Date:
Page:

References :

1. Artificial Intelligence : A modern Approach, Stuart Russel & peter norvig

2. Artificial Intelligence : Knight, Rich & S.B. Nair

Scanned with CamScanner

Date:
Page:

Artificial Intelligence

Er. Kishor Kumar Giri
MSc. Information System Engineering

Part - 1

unit 1 - 4

Scanned with CamScanner

1.

Date.....
Page.....

" There are three kinds of intelligence: one kind understands things for itself, the other appreciates what others can understand, the third understands neither for itself nor for others. The first kind is excellent, the second is good, and the third kind is useless.

— Niccolo Machiavelli (1469–1527)
Italian diplomat, political philosopher

Scanned with CamScanner

Unit-1 Introduction

Date:
Page:

1.1. Artificial Intelligence (AI), AI perspective

Intelligence → Intelligence is defined as the ability to acquire, understand and apply knowledge or the ability to ~~exersi~~ exercise and reason.
It includes all the knowledge which we have acquired through study and experience, thought, imagination, ability to read, write, memorize and recall facts, express and ~~fee~~ feel emotions.

Formally, intelligence can be defined as follows:

- It is ability to learn from past experiences.
- It is ability of applying knowledge successfully in new situation.
- It is the process of thinking abstractly (to draw conclusion)
- It is ability of reasoning to solve problems
- It has answers for curiosity, creativity and expressiveness.

Scanned with CamScanner

Artificial Intelligence → AI is the branch of computer science concerned with the study of creation of computer system that exhibit some form of intelligence.

It is the study which makes the computer systems to do things which at the moment people do better.

AI can be defined as:

→ System that learn new concepts and tasks (Learning)

→ System that can reason and draw useful conclusions about the world (Reasoning)

→ System that can understand natural language. (NLP)

→ System that perform other types of tasks that require human type of intelligence.

Assignment → Difference between AI & NI

Scanned with CamScanner

Task domain of AI

* Mundane Tasks

perception → vision, speech
Natural language → understanding
→ Generation
→ Translation
Commonsense Reasoning
Robot control

* Formal Tasks

Games → chess, Backgammon, checker, go
Mathematics → Geometry, logic, Integral calculus
proving properties of programs

* Expert Tasks

Engineering → Design, fault finding, Manufacturing planning
Scientific analysis
Medical diagnosis
Financial analysis

Scanned with CamScanner

Characteristics of AI

Symbolic reasoning → Reasoning about the objects represented by symbols, their properties and relationships, not just numerical calculations.

Knowledge → General principles are stored in the program and are used for reasoning.

Search → A weak method for finding a soln to a problem when no direct method exists.

Flexible control → Direction of processing can be changed with the facts in the environment.

Footnotes:

undecidability → The non-existence of an algorithm or the impossibility of proving or disproving a statement within a formal system.

interactability → quality of being interactable (hard to deal)

motor control → process by which human & animal organize and execute their actions.

Syllogism → an instance of reasoning in which a conclusion is drawn from two given or assumed propositions (premises)

Scanned with CamScanner

Foundations of A.I.

philosophy → logic, reasoning, foundations of learning, language and rationality

Mathematics → Formal representation and proof algorithms and probability

Psychology → Adaptation, learning, thinking phenomenon of perception and control.

Economics → Formal theory of rational decisions, operations research and to determine the max^m payoff.

Linguistics → Knowledge representation and grammer.

Neuroscience → physical substrate for mental activities.

Control theory → Stability, optimal agent design and the study of artifacts which operate under their own control.

Scanned with CamScanner

Definition of AI

Dimensions of AI

Different definitions of AI are given by different writers / books.
Definition of AI are categorized in two dimensions. The definn on the top are concerned with thought processes and reasoning, whereas the ones on the bottom address behaviour.

Thinking Humanly	Thinking Rationally
"The exciting new effort to make computers think.... machine with minds, in the full & and literal sense" (Haugeland, 1985)	"The study of mental faculties through the use of computational models". Charniak and McDermott, 1985
"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, ~~lar~~ learning.." (Bellman, 1978)	"The study of the computations that make it possible to perceive, reason, and act. Winston, 1992
"The art of creating machines that perform functions that require intelligence when performed by people." Kurzweil, 1990	"Computational Intelligence is the study of the design of intelligent agents". Poole et al 1998
The study of how to make computers do things at which, at the moment, people are better" Rich & Knight 1991	AI ... is concerned with intelligent behaviour in artifacts." Nilsson 1998

Scanned with CamScanner

Acting humanly : The turing test approach

The turing test approach proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence. A computer passes the test if a human interrogator, after posing some written questions, cannot tell ~~whether~~ whether the written responses comes from a person or from a computer.

The computer would need to possess the following capabilities to pass the turing test.

Natural language processing → to enable it to communicate successfully in english.

Knowledge representation → to store what it knows or hears

automated reasoning → to use the stored information to answer questions and to draw new conclusions;

Machine learning → To adapt a new circumstances and to detect and extrapolate patterns.

Turing test deliberately avoided direct physical interaction between the interrogator and the computer, because physical simulation is unnecessary for intelligence.

Scanned with CamScanner

Total turing test → Total turing test includes a video signal so that the interrogator can test the subject's perpectual abilities, as well as the opportunity for the interrogator to pass physical objects "through the hatch".

To pass the total turing test, computer will need

⇒ Computer vision → to perceive objects, and

⇒ robotics → to manipulate objects and move about.

Thinking humanly : The cognitive modeling approach

If we are going to say that the given program thinks like human, we must have some way of determining how humans think. We need to get inside the actual working of human mind.

There are two ways of doing that

1. Through introspection → Catch our thoughts while they go by.

2. Through psychological experiments → once we have precise theory of mind, it is possible to express the theory as computer science.

→ Cognitive science brings computer models from AI & experimental techniques from psychology to construct testable theories

Scanned with CamScanner

Thinking Rationally: The law of thought approach.

Aristotal was one of the first who attempt to codify the right thinking that is irrefutable reasoning process. He gave syllogisms that always yielded correct conclusion when correct premises are given.

for example: Ram is a man
All men are mortal
$\Rightarrow$ Ram is mortal

These law of thought were supposed to govern the operation of mind: The study initiated the field of logic. The logicist tradition in AI hopes to create intelligent systems using logic programming.

problems with this approach $\rightarrow$

1. It is not easy to take informal knowledge and state in the formal terms required by logical notation, particularly when knowledge is not 100% certain.

2. Solving problem principaly is different from doing it in practice. Even problems with certain dozens of facts may exhaust the computational resources of any computer unless it has some guidance as which reasoning step to try first.

Scanned with CamScanner

Acting Rationally : The rational Agent Approach

Agent is something that acts.
Computer agent is expected to have following attributes.

→ Autonomous control
→ perceiving their environment
→ persisting over a prolonged period of time.
→ Adapting to change.
→ And capable of taking one another's goal.

Scanned with CamScanner

Brief History of AI

1943 : Warren Mc Culloch and Walter pitts
— A model of Artificial boolean neurons to perform computations.

— first steps towards connectionist computation & learning (Hebbian learning)

1950 : Alan Turing's " Computing Machinary & Intelligence.
— First complete vision of AI

1956 : Dartmouth workshop
— .

1952 — 1969

— Newell and simon ⇒ General problem solver
- Imitation of human problem solving

1952 - Arthur Samuel ⇒ investigated game playing (checker)

1958 - John McCarthy ⇒ Lisp language

1958 - Marvin Minsky ⇒ microwords

Scanned with CamScanner

1966 – 1973 – collapse in AI research

- unrealistic prediction
- ~~lacked~~ some system lacked scalability
- Combinatorial explosion in search

1969 – Minsky & Papert ⇒ perceptrons

1969 – 197 ⇒ AI revival through Knowledge Based systems

MYCIN → to diagnose blood infection

- Introduction of uncertain reasoning

Increase in Knowledge representation research
R1 at DEC – 1982
- production – rule based system.

Connectionist revival – 1986 – present)
⇒ Return of Neural Networks

Scanned with CamScanner

1987 — present : AI becomes science

- speech recognition
- Neural Networks
- uncertain reasoning & expert systems

1995 — present : Emergence of intelligent agents.

- The whole agent problem
- How does an agent act

Scanned with CamScanner

Applications of AI

- Game playing
- speech recognition
- Understanding Natural language
- Computer vision
- Expert system
- E-Commerce
- Heuristic classification

And many more areas.

[page -16 to 23 intentionally left blank]

Scanned with CamScanner

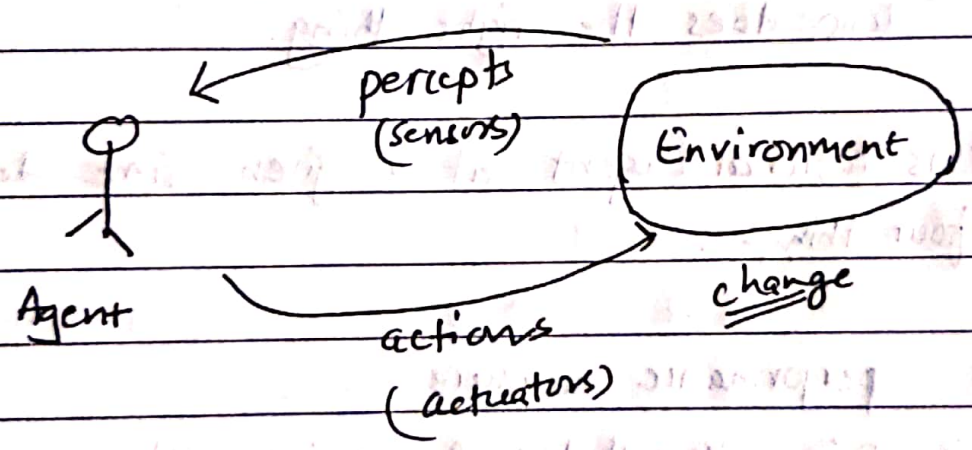
Intelligent Agents

Introduction of agents

→ An intelligent agent perceives its environment via sensors and acts rationally upon that environment with its effectors (actuators). Hence an agent gets percepts one at a time, and maps this percept sequence to actions.

properties of agent

→ Autonomous
→ Interacts with other agents plus the environment
→ Reactive to the environment
→ pro-active (goal-oriented



Agent terms

performance measure → it is the criteria, which determines how successful an agent is.

Behaviour of an agent → It is the action that agent performs after any given sequence of percepts.

Scanned with CamScanner

percepts → It is agents perpectual input at a given instance.

percept sequence → it is the history of all that an agent has perceived till date.

Agent function → It is a map from the percept sequence to an action.

Rationality → Rationality is the status of being reasonable, sensible, and having good sense of judgement.

Rational agent → A rational agent is the one, who does the right thing.

What is rational agent at a given time depends on four things:

→ performance measure
→ Prior knowledge of environment
→ Actions.
→ percepts sequence to date (sensors)

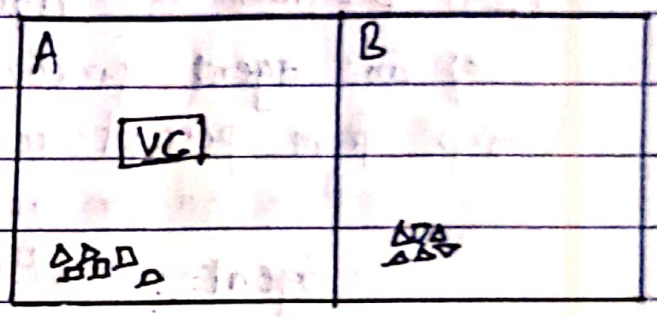
Scanned with CamScanner

Rational Agent

→ A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date and prior environment knowledge.

→ A rational agent is the one who does the right thing.

An Example of agent: The vacuum cleaner world.



Environment: square A and B

percepts : [location and contents] [A, dirty]

Actions : left, right, suck and no-op.

Scanned with CamScanner

Structure of Intelligent Agent

→ The intelligent agent structure is the combination of Agent Function, Architure & Agent program.

Architecture → Architecture is the machinery on which the agent executes its action. It is a device with sensors and actuators, for example: a robotic car, a camera, a pc.

Agent program → Agent program is an implementation of an agent function. An agent function is a map from percept sequence (agent perceived history)

Agent = Architecture + Agent program

Forms of Agent

Human Agent → A human agent uses eyes, Nose, Tongue and other sensory organs as sensors to percept information from the environment and uses limbs and vocal tract as actuators to perform an action based on the information.

Robotic Agent → Robotics Agents uses cameras and infrared radars as sensors to record information from the Environment and it uses reflex motors as actuators to deliver output back to environment.

Scanned with CamScanner

software agent → S/w agent uses keypad strokes, audio commands as input sensors and display screen as actuators : Siri, alexa.

properties of Intelligent Agent

→ properties can be classified into internal and external

Internal properties :

Learning / reasoning : An agent has the ability to learn from previous experience and adapt the change.

Reactivity : An agent must be capable of reacting appropriately to influences or information from its environment.

Autonomy : An agent must have both control over its actions and internal states.

Goal oriented : An agent has well-defined goals and gradually influence its environment and so achieve its own goals.

External properties

Communication : Cooperation of several agents permits faster and better solutions for complex tasks that exceeds the capabilities of a single agent.

mobility → An agent may navigate within electronic

Scanned with CamScanner

communication networks.

character → an agent may demonstrate an external behavior with many human characters as possible.

PEAS Description of Agents

To design a rational agent we must specify its task & environment. Task environment means: PEAS description of the environment

→ Performance
→ Environment
→ Actuators
→ Sensors

Example: Fully automated taxi

Performance: safety, destination, profits, legality, comfort

Environment: streets/freeways, other traffic, pedestrians, weather ...

~~Actuators~~ Sensors: Video, sonar, speedometer, engine sensors, keyboard, GPS etc.

~~Sensors~~ Actuators: steering, ~~sonar~~, ~~speedometer~~, ~~engine~~ accelerating, brake, horn, speaker/display

Assignment: Medical diagnosis system, part-picking robot, humans.

Scanned with CamScanner

Environment types

Accessible / Inaccessible (Fully observable / partially observable)

→ If an agent's sensors give it access to the complete state of the environment needed to choose an action, the environment is accessible.

Example → Room with temperature as state [accessible]

→ Such environment are convenient, since the agent is freed from the task of keeping track of the changes in the environment.

Example: Event on earth [inaccessible]

Deterministic / Non-deterministic (Stochastic)

→ An environment is deterministic if the next state of the environment is completely determined by the current state of the environment and the action of the agent.

→ In an accessible and deterministic environment the agent need not to deal with uncertainty.

Episodic / Nonepisodic

→ An episodic environment means that subsequent episodes do not depend on what actions occured in previous episodes.

→ Such environment do not require the agent to plan ahead.

Scanned with CamScanner

Static / Dynamic

- A static environment does not change while the agent is thinking
- In a static environment the agent need not worry about the passage of time while he is thinking, nor does he has to observe the world while he is thinking.

→ In static environments the time it takes to compute a good strategy doesnot matter.

Example → cross word puzzle [static], car driving [dynamic]

Discrete / Continuous

→ If the number of distinct percepts and actions is limited, the environment is discrete, otherwise it is continuous.

self driving car [continuous], chess game [discrete]

with/without rational adversaries (Single agent / Multi-agent)

→ If an environment does not contain other rationally thinking, adversary agents, the agent need to worry about strategic, game theoretic aspects of the environment.

→ most engineering environments are without rational adversaries, whereas most social and economic systems get their complexity from the interactions of (more or less) rational agents.

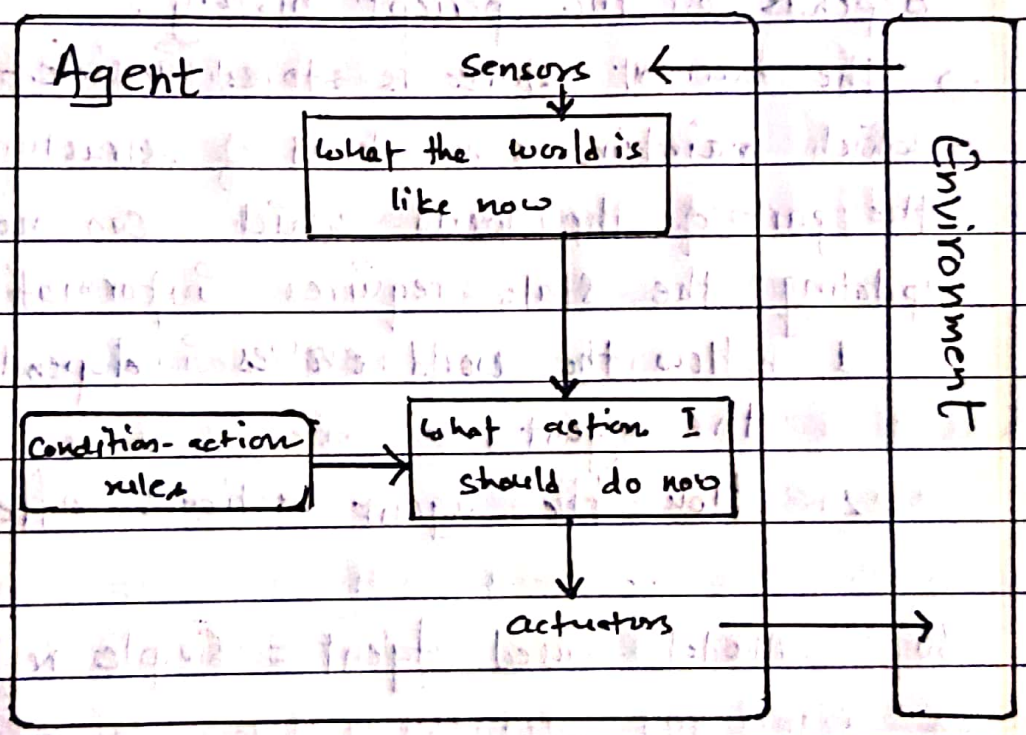
Scanned with CamScanner

Types of Agents

Simple Reflex Agent

→ The agent selects the actions on the basis of current situation ignoring all the percept history. Eg: Eye starts blinking when something approaches to it.

→ It is easily implementable with if then (condition-actions rules) construct available in programming. It is also called table lookup approach of agent design.



problems

→ very limited intelligence

→ no knowledge of non-perpectual part of states

→ usually too big to generate & store.

→ if any change occurs in environment, then collection of rules need to be updated.

Scanned with CamScanner

Model-based Reflex Agents

→ The model based reflex agent considers the knowledge of the world (i.e How it works and also the possible actions of the agent) and then acts accordingly.

→ A model based agent can handle partially observable environments by use of model about the world.

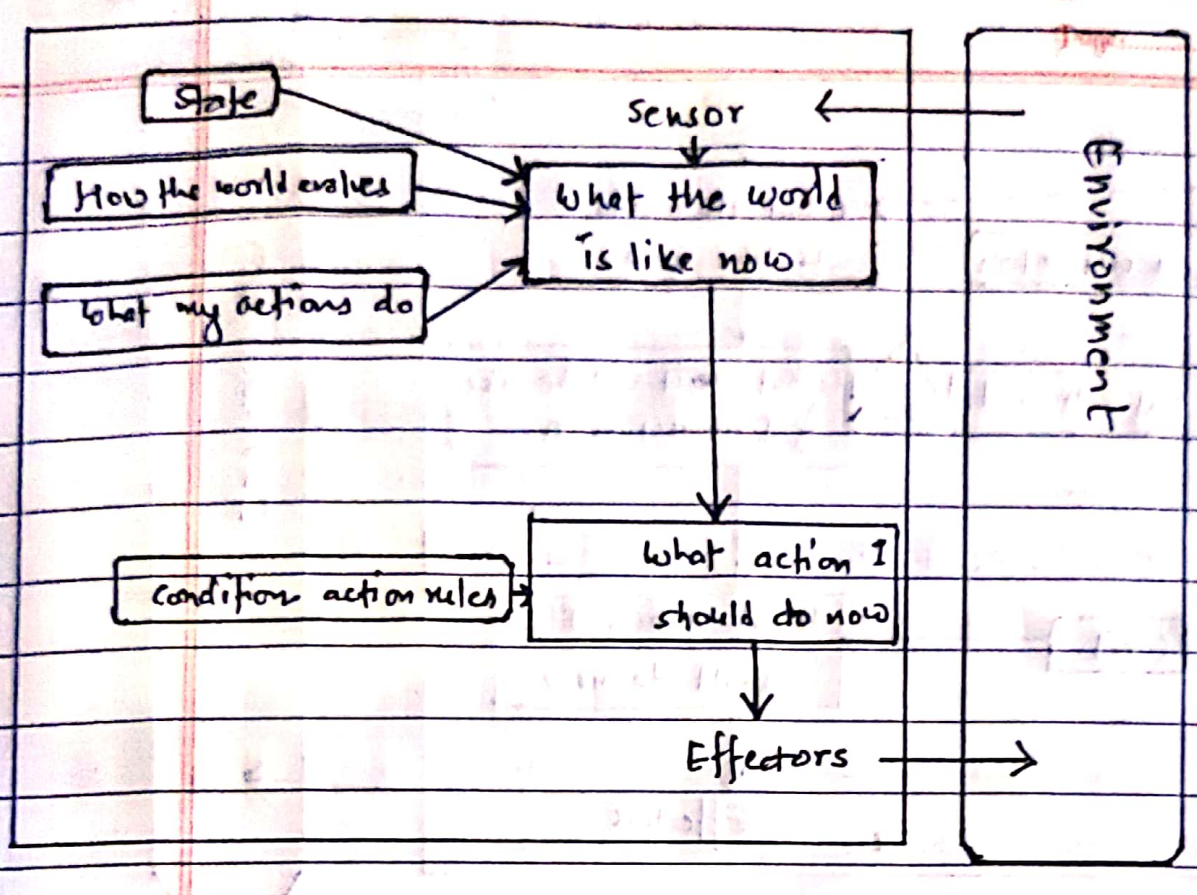
→ The agent has to keep track of internal state which is adjusted by each percept and that depends on the percept history.

→ The current state is stored inside the agent which maintains some kind of structure describing the part of the world which can not be seen.

Updating the state requires information about:

1. How the world evolves independently from the agent,
2. How the agent actions affect the world.

So, Model based Agent = Simple reflex agent + model



Goal Based Agents

→ These kind of agents take decision based on how far they are currently from their goal.

→ Their every action is intended to reduce its distance from goal.

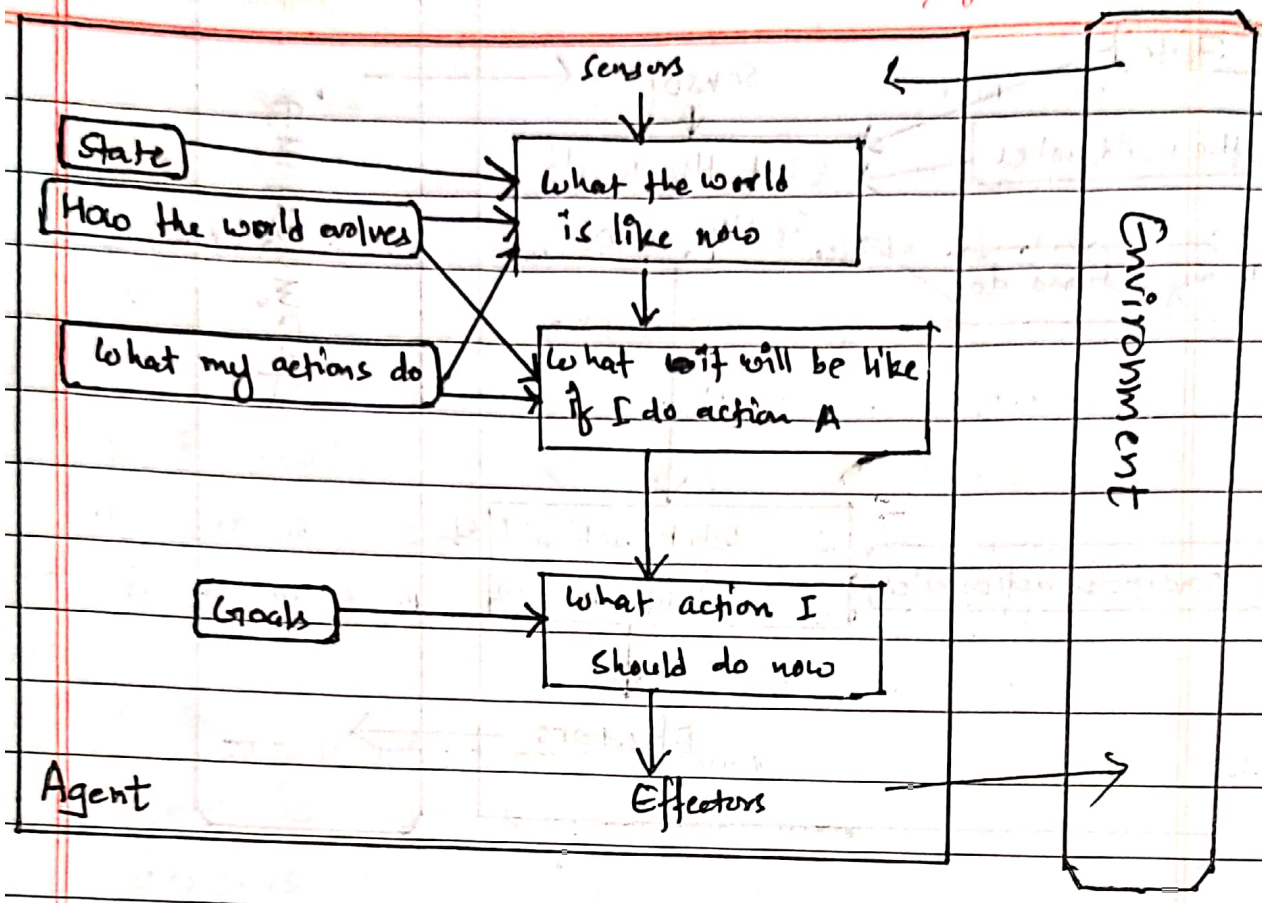
→ This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.

→ The knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible.

→ They usually require search & planning

→ The goal based agent's behavior can easily be changed.

Scanned with CamScanner



Goal based Agent = Model based Agent + Goal

Utility based Agents

→ When there are multiple possible alternatives, then to decide which one is best, utility based agents are used.

→ They choose actions based on preference (utility) for each state.

→ Sometimes achieving the desired goal is not enough.. We may look for quicker, safer, cheaper trip to reach destination.

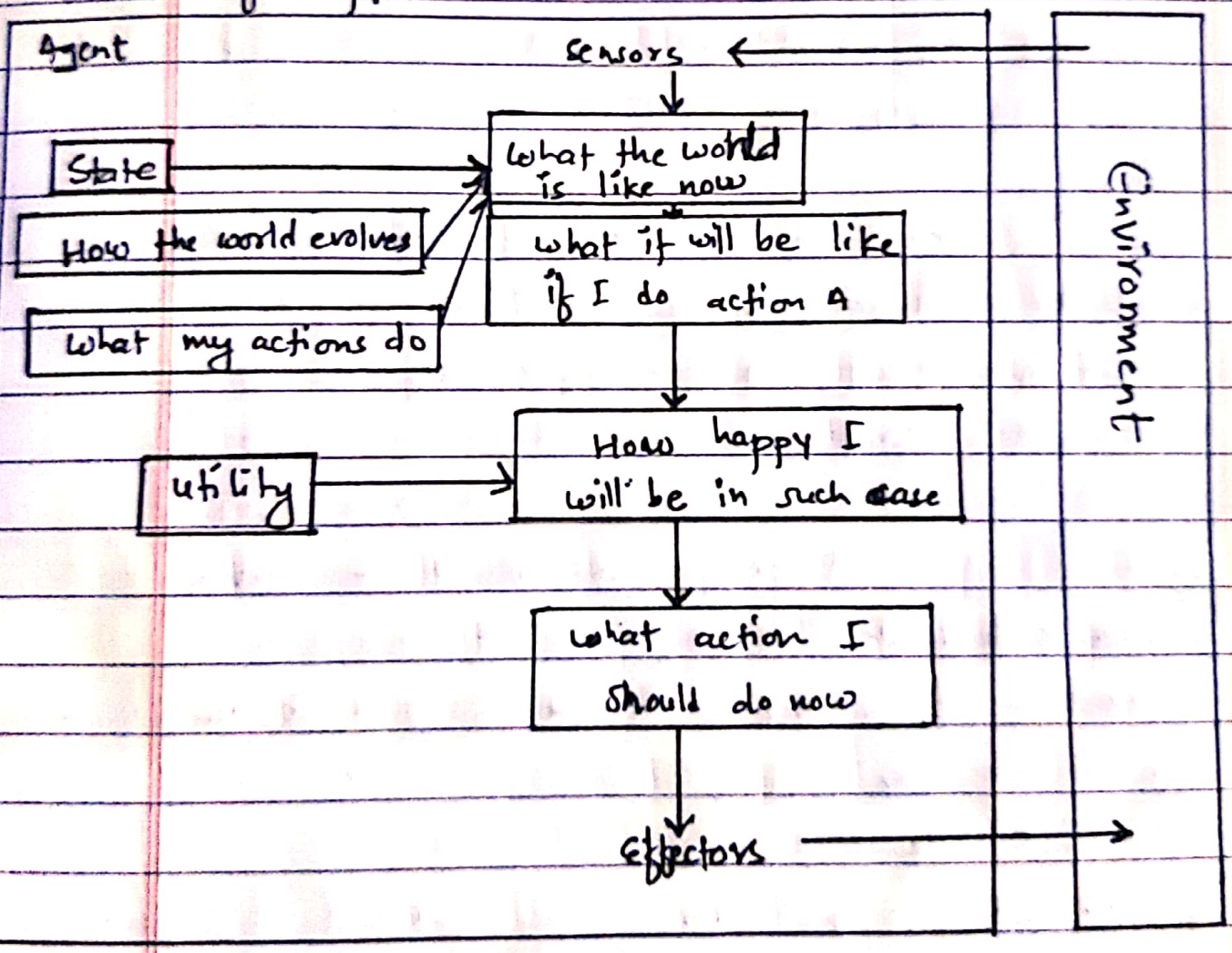
→ Agents happiness should be taken into consideration.

Scanned with CamScanner

→ utility ~~based a~~ describes how "happy" the agent is.

→ Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility.

→ A utility function maps a state onto a real number which describes the associated degree of happiness.



utility based agent = Goal based agent + utility

prepared by: Er. Kishor K Giri

Scanned with CamScanner

unit-3

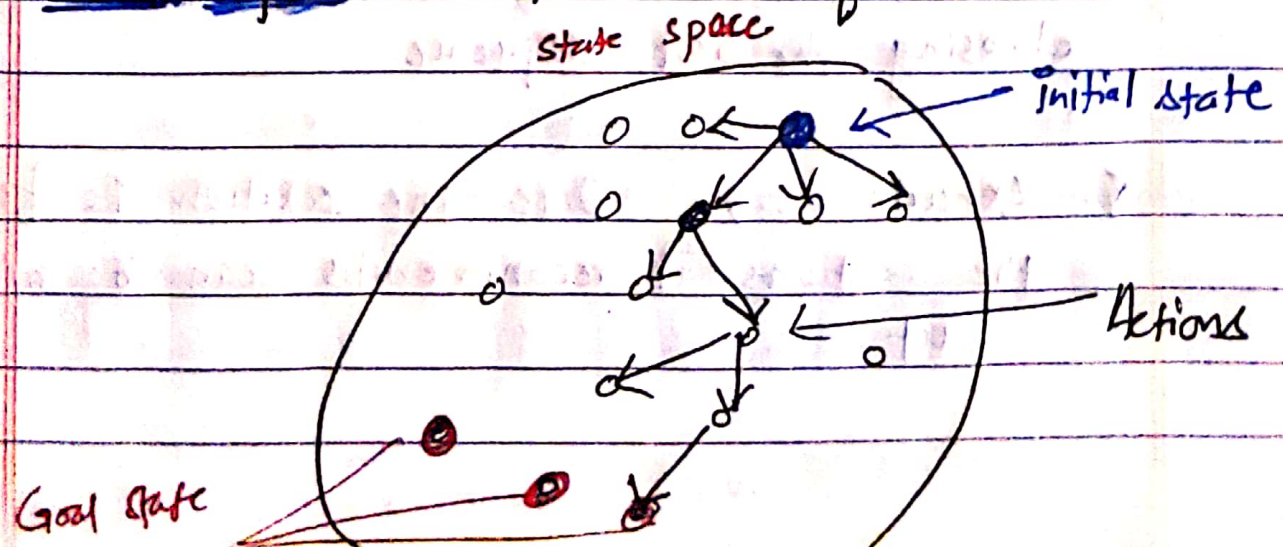
problem solving by Searching

Search → Search is an important aspects of AI because in many ways, problem solving in AI is fundamentally a search.

→ Search in AI is the process of navigating from a starting state to a goal state by transitioning through intermediate state.

Almost any AI problem can be ~~formulated~~ defined in these terms

State → A potential outcome of problem.
Transition → The act of moving between states.
starting state → Where to start starting search from.
Intermediate state → The states between the starting state and the goal state that we need to transition to.
Goal state → The state to stop searching.
Search space → A collection of states.



Scanned with CamScanner

problem as state space search

→ If each state is represented by a node and each possible change is represented by link, then a problem can be represented as a state space, with solution corresponding to a path from the initial state to final state.

Four general steps in problem solving

1. Goal Formulation → Formulation of goal based on the current situation and the agent's performance measure.

2. problem formulation → It is the process of deciding what actions and states to consider, given a goal.

3. Search method → Search method determine the possible sequence of actions that lead to the states of known values and then choosing the best sequence.

4. Execute → Once the solution is found the actions it recommends can be carried out.

Scanned with CamScanner

Date:
Page:

problem Formulation

problem formulation involves decides what actions and states to consider, given the goal.
We need to abstract the state details from the representation.
A problem can be defined formally by 4 components

1. Initial state → Starting point from which the agent sets out or starts solving the problem

2. Actions or successor function → This is the description of the set of possible actions available to agent.
→ It is sometime called operator which move the agent from one state to another.

3. Goal Test → It determines if a given state is the goal state or not.

4. Path cost → It determines the expenses of the agent for executing the actions in a path.
→ It is the sum of the costs of the individual actions in a path from start state to goal state.

Scanned with CamScanner

Well defined problems

Any problem in which the initial state or starting position, the allowable operations, and the goal states are clearly specified, and a unique solution can be shown to exist.

- Tower of Hanoi, water-jug problem etc

ill defined problems

- The problem may be solved by seeing the problem differently.
- There is no real mechanism for dealing with these type of problems.

- any problem in which either the starting position, the allowable operations, or the goal state is not clearly specified is called ill defined problem.
- eg: over population.

Types of problems

1. Real world problems

- Real world problems are real life problems such as route-finding problem, airline travel problem etc.

other examples:

- Touring problems
- VLSI layout
- Automatic assembly sequencing
- protein design.

Scanned with CamScanner

Types of problem .....

1. Toy problem → It is intended to illustrate or exercise various problem solving methods. It is used by different researchers to compare the performance of algorithm.
Example : chess, 8-queens, waterjug problem etc.

Example 1 The 8-puzzle

7	2	4
5		6
8	3	1

start state

	1	2
3	4	5
6	7	8

Goal State

- Figure above, consists of 3X3 board with eight numbered tiles and blank spaces.
- A tile adjacent to blank space can slide into the space.
- The objective is to ~~get~~ reach specified goal state, as shown in the figure right side.
- The standard formulation is as follows.

States → A state description specifies the location of each of the eight tiles and the ~~blank~~ blank in one of the nine squares.

Scanned with CamScanner

Initial state → Any state can be designated as the initial state.

Actions → The simplest formulation defines the actions as movements of the blank spaces left, Right, Up or Down.

Goal test → This checks whether the state matches the goal configuration or not.

Path cost → Each step cost 1, so the path cost is the no. of steps in the path.

Example 2 Vacuum World

- The vacuum world problem can be formulated as follows

States → The state is defined by both agent location and the dirt locations. The agent is in one of two locations, each of which might or might not contain dirt. Thus there are $2 \times 2^2 = 8$ possible world states.

For larger environment $= n \cdot 2^n$ states.

Initial state → Any state can be designated as initial state.

Scanned with CamScanner

Actions → Each state has just 3 actions: Left Right, and Suck. Larger environment might include up & down.

Transition model → The actions have their expected states, ~~exp~~ except moving left in the leftmost square, moving Right in the rightmost square, and sucking in a clean square have no effect.

Goal Test → This checks whether all the square are clean or not

Path cost → Each step costs 1, so the path cost is the number of steps in the path.

Example 3 8-queens problem

States → Any arrangement of 0 to 8 queens on the board is a state.

Initial state → No queens on the board.

Actions → Add a queen to any empty square

Transition model → Returns the board with a queen added to the specified square.

Goal test → 8 queens are on the board none attacked.

Another formulation do your self.

Scanned with CamScanner

Measuring problem solving performance

- Before we get into the design of specific search algorithms, we need to consider the criteria that might be used to choose among them.
- We can evaluate an algorithm's performance in four ways.

Completeness → Is the algorithm guaranteed to find a solution when there is one?

Optimality → Does the strategy find the optimal solution?

Time Complexity → How long does it take to find a solution?

Space complexity → How much memory is needed to perform the search?

Complexity is expressed in terms of three quantities

b → the branching factor or maximum number of successor of any node.

d → the depth of the shallowest goal node [no. of steps along the path from the root]

m → max^m length of any path in the state space.

Scanned with CamScanner

Types of search algorithms

Searching is a process to find the solution for a given set of problems.

Two broad classification

1. Uninformed search / blind search
2. Informed search / heuristically search

Breadth First Search

- BFS is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, and then their successors, and so on.
- The shallowest node is choosen for expansion, using a FIFO queue for the frontier.
- Thus, the new nodes (which are always deeper than their parents) go to the back of the queue, and old nodes, which are shallower than the new nodes, get expanded first.

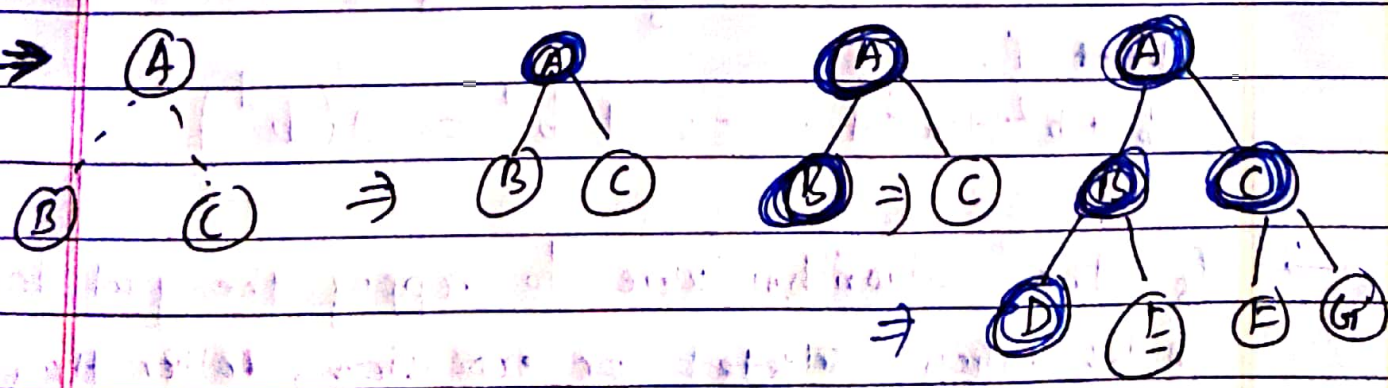


fig: BFS on simple binary tree.

Scanned with CamScanner

performance measure

Completeness

- BFS always finds a solution if shallowest goal node is at some finite depth d and if b is finite.

Time complexity

- Assume, searching a uniform tree where every state has b successors.
- The root of the search tree generates b nodes at the first level, each of which generates b more nodes, for a total of b^2 at second level.
- Each of these generates b more nodes, yielding b^3 nodes at the third level and so on.
- Suppose that solution is at depth d.
- In the worst case, it is the last node generated at that level.
- Then the total no. of nodes generated ~~at that level~~ is

$$b + b^2 + b^3 + \cdots + b^d = O(b^d)$$

→ If the algorithm were to apply the goal test to nodes when selected for expansion, rather than when generated, the whole layer of nodes at depth d would be expanded before the goal was detected and the time complexity would be $= O(b^{d+1})$

Scanned with CamScanner

O notation → denotes asymptotic upper bound, pronounced as "big Oh"

Space Complexity

- For BFS, every node generated remains in memory
- There will be $O(b^{d-1})$ nodes in the explored set, and $O(b^d)$ nodes in frontier, so the space Complexity is $O(b^d)$ i.e. it is dominated by the size of frontier.

Optimality → Not optimal but finds solution ~~if all~~ with shortest path length (shallowest solution).

→ If each path cost doesnot have same path cost shallowest solution may not be optimal.

Disadvantages / problems

- High memory requirements
- Exponential-complexity search problems cannot be solved by uninformed methods for any but small instances.

please refer norvig's book page 85 for example with $b=10$.
* for time & space requirement problem.

Applications → social networking websites, search engine, GPS navigation, Broadcasting in N/w.

Scanned with CamScanner

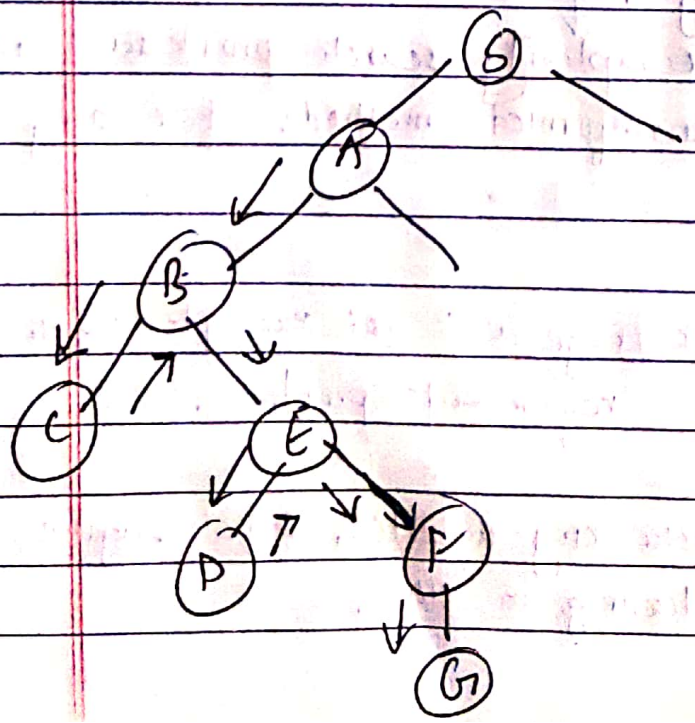
Depth First Search

→ DFS always expands the deepest node 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 are expanded, they are dropped from the frontier, so then the search "backs up" to the next level deepest node that still has unexplored successors.

→ DFS uses a stack data structure to keep track of vertices. Fringe is implemented as a LIFO queue (stack).



Scanned with CamScanner

performance measure

Completeness → If search space is infinite and search space contains loops then DFS may not find solution.

Time complexity → It is bounded by the size of the state space (which may be infinite).

let $m \Rightarrow max^m$ depth of search tree
$b \Rightarrow$ successor of root

worst case ⇒ expand all except the last node at depth m

Total nodes generated : $O(b^m)$

Space complexity ⇒ It needs to store only a single path from the root node to a leaf node, along with the remaining unexpanded sibling nodes for each node on the path.

Total no. of nodes in memory ⇒ $O(bm)$

Scanned with CamScanner

Optimality ⇒ Not optimal.
Solution may be a high cost solution, there may exist low cost solution.

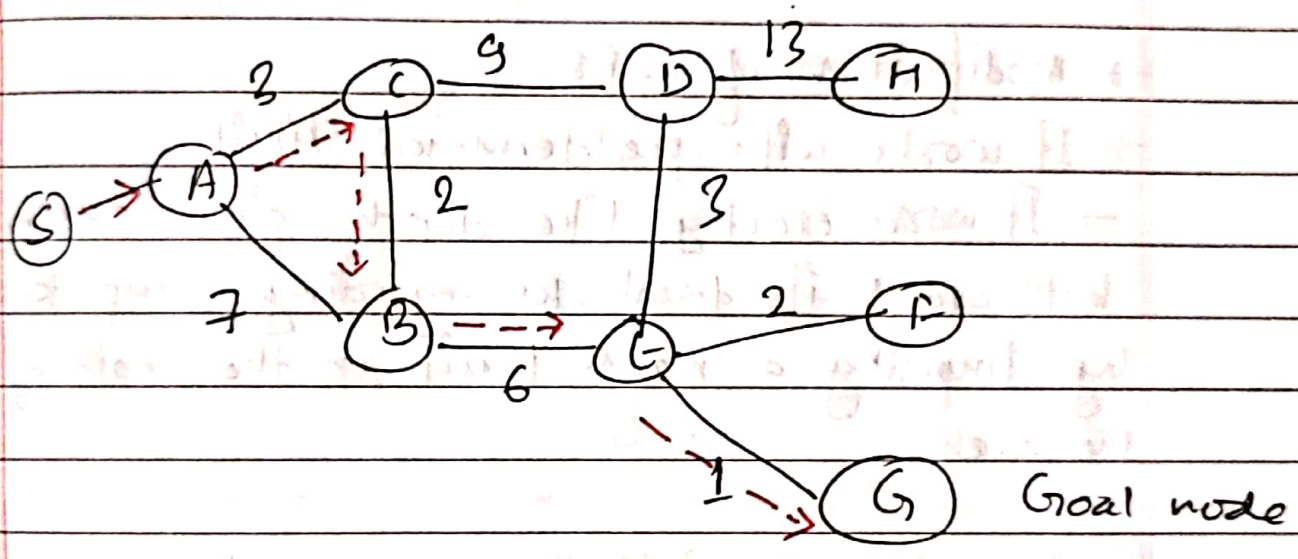
Applications

- Detecting cycle in graph
- path finding
- Topological sorting
- solving puzzle with only one solution.

Scanned with CamScanner

Uniform cost search / cheapest first Search

→ It is used for weighted Tree / Graph traversal.
→ The uniform cost search expands the node with least path cost.
→ Node expansion is based on path costs.
→ priority queue is used for implementation i.e. high priority is given to minimum cost.



path to goal node.

$A \xrightarrow{3} C \xrightarrow{2} B \xrightarrow{6} E \xrightarrow{1} G \Rightarrow 12$ (cost

If the goal node was in H, then this algorithm uses backtracking to go to H.

path. $A \xrightarrow{3} C \xrightarrow{2} B \xrightarrow{6} E \xrightarrow{1} G \xrightarrow{1} E \xrightarrow{2} F \xrightarrow{2} E \xrightarrow{3} D \xrightarrow{13} H$

Total cost ⇒ 33

Scanned with CamScanner

optimality => optimal solution, because least path cost is choosen.

problem => may stuck in infinite loop.

Depth limited search

→ Modification of DFS.
→ It works with predetermined limit.
→ It works exactly like depth first search, but avoids its drawbacks regarding completeness by imposing a maxm limit on the depth of the search.

→ It solves the infinite-path problem of DFS, yet it introduces another source of problem if we are unable to find good guess of l.

Advantage
→ Memory efficient

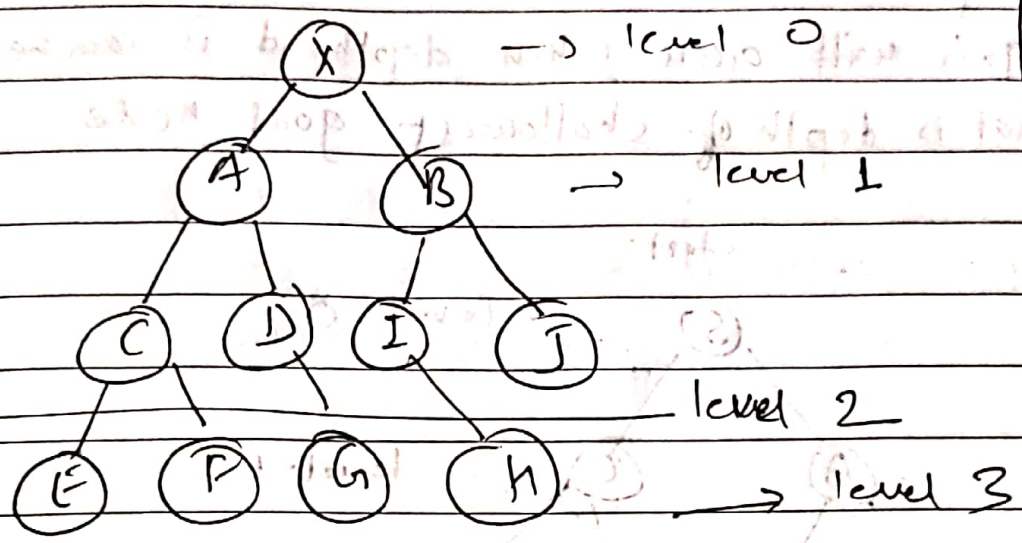
Disadvantage
→ can be terminated without finding solution
→ Not optimal.

Scanned with CamScanner

Time complexity $\Rightarrow O(b^{l})$

Space complexity $\Rightarrow O(b^{l})$

Completeness $\Rightarrow$ incomplete result if $l < d$.



Depth $d = 2$

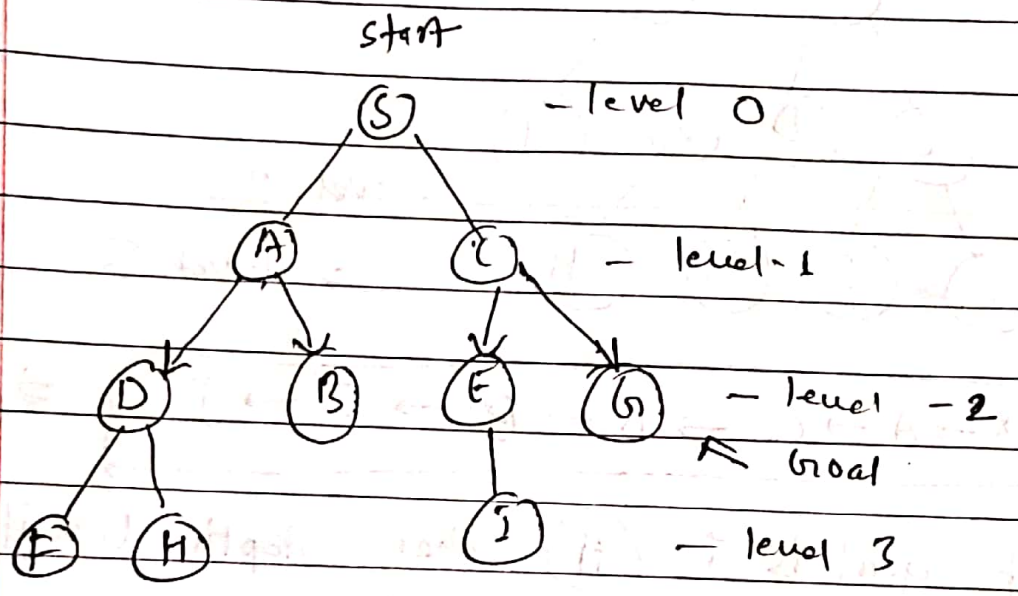
$X \rightarrow A \rightarrow C \rightarrow D \rightarrow B \rightarrow I \rightarrow J$ $\Rightarrow$ path

If goal is in (H) then depth limit = 2 will not give any solution.

Scanned with CamScanner

Iterative deepening Depth first search

→ It is the combination of both DFS & BFS.
→ In this method best depth limit is found out by gradually increasing limit - first 0, then 1, then 2, and so on until the goal is found.
→ This will occur when depth d is reached; that is depth of shallowest goal node.



1st iteration, d = 0 [S]
2nd iteration, d = 1 [S → A → C]
3rd iteration d = 2 [S → A → D → B - C → E - G]

performance measure

Completeness → complete when branching factor b is finite.

Time complexity → Algorithm seems costly due to repeated generation of certain states.

Node generation:

level d : once
level d-1 : 2
level d-2 : 3
⋮
level 2 : d-1
level 1 : d

Total no. of nodes generated.

$$d \cdot b + (d-1) b^2 + (d-2) b^3 + \cdots + 1 \cdot b^d = O(b^d)$$

Space complexity

→ It needs to store only a single path from the root node to a leaf node, along with remaining unexpanded sibling nodes for each node on the path.

- Total no. of node in memory:

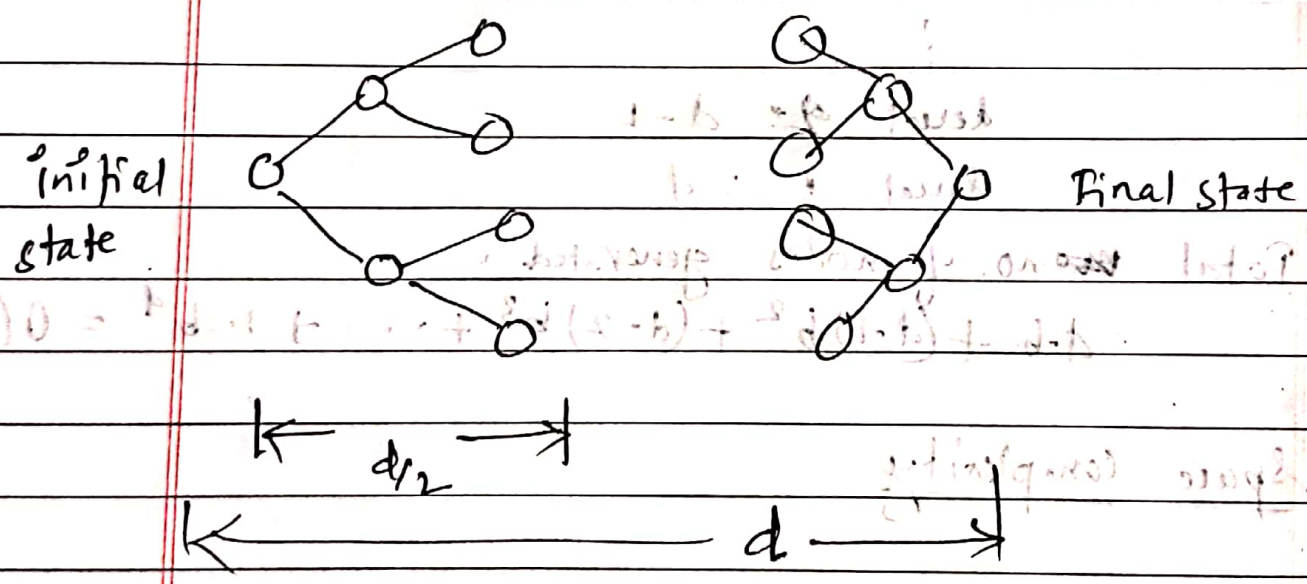
$$1 + b + b + b + \cdots \quad b \cdot d \text{ times} = O(bd)$$

Scanned with CamScanner

Optimality → Yes if path cost is non-decreasing function of the depth of the node. (when the steps are of the same cost.)

Bidirectional Search

- This technique simultaneously search from one initial node to goal and other goal node to initial, stopping when two meet.



performance measure

Completeness → yes

optimality → yes (if done with correct strategy eg: BFS)

Scanned with CamScanner

Time complexity → $O(b^{d/2})$

Space complexity → $O(b^{d/2})$

Drawbacks of Uninformed Search

→ Doesnot exploit the structure of the problem.

→ One may prefer to use a more flexible rule that takes advantages of what is being discovered on the way, and hunches about what can be a good move.

Scanned with CamScanner

Informed / Heuristic search

→ Heuristic search tries to optimize a problem using heuristic function.
→ Tries to solve problem in minimum steps/cost.

Heuristic function → It is a function H(n), that gives an estimation on the cost of getting from node 'n' to the goal state.
→ It is the additional knowledge of the problem.

Heuristic search is useful in tough problems which.
- could not be solved any other way
- solutions take infinite time or very long time to compute.

→ Heuristic information helps to decide which node to expand next, instead of doing the expansion in a strictly breadth-first search or depth first search.

Scanned with CamScanner

→ In the course of expanding a node, deciding which successor to generate, instead of blindly generating all possible successor at one time.

→ Deciding certain nodes should be discarded, or pruned from the search space.

Why heuristic search?

- Blind search may be too resource intensive (both time & space)
- Even if blind search will work we may want a more efficient search method.

- Uses domain specific information to improve search pattern.

→ Define a heuristic function, h(n) that estimates the "goodness" of a node n
→ specifically, h(n) = estimated cost (or distance) of minimal cost path from n to goal state.

→ The heuristic function is an estimate, based on domain specific information that is computable from the current state description, of how close we are to a goal.

Scanned with CamScanner

Best first search

- The idea of Best first search is to use an evaluation function to decide which adjacent is most promising & then explore.

Idea: use an evaluation function $f(n)$ that gives an indication of which node to expand next for each node.

- usually gives an estimate to the goal
- the node with the lowest value is expanded first.

Special cases: based on evaluation function
- Greedy best-first search
- A* search.

Scanned with CamScanner

Greedy best-first search

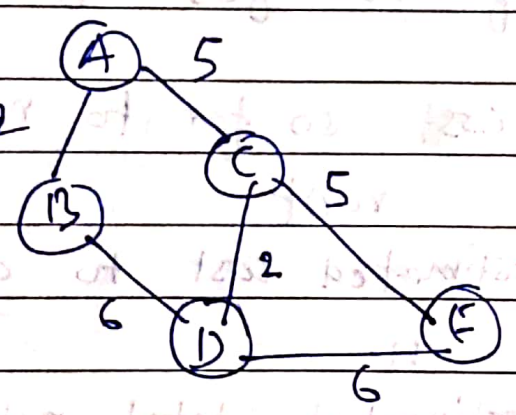
→ Greedy best-first search uses only the heuristic.

Evaluation function f_n = heuristic $h(n)$

→ Greedy best-first search expands the node that appears to be closest to goal.

Space complexity = $O(b^m)$
time " = $O(b^m)$
Optimality ⇒ not optimal
Completeness ⇒ not complete (may oscillate between two nodes)

Example:



Heuristic value

A → G ⇒ 10
B → G ⇒ 8
D ⇒ G ⇒ 3
C → G ⇒ 5
E → G ⇒ 0

Path

A → C → E

Actual cost ⇒ 10 ∴ $f(n) = h(n)$
Heuristic f^n ⇒ 10

Scanned with CamScanner

Date:

A* search: A better Best-First Strategy

Admissible Heuristic → A heuristic is admissible if it never overestimates the cost of reaching the goal.
→ Also known as optimistic heuristic.

A* search uses a heuristic function h(n) as well as the path cost to the node g(n).
- The h(n) part of the f(n) must be admissible heuristic; that is it must not overestimate the distance to the goal.

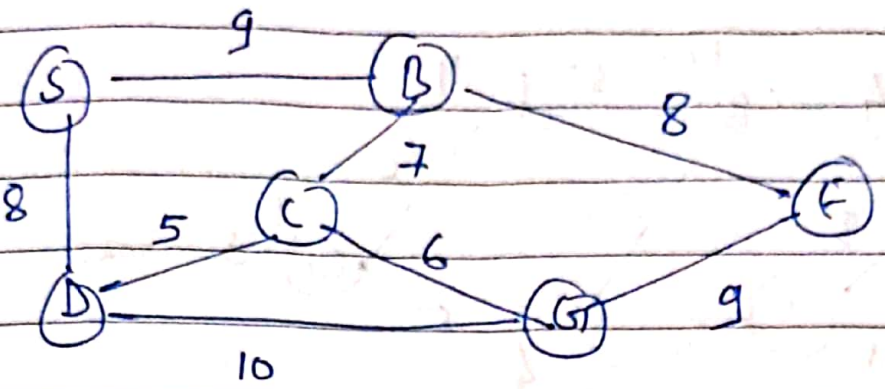
Evaluation function $f(n) = g(n) + h(n)$
where,
g(n) = cost so far to reach n from root
h(n) = estimated cost to goal from n
f(n) = estimated total path cost through n to goal.

→ It is informed under reasonable assumptions optimal & complete.

Scanned with CamScanner

Date:
Page:

Example:



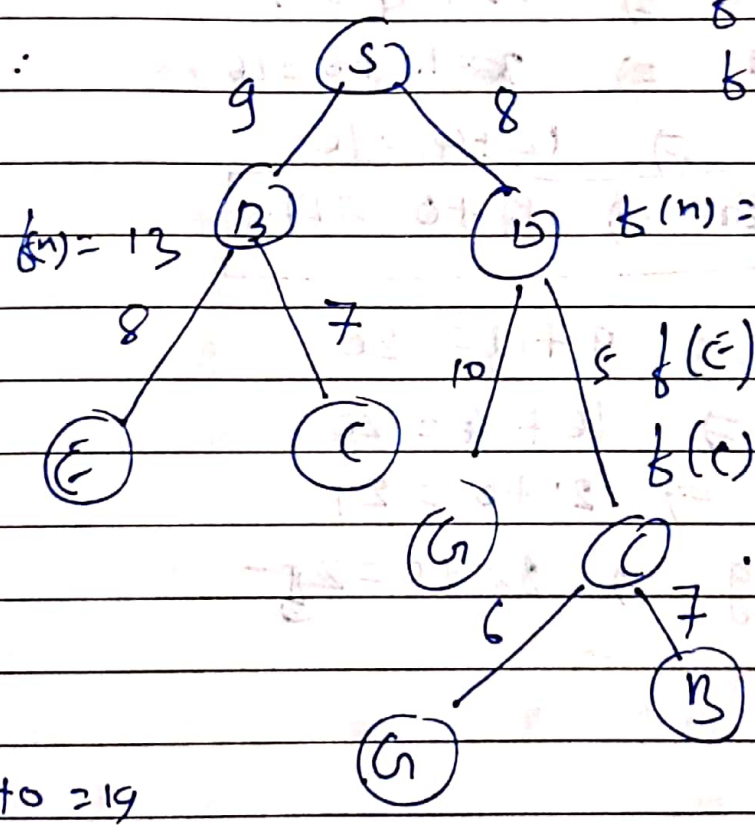
straight line distances to node G (goal node) from other nodes:

$S \to G = 12$
$B \to G = 4$
$E \to G = 7$
$D \to G = 6$
$C \to G = 3$

path
S — D — G

$f(B) = 9+4 = 13$
$f(D) = 8+6 = 14$

step 1:



$f(E) = 17 + 7 = 24$
$f(C) = 16 + 3 = 19$

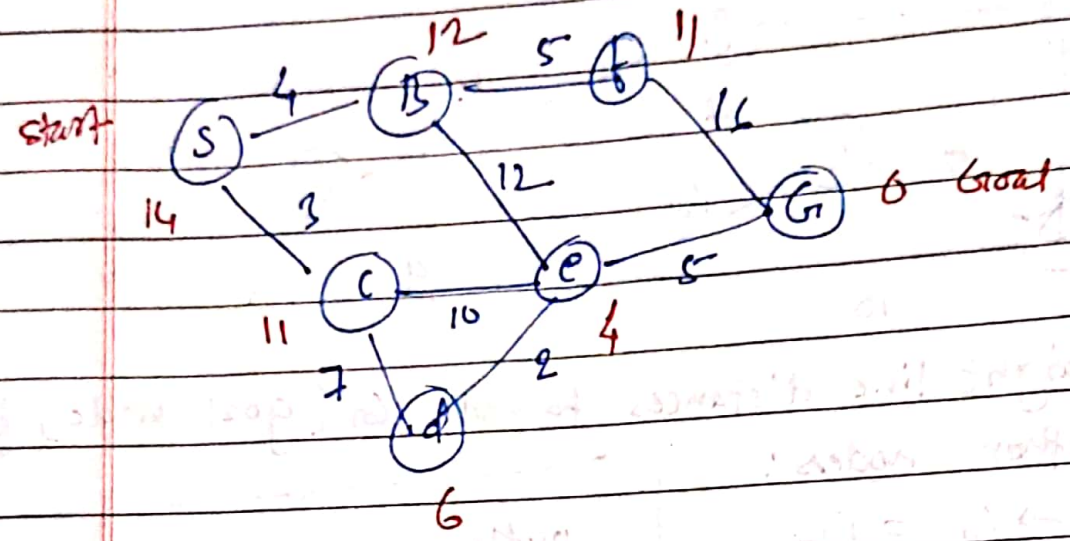
$f(G) = 18 + 0 = 18$
$f(C) = 13 + 3 = 16$
$f(G) = 19 + 0 = 19$
$f(B) = 20 + 4 = 24$

$f(G) = 19 + 0 = 19$
$f(C) = 24 + 4 = 24$

Scanned with CamScanner

Date:
Page:

Example 2.



S → B
4+12
= 16

S → C
3+11
= 14

SC → e ⇒ 13 + 4 = 17
SC → d ⇒ 10 + 6 = 16
SCde ⇒ 12 + 4 = 16
SCde G ⇒ 17 + 0 = 17

SBf ⇒ 9 + 11 = 20
SBe ⇒ 16 + 4 = 20
SBeg ⇒ 21 + 0 = 21
SBfg ⇒ 25 + 0 = 25

Scanned with CamScanner

Evaluation :

Completeness → yes, always gives solution
optimality → gives optimal sol^n when heuristic f^n is admissible

Time complexity → $O(b^d)$
space complexity → It keeps all generated node in memory. Hence space is the major problem not time.

Admissible heuristic

- A heuristic function is admissible if it is no more than the lowest-cost path to the goal.
- In other words, a heuristic is admissible if it never overestimates the cost of reaching the goal.
- An admissible heuristic is also known as an optimistic heuristic.

If estimated distance $h(n)$ never exceed the true distance $h^*(n)$ between the current node to goal node, the A* algorithm will always find a shortest path. This is known as the

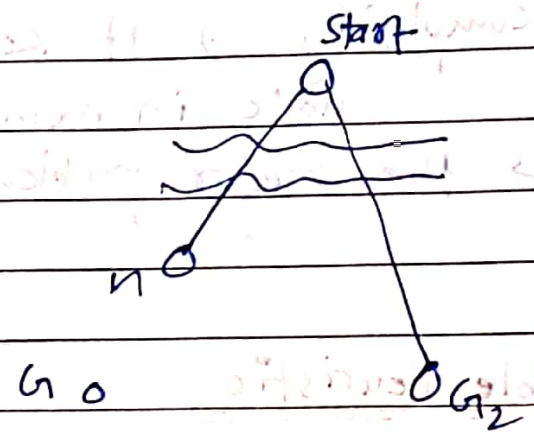
Scanned with CamScanner

admissibility of A* algorithm and h(n) is admissible heuristic.

A* properties

Tree-search version

Theorem: A* is optimal if h(n) is admissible



Suppose suboptimal goal G2 in the queue. Let n be an unexpanded node on a shortest path to optimal goal G and C^* be the cost of optimal goal node.

$$f(G2) = h(G2) + g(G2)$$

$$f(G2) = g(G2), \text{ since } h(G2) = 0$$

$$f(G2) > C^* \quad \text{——— (1)}$$

Again, since h(n) is admissible, it doesnot overestimate the cost of completing the

solution path.

$f(n) = g(n) + h(n) \leq C^*$ — (2)

Now from (1) & (2)

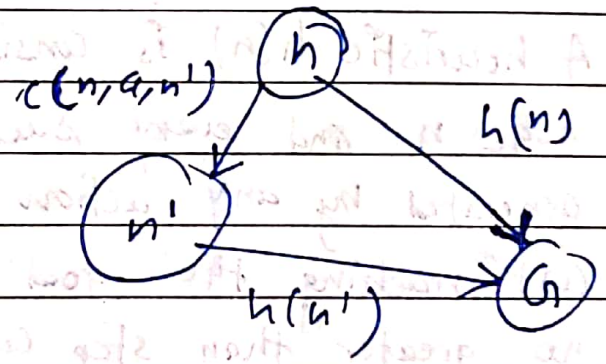
$f(n) \leq C^* < f(G2)$

Since $f(G2) > f(n)$, A* will never select G2 for expansion.

Thus A* gives us optimal sol^n when heuristic function is admissible.

graph search version

Theorem: If $h(n)$ is consistent, then the values of $f(n)$ along the path are non decreasing.



Suppose n' is successor of n; then

$g(n') = g(n) + c(n, a, n')$ for some action a,

Scanned with CamScanner

$$\boxed{f(n) = g(n) + h(n)}$$

we have,

$$f(n') = g(n') + h(n')$$
$$f(n') = g(n) + c(n, a, n') + h(n') \quad \text{--- (1)}$$

A heuristic is consistent if

$$h(n) \leq c(n, a, n') + h(n') \quad \text{--- (2)}$$

Now from (1) & (2)

$$f(n') = g(n) + c(n, a, n') + h(n') \geq$$
$$g(n) + h(n) = f(n)$$

<u>Consistency</u> / Monotonicity

A heuristic $h(n)$ is consistent if, for every node n and every successor n' of n generated by any action a, the estimated cost of reaching the goal from n is no greater than step cost of getting to n' plus the estimated cost of reaching the goal from n'

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

Scanned with CamScanner

Date:
Page:

Hill climbing search

- Hill climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead where to go next.

→ It is simply a loop that continually moves in the direction of increasing value - that is, uphill.

Algorithm

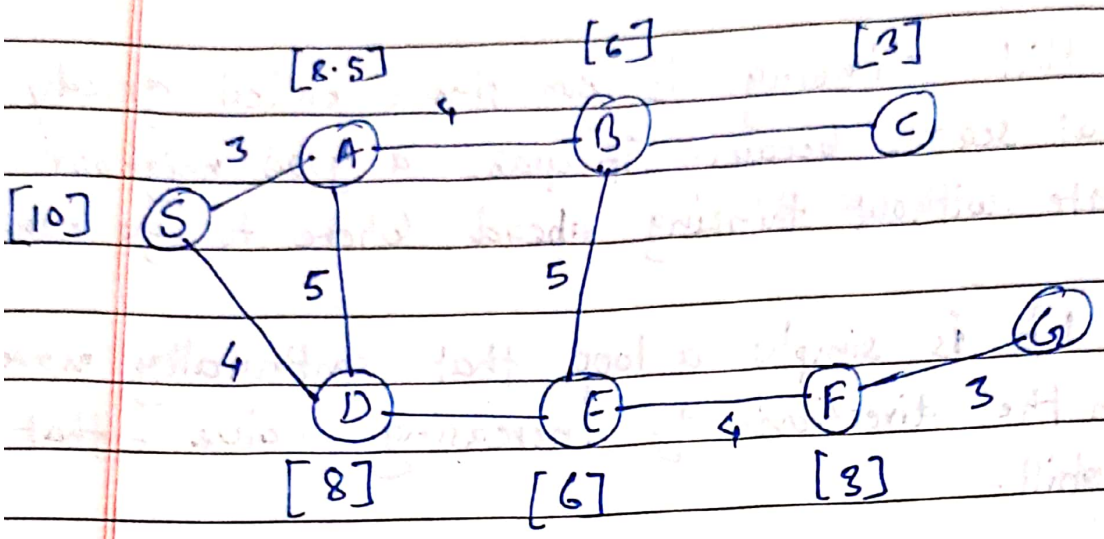
1. Evaluate the initial state, if it is goal state then Terminate. Otherwise, current state is initial state.

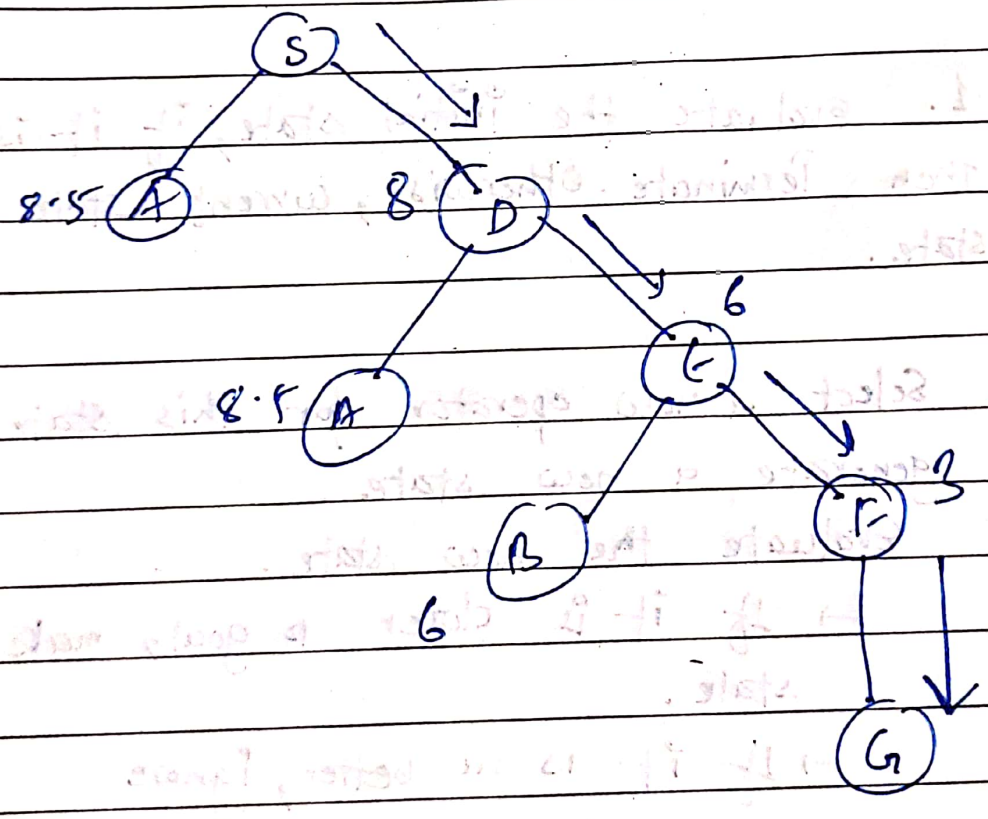
2. Select a new operator for this state and generate a new state.
3. Evaluate the new state.
   → If it is closer to goal, make it current state.
   → If it is no better, ignore

4. If the current state is goal state or no new operators available, Terminate. Otherwise repeat from 2

Scanned with CamScanner

Example:

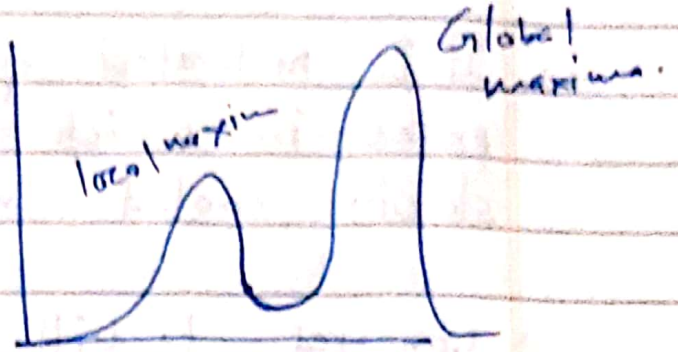




Scanned with CamScanner

problems with Hill Climbing

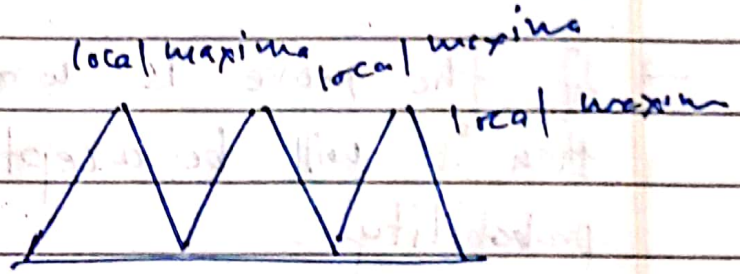
Local Maxima



→ A local maximum is a peak that is higher than each of its neighbouring state but lower than the goal maximum.

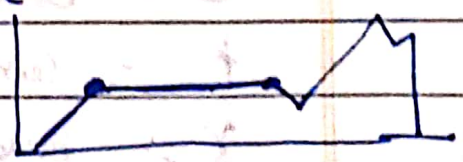
→ It gets stuck at local maxima when ~~reches~~ reaches a position where there are no better neighbour. It is not a guarantee that we have found the best solution.

Ridges



- It is sequence of local maxima.
- Algorithm stucks getting local maxima all the time.

plateaux → A plateau is a flat area of the state-space landscape.



It can be flat local maximum from which no uphill exit exists.

Scanned with CamScanner

Simulated Annealing

→ It is motivated by the physical annealing process in which material is heated & slowly cooled into a uniform structure.

→ Compared to hill climbing simulated annealing allows downwards steps.

→ SA also differs from hill climbing in that a move is selected at random and then decides whether to accept it.

→ If the move is better than its current position then SA will always take it.

→ If the move is worst (lesser quality) then it will be accepted based on some probability.

- The probability of accepting a worst state is given by the eqn.

$$P = \text{exponential}(-c/t) > r$$

where,

c = change in the evaluation fn

t = current value

r = random no. betn 0 and 1

Scanned with CamScanner

Advantages

- Easy to code for complex problem
- Gives good solution
- statistically guarantees to find optimal sol^n.

Disadvantages

- Slow process
- Can't tell whether an optimal sol^n is found.
- to check some other method is also required.

Simulated annealing	Hill Climbing
- annealing schedule is maintained	X
- Moves to worst state may be accepted	X
- Best state found so far is maintained.	X

Scanned with CamScanner

Game playing

→ Game playing is an important domain of AI.

→ Game playing require the knowledge to be provided is the rules, legal moves and the conditions of winning or losing the game.

→ Both players try to win the game. So both of them try to make the best move possible at each turn.

→ Search technique like BFS are not accurate for this as branching factor is very high, so searching will take a lot of time.

→ So we need another search procedure that improves

Generate procedure so that only good moves are generated.

Test procedure so that the best move can be explored first.

Scanned with CamScanner

Games are a form of multi-agent environment
- what do other agents do and how do they affect our success?
- co-operative vs competitive multi-agent environments
- Competative multi-agent environments give rise to adversarial search often known as games.

Game - adversary

- Solution is strategy
- Time limits force an approximate solution
- Evaluation function evaluate "goodness" of game position.

Examples: chess, checker etc.

Examplary Game : Tic - tac - toe

- There are two players denoted by X or O.
- They are alternatively writing their letter in one of the 9 cell of a 3x3 board.
- The winner is the one who succeeds in writing three letters in line.
- The game begins with empty bord. It ends in a win for one player, and a loss for the other, or possibly in a draw.

Scanned with CamScanner

- A complete tree is the representation of all possible plays of the game.

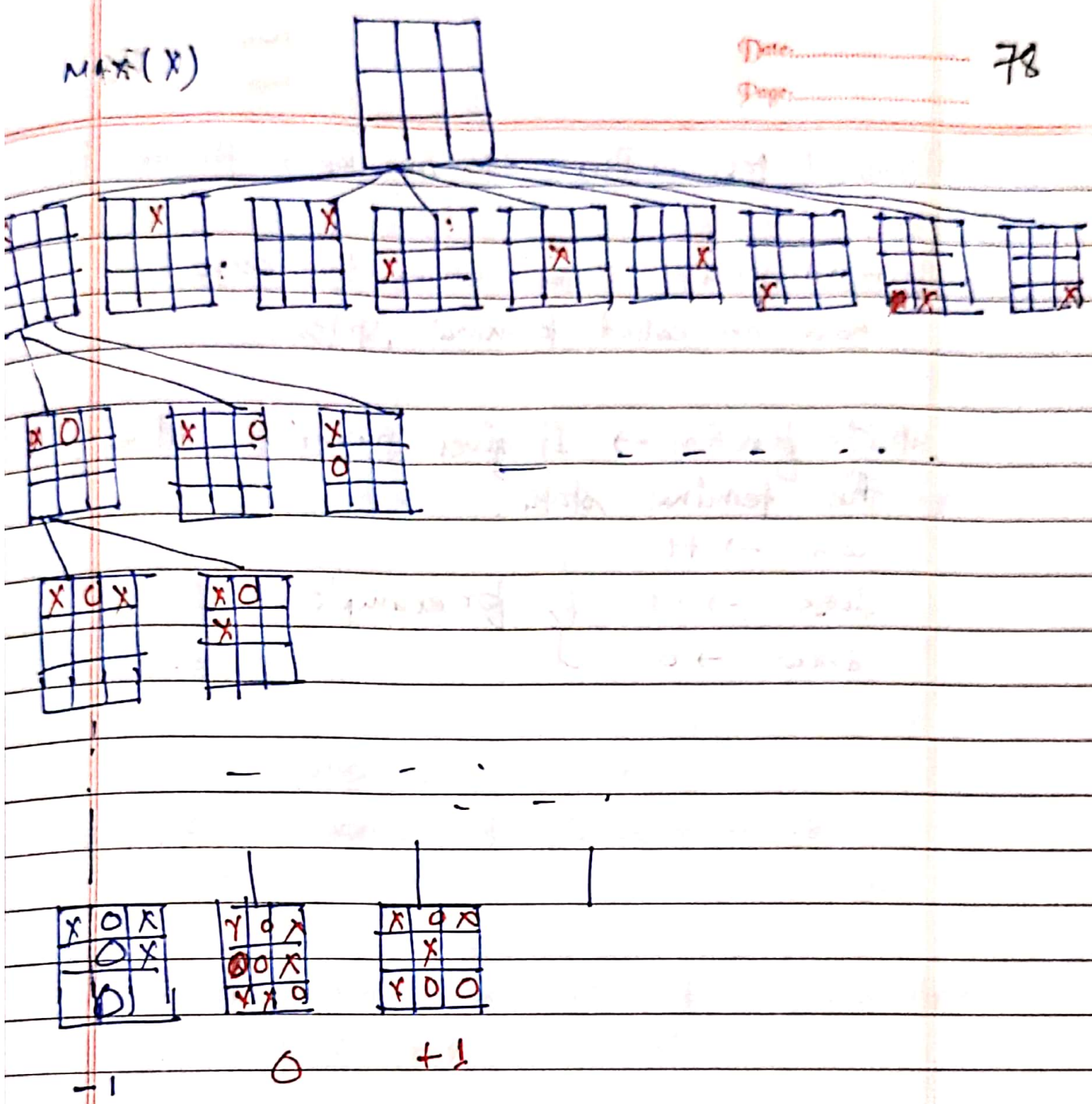
- The root node is the initial state, in which it is the first player's turn to move (the player X)

- The successors of the initial state are the states the player can reach in one move, their successors are the states resulting from the other player's possible replies, and so on.

- Terminal states are those representing a win for X, loss for X or draw.

- Each path from the root node to a terminal node gives a different complete play of the game.

Scanned with CamScanner



A game can be formally defined as a kind of search problem as :

Initial state : It includes the board position and identifies players to move.

Successor function : It gives a list of (move, state) pairs each indicating a legal move and resulting state.

Scanned with CamScanner

Terminal test → This determines when the game is over.

Terminal states → states where the game is ended are called terminal states.

utility function → It gives numerical value of the terminal state.

win → +1
loose → -1
draw → 0

} for example.

Scanned with CamScanner

The Mini-max Search

It is a specialized search algorithm that returns optimal sequence of moves for a player in zero sum game.

- Recursive / backtracking algorithm which is used in decision making and game theory.

→ Algorithm computes minmax decisions for current state.

- There are two players
(1) MAX - selects the maximum value
(2) MIN - " " minimum value.

For the exploration of the game tree depth first search is used.

Example: Chess, checker, tic-tac-toe

The worst initial values for max & min are $-\infty$ to ∞.

properties

(1) Complete → definitely finds sol^n if exists

Scanned with CamScanner

② optimal

③ Time complexity $= O(b^m)$

$b \rightarrow$ branching factor of game tree

$m \rightarrow$ maximum depth

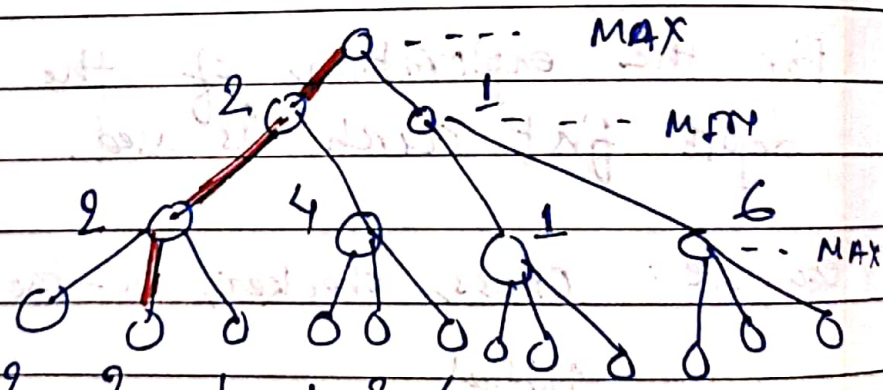
④ Space complexity $= O(b^m)$

Limitations

- Slow for complex games such as chess.

35 choices/move

Example:



utility $\rightarrow$ -2 2 1 1 3 4 -1 1 0 2 5 6

Initial values $\left[\begin{array}{l} max = -\infty \\ min = \infty \end{array}\right]$

MAX TURN

$(-2, \alpha) \rightarrow (-2, 2) \rightarrow (2, 1) \rightarrow 2$

$(1, \alpha) \rightarrow (1, 3) \rightarrow (3, 4) \rightarrow 4$

$(-1, \alpha) \rightarrow (-1, 1) \rightarrow (1, 0) \rightarrow 1$

$(2, \alpha) \rightarrow (2, 5) \rightarrow (5, 6) \rightarrow 6$

MIN TURN

Min will try to minimize. So it will choose.

$(2, 4) \rightarrow 2$

$(1, 6) \rightarrow 1$

MAX TURN

Max will try to maximize, so it will choose

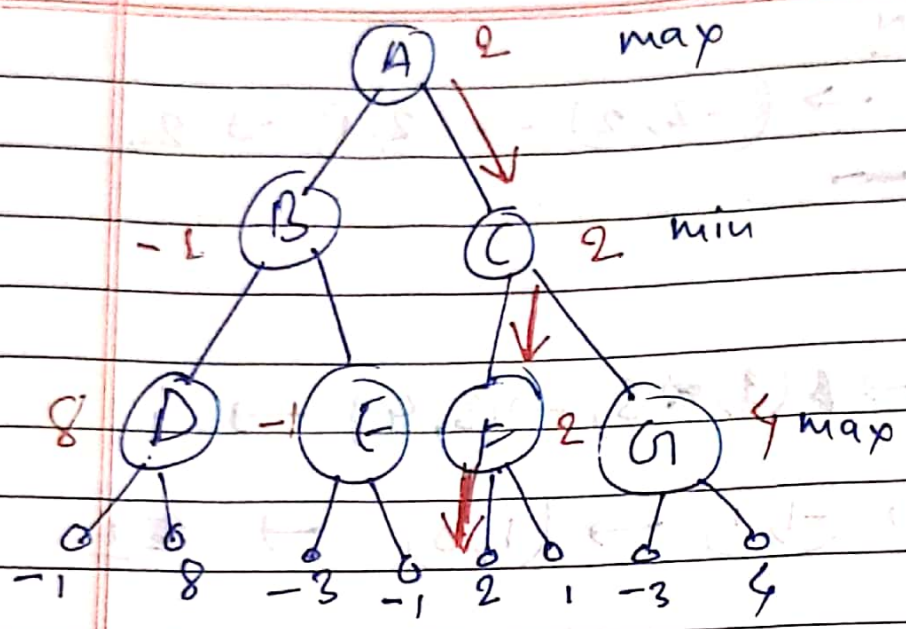
$(2, 1) \rightarrow 2$

∴ So path for Max to win the game is show with red line. It will give maximum benifit to max player

$\therefore \boxed{2 - 2 - 2}$

max - min - max

Scanned with CamScanner



path A – C – F
2 – 2 – 2 will give maximum benifit to the max player.

Scanned with CamScanner

Alpha beta pruining

when alpha-beta pruining is applied to a standard minimax tree, it returns the same move as minmax would, but prunes away branches that can not possibly influence the final decision.

→ Alpha beta pruining can be applied to trees of any depth, and it is often possible to prune entire sub-trees rather than just leaves.

→ It uses two parameters, alpha & beta.

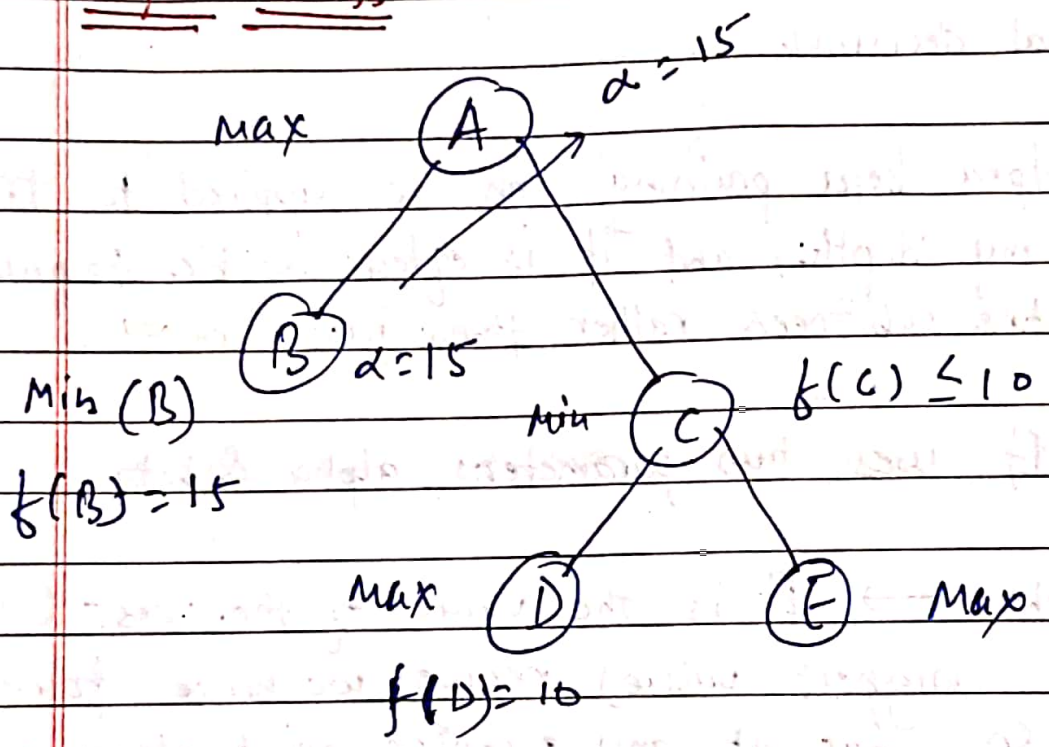
Alpha → It is the value of the best (i.e. highest value) choice we have found so far at any choice point along the path for MAX.

Beta → It is the value of the best (lowest-value) choice we have found so far at any choice point along the path for min.

Alpha-beta search updates the values of alpha and beta as it goes along and prunes the remaining branches at a node as soon as the value of

current node is known to be worse than the current alpha or beta for MAX or MIN respectively.

Alpha cutoff



→ The mini-max method uses a depth-first search strategy in evaluating the decendants of a node.

→ It will therefore estimate first the value of the node B.

→ let us suppose f(B) = 15, evaluated either by using a static evaluation f^n or by backing up from descendants omitted in the figure.

Scanned with CamScanner

If the max will move to B then it is guaranteed to achieve 15.

- Therefore 15 is the lower bound for the achievement of the max player.

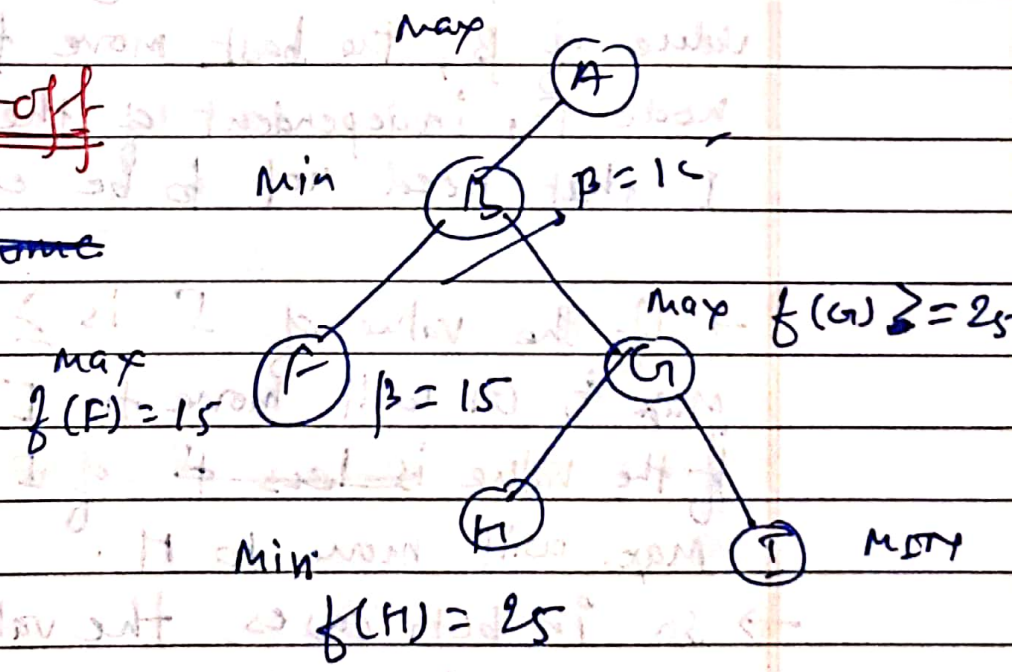
$\therefore \alpha = 15$ (at node B)

- This value is transmitted upward to node A and will be used for evaluating the other possible moves from A.

- To evaluate the node C, its left-most ~~child~~ child D has to be evaluated first
- Assume $f(D) = 10$
- Because this value is less than α, the best move for max is to ~~next~~ node B, independent of the value of node E.

A beta cutoff

~~let us assume~~



Scanned with CamScanner

- let us assume that the value of F has been evaluated to 15.
- From the point of view of min, this is the upper bound for the achievement of min

→ Therefore the value of B at the node F is 15.

→ This value is transmitted ~~to~~ upward to the node B and will be used for evaluating the other possible moves from B.

→ To evaluate the node G, its left most child F is evaluated first.
Suppose f(H) = 25.
Because this value is greater than the value of B, the best move for min is to node F, independent of the value of node I that need not to be evaluated.

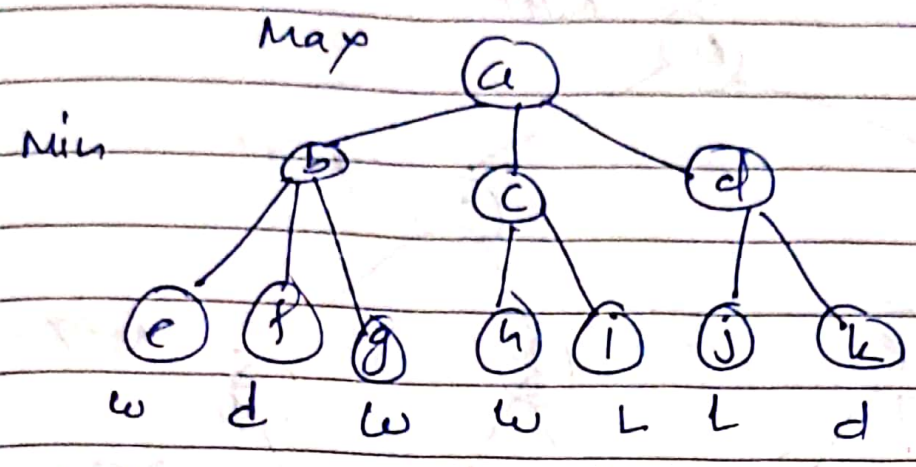
→ If the value of I is ≥ 25 then Max in G will move to I. otherwise, if the value ~~is less~~ of I is less than 25, Max will move to H.

→ So in both cases the value ~~of~~ obtained by max is at least 25 which is greater than B.

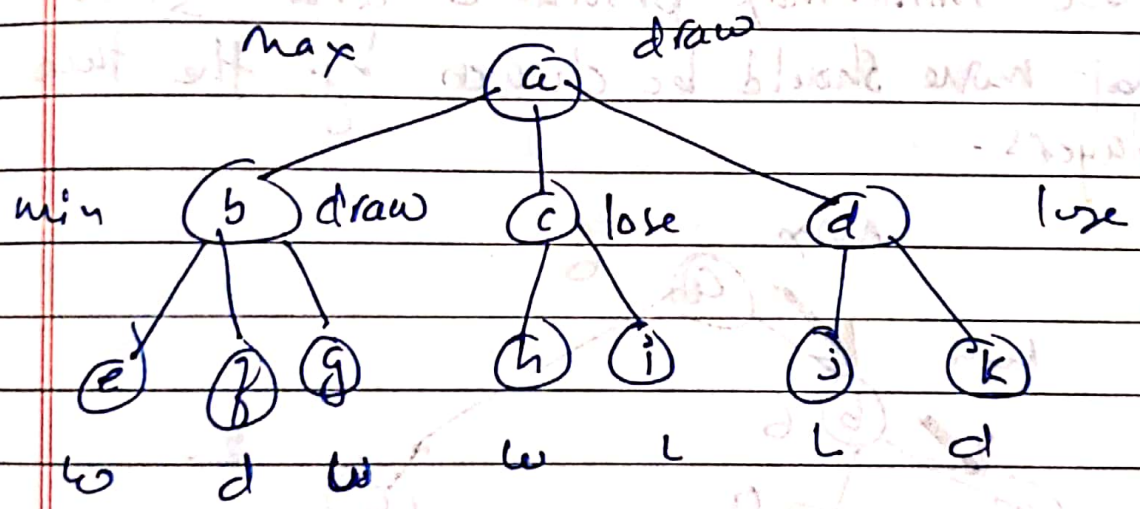
Scanned with CamScanner

MIN-MAX Example

Consider the following game tree (drawn from the point of view of Maximizing player.



What move should be chosen by the Max player, and what should be the response of min player, assuming both are using the mini-max procedure?

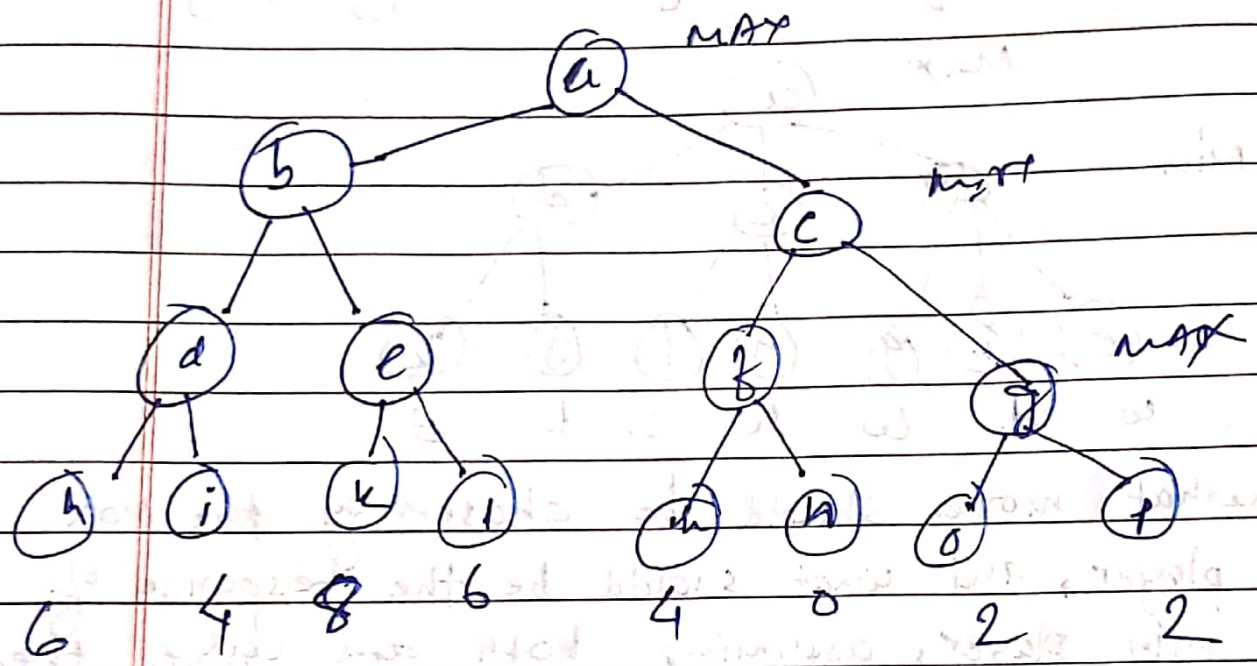


Max will move to b & min will respond by moving to f.

Scanned with CamScanner

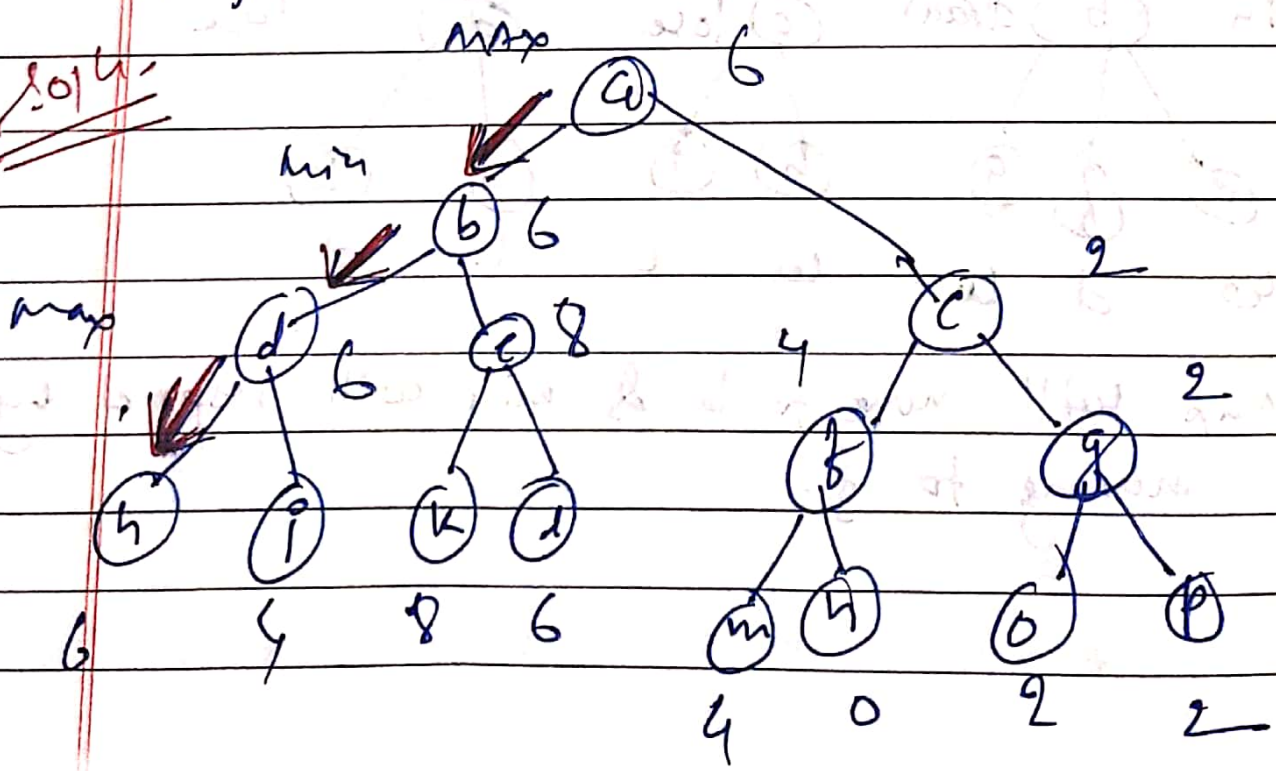
Example 2.

Consider the following game tree (drawn from the point of view of maximizing player)



(a) Use mini-max procedure and show what move should be choosen by the two players.

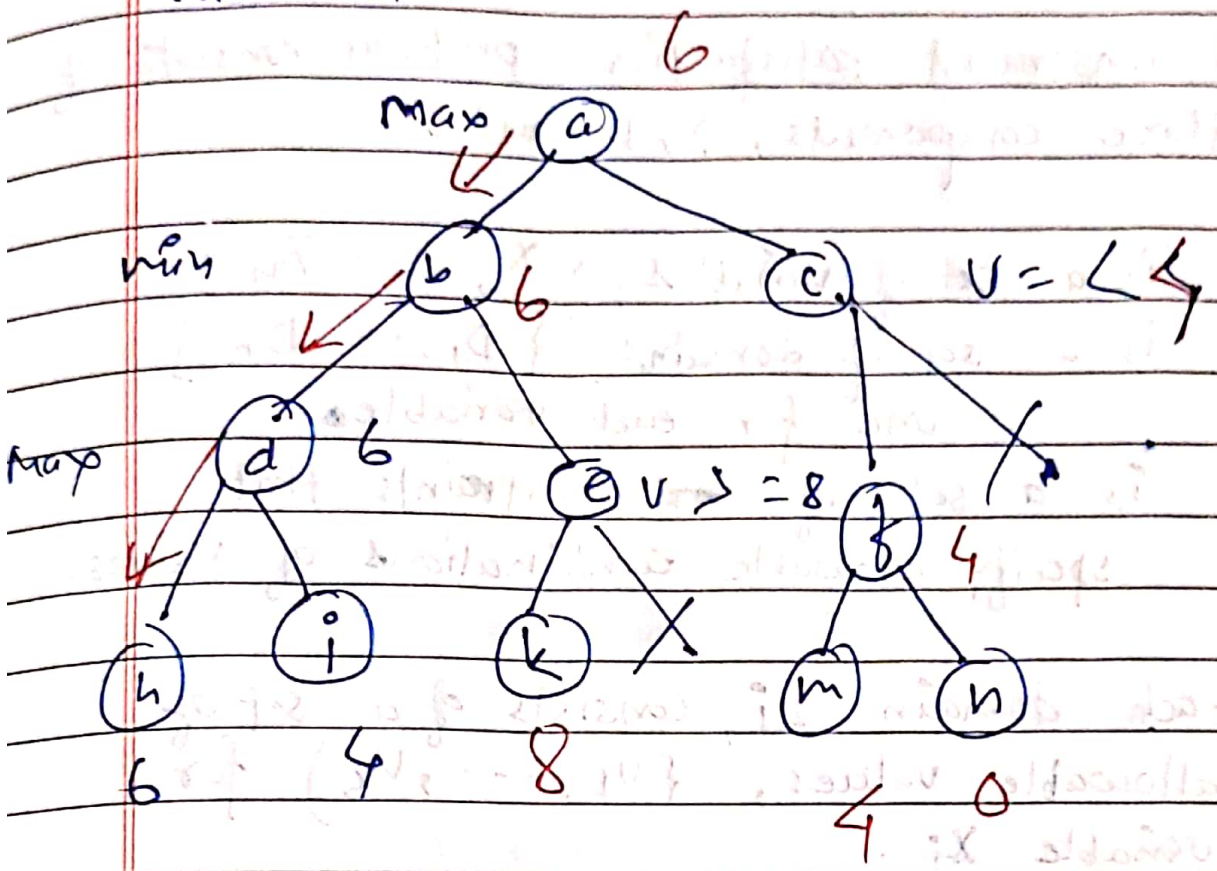
Soln:



Scanned with CamScanner

Date:

(b) Use the alpha-beta pruning procedure and show what nodes would not need to be examined



Constraint satisfaction problem

A constraint satisfaction problem consists of three components, X, D, and C

X is a set of variables, $\{X_1, \ldots, X_n\}$
D is a set of domains, $\{D_1, \ldots, D_n\}$ one for each variable.
C is a set of constraints that specify allowable combinations of values.

→ Each domain D_i, consists of a set of allowable values, $\{v_1, \ldots, v_k\}$ for variable X_i.
→ Each constraint C_i consists of a pair (scope, rel) where
scope → tuple of variables that participate in the constraint and
rel → is a relation that defines the values that satisfy the constraint.

Scanned with CamScanner

Example: Map colouring problem

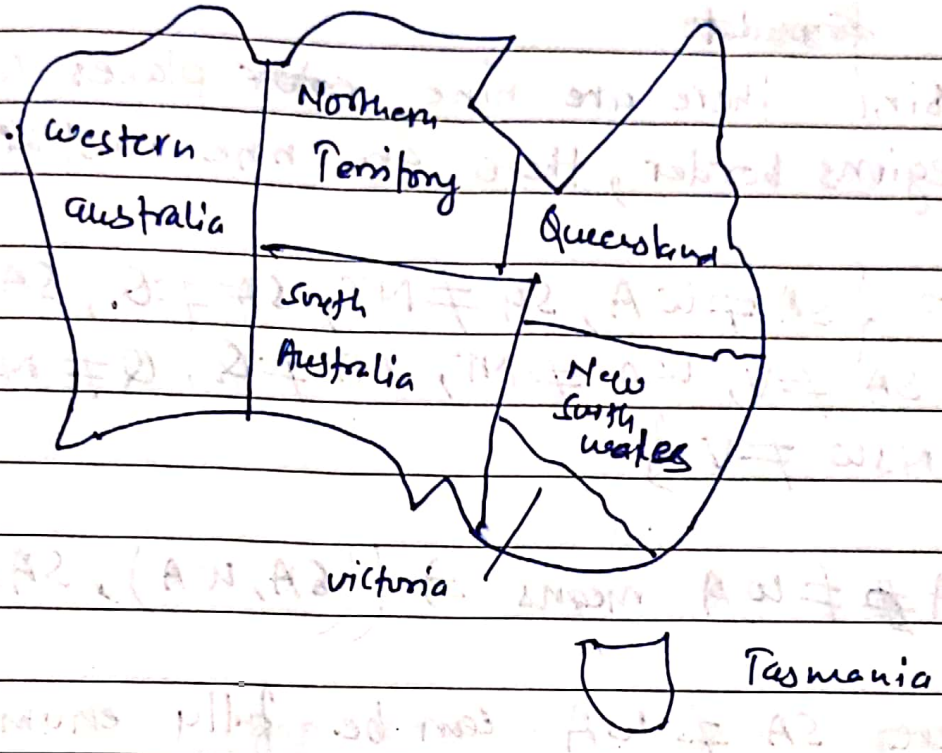


Fig: The principle states and territories of ~~and australia~~ australia.

We are given the task of colouring each region either red, green, or ~~bl~~ blue in such a way that no neighbouring regions have the same color.

→ To formulate this as a CSP, we define the variables to be the regions.

$$X = \{ WA, NT, Q, NSW, V, SA, T \}$$

The domain of each variable is the set

$$D_i = \{ red, green, blue \}$$

Scanned with CamScanner

The constraints require neighbouring regions to have distinct colors.

→ ~~Formulate~~

→ Since there are nine ~~color~~ places where regions border, there are nine constraints:

$C = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}$

$SA \neq WA$ means $\Rightarrow ((SA, WA), SA \neq WA)$

where $SA \neq WA$ can be fully enumerated in turn as

(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green) }

There are many possible solutions to this problem, such as.

$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = red\}$

Scanned with CamScanner

Date:.....

Page:.....

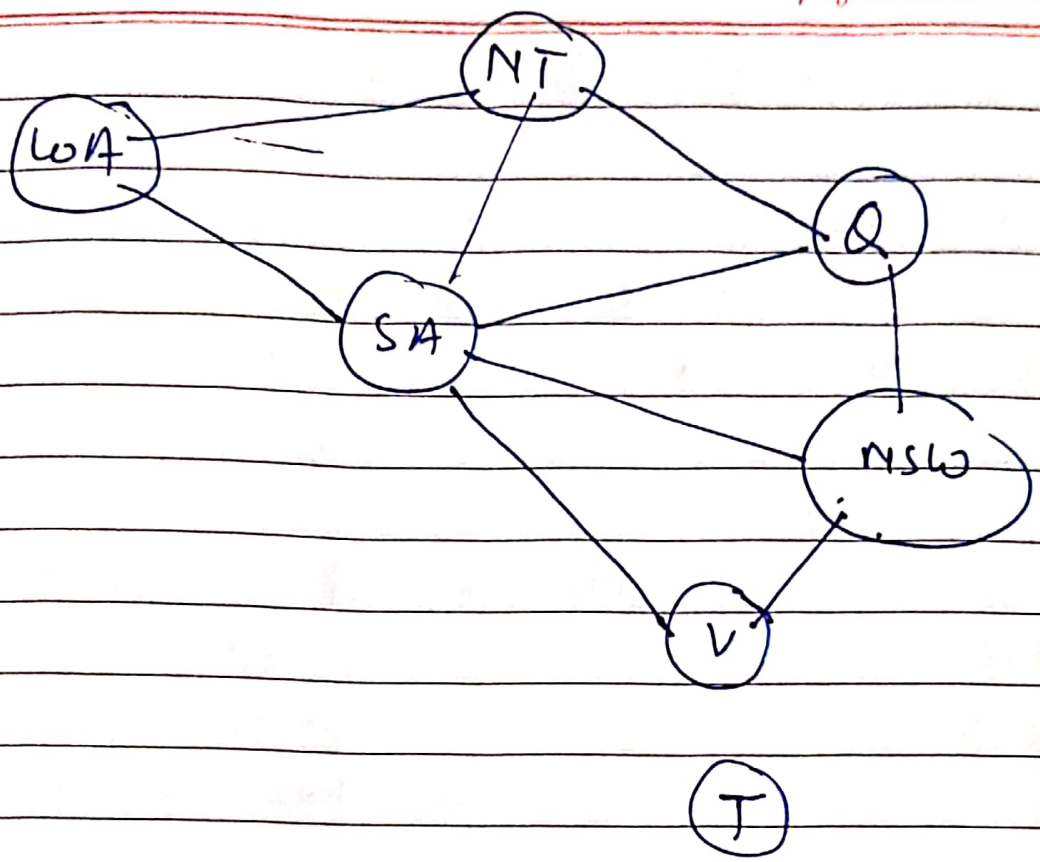


Fig: Map Coloring problem represented as a constraint graph.

Scanned with CamScanner

Unit-4

Knowledge Representation

→ Knowledge is what we know as facts.

→ Knowledge is the information about particular domain that can be used to solve problems in that domain.

→ Knowledge is important in problem solving by computer because to solve a problem by a computer, it first must be represented in computer understandable form.

- ~~To solve problems, it requires full knowledge about the problem~~

→ The process of converting informal knowledge into computer understandable form is known as knowledge representation scheme.

- A representation scheme specifies the form of the knowledge.

→ A knowledge base is the representation of all ~~the~~ of the knowledge that is stored by an agent.

Scanned with CamScanner

Issue in Knowledge Representations

1. Selection of important attributes ⇒ How to find the important attribute from the bank of information about an object or world.

2. Relationship → What about the relationship between the attributes of an object.

3. Granularity → At what level should the knowledge be represented and what are the primitives.

Scanned with CamScanner

Knowledge Representation System and its properties

→ Knowledge representation system is a formalized structure and sets of operations that embodies the descriptions, relationships, and procedures provided in an AI system.

It basically consists of two components :

1. Knowledge base
2. Inference methods.

A good knowledge representation system should consists of following properties.

(1) Representational adequacy ⟹ The scheme is rich enough to express the knowledge needed to solve the problem.

- In other words, the ability to represent all of the kinds of knowledge that are needed in a certain domain.

(2) Inferential adequacy ⟶ The ability to represent all ~~kind~~ of the kinds of inferential procedures

Scanned with CamScanner

③ Inferential efficiency → The ability to represent efficient inference procedures.

④ Acquisitional efficiency → The ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Types of Knowledge Representation schemes

Semantic nets

→ It is a graphical notation for representing knowledge in interconnected nodes pattern.

→ It is a directed or undirected graph consisting of vertices, which represent concepts, and edges, which represent semantic relations between concepts.

→ The meaning of concepts emerges from how it is connected to other concepts.

Scanned with CamScanner

Semantic network can
→ Show natural relationship between objects / concepts
→ be used to represent declarative / descriptive knowledge.

→ Knowledge is represented as a collection of concepts, represented by nodes.
→ Thus, semantic networks are constructed using nodes linked by directional line called arcs.

A node can represent a fact description
- physical object
- concept, attribute, values
- event

An arc (or link) represents relationship between nodes. There are some standard relationship types
'Is-a' (instance relationship) ⇒ represent class / instance relationships
'Has-a' (part-subpart relationship) ⇒ identify property relationships

Scanned with CamScanner

Components of semantic Network

The fundamental components of semantic N/W are :

lexical part → Nodes - denoting objects
links - denoting relations betn objects
labels → denoting particular objects & relations

Structural part → the link and nodes form directed graphs. The ~~the~~ labels are placed on the links and nodes.

Semantic part → meanings are associated with the link and node labels.

procedural part → Constructors allow creation of new links and nodes
→ Destructors allow the deletion of links and nodes.

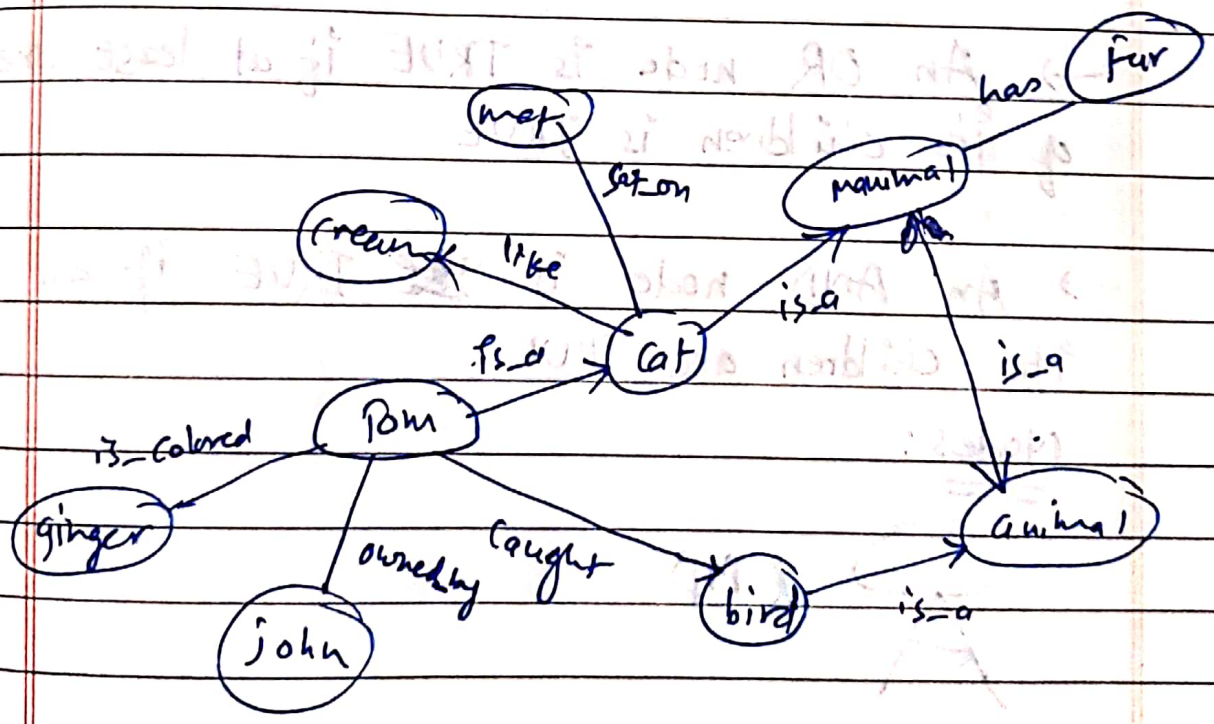
- writers allow the creation and alteration of labels
- readers can extract answers to questions.

Scanned with CamScanner

Date:
Page:

Example:

Tom is a cat
Tom caught a bird
Tom is owned by John
Tom is ginger in color.
Cats like cream
The cat sat on the mat
A cat is a mammal
A bird is an ~~mammal~~ animal
All mammals are animals
Mammals have fur



Scanned with CamScanner

Semantic network as AND/OR TREE

→ An AND/OR tree is a tree whose internal nodes are labelled either "AND" or "OR".

→ A valuation of an AND/OR tree is an assignment of "TRUE" or "FALSE" to each of their leaves.

→ Given a tree T and a valuation over the leaves of T, the values of the internal nodes and of T are defined recursively in the obvious way:

→ An OR node is TRUE if at least one of its children is TRUE

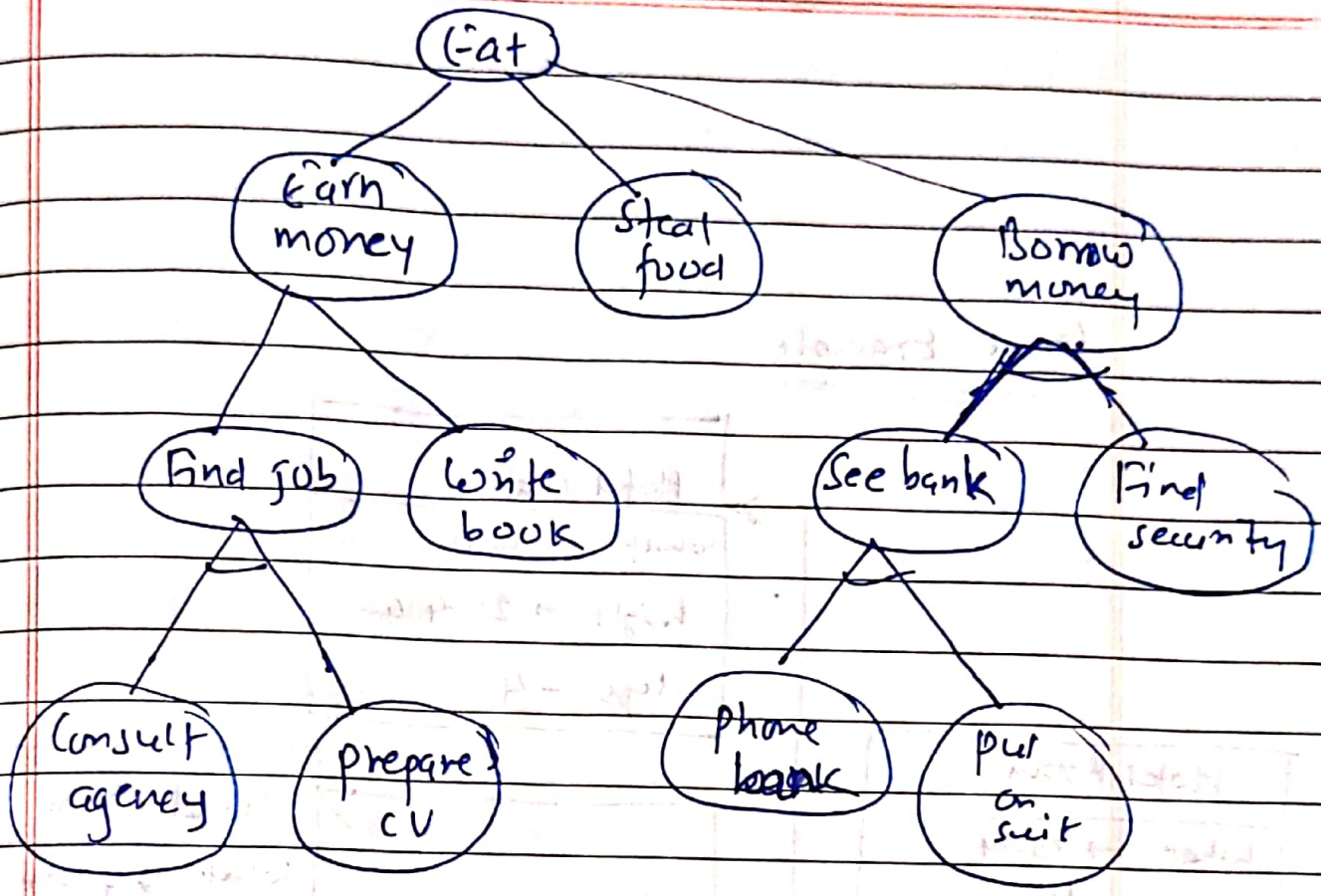
→ An AND node is ~~TRUE~~ TRUE if all its children are TRUE

Nodes:

AND

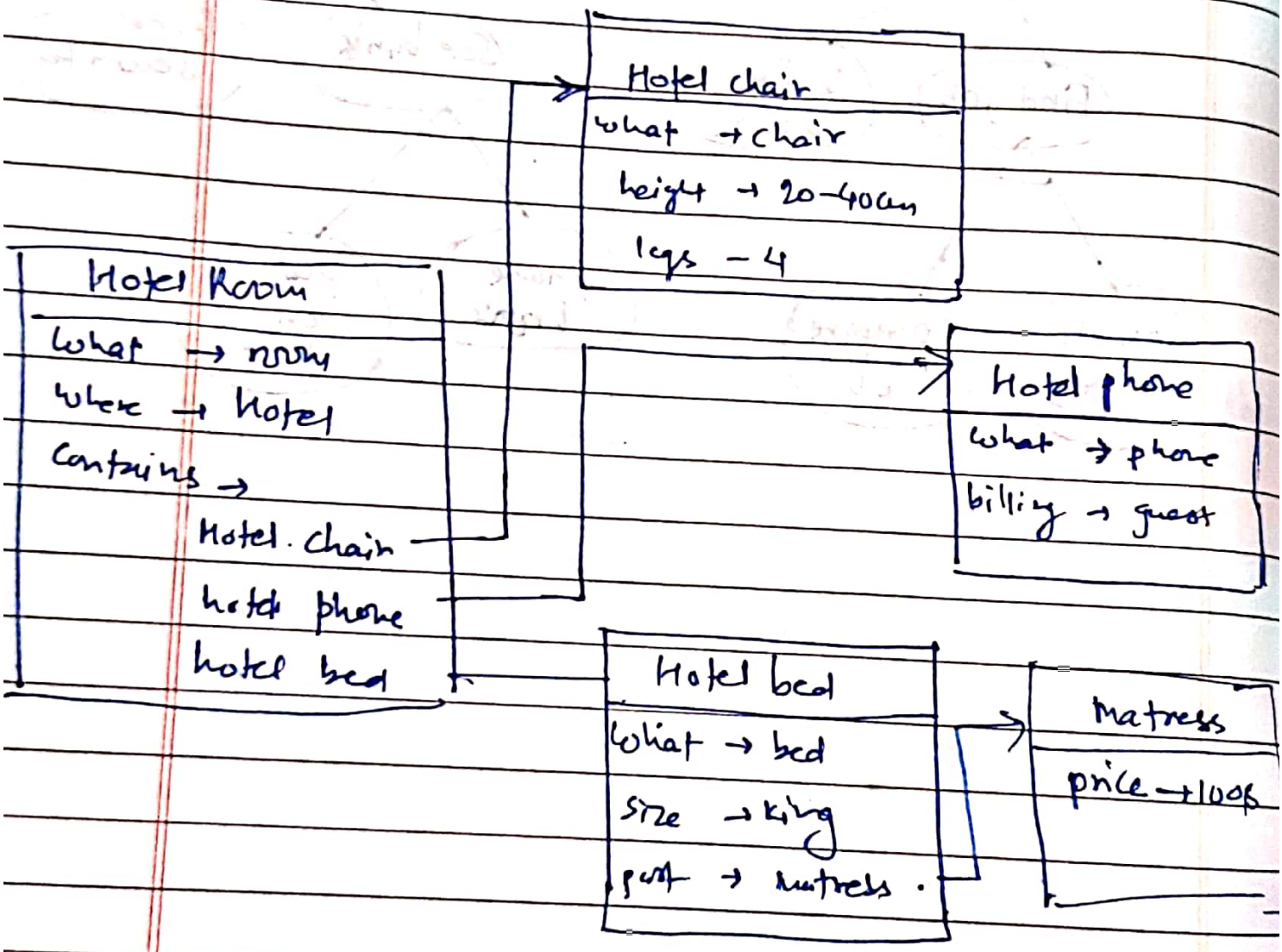
OR

Scanned with CamScanner



Scanned with CamScanner

Frame Example



Scanned with CamScanner

Frame based Knowledge Representation

→ A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world.

→ Frames are the AI data structure which divides knowledge into substructures by representing stereotype situations.

→ Frame are semantic net with properties

→ ~~These slots~~

→ It consists of a collection of slots and slot values.

→ These slots may be of any type and sizes. Slots have names & values which are called facets.

Components of a Frame

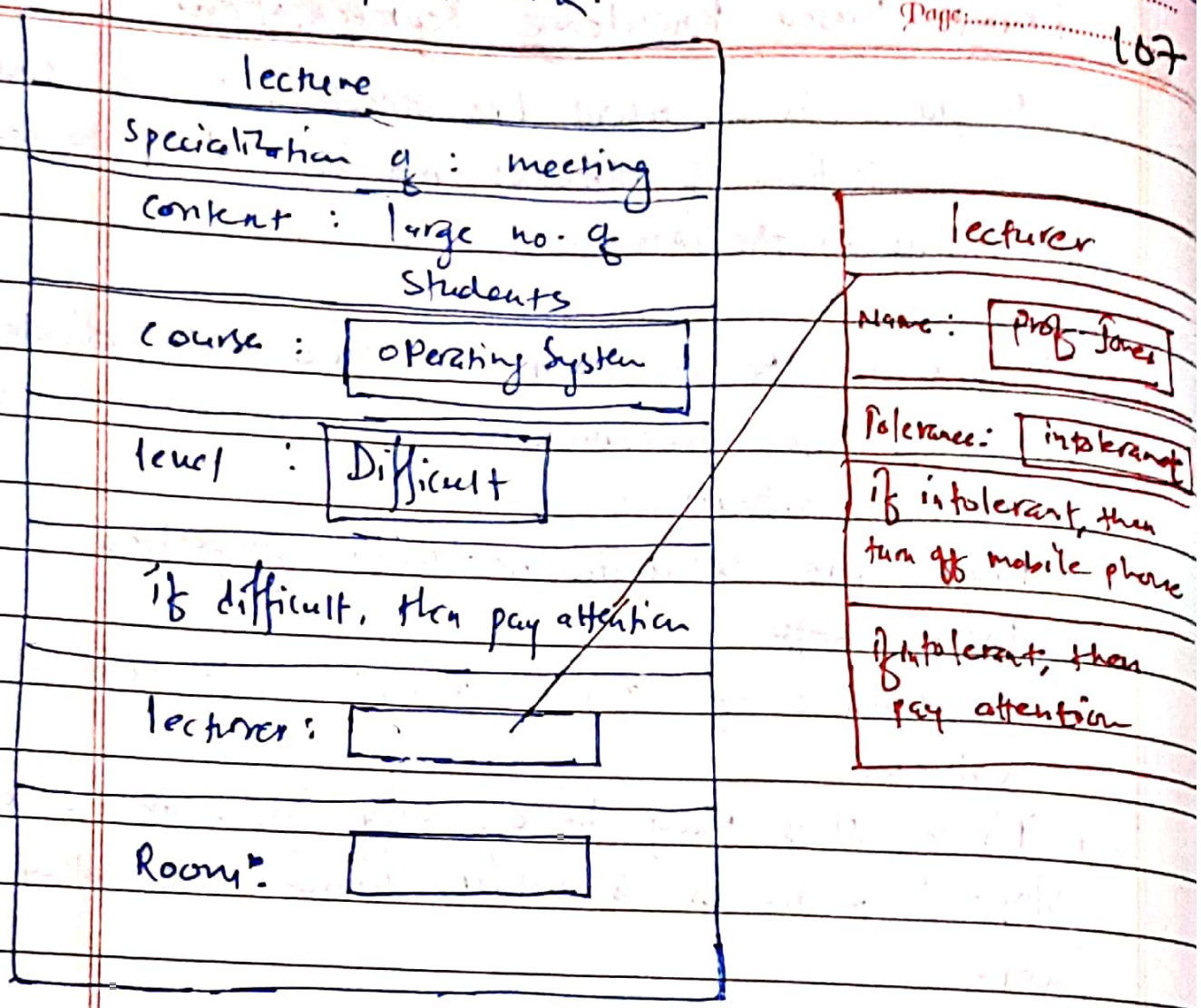
- Frame Name
  - attributes (slots)
  - values (filler : list of values, range, string etc)

Simple frame

Slot name	Filler
name	Asterix
height	small
weight	low
profession	warrior
armour	helmet
intelligence	very high

Scanned with CamScanner

following fig. show the two frame : lecture & lecturer connected with a link.



→ More natural support of values than semantic nets (each slots has constraints describing legal values that a slot can take)

→ similar to OO paradigm

→ Can be easily implemented using object-oriented programming technique.

→ inheritance is easily controlled.

Scanned with CamScanner

Conceptual Dependencies

→ Conceptual Dependency (CD) is a theory of natural language processing which mainly deals with representation of semantics of a language.

→ Developed by ~~shank~~ schank in 1973 to 1975 to represent the meaning of Natural language sentences

→ CD representation of a sentences is not built using the words in the sentences rather ~~is~~ built using conceptual primitives which gives the intended meaning of words.

→ It helps in drawing inferences
→ It is independent of the language.

The main motivation for the development of CD as a KR technique:

→ To construct computer program which can understand natural language.

→ To make inference from the statements and also to identify conditions in which two sentences can have similar meaning

Scanned with CamScanner

Conceptual dependency primitives

→ There are four conceptual categories

ACT → Actions (Real world actions)
PP → Real world objects (picture producer)
AA → Attributes of Action (action aiders)
PA → Modifiers of PP's (picture Aiders)

Standard CD primitives.

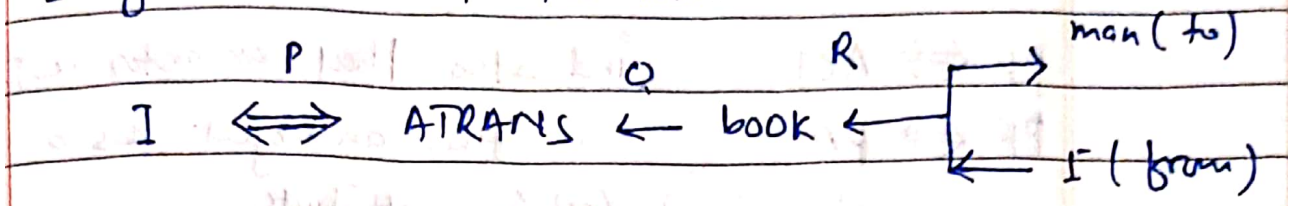
ATRANS → transfer a relationship (give)
PTRANS → transfer physical location of an object (go)

PROPEL → apply physical force to an object (push)
MOVE – move body part by owner (kick)
GRASP → grab an object by an ~~ani~~ actor (grasp)
INGEST → ingest an object by an animal (eat)
EXPEL → expel from an animal's body (cry)
MTRANS → transfer mental information (~~tell~~ ~~decide~~ tell)
MBUILD → mentally make new information (decide)
CONC → conceptualize or think about an idea (think)
SPEAK → produce sound (say)
ATTEND → focus sense organ.

Scanned with CamScanner

Example:

I gave a book to the man.



Above representation is same for different saying with same meaning

→ I gave the man a book;
→ The man got book from me,
→ The book was given to man by me etc.

CD notations.

Arrows indicate directions of dependency
Double arrow indicates two way link between actor and action

O – for the object case relation
R – for the recipient case relation
P – for past tense
D – Destination.

Scanned with CamScanner

Conceptual Dependency syntax rules

PP ⟺ ACT — indicates that an actor acts.

PP ⟺ PA — " that an object has a certain attribute

ACT ←O— PP — indicates the object of an action

ACT ←R— PP, ACT ←R— PP — indicates the recipient and doner of an object

ACT ←D— PP, ACT ←D— PP — indicates the direction of an object within an action

ACT ←I— ⇕ — indicates the instrumental conceptualization for an action

X ⇑ Y — indicates that conceptualization X caused conceptualization Y

PP ⇐ PA2, PP ⇐ PA1 — indicates a state change of an object

Scanned with CamScanner

Some basic conceptual dependency

1. PP $\Leftrightarrow$ ACT

John ran John $\overset{p}{\Leftrightarrow}$ PTRANS

2. PP $\Leftrightarrow$ PA

John is tall John $\Leftrightarrow$ height (> average)

3. PP $\Leftrightarrow$ PP

John is a doctor John $\Leftrightarrow$ doctor

4. PP $\leftarrow$ PA

A nice boy boy $\leftarrow$ nice

5. ACT $\overset{o}{\leftarrow}$ PP

John pushed the cart John $\overset{p}{\Leftrightarrow}$ PROPEL $\overset{o}{\leftarrow}$ cart

7. ACT $\xleftarrow{R}$ PP, PP

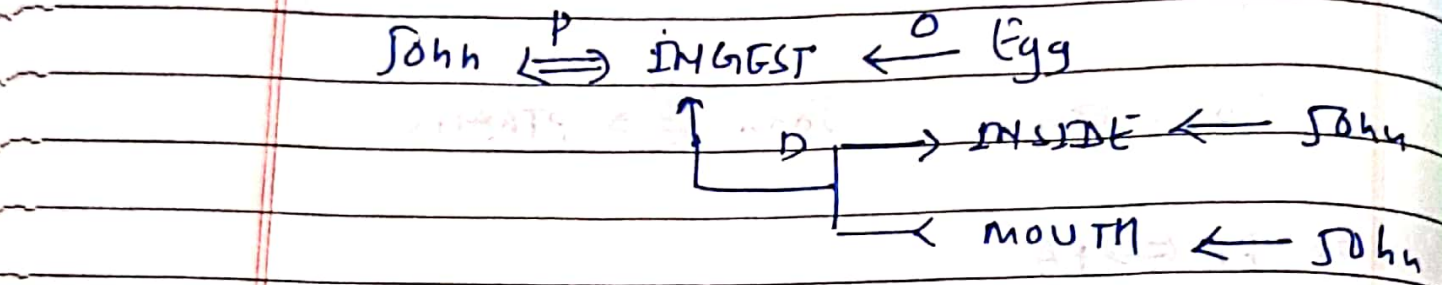
John took book from mary

John $\overset{p}{\Leftrightarrow}$ ATRANS $\xleftarrow{R}$ John, mary

ATRANS $\xrightarrow{o}$ book

Example

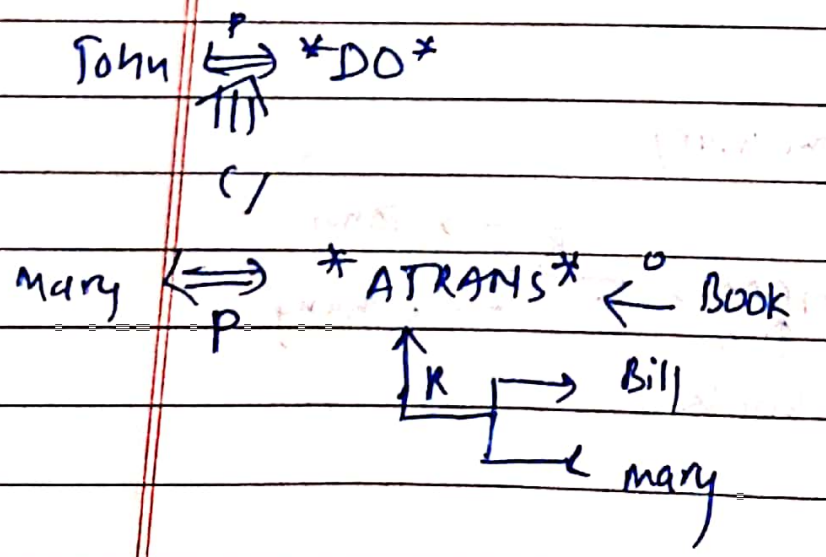
John ate the egg



←	⇒	the direction of dependency
↔	⇒	The agent - verb relationship
p	⇒	past tense
INGEST	⇒	A primitive act of the theory
O	⇒	Object relation
D	⇒	the direction of the object in action.

Example:

John prevented Mary from giving a book to Bill



p	⇒	past
f	⇒	future
t	⇒	transition
k	⇒	continuing
c	⇒	conditional
/	⇒	negative
?	⇒	interrogative
p/f	⇒	present

Scanned with CamScanner

Problems with CD representation

→ It is difficult to construct original sentence from its corresponding CD representation

→ Can be used as general ~~representation~~ model for KR.

→ Rules are to be carefully ~~defined~~ designed for each primitive action in order to obtain semantically correct interpretation.

→ Many verbs may fall under different primitive ACTs, and it becomes difficult to find correct primitive in the given context.

→ The CD representation becomes complex requiring lot of storage for many simple actions.

Scanned with CamScanner

Script structure

A script is a knowledge representation scheme similar to a frame. but instead of describing an object, the script describes a sequence of events.

→ It was designed by ~~schank~~ schank in 1974.

→ It is a structured representation describing a stereotyped sequence of events in a particular context.

→ It is a means of organizing conceptual dependency structures.

→ Used in natural language understanding for knowledge base.

Script Components

Each script contains the following main Components

1. Entry conditions → Must be satisfied before events in the script can occur.

Scanned with CamScