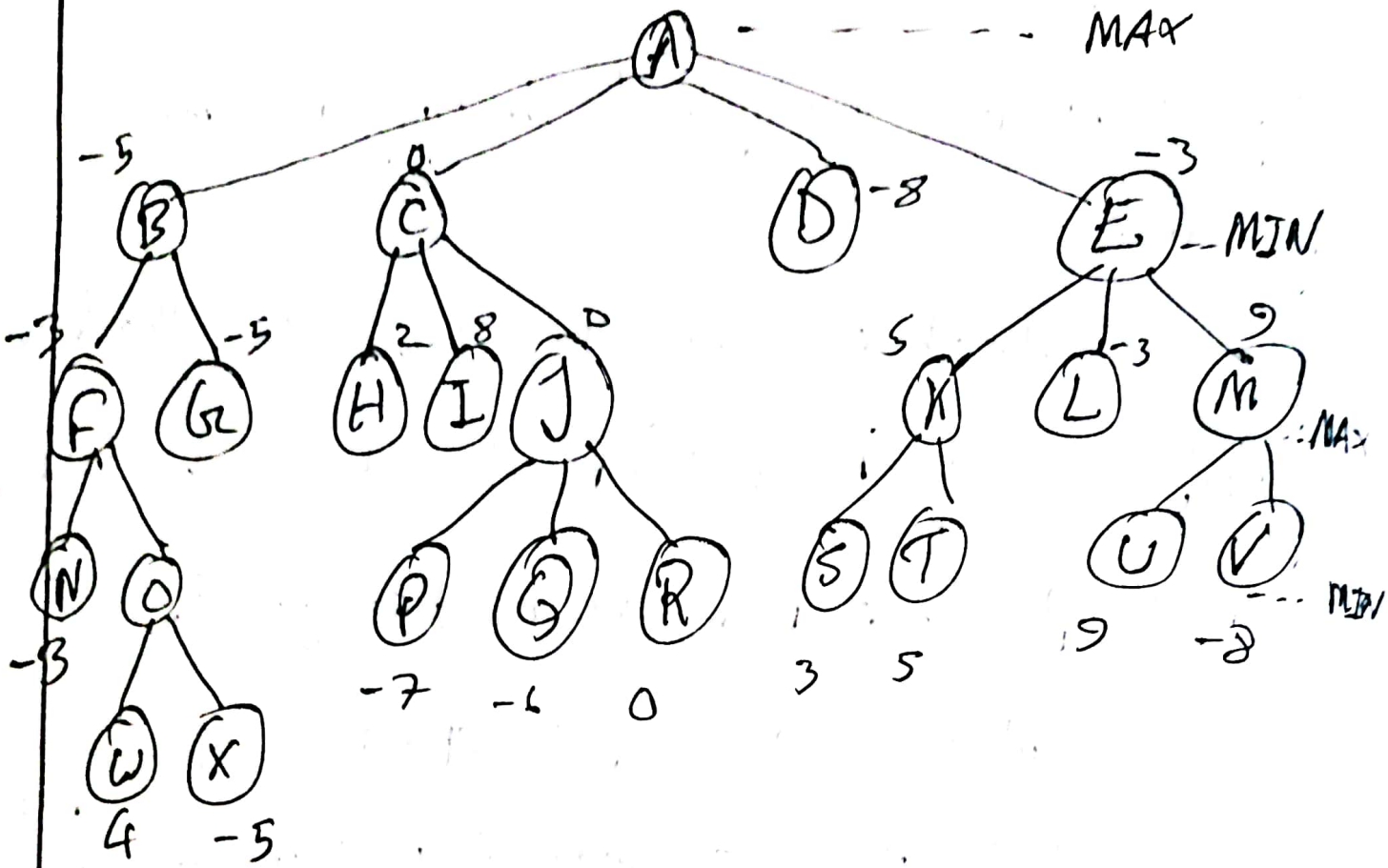
(ii) Apply utility (Payoff) function to each leaf.

(iii) Back-up values from leaves through branch node:
- a Max node computes the max of its child values
- a Min node computes the min of its child values.

(iv) At root: choose the move leading to the child of highest value.

Game Tree:





A -> C -> J -> R

Properties of minimax

- Complete? Yes.
- Optimal? Yes.
- Time complexity? (b^m) — numb of node.
- Space complexity? $O(bm)$

Static (Heuristic) Evaluation function:

An Evaluation function:

- Estimate how good the current board configuration is for a player.
- Chess: Value of all white pieces.
  - value of all black pieces.

For chess, typically linear weight sum of features

$$Eval(S) = w_1 f_1(S) + w_2 f_2(S) + \dots + w_n f_n(S)$$

↳ Heuristic is $E(n) = M(n) - O(n)$

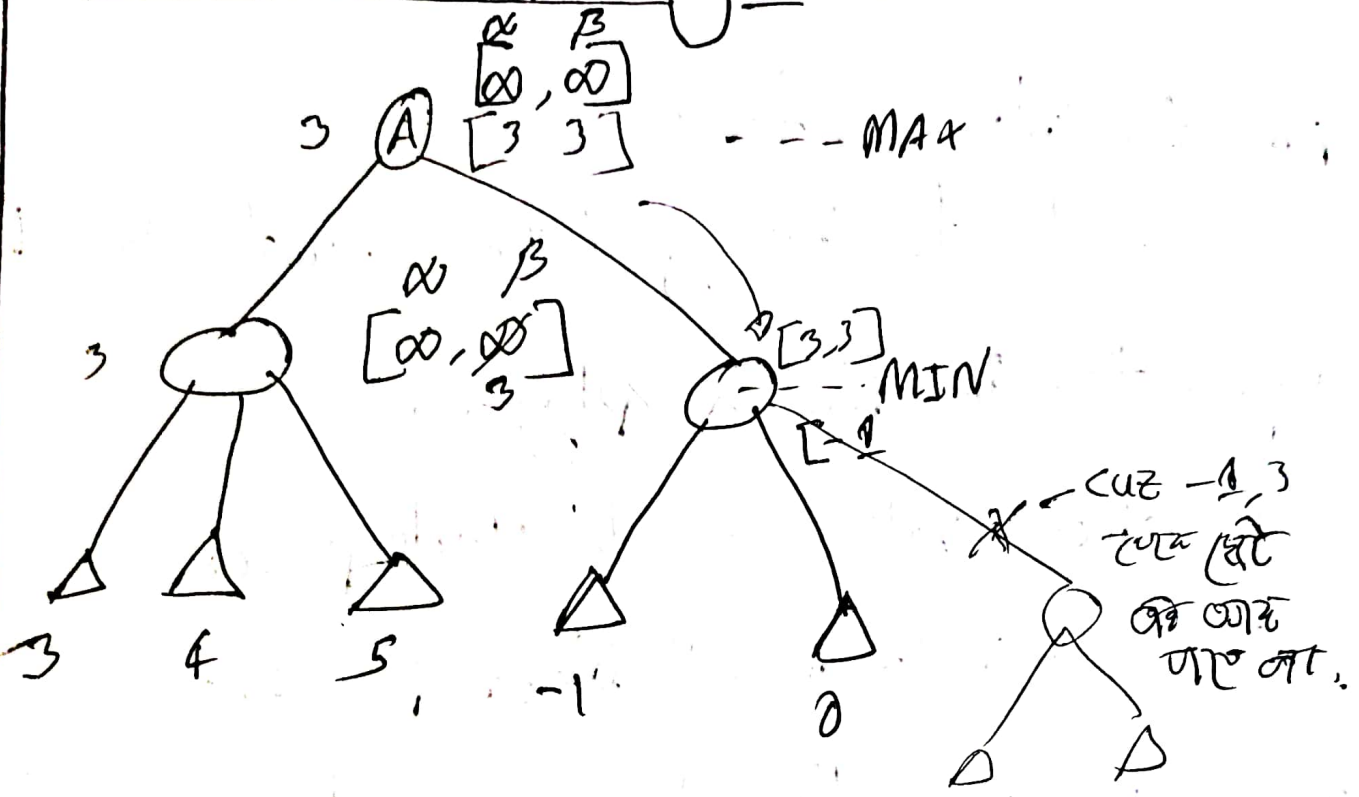
where, $M(n)$ in the total of my possible winning line.

$O(n)$ in total of opponent possible winning line,

$E(n)$ in the total Evaluation for staten.

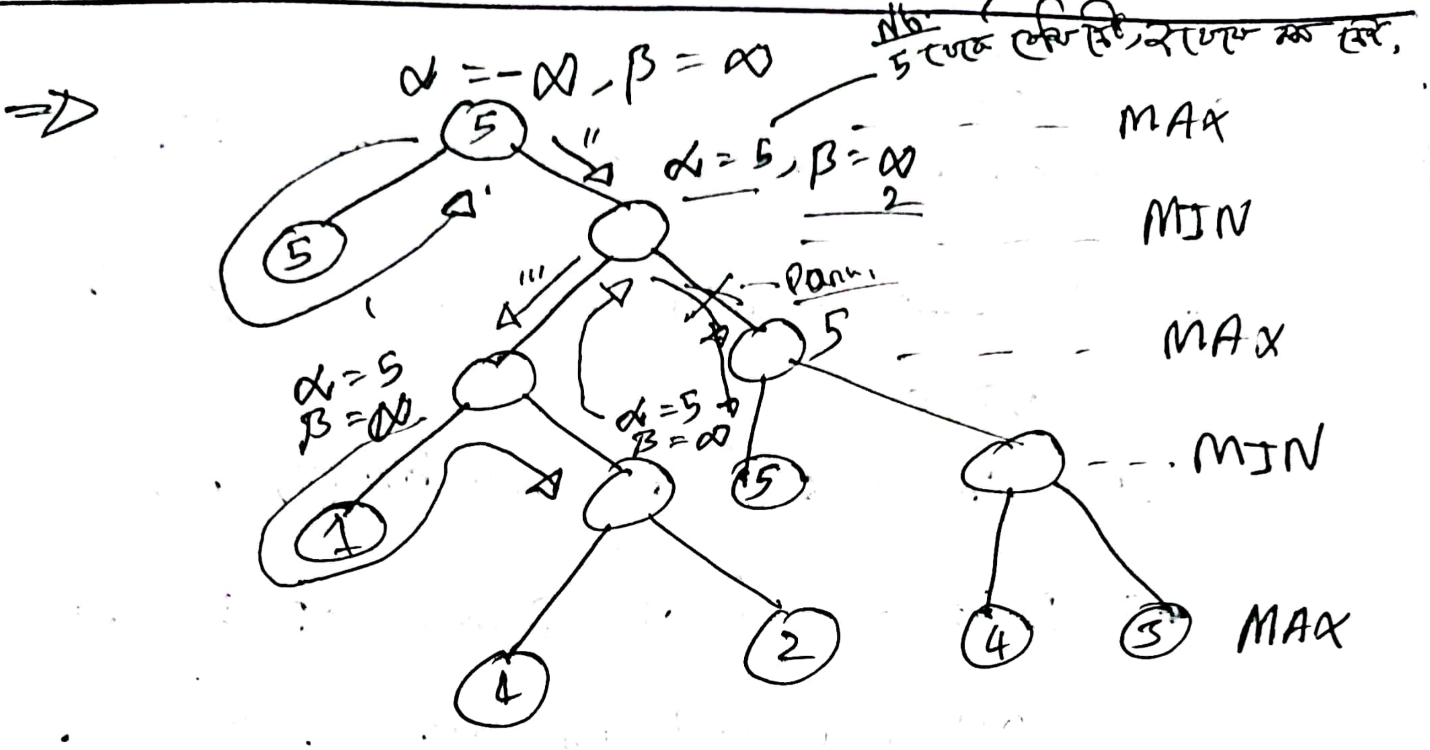
↳ update version of min-max algo.
- dfs

⊛ ✻ Alpha - Beta pruning : (DFS).



□ What is the mini-max value of the root node for the game tree below? Assuming that it performs a dfs that always generates the left most child node first and a bom (and max min) of Max (and Min) corresponds to a value of $-\infty$ (and ∞, respectively) determines the alpha and beta values of the remaining node (s).

So, we got max-min value.



the minimax value is 5.

NOTE: In minimax we have to explore every nodes. The time complexity will be $O(b^d)$; So we can cut off the search by exploring less no. of nodes.

condition ⇒ $\alpha \geq \beta$ → then we will prune ✓

- Max ⇒ α
- Min ⇒ β.

chap - 6

Constraint Satisfaction problem (CSP)

* CSP consists of three components

V, D, C

→ V is a set of variable $\{V_1, V_2, \ldots V_n\}$

→ D is a set of Domain $\{D_1, D_2, \ldots D_n\}$ (types)

one for each variable.

→ C is set of constraints that specifies combination of values,

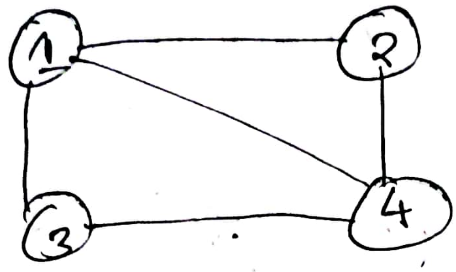
$$c_i = (\text{scope}, \text{rel})$$

Where, scope is set of variables that participate in constraint.

Rel is relation that defines the value that variable can take

Graph coloring problem:

• BackTracking in intelligent way.



V = {1, 2, 3, 4}

D = {Red, Green, blue}

C = {1≠2, 1≠3, 1≠4, 2≠4, 3≠4}

← intell →

	1	2	3	4
Initial Domain	R,G,B	R,G,B	R,G,B	R,G,B
1 = R	R	G B	G B	G B
2 = G	R	G	B	B
3 = B	R	G	B	B (empty)
3 = G	R	G	G	B

back track

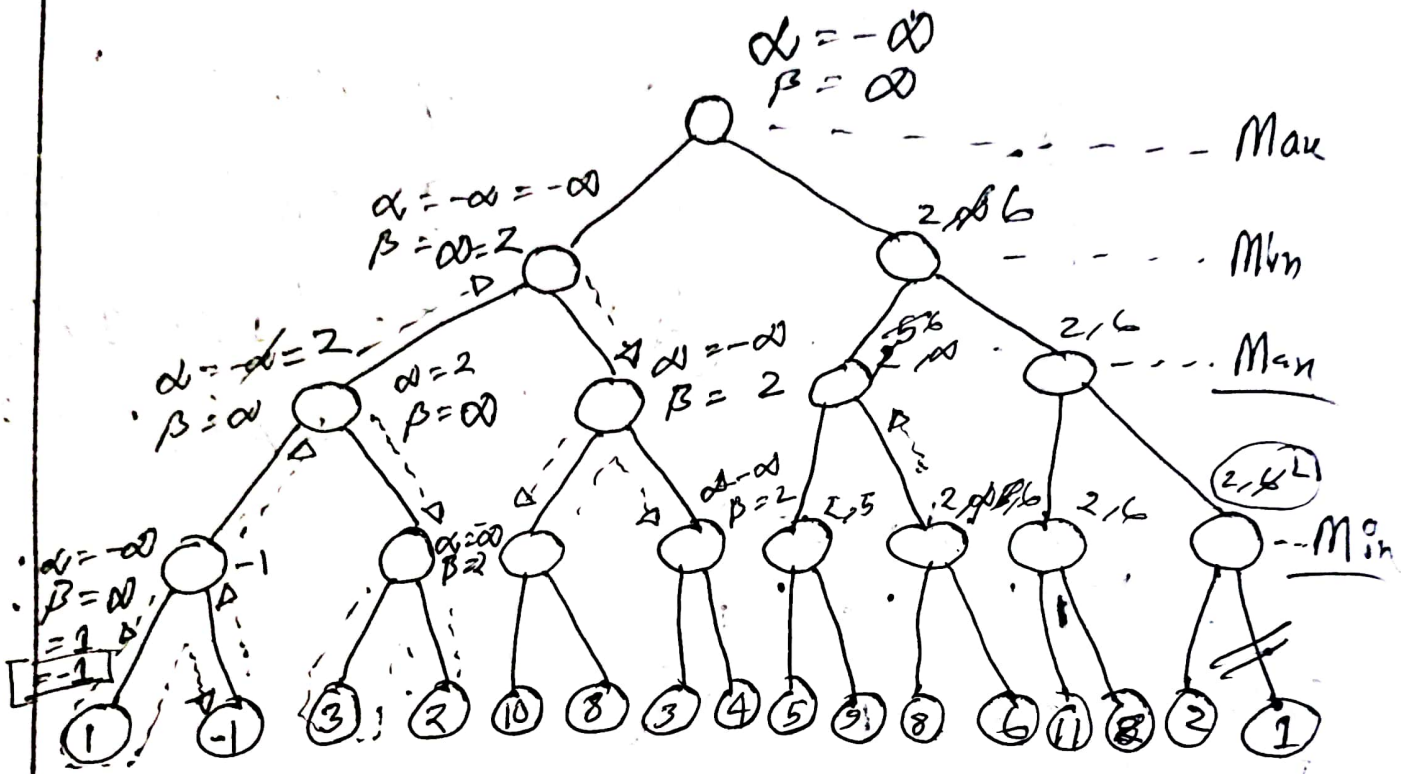
[Note: Here, 1≠2, 1≠3, 1≠4 so, the colon in intell 2,3,4 will GB, GB, GB.]

[NOTE: here, 1 is fixed for R]

[here, in (4) is used backtrack cause this empty]

here, no neighbour colour is not same.

ex: (Alpha-beta pruning)



condition:

here, α = value of best choise so far for Max (highest)

β = value of best choise so far for Min. (lowest)

prune when, $\alpha \geq \beta$

✓(CSP)

Crypto arithmetic problem:

→ Digits that can be assigned to a word/alphabet

→ Rang - (0-9)

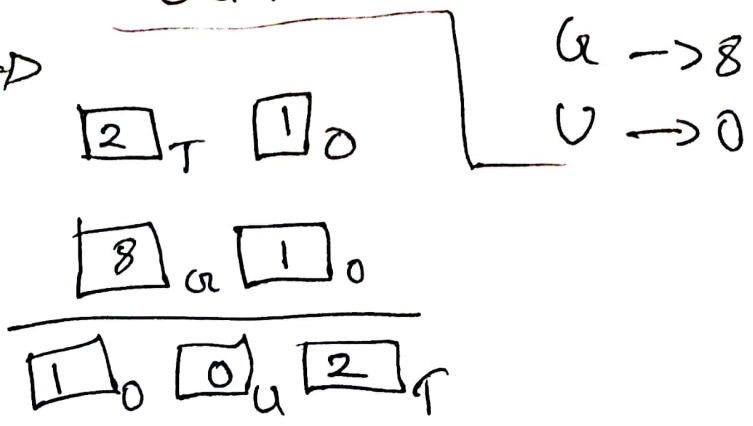
→ Construction:

* No two letter have same value
* sum of digit must be as shown in problem.
* there should be only one carry-forward.

ex: ② TO
GO
OUT

↑ -> 2
O -> 1
G -> 8
U -> 0

→



(3)

```
  S E N D
  M O R E
---------
M O N E Y
```

⇒

```
      6
  9   5   8   7
  S   E   N   D
  1   0   8   5
  M   O   R   E
---------------
  1   0   6   5   2
  M   O   N   E   Y
```

Carry

(?)

```
  S O M E
+ T I M E
---------
S P E N T
```

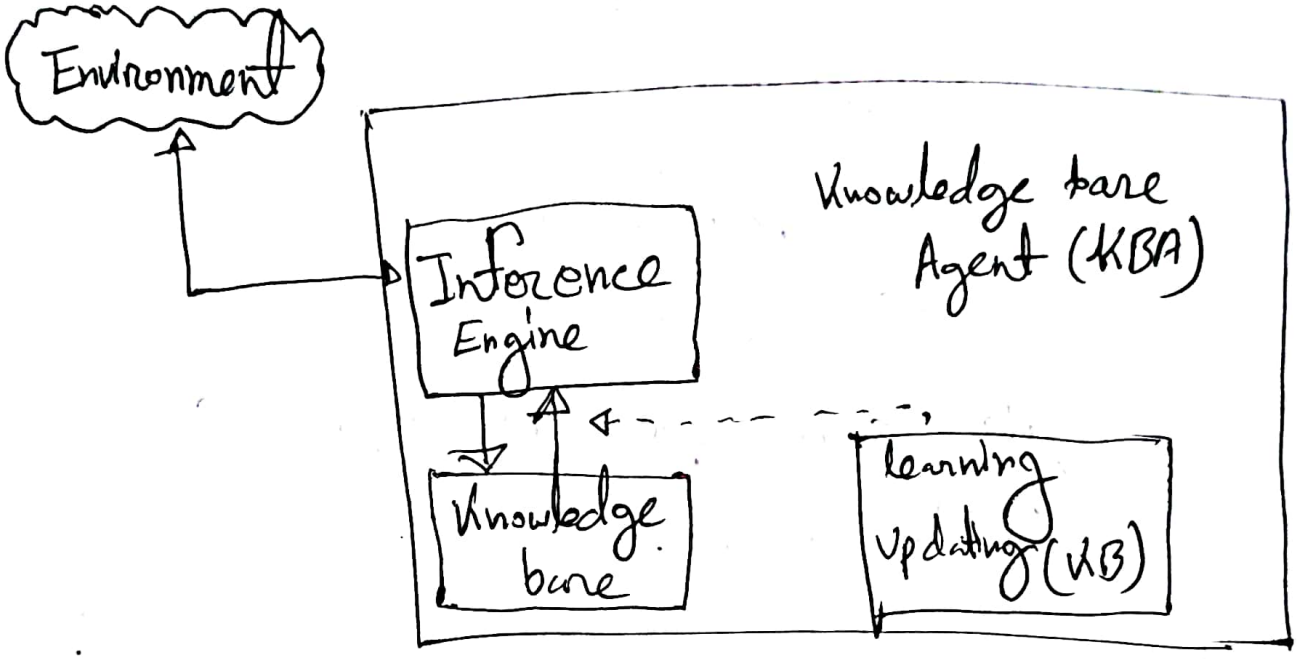
⇒

```
    2
  1   9   3   4
  S   O   M   E
  1   5   3   4
  T   I   M   E
---------------
  1   0   6   8
  S   P   E   T
```

Chap-8

final

Knowledge base and Agent:



Operation perform by KBA:

(i) Tell: This operation tells KB agent that precise from the environment.

(ii) ASK: This operation tells → what action it should perform.

(iii) Perform/Action: If perform the selected ACTION.

Formal Symbol System:

- Ontology.
- Representation.
- Inference.

- Symbol correspond to things / ideas in the world.
- Pattern matching & rewrite corresponds to inference.

Ontology → (what exists in world)
(∧ ∨ !) ⇒ defines meanings of symbol.

eg: logic representation ontology.
{∧: and, ∨: or, !: not}

Representation: Syntax Vs Semantics.

C: if (a>0) {...}
else {...}

✓ Inference:

Schema Vs Mechanism

Proof steps search strategy

✓ Knowledge base Agent:

- KB = Knowledge Base.
  - A set of sentences or facts
  - A set of statement in a logic language

- Inference
  - Deriving new sentences from old.

- A simple model for reasoning
  - Agent is told or perceives new evidence.
  - Agent then infer new facts to add the KB.

Eg: A, B logic ⊢ A ∨ B infer
A in Trainee ⊢ T.

Approaches to Knowledge Representation:

(i) Inferential Knowledge.

→ EWU in a university, Propositional logic

→ All student enroll in CSE, Predicate logic.

(ii) Inheritable Knowledge:

→ OOP.

(iii) Simple Relational Knowledge:

→ DBMS.

(iv) Procedural Knowledge

→ USE of small code with If-else condition.

(1) logical Representation:

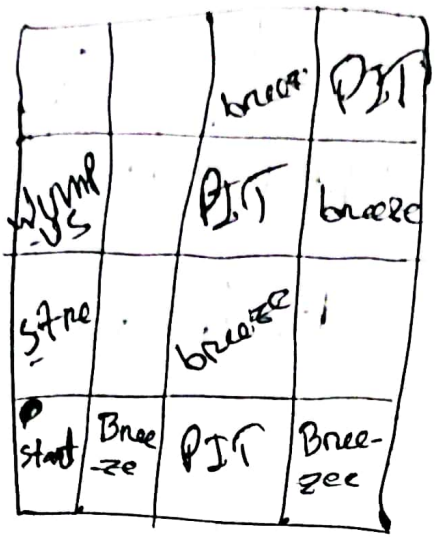
Types of logic:

* * propositional logic
  * Predicate logic (first order logic, first order predicate calculas).

Predicate:
- For all x: $\forall x$
- For one x: $\exists x$

* Fuzzy logic
* Probability
* Temporal logic.
* Moral logic.

Wumpus world PEAS description:



*Performance measure:

- gold: +1000, death: -1000
- -1 per steps, -10 for using the arrow

* Environment

- Squares adjacent to wumpus are smelly.
- Squres adjacent to pit are breezy.

- Glitter iff gold in in the same square.
- shooting kills. wumpone if u are facing it.
- shooting uses up only the arrow.
- Grabbing picks up gold if in same square.

* Sensor : stench, Breeze, Glitter, Scream
* Actuators left turn, Right turn, forward, Grab, Release, shoot.

□ We used logical reasoning to find the gold.

Entailment: It means that one thing follows from another: $KB \models \alpha$.

eg: "Mary is Sue's sister and Amy is sue daughter"

entail - "Mary is Amy's aunt."

Models:

We say m is model of a sentence α if α is true in m.

$M(\alpha)$ is the set of all model of α.

Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$

ex: Brazil won the WC in 2002 and Italy won in 2006.

α = Brazil won

Think of KB and α as collection of constraints and model m as possible states.

$M(KB)$ are the sol of KB &

$M(\alpha)$ the sol of α

Then $KB \models \alpha$ when all sol to KB are also sol to α.

Final

Class-2

Recap:

#Propositional logic:

syntax: The propositional symbols P_1, P_2 are sentences -

- ✓ If S is a sentence, $\neg S$ is a sentence (negation)
- ✓ If S_1 & S_2 are sentence, $S_1 \wedge S_2$ is a sentence (conjunction)
- ✓ " " " , $S_1 \vee S_2$ is a sentence (disjunction).
- ✓ " " " , $S_1 \Rightarrow S_2$ is a sentence (implication)
- ✓ " " " , $S_1 \Leftrightarrow S_2$ " " " (biconditional).

semantics:

Each model/world specifies true or false.

for each proposition symbol.

For example, P_1 or P_2 is false.

with respect to

Rules for evaluating truth w.r.t a model, m:

"$\neg S$" is true iff S is false.

$S_1 \wedge S_2$ is true if S_1 is true and S_2 is true.

$S_1 \vee S_2$ " " " or " " ".

$S_1 \Rightarrow S_2$ " true S_1 is false or S_2 is true

$S_1 \Leftrightarrow S_2$ " false iff S " true and S_2 " false.

$S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and

$S_2 \Rightarrow S_1$ " true.

Suppose, P_1 = true
P_2 = true
P_3 = false

For arbitrary sentence,

$\neg P_1 \wedge (P_2 \vee P_3)$ = false $\wedge$ (true $\vee$ false)
= false $\wedge$ true = false

Truth table:

in always true (implication) when premise are false

P	Q	¬P	P ∧ Q	P ∨ Q	P ⇒ Q	P ⇔ Q
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	T	T	T	T

Logical equivalence:

Two sentences are logically equivalent iff they are true in same model:

$\alpha \equiv \beta$ if $\alpha \models \beta$ & $\beta \models \alpha$

→ $(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ Commutativity of $\wedge$

→ $(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ " of $\vee$

→ $((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \vee \gamma))$ associativity of $\wedge$

→ $((\alpha \wedge \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ ... = of ∨

→ $\neg(\neg\alpha) \equiv \alpha$ double negation elimination.

→ $(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition.

→ $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination.

→ $(\alpha \Leftrightarrow) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination.

→ $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de morgan

→ $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de morgan

→ $(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \vee \gamma))$

distributivity of ∨ ∧ over ∨

→ $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of ∧ over ∨.

Validity and satisfiability :

A sentence is valid if it is true in all models

eg: True, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \models B$

validity : is connected to inference via the deduction theorem $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid.

A sentence is satisfiable if it is true in some model.

A sentence is unsatisfiable if it is false in all models.

Satisfiability : is connected to inference via following :

KB $\models \alpha$ iff $(KB \wedge \neg\alpha)$ is unsatisfiable.

(There is no model for which KB = true & α = false)

Problem:

Find whether the meaning of the statement is true or false -

"If the earth moves round the sun or the sun moves round the earth, then copernicus might be mathematician but was not an astronomer."

(implication) (but) (not)

$\Rightarrow$ P = The earth moves round the sun = TRUE(T)

Q = Sun " " " earth = F.

R = copernicus might be a mathmatician = T

S = " was an astronomer = T

~~P∨Q~~ Propositional logic :

$(P \vee Q) \Rightarrow (R \wedge \neg S)$

$= (T \vee F) \Rightarrow (T \wedge \neg T)$

$= T \Rightarrow (F \wedge F)$

$= T \Rightarrow F$

$= F$.

Inspite of having French nationality,
B. Russel was a critic of imperialism,
then either he was not a bachelor or
he was a mathematician.

$\Rightarrow P$ = B. Russel has French nationality $= T$.

q = B. Russel was a critic of imperialism $= T$.

R = B. Russel was a bachelor $= T$.

S = B. Russel was a mathematician $= T$.

$(P \wedge Q) \Rightarrow (\neg R \vee S)$

$= (T \wedge T) \Rightarrow (\neg T \vee T)$

$= \{T \Rightarrow (F \vee T)$

$= T \Rightarrow T$

$= T.$

To Prove that, $\neg(P \vee (\neg P \wedge Q)) \equiv \neg P \wedge \neg Q$
$\equiv \neg(P \vee Q)$

Sol: $\neg(P \vee (\neg P \wedge Q))$

$= \neg P \wedge \neg(\neg P \wedge Q)$ [De morgan]

$= \neg P \wedge (P \vee \neg Q)$ [Double negation & De morgan]

$= (\neg P \wedge P) \vee (\neg P \wedge \neg Q)$ [Distributive property]

$= F \vee (\neg P \wedge \neg Q)$

$= \neg P \wedge \neg Q$

$= \neg(P \wedge Q)$ =

✓ Inference :

NOTE: KB Agent
↳ Knowledge representation.
- propositional logic

new logic using old logic on evidence

so, generating the conclusion from evidence and facts is termed as inference.

- inference rules: are the templates for generating valid arguments. Inference rules are applied to derive proofs of AI, and the proof is a sequence of the conclusion that leads to the goal.

* implication : $P \Rightarrow Q$
* converse : $Q \Rightarrow P$ (opposite of implication)

* Contrapositive : $\neg Q \Rightarrow \neg P$
* Inverse : $\neg P \Rightarrow \neg Q$ (opposite of contrapositive)

P	Q	$P \Rightarrow Q$	$Q \Rightarrow P$	$\neg Q \Rightarrow \neg P$	$\neg P \Rightarrow \neg Q$
T	T	T	T	T	T
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	T

From truth table, we can prove

$P \rightarrow Q$ in equation to $Q \Rightarrow \neg P$ and $Q \Rightarrow P$ is equivalent to $\neg P \Rightarrow \neg Q$.

Class-3

Let p, q and r be the proposition

p: Grizzly bears have seen in the area.

q: Hiking in safe on the trail

r: Berries are ripe along the trail.

Write - these proposition using p, q, r and logical connection, -

(a) Berries are ripe along the trail but grizzly bears have not been seem in the area

P. Logic $= r \wedge \neg p$

(b) Grizzly bears have not been seen in the area and hiking on the trail in safe but berries are ripe along the trail.

P. logic $= \neg p \wedge q \wedge r$.

(c) If berries are ripe along the trail, hiking is safe if and only if grizzly bears have not been seen in the area.

$$P.\text{logic} : r \Rightarrow (q \Leftrightarrow \neg p)$$

(d) It is not safe to hike on the trail but grizzly bear have not been seen in the area and the berries along the trail are ripe.

$\Rightarrow \neg q \wedge \neg p \wedge \neg r$.

☑ Types of inference rules –

① Modus ponens: One of the most essential law of inference is the modus ponens rule which asserts that if P and $P \Rightarrow Q$ are both true, we can infer that Q will be true as well

Notation of modus ponens = $\frac{P \Rightarrow Q, P}{Q}$

ex- "If I am sleepy then I go to bed"

$P \Rightarrow Q$

I am sleepy = P premise

I go to bed = q conclusion.

Hence, we can say that if $P \Rightarrow Q$ is true and P is true then Q will be true

P	Q	$P \Rightarrow Q$
0	0	0
0	1	1
1	0	0
1	1	1

(ii) Modus Tollens: If $P \Rightarrow Q$ is true and $\neg Q$ is true then $\neg P$ will also true.

Notation modus Tollens = $\dfrac{P \Rightarrow Q, \neg Q}{\neg P}$

Premise: I do not go to the bed $\neg Q$

conclusion: which infers that I am not sleepy $\neg P$

✓

$\neg P$	$\neg Q$	$P \Rightarrow Q$	$\neg P$	$\neg Q$
0	0	0	1	1
0	1	1	1	0
1	0	0	0	1
1	1	1	0	0

(iii) Hypothetical syllogism:

If $P \Rightarrow R$ is true whenever $P \Rightarrow Q$ is true and $Q \Rightarrow R$ is true

"If you have my home key then you can unlock my home." $P \Rightarrow Q$

.
.
.

"If you can unlock my home then you can take my money" $Q \Rightarrow R$

Conclusion: "If you have my home key then you can take my money" $P \Rightarrow R$

$$\frac{P \Rightarrow Q,\ Q \Rightarrow R}{P \Rightarrow R}$$

(iv) Disjunctive syllogism: If $P \vee Q$ is true, and $\neg P$ is true, then Q will be true.

Notation of Addition = $\frac{P \vee Q, \neg P}{Q}$

Example: "Today is sunday or monday" $P \vee Q$

"Today is not sunday" $\neg P$

Conclusion: Today is monday.

(v) Addition: "If P is true then P or Q will be true" Notation: $P / P \vee Q$

example: "I have a vanilla icecream" P

"I have a chocolate icecream" Q

Conclusion: I have vanilla or chocolate icecream. $(P \vee Q)$ $P \vee Q$

(vi) Simplification: If $P \wedge Q$ is true, then Q or P will also be true.

Notation: $P \wedge Q / P$ or $P \wedge Q / Q$

(vii) Resolution: If P or Q and $\neg P \wedge R$ is true then $Q \vee R$ will also be true.

Notation: $$\frac{P \vee Q, \neg P \wedge R}{Q \vee R}$$

(4 Jan)

KB for Wumpus world:

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
1,1	2,1	3,1	4,1

A
G
OK
P
S
V
W
B

Atomic proposition variable for wumpus world:

Let, P_{ij} be true if ~~agent perceive breeze~~ there is a pit in the room [i,j]

* Let $B_{i,j}$ true if agent perceive breeze in [i,j]. (dead or active)

* Let $W_{i,j}$ be true if there is wumpus in the square [i,j]

✓ Let $S_{i,j}$ be true if agent percieve stench is the square $[i,j]$

✓ Let $G_{i,j}$ be true if gold in the square $[i,j]$

✓ Let $OK_{i,j}$ be true if there room is safe

propositional rules - for the wampus world.

$R1: \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

$R2: \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \vee \neg W_{2,2} \wedge \neg W_{3,1}$

$R3 \quad \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$

$R4 \quad \neg S \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$

#Prove that wumpus in in room [1,3]

Sol:- 1. Apply modules ponen with

$\neg S_{1,1}$ and R_1

$\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

2. Apply and elimination rules:

... $\neg W_{1,1}$, $\neg W_{1,2}$, $\neg W_{2,1}$

-> $\neg S_{2,1}$ and R_2

$\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$

$\neg W_{1,1}$, $\neg W_{2,1}$ $\neg W_{2,2}$, $\neg W_{3,1}$

-> $\neg S_{1,1}$ and R_3

$\neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \vee \neg W_{2,2} \wedge \neg W_{1,3}$.

$\neg W_{1,1}$, $\neg W_{1,2}$, $\neg W_{2,2}$, $\neg W_{1,3}$.

-> $S_{1,2}$ & R4 : $S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
$W_{1,2}, W_{2,2}, W_{2,1}$.

Apply unit resolution on $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,2}$ and $\neg W_{1,1}$

we will see $W_{1,3} \vee W_{1,2} \vee W_{2,2}$

$W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$ $\quad \neg W_{1,1}$

* Apply unit resolution -

$W_{1,3} \vee W_{1,2} \vee W_{2,2}$ $\quad \neg W_{2,2}$

$W_{1,3} \vee W_{1,2}$

* Apply -

$W_{1,3} \vee W_{1,2}$ $\quad \neg W_{1,2}$

$W_{1,3}$

Day-5

Predicate

Knowledge representation and Reasoning :

- logic
  - Propositional - T/F
  - Predicate
- Rules - if / then
- Semantic net → graph
  - meaning
  - graph

Knowledge based Agents :-

Predicate logic :

syntax
semantics
meaning

∧ conjunction
∨ disjunction
¬ negation
→ implies
↔ bi-conditional

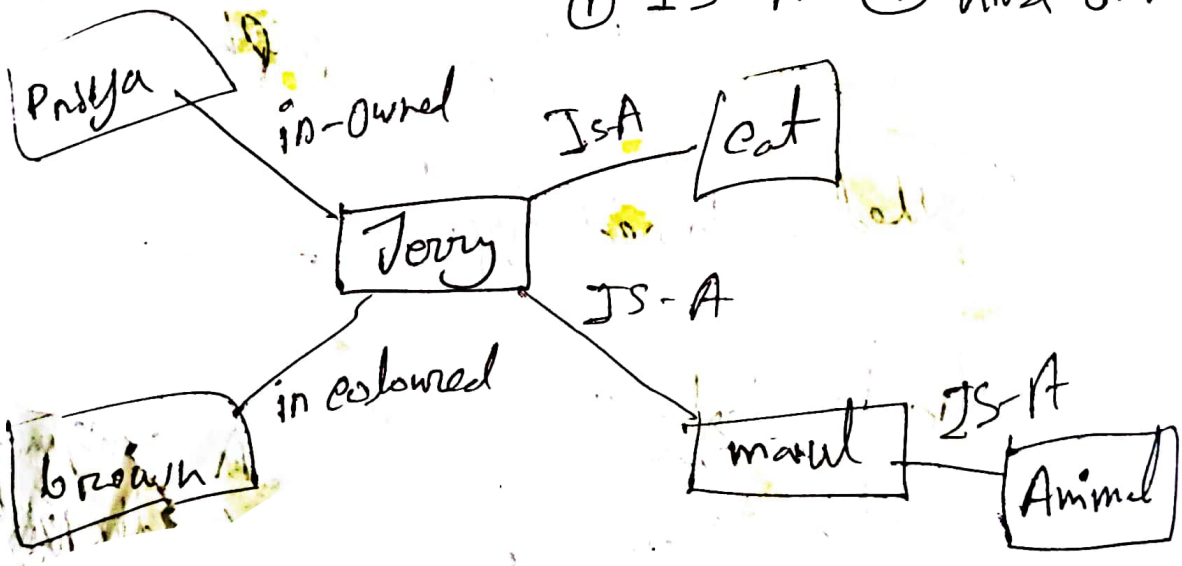
Symantic network representation :

statement : (a) Jerry is a Cat.

(b) Jerry is a mammel

(c) Jerry is owned by riya

Network representation : Two types relation →
(i) Is-A (ii) Kind of.



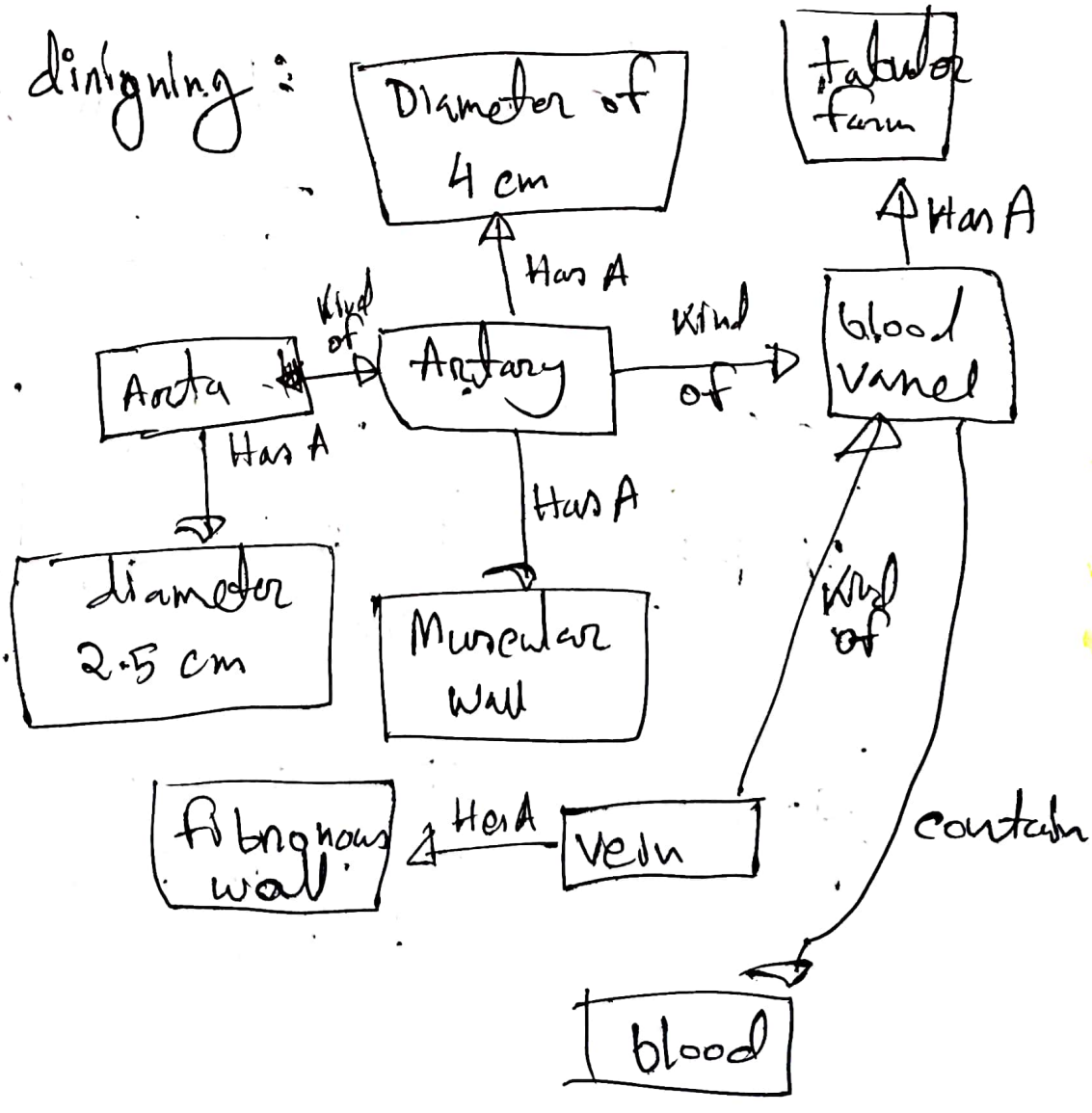
statement : The aorta is a particular kind of artary which has a diameter of 2-5 cm.

An artary is a kind of blood vessel. An artary always has a muscular wall and generally has

a diameter of 4 cm. A vein in a kind of blood vessel but has a fibrous wall. Blood vessels all have tabular form and contain blood.

↓ relationship.

Network diagram:



2 types not. design ⇒

① Frame ✗ ⟶ DBMS System (Attribute, no

② Production Rule: If condition then Action

statement: If (at the bus stop AND bus arrives)
then (get into the bus)

Predicate logic: ∀ - universal quantifier
∃ - Existential quantifier

statement	when true	when false
∀n P(n) ⌊OR	P(n) is true for every n	There is an n for which P(n) is false
∃n P(n) ⌊and	There is an n for which P(n) is true	P(n) is false for every n.

express the statement:

"Every student in the class has studied calculus" using predicate and quantifier.

student predicate $S(x)$

calculus predicate $C(x)$

statement: $\forall x(S(x) \Rightarrow C(x))$

"All lion are fierce"

"some lions do not drink coffee"

"some fierce creatures do not drink coffee"

$P(x)$ = "x is a lion"

$Q(x)$ = "x is fierce"

$R(x)$ = "x drinks coffee".

s_1: $\forall (P(x) \Rightarrow Q(x))$

s_2: $\exists x (P(x) \wedge \exists R(x))$

s_3: $\exists x (Q(x) \wedge \neg R(x))$

Examples:

* All birds fly.

$$\forall x \; bird(x) \rightarrow fly(x)$$

* Not all student like both mathematics and Science.

Predicate question like (x, y)

where x = student and y = subject

$$\exists x \; [student(x) \rightarrow like(x, Mathematics) \wedge like(x, Science)]$$

Logical inferance

→ Notion of entailment

→ Model checking: x enumerate all possible models and check whether α is true.

→ Sound (or truth preserving):

- i is a sound iff whenever $KB \models \alpha$ i if is also true that $KB \models \alpha$.
- Model checking is sound.

→ Complete

Proof Method

→ Application of inference rules

Propositional FOL

legitimate (sound) generation of new sentences from old.

- Resolution - Proof by contradiction.
- forward and backword chaining.

→ Model checking:

Searching through truth assignments

* Improved backtracking (DPLL)
* Heuristic search in model space: walk sat

(iv) Conjunctive normal form :

$P \rightarrow Q \equiv \neg P \wedge Q$

We would like to prove : $KB \neq \alpha$

equivalent to - $KB \wedge \neg \alpha$, unsatisfied.

$KB \wedge \neg$ into conjunctive normal form (CNF)

↓

A "conjunction of disjunction"

Resolution : → Algorithm works using proof of contradiction.

→ To show $KB \neq \alpha$, we show that $KB \wedge \neg \alpha$ is not satisfied.

→ Apply resolution to $KB \wedge \neg \alpha$ in CNF and resolve pairs with complementary literal

★ Statement: if the unicorn in mythical, then in ~~not~~ immortal, but if it in not mythical, it in a mamanal. If the unicorn in either immortal on a mammal, then it in horned.

Prove: the unicorn in horned.

Sol: M = mythical, I = immortal A = mammal, H = horned.

(i) $M \Rightarrow I$, CNF: $\neg M \vee I$

(ii) $\neg M \Rightarrow A$, CNF: $M \vee A$

(iii) $I \vee A \Rightarrow H$, CNF: $\neg(I \vee A) \vee H$.

Statement: (i) If it is sunny & warm day you will enjoy. [Sunny ∧ Warm => enjoy] → Prop. logic

(ii) If it is raining you will get wet.

[Raining ⇒ wet]

(iii) If it is a warm day [warm]

(iv) It is raining [Raining]

(v) It is sunny [Sunny]

Prove: you will enjoy.

Sol: (i) Sunny ∧ warm => enjoy

≡ ¬(Sunny ∧ Warm) ∨ enjoy.

≡ ¬ Sunny ∨ ¬ Warm ∨ enjoy.

(iii) ¬Raining => wet

≡ ¬Raining ∨ wet

Draw resolution graph: ¬enjoy ¬Sunny ∨ ¬Warm ∨ enjoy [Stmt (i)

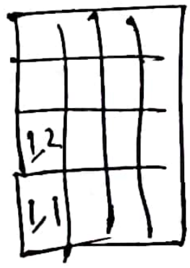
¬Sunny ∨ ¬ warm [Stmt 3

¬Sunny

$\neg$Sunny Sunny [statement 5]

⟹ contradiction.

Wampus Prev.



Rules / statement:

$R_1 : \neg P_{1,1}$

$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$R_4 : \neg B_{1,1}$

$R_5 : B_{2,1}$

Is there a pitfall in [1,2] ?

Sol: we have $KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$,

we want to prove $\neg P_{1,2}$

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$\equiv (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

by biconditional elimination R2.

$\equiv (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$ by AND elimination

$\equiv \neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})$ by contrapositions.

$\equiv \neg (P_{1,2} \vee P_{2,1})$ by Modus Ponens.

$\equiv \neg P_{1,2} \wedge \neg P_{2,1}$

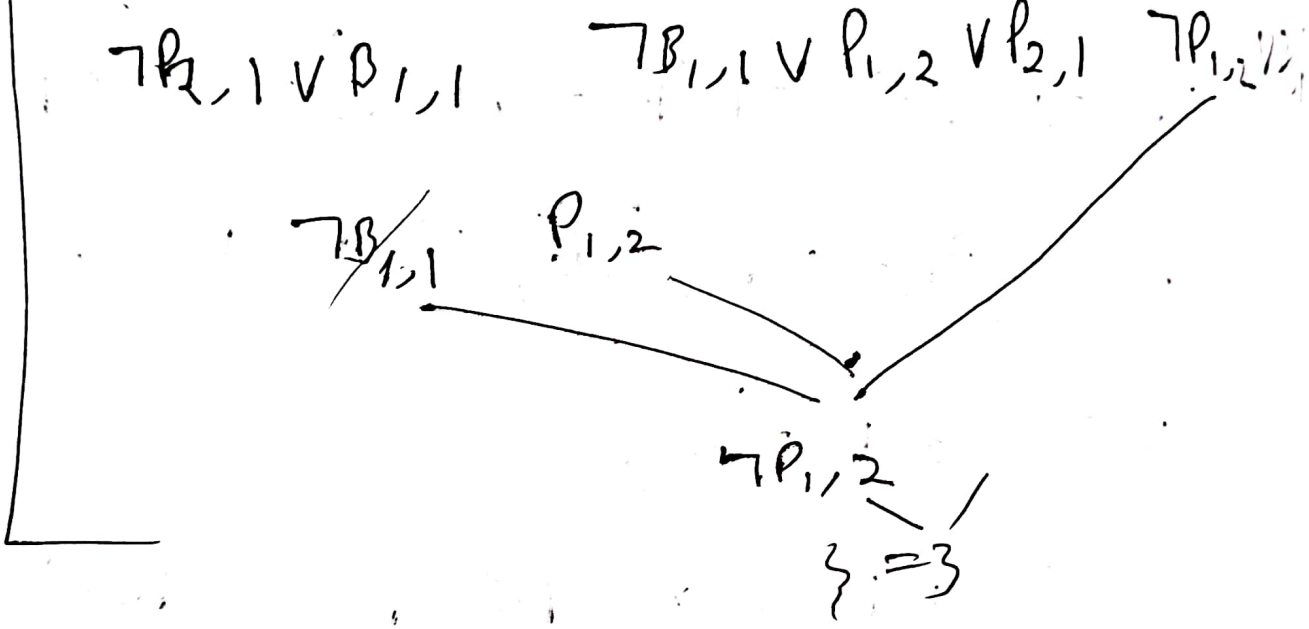
That is neither no pit found both place.

Resolution: $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

$\alpha = \neg P_{1,2}$

(HW) $\Rightarrow$!

Resolution graph:



Inference in first order logic:

#Rules of FOL:

* Universal Generalization: $\frac{P(c)}{\forall x\, P(x)}$

If we want to prove that every element

Example: Let represent, P(c): "A byte contain 8 bits"

so "All bytes contain 8 bits" for $\forall x\, P(x)$, it will also be true.

* Universal Instantiation (elimination) (UI):

→ A valid inference in universal instantiation.

$(KB \neq \alpha)$ $\frac{\forall x\, P(x)}{P(c)}$

→ We can infer any phrase by replacing a ground word for the variable.

→ The UI rule say that we can infer any sentence P(c) by substituting a ground term c (a constant

within domain x) from $\forall x P(x)$ for any object object in the universe of discourse.

exam: If "Every person likes ice cream", $\forall x P(x)$

so we can infer that

"John likes ice-cream", $P(c)$

exam: All Kings who are greedy are Evil.

FOL statement: $\forall x \; King(x) \wedge greedy(x) \Rightarrow Evil(x)$

Infer statement: $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

/ $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

$Father(John) = Richard$

According to UI: $King(Father(John)) \wedge Greed(Father(John)) \Rightarrow Evil(Father(John))$.

(3) ☐ Existential Instantiation: This rules state that for a new constant symbol c, one can deduce P(c) from the formal given in the form of $\exists x\, P(x)$.

It can be represented as, $\frac{\exists x\, P(x)}{P(C)}$

example: FOL ~~example statement~~: $\exists x$ Crown(x) ∧ OnHead(x, John)

so we can infer: Crown(K) ∧ OnHead(K, John)

☐ Unification: It is the process of finding substitute that makes two seperate logical atomic expression identical.

It accepts two literals as input and use substitution to make them identical.

Let ϕ_1 and ϕ_2 be two atomic sentence and to be a unifier such that $\phi_1 \alpha \phi = \phi_2 \alpha$; then

Unify (ϕ_1, ϕ_2)

Example: find the most general unifier (MGU) for

Unify {King(x), King(John)}

let ϕ_1 = King(x), ϕ_2 = King(John)

for two different expression, P(x, y) & P(a, f(z))

$\phi_1 = P(x, y)$

$\phi_2 = P(a, f(z))$

substitute x with a, y with f(z); it will be represented x/a and f(z)/y.

final substitute set will be: P(x/a, f(x)/y).

Unification Algorithm: step 1: Begin by making the substitute set empty.

step 2: Unify atomic sentence in a recursive manner: (i) check for expression that are identical.

(ii) If one expression is a variable V_i, & the other is a term t_i, what does not contain variable V_i, then (a) substitute t_i/V_i in the existing substitute.

(b) Add t_i/V_i to the substitution set list

Rule 3: ⓒ If both the expression are function, then function name must be similar, and the number of arguments must be same for both expression.

Q1: Find the most general unifier (MGU) of
{ P(f(x), g(y)) and P(x, x) }

Sol: $\psi_1 = P(f(x), g(y))$, $\psi_2 = P(x, x)$

Subset: {f(a)/x}

$\psi_1 = P(f(a), g(y))$, $\psi_2 = P(f(a), f(a))$

subset: {f(a)/g(y)} - Unification failed.

unification not possible for this expression.

Q2: Unify (Known(Richard, x), Known(Richard, John))

Sol: ψ_1 = Known (Richard, x), ψ_2 Known (Richard, John)

Substitution: {John/x}

S_1 = {Known(Richard, John), Known(Richard, John)}

✓ Unify logic: ✓ P(x,y) P(z,z) ⇒ Unify {y/z}
⇒ P(x,y) P(x,z) (✓)

✓ if P(x,y) P(z,z) then not possible = (✗)

---

successful unification.

Unifier: {John/x}

Q3: Unify {P(b,x,f(g(z))) & P(z,f(y),f(y))}

Sol: $S_0 \Rightarrow$ {P(b,x,f(g(z))), P(z,f(y),f(y))}

subst: {b/z}

$S_1 \Rightarrow$ {P(b,x,f(g(b))), P(b,f(y),f(y))}

~~$S_2 \Rightarrow$ {f(y)/x}~~ subst: {f(y)/x}

$S_2 \Rightarrow$ {P(b,x,f(g(z))), P(b,x,f(y))}

subst: {g(b)/y}

$S_3 \Rightarrow$ {P(b,f(g(b)), f(g(b))); P(b,f(g(b)), f(g(b))}.

Unifier = {b/z, f(y)/x, g(b)/y}

Q4: Unify {P(x, x) and P(z, f(z))}

Unifier = {x/z}

So => {P(z/z), P(z, f(z))}

unification failed.

Steps for Resolution :-

1. Conversion of facts into FOL
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by
4. Draw resolution graph (Unification)

Ex problem :-

(1) Anything anyone eats and not killed is food

(2) Alex eats peanuts and still alive

(3) John likes all kind of food

(4) (3) Harry eats everything that Alex eat

(5) Apple and vegetable are food

prove by resolution that John likes

Step 01 (a) $\forall x \forall y$ eats $(x, y) \wedge \neg$ killed $\Rightarrow$ food (y)

Step 01 (b) eats (Alex, peanuts) $\wedge$ alive (Alex)

(c) $\forall x$ eats (Alex, x) $\Rightarrow$ eats (Harry, x)

(d) $\forall x$ food $(x) \Rightarrow$ likes (John, x)

(e) food (Apple) $\wedge$ food (vegetables)

added predicate

(f) $\forall x$ $\neg$ killed $(x) \Rightarrow$ alive (x)

(g) $\forall x$ alive $(x) \Rightarrow \neg$ killed (x)

(h) likes (John, peanuts)

Step 02 :- Conversion of FOL into CNF

(a) $\forall x \forall y \neg$ [eats $(x, y) \wedge$ { $\neg$ killed] $\vee$ food (y)

(b) Eats (Alex, peanuts) $\wedge$ alive (Alex)

(c) $\forall x : \neg$ eats (Alex, x) $\vee$ eats (Harry, x)

(e) $\forall x \cdot \neg food(x) \vee likes(John, x)$

(f) $food(Apple) \wedge food(vegetables)$

(b) $\forall x \cdot [\neg killed(x)] \vee alive(x)$

(g) $\forall x \cdot \neg alive(x) \vee \neg killed(x)$

(h) $likes(john, peanuts)$

* By dropping universal quantifiers -

1. $\neg eats(y, z) \vee killed \vee food(z)$

2. $\neg eats(Alex, w) \vee eats(Harry, w)$

3. $\neg food(x) \vee likes(John, x)$

4. $\neg food(Apple)$

5. $food(vegetables)$

6. $eats(Alex, peanuts)$

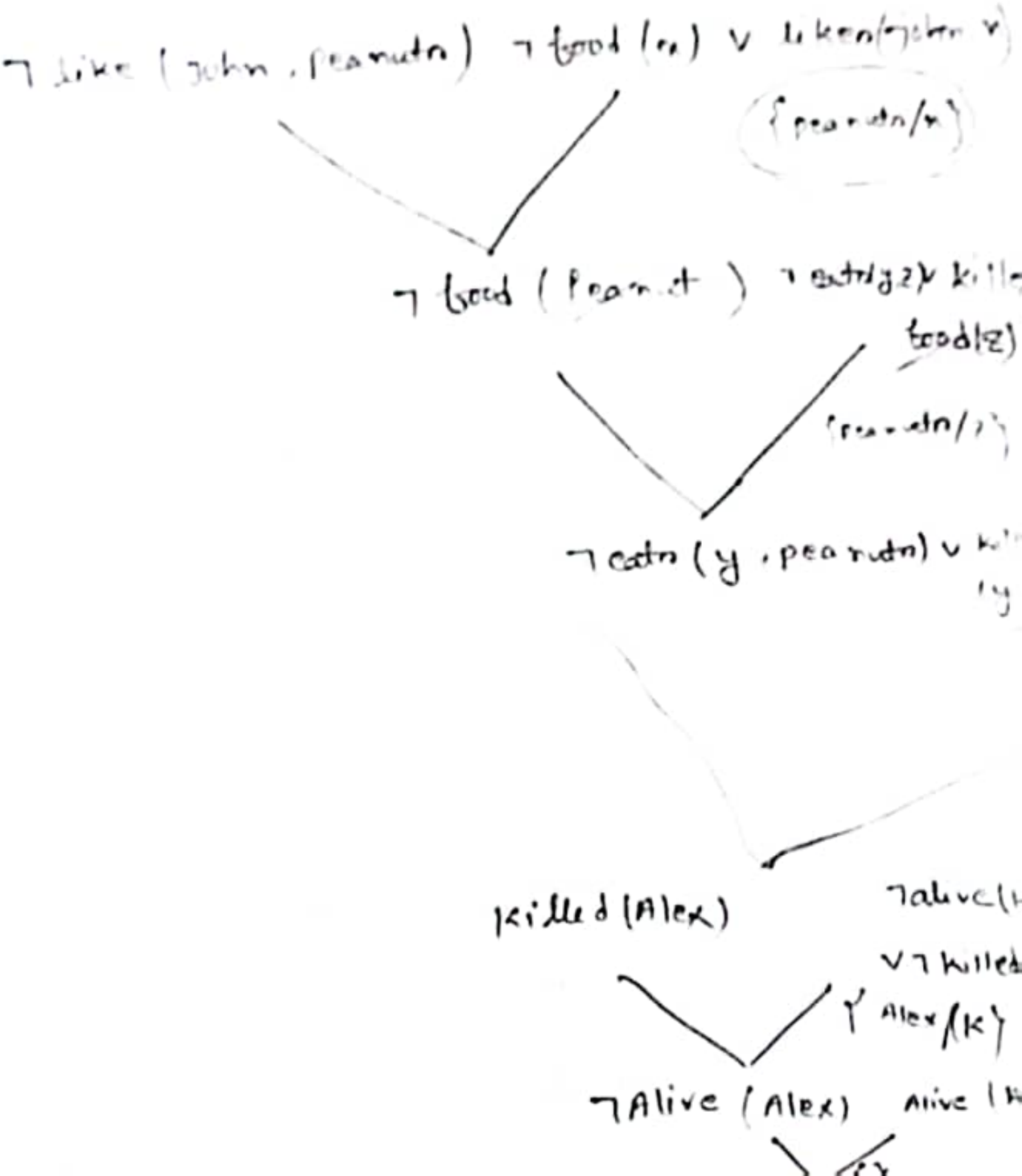
7. $alive(Alex)$

8. $killed(g) \vee alive(g)$

¬ alive (k) ∨ ¬ killed (k)

To like (John, peanuts)

Step 3



(a) Inference engine -

1. Forward chaining
2. Backward chaining

Terms:-

Definite clause - A clause which is a disjunction of literals with exactly one positive literals is known as a definite clause or strict horn clause

eatn (Alex, peanuts)

{ Alex/y }

Horn clause - A clause which is disjunction of literals with at most one positive literal is known as horn clause

For example $(\neg P \vee \neg Q \vee K)$ it has only one positive literal K

It is equivalent $P \wedge Q \Rightarrow K$

Example:- Forward chaining:-

Prob "As per the law, it is a crime for an American to sell weapons to hostile nations Country A, an enemy of Amer has some missiles and all the missiles were sold to it by Robert who is an Am -rican citizen".

prove that using forward chaining:-

" Robert is criminal"

Soln: Fstn conversion into FOL:-

(a) It is a crime for an american to sell weapons to hostile nations.

(a) Statements

[illegible] American (p) ∧ weapon (q) ∧ hostile (r) ∧ sells (p, q, r)

⇒ Criminal (p) ... (1)

(b) Country A has some missiles

Statement -

Owns (A, p) ∧ Missile (p)

Owns (A, T_1) ... (2)

Missile (T_1) ... (3)

(c) All of the missiles were sold to Country C by Robert Statement :-

Missile (p) ∧ Owns (A, p) ⇒ Sells (Robert

(d) Missiles are weapons

Statement :

Missiles (p) ⇒ Weapons (p) ... (5)

(e) Enemy of America is known as hostile

Statement

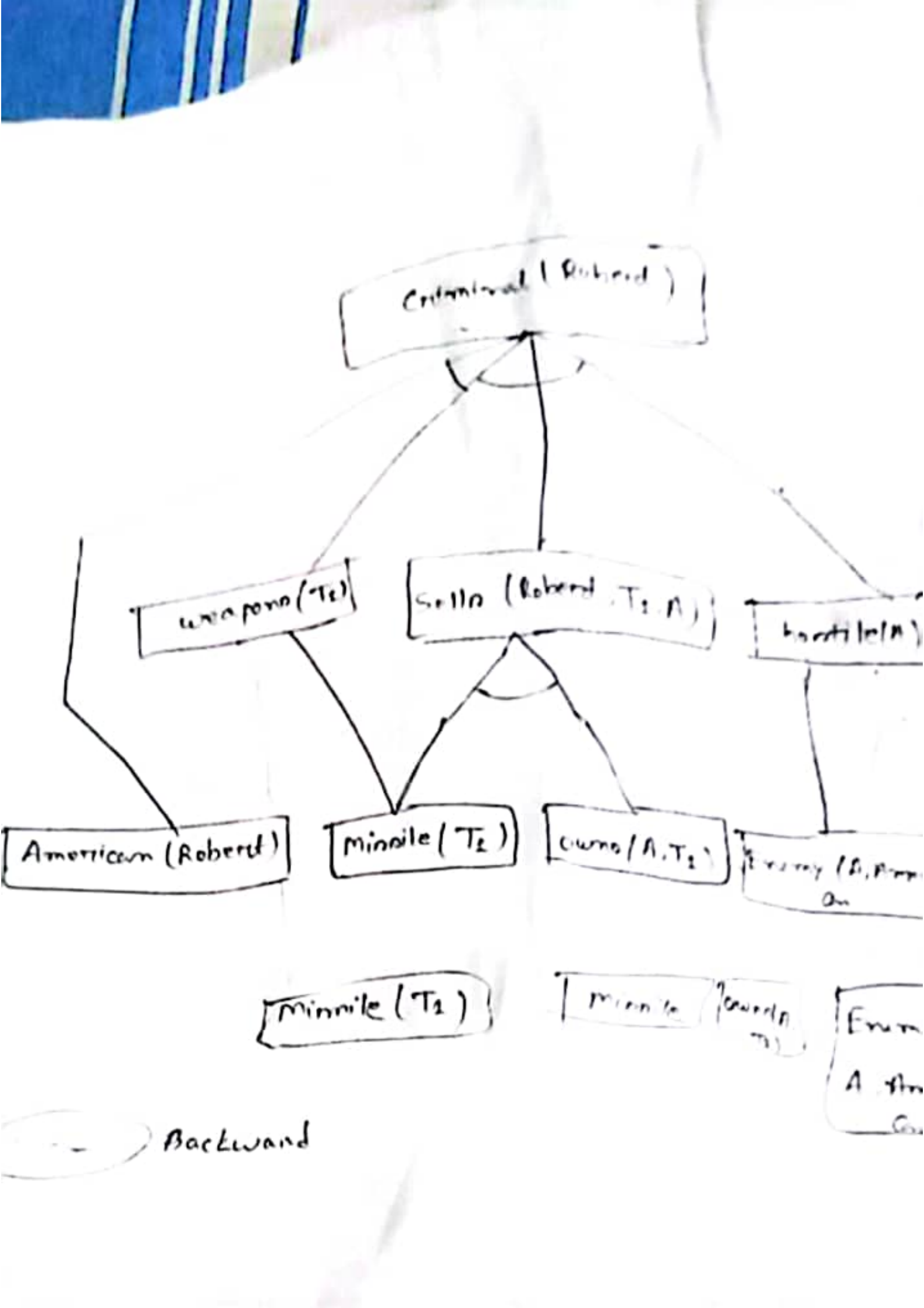
Enemy (P, America) → Hostile (P). ... (6)

(6) Country A is an enemy of America.

Enemy (A, America) ... (7)

(7) Robert is American

American (Robert). ... (8)


Criminal (Robert)
weapons (T2)
Sells (Robert, T2, A)
hostile(A)
American (Robert)
Missile (T2)
owns (A, T2)
Missile (T1)
Backward

(last-second class.)

(Sunday)

ML general setting: * set of possible instances, x

* " " labels, y

* unknown target func, $f: x \to y$

* set of function hypotheses, $H = \{h \mid h: x \to y\}$ each h in decision tree.

Input: Training examples of unknown target function of

$$\{(x_i, y_i)\}_{i=1}^{n} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

output: hypothesis $h \in H$ that best approximates f.

Patient:

Age	Edu	Prof	D
65	High	Driver	Y
30	PHD	Program	N

How would we choose which feature to split the data?

Random: choose any feature at random.

least value: choose the feature with the fewest possible values.

Most-values: choose the features with most possible value

(impurity → mixed) ← Pos/neg classes

Max Gain: choose the feature with the largest expected (information gain.) ✓✓

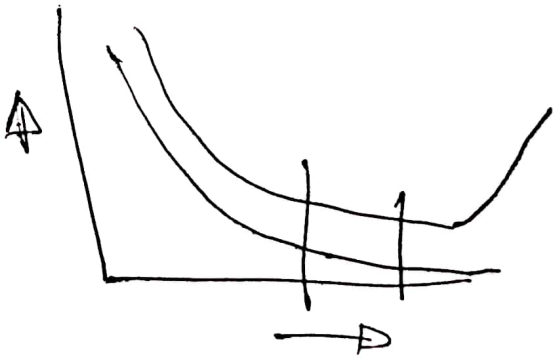
Learning bias: Ockham's Razor

ML → "The simplest consistent explanation is the best."

✓ Entropy (to measure impurity) $H(Y) = -\sum P(X=i) \log_2 P(X=i)$

Info gain $(X, \text{High BP}) = \text{Info}(X) - \text{Info}(X \mid \text{High BP})$

✓ Overfitting & Underfitting:



- K fold cross validation Method

syllabus

✓ Propositional logic
- → T/F
- → rules...
- → inference
- → resolution

✓ Predicate logic
✓ Unification applying resolution.
✓ backward/forward chaining. (robert cross)
✓ info Gain] ML.