Reg No: 1BGAEC9064

Spoorthi Hebbar

Date Page

AI ASSIGNMENT

(1) Q. Define AI. Explain its applications.

Artificial intelligence is the simulation of human intelligence processes by machines esp. computer systems. These processes include learning, reasoning & self correction.
Ex: Speech recognition / machine vision.

Applications:

→ Diagnosis: Medical diagnosis programs based on probabilistic analysis have been able to perform at the level of an expert physician in several areas of medicine

→ AI in gaming: The AI machines can play strategic games like chess where machine needs to think of a large number of possible positions

→ AI in finance: The finance industry is implementing automation, chatbot, adaptive intillegence, algorithm trading and ML into financial processes

→ AI in data security: The security of data is crucial for every company and cyber attacks are growing rapidly AI can be used to make the data more safe & secure

→ AI in Social media: sites have billions of user profiles which need to be stored & managed efficiently. AI can organize & manage massive amounts of data & also analyse the data to identify latest trends, hashtag & requirements of different users.

(1) Q What is state space search? Explain its strategies.

State space search is a process used in the field of computer science, including AI in which successive configurations or states of an instance are considered with the intention of finding a goal state with a desired property.

Problems are often modelled as a ~~set of~~ statespace, a set of states a problem can be in. These sets form a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

It is implicit, ie the nodes are generated as they are explored & typically discarded thereafter. A solution is either a goal state itself or a path from some initial state to the goal state.

Strategies:

a Forward Chaining / Data driven Search: Takes facts of problem and applies rules or legal moves to produces new facts leading to goal.

b Backward Chaining / Goal driven search: uses information about goal, finds rules required to produce a goal & conditions necessary for those rules. These conditions become new subgoals.

c. Backtrack, Algorithm: stores unprocessed states, dead states & nodes to keep track of which have been searched, which are useful & not useful to avoid repetitive looping.

d BFS: It starts from the initial state & explores all the neighboring nodes at the present depth before

moving on to the nodes at next depth level.

c DFS : It also start at the initial state & explores as far as possible along each branch before backtracking.

(1) a Implement Heuristic evaluation functions with Example.

A Heuristic is a function that ranks alternatives in various search algorithms at each branching step basing on an available information in order to make a decision which branch is to be followed during a search.

Ex: The 8-puzzle

7	2	4
5		6
8	3	1

Start state

	1	2
3	4	5
6	7	8

Goal state

The objective of the puzzle is to slide the tiles horizontally or vertically into the empty space until the configuration matches the goal state. The average cost for a randomly generated 8 puzzle instance is about 22 steps.

The branching factor is about 3 (when the empty tile is in centre there are 4 possibilities & when its in corner there is 2, when it is along edge it is 3). This means an exhaustive search to depth would look at 3×22 states.

By keeping track of repeated states, we could cut this down to only $9!/2$ states that is reachable.

~~$f(N) \neq$~~ $h(N) \geq 0$. The value is independent of current tree.

$f(N) = g(N) + h(N)$. $g(N) \rightarrow$ cost

$f(N) = h(N)$ = number of misplaced tiles.

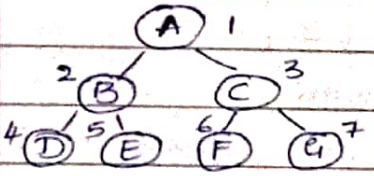
$f(N) = \Sigma$ distances of numbered tiles to their goals.

which is $\frac{9!}{2}$.

(1) Q Explain DFS and BFS with example

BFS is a simple strategy in which the root node is expanded first, then all the successors of the root node and then their successors and so on. In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level is expanded.

Breadth first search is an instance of the general graph search algorithm in which the shallowest node (unexpanded) is chosen for expansion. This is achieved using a FIFO queue for the frontier. Thus, new nodes go back to the back of the queue & old shallower nodes get expanded first

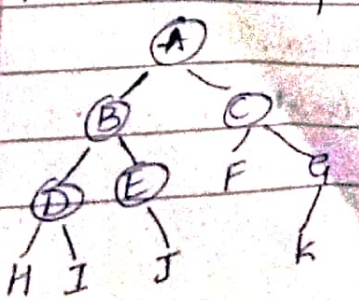


| A | B | C | D | E | F | G |

Depth first search.

DFS always expands the deepest nodes in the current frontier of the search tree. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. As the nodes get expanded they are dropped from the frontier, so then the search backs up to the next deepest node that still has unexplored successors.

DFS uses an LIFO Queue i.e most recently generated node is chosen for expansion This must be the deepest node bcuz it is one deeper than its parent



A B D H I E J C F G K

Date _____ Page _____

(2) Q Write the pseudo code definition for a alpha beta pruning

```
function AlphaBeta (state) returns an action
  v ← MaxValue(state, -∞, +∞)
  return the action in Actions(state) with value v

function MaxValue (state, α, β) returns a utility value
  if TerminalTest (state) then return Utility (state)
  v ← -∞
  for each a in Actions(state) do
    v ← MAX (v, Minvalue (Result (s, a), α, β))
    if v ≥ β then return v
    α ← Max(α, v)
  return v

function MinValue (state, α, β) returns a utility value
  if TerminalTest (state) then return Utility (state)
  v ← ∞
  for each a in Actions(state) do
    v ← Min(v, MaxValue (Result (s,a), α, β))
    if v ≤ α then return v
    β ← Min (β, v)
  return v.
```

(2) Q Explain the minimax algorithm.

```
function minimax (node, depth, maxPlayer)
if depth == 0 or node is a terminal node then
return static evaluation of node

If MaxPlayer then
maxEva = -∞
for each child of node do
eva = (minimax (child, depth-1, false)
maxEva = max (maxEva, eva)
return maxEva

else
minEva = +∞
for each child of node do
eva = minimax (child, depth-1, true)
minEva = min (minEva, eva)
return minEva
```

Minimax algorithm is a recursive algo used in decision making & game theory. In this algo two players play the game one is called max & the other min. Both players fight as the opponent gets minimum benefit & they get max profit.

Minmax algo performs a DFS for the exploration of complete game tree. The algo proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion

Q 8 Explain Constraint satisfaction problems with ex

A problem that is solved when each variable has a value that satisfies all the constraints on the variable is called CSP.

A CSP consists of three components X, D, and C:

X is a set of variables $\{X_1 \dots X_n\}$

D is a set of domains $\{D_1 \dots D_n\}$ one for each X

C is a set of constraints that specify allowable combination of values

Each D_i has set of allowable values $\{v_1 \dots v_n\}$ for variable X_i

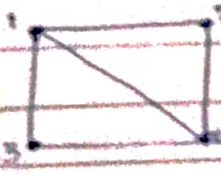
C_i = (Scope, rel)

Scope is a set of variables that participate in constraint

Rel is relation that defines the values can take

$C_1 = ((v_1, v_2), (v_1 \neq v_2))$

Ex: Map Coloring



V = {1, 2, 3, 4}

D = {Red, Green, Blue}

C = {1≠2, 1≠3, 1≠4, 2≠4, 3≠4}

(Without Heuristic we have 3^4 assignments)

	1	2	3	4
Int	RGB	RGB	RGB	RGB
1=R	R	GB	GB	GB
2=G	R	G	GB	B
3=B	R	G	B	Empty(X) Backtrack
3=G	R	G	G	B

Without CSP 3^5 assignments

With Heuristic 2^4 assignments.

(The centre with highest degree is assigned)

(Not QP)

(2) Q Explain Hill Climbing algorithm.

It is a loop that continually moves in the direction of increasing value (uphill). It ~~reaches~~ terminates when it reaches a peak where no neighbour has higher value. It doesnot maintain a search tree so the data structure for the current node only records the state & value of the obj fution. It doesnot look ahead beyond the immediate neighbours of current state (No backtracking) (also called greedy local search).

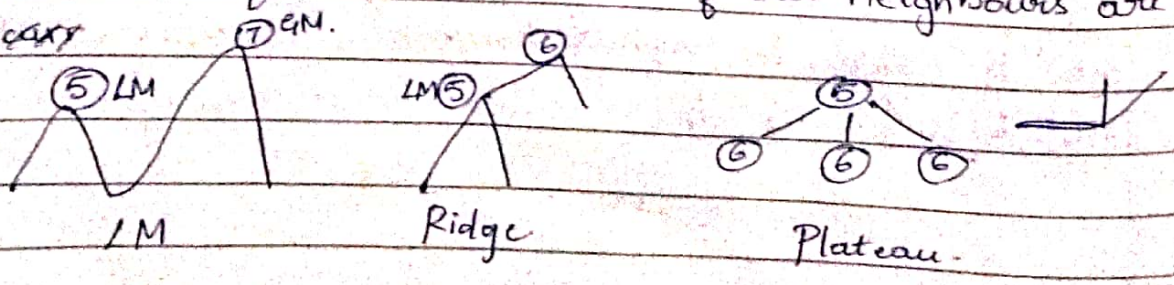
function HillClimbing(problem) returns local maximum state
current ← Makenode(problem, Initialstate)
loop do
neighbor ← Highest valued successor of current
if neighbour.value ≤ current.value then current ← neighbour.
return current.State

Disadvantages

→ Local maxima: It can be a peak higher than each of its neighboring states but lower than global maximum. Hill climbing algo gets stuck after reaching local maximum hence cant go to global max

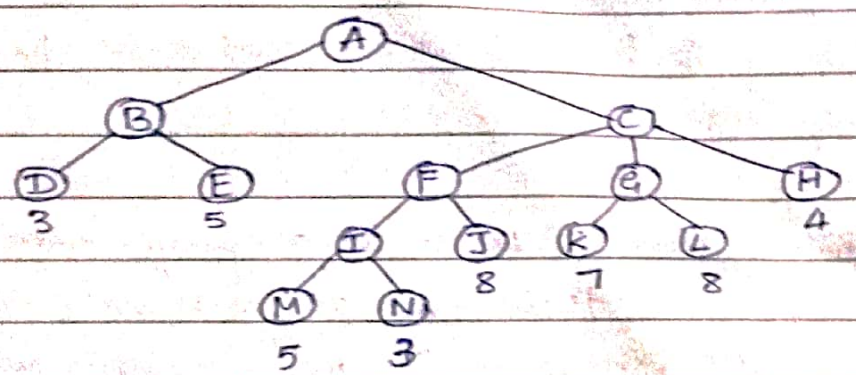
→ Ridges: Ridges result in sequence of local maxima that is difficult for greedy algos to navigate.

→ Plateaux: It can be flat local maximum from which no uphill exists or a shoulder from which progress is possible but is stuck for ex when value of all neighbours are equal ~~by chance~~



Date Page

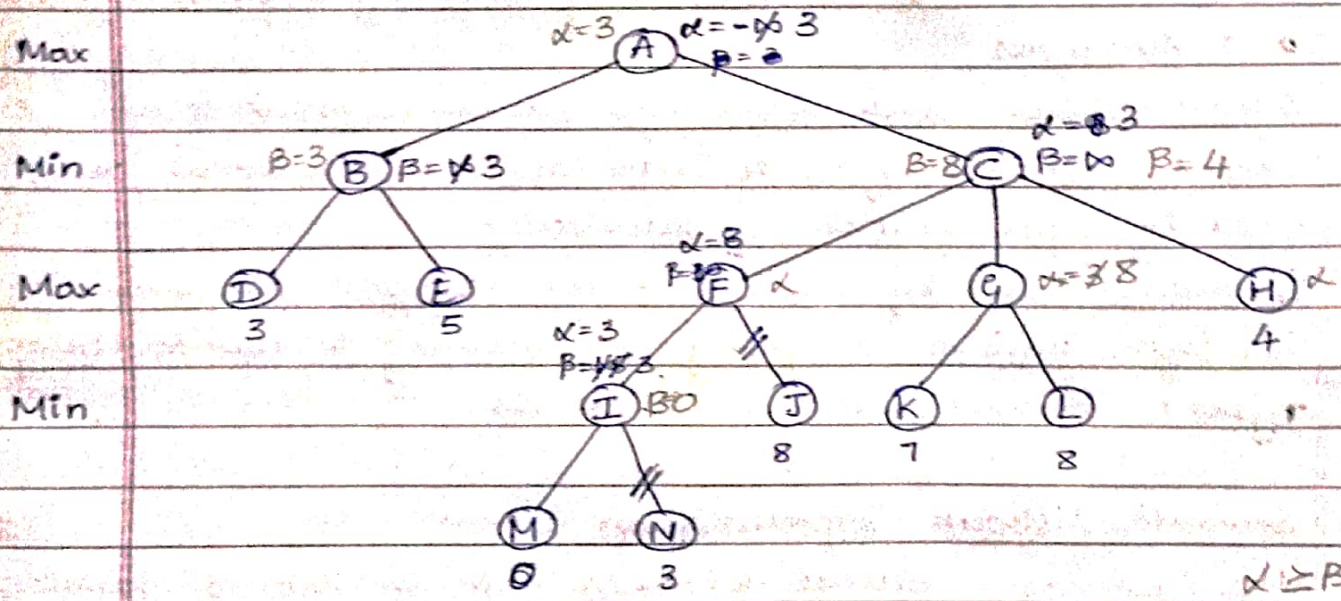
(2) Q. Perform a left to right α-β pruning on the tree



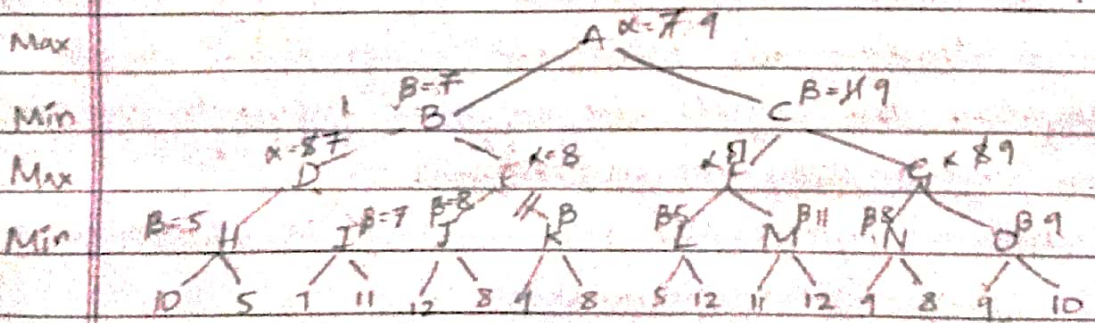
$\alpha = -\infty$

$\beta = \infty$

$\alpha \geq \beta$ (prune)



$\alpha \geq \beta$ -prune



(U3) Q Explain different methods of representing knowledge

Knowledge representation in AI is concerned with AI agents thinking & how thinking contributes to intelligence behaviour of agents.

4 major ways:

1. Logical Representation: It is a language with some concrete rules which deals with propositions & has no ambiguity in representation. It draws conclusion based on various condition. It consists of precisely defined syntax and semantics.

Syntax: Rules which decide how we can construct legal sentences in the logic, it determines which symbol can be used in representation of knowledge

Semantics: Rules by which we can interpret the sentence in the logic, involves assigning a meaning to each sentence.

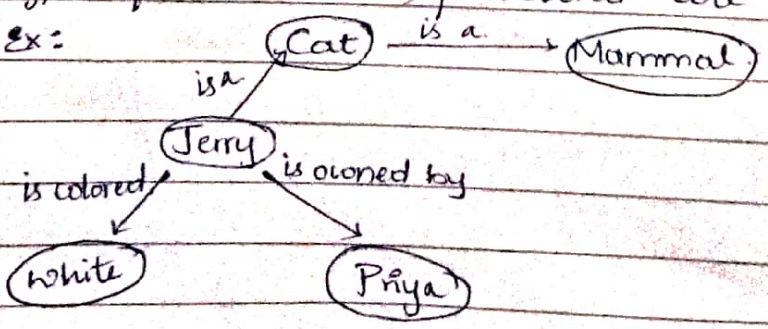
Types: Propositional & Predicate.

2 Semantic Network Representation.

Here we can represent the knowledge in form of graphical networks. This n/w consists of nodes representing objects & arcs which describe the relationship b/n those objects. They categorize the objects in different forms & can also link those objects. Easy to understand.

Types of relations represented are usually is-a/inheritance

Ex:



Date Page

3. Frame Representation

It's a record like structure which consists of a collection of attributes & its values to describe an entity in the world. Frames are AI data structures which divides knowledge into substructures by representing stereotypes, situations. It consists a collection of slots & slot values. They have names & values which are called facets

Frame system consists a collection of frames which are connected to form a knowledge base.

Ex: Peter is an engineer & his age is 25 He lives in London, England.

Slots	Filler
Name	Peter
Profession	Engineer
Age	25
City	London
Country	England

4. Production Rules

This system consists of (condition, action) pairs i.e "If condition then action". It has mainly 3 parts

1) Set of production rules : Agent checks for condition. If true then action is carried out. This process is called recognize act cycle.

2) Working m/m : contains the description of current state of problem solving & rule can write knowledge to the m/m. This may match / fire other rules

3. Recognize act cycle : If there is a new (state) situation then multiple production rules will be find together rising a conflict

Ex. IF (at bus stop AND bus arrives) THEN (get into the bus).

(3) Q Explain Frames & slots.

A Frame consists a collection of attributes & its values to describe an entity in the world

Frames are an AI data structure used to divide knowledge into substructures by representing "stereotyped situations"

They are stored as ontologies of sets.

It consists of collection of slots & slot values.

Each piece of information about a particular frame is held in a slot. The information can contain

- Facts or Data : Values are called facets.
- Procedures : IF-NEEDED : deffered evaluation
  IF-ADDED : updates linked evaluation
- Default values : for data / for prodedures
- Other frames or subframes

All these frames related together forms a knowledge base / database.

Ex

Slots	Values
Name	ABC
Age	12
Location	XYZ
Profession	Doctor.

(3) Q Explain scripts and its components

A script is a structured representation describing a stereotyped sequence of events in a particular context. Scripts are used in NL understanding systems to

Date Page

Organise a knowledge base in terms of the situations that the system should understand. Scripts use a frame like structure to represent the commonly occuring experience like going to movies, shopping etc.
Thus a script is a structure that prescribes a set of circumstances that could be expected to follow on from one another.

Components

* Entry condition : There are basic condition which must be fulfilled before events in the script can occur
* Results : Condition that will be true after events in script occurred
* Props : Slots representing objects involved in events
* Roles : Actions that the individual participants perform
* Track : Variations on the script.
* Scenes : The sequence of events that occur.

Ex

Symbol	Meaning	Example
ATRANS	transfer a relationship	give
PTRANS	transfer physical location of object	go
PROPEL	applying force	push
MOVE	move body part by owner	kick
GRASP	Grab object	hold
MTRANS	Transfer mental info	tell
MBUILD	mentally make new info	decide
CONC	conceptualize	think
SPEAK	produce sound	say

(3) Q Explain unification algo in FOL.

Finding substitutions that make different logical expressions look identical is called unification.

UNIFY $(P(x, F(y)), P(a, F(g(z)))) = \{a/x, g(z)/y\}$

UNIFY $(P(x, a), P(y, b)) = \{y/x, b/a\}$.

Algo : Unify (L_1, L_2)

Step 1 : If L_1 ~~and~~ or L_2 is a variable or constant then:

a) If L_1 and L_2 are identical then return NIL

b) Else if L_1 is a variable
then if L_1 occurs in L_2 then return FAIL
else return $\{(L_2/L_1)\}$

c) Else if L_2 is a variable
If L_2 occurs in L_1 then return FAIL
else return $\{(L_1/L_2)\}$

d) Else return FAIL

Step 2 : If the initial predicate symbol in L_1 & L_2 are not same then return FAIL

Step 3 : If L_1 & L_2 have different number of arguments then return FAIL

Step 4 : Set Substitution set θ to NIL

Step 5 : For $i=1$ to no of arguments in L_1

a) Call unify function $(L_1(i), L_2(i), S)$

b) If S = failure then return Failure

c) If $S \neq$ NIL

a. Apply S to remainder of both L_1 & L_2

b. θ = APPEND (S, θ)

Step 6 : Return θ.

Date Page

(4) Q Differentiate between forward and backward chaining

	Forward Chaining	Backward Chaining
*	It starts from known facts & applies inference rule to extract more data until it reaches goal	It starts from the goal & works backward through inference rules to find required facts the support the goal
*	Bottom up approach	Top down approach
*	Data driven inference technique	Goal driven technique
*	Applies a BFS strategy	Applies DFS strategy
*	Tests for all available rules	Tests only few rules that infer the goal
*	Can generate infinite no of possible conclusions	Generates finite number of possible conclusions
*	Suitable for planning, control, monitoring & interpretation	Suitable for diagnostic, debugging, prescription etc
*	Slow as it checks all the rules	Fast as it cheks fewer rules

(4) Q (NSP) What are the different methods of reasoning? Explain any two of them.

* Deductive reasoning
* Inductive reasoning
* Abductive reasoning
* Common sense reasoning
* Monotonic reasoning
* Non monotonic reasoning.

1. Deductive reasoning: is deducing new information from logically related known information. It is the form of valid reasoning, which means the argument's conclusion

must be true when the premises are true.
Deductive reasoning is a type of propositional logic in AI that requires various rules and facts. It is sometimes refferred to as top down reasoning. In deductive reasoning the truth of premises guarantees the truth of the conclusion. It starts from the general premises to the specific conclusion, ~~as~~

Ex. P1: All humans eat vegetables
P2: Suresh is human
Conclusion: Suresh eats vegetables.

Inductive Reasoning: is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalisation. It starts with the series of specific facts or data and reaches to a general statement or conclusion. It is also known as cause-effect reasoning or bottomup reasoning.
We use historical data or various premises to generate a generic rule for which premises support the conclusion
Truth of premises donot guarantee the truth of conclusion
Ex: All the pigeons we have seen in the zoo are white
Conclusion: Therefore we can expect all the pigeons to be white.

(4) Q (NBP) Explain Hierarchial planning with algorithm

function Hierarchial_search(problem, hierarchy) returns a solution, or failure
frontier ← a FIFO queue with [Act] as the only element

Date Page

loop do
if Empty?(frontier) then return failure
plan ← POP(frontier)
hla ← the first HLA in plan, or null if none
prefix, suffix ← the action subsequences before & after hla in plan
outcome ← RESULT(problem, initial_state, prefix)
If hla is null then
  If outcome satisfies problem.GOAL then return plan
else for each sequence in REFINEMENTS(hla, outcome, hierarchy) do
  frontier ← INSERT(APPEND(prefix, sequence, suffix), frontier)

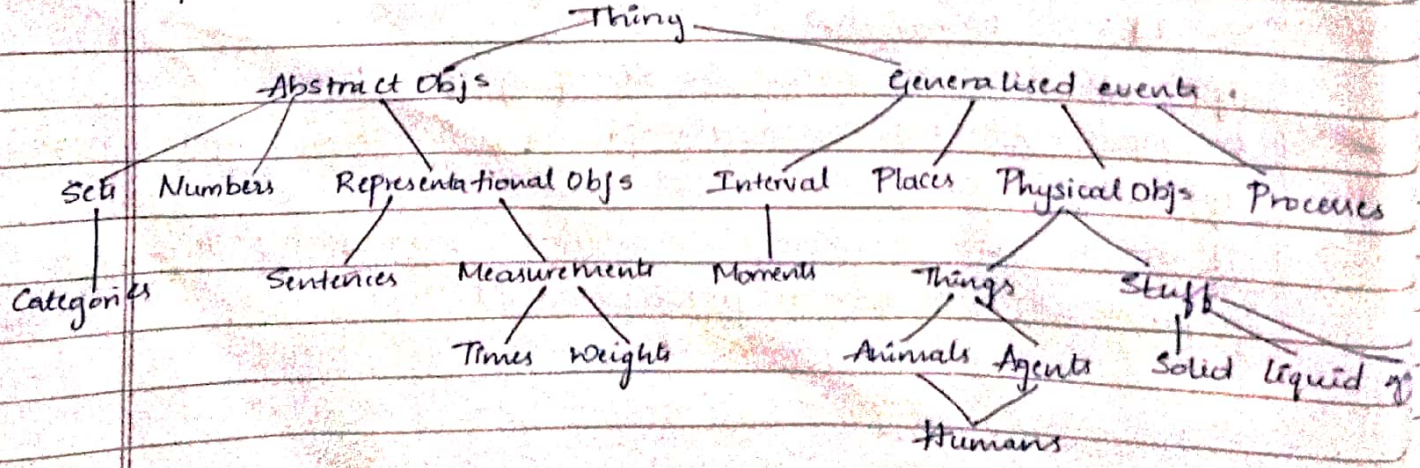
The initial plan supplied to the algorithm as [ACT].
The REFINEMENTS function returns a set of action sequences, one for each refinement of the HLA whose preconditions are satisfied by the specified state, outcome.

(14) Q Explain graph plan algorithm

function GRAPHPLAN(problem) returns solution or failure
graph ← INITIAL_PLAN_GRAPH(problem)
goals ← CONJUNCTS(problem.GOAL)
nogoods ← an empty hash table
for tl = 0 to ∞ do
if goals all non-mutex in St of graph then
solution ← EXTRACT_SOLUTION(graph, goals, NUMLEVELS(graph), nogoods)
if solution != failure then return solution
if graph and nogoods have both leveled off then return failure
graph ← EXPAND_GRAPH(graph, problem)

The GRAPHPLAN algorithm repeatedly adds a level to a planning graph with EXPANDGRAPH
Once all the goals show up as non mutex in graph, GRAPHPLAN calls EXTRACTSOLUTION to search for a plan that solves the problem
If that fails it expands another level & tries again, terminating with failure when there is no reason to go on

(4) & (NAP) Explain Ontological engineering with diagram
The representations concentrating on general concepts such as Events, Time, Physical Objects and beliefs that occur in many different domains is called ontological engineering.
Ontology engineering aims at making explicit the knowledge contained within software applications & enterprises for a domain
It helps in solving the interoperability problems brought about by semantic obstacles.
Ontological engineering is a set of tasks related to the development of ontologies for a particular domain
The general framework of concepts is called upper ontology because of convention of drawing graphs with general concepts at the top & more specific concepts below them.



Date Page

(5) Q. What is explaination based learning? Explain with Example (NOT)

Explaination based learning is a form of ML that exploits a very strong or even perfect domain theory in order to make generalisatio-ns of form concepts from training examples.

EBL works by finding a way to deduce each training example from the system's existing database of domain theory.

Having a short proof of the training example extends the domain theory database enabling the EBL system to find & classify future examples that are similar to the training example very quickly

EBL software takes 4 inputs :

a hypothesis space (set of all possible conclusions)

a domain theory (axioms about a domain of interest)

training examples (specific facts that rule out some possible hypothesis)

operationality criteria (criteria for determining which features of domain are efficiently recognisable)

Ex: Program that learns to play chess through example

A specific chess position that contains an important feature such as "Forced loss of black queen in 2 moves" includes many irrelevant features such as the specific scatting of pawns on the board

EBL can take single training example & determine what are relevant features in order to form generalisation.

(5) Q What is reinforcement learning? Explain a) Passive b) A reinforcement learning.

It is an area of ML concerned with how s/w agents ought to take actions in an environment in order to maximize the notion of cumulative reward

In reinforcement learning there is no answer key but the agent decides what to do to perform the given task. In absence of training dataset it is bound to learn from its experience

Passive

In this learning, the agents policy is fixed and the task is to learn the utilities of states. It could also involve learning a model of the environment.

In passive learning, the agents policy is fixed ie the states always execute the action specified.

Its goal is simply to learn the utility function

Passive learning is similar to policy evaluation task. The difference is that passive learning agent doesnot know

* The transition model $T(s, a, s')$ which specifies the probability of reaching state s' from state s after doing action a
* Or the reward function $R(s)$ which specifies the reward of each state.

Active

It's the agents decision to perform a certain task as there is no fixed policy that it can perform

Active agent must decide what actions to do. It must be modified to handle this freedom

Date Page

The modifications are

* The agent will need to learn a complete model with outcome probabilities for all actions.
* Next take into acct the fact that agent has freedom, the utilities it needs to learn are those defined by the optimal policy $U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$

Having obtained the utility function, the agent can extract an optimal action by one step look ahead to maximize the reward. Alternatively, if it uses policy iteration, the optimal policy is already available so it executes the action the policy recommends.

(5) Q Distinguish between supervised & unsupervised learning

Supervised Learning	Unsupervised learning
→ Algorithms are trained using labeled data	Algorithms are trained using unlabelled data
→ Supervised learning model takes direct feedback to check if it is predicting correct o/p or not	These models donot take any feedback.
→ Supervised learning model predicts the output	Unsupervised model finds the hidden patterns in data
→ Input data is provided to the model along with the output	Only input data is provided to the model
→ The goal is to train the model so that it can predict the o/p when new data is given	The goal is to find the hidden patterns & useful insights from the unknown dataset
→ Categories: Classification, Regression	Categories: Clustering, Associations
→ Accurate	Less accurate.

(NRP)

(5) Q Explain learning of decision tree algo & draw its learning curve.

A decision tree represents a function that takes a vector of attribute values as input & returns a decision.

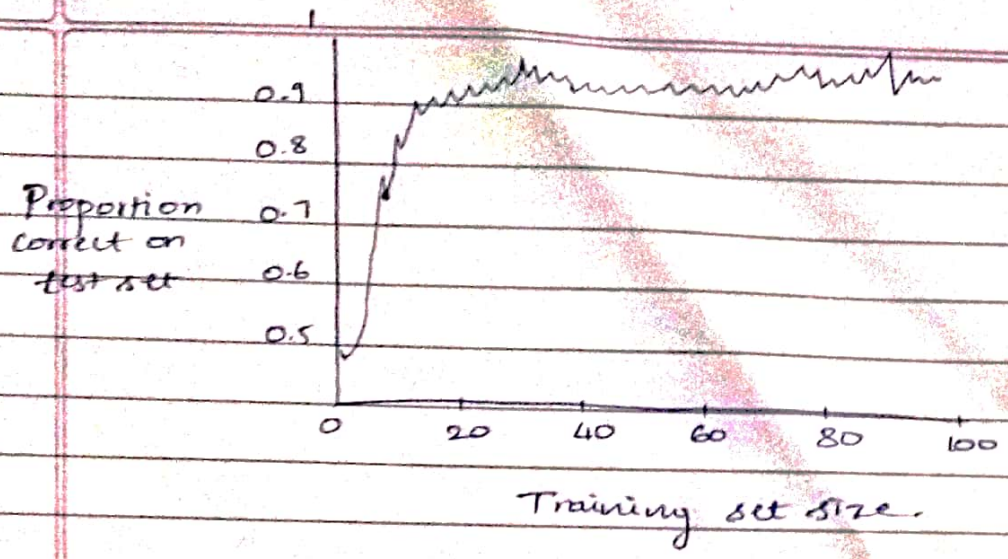
function DECISION TREE LEARNING (examples, attributes, parent_examples)
returns a tree

if examples is empty then return PLURALITY VALUE(parent_examples)
else if all examples have the same classification then return the classification
else if attributes is empty then return PLURALITY VALUE(examples)
else
A ← $\text{argmax}_{a \in \text{attributes}}$ IMPORTANCE (a, examples)
tree ← a new decision tree with root test A
for each value v_k of A do
exs ← {e : e ∈ examples & e.A = v_k}
subtree ← DECISION TREE LEARNING (exs, attributes − A, examples)
add a branch to tree with label (A = v_k) and subtree subtree
return tree

* PLURALITY VALUE selects most common o/p among a set of examples
* IMPORTANCE returns the attribute that is better than the rest ie has maximum gain.

Learning curve for decision tree :

For each size we repeat the splitting 20 times & average the results of 20 trials, the curve shows that as the training set size grows the accuracy ↑ses.



Spoorthi Hebbar
18GAEC9064.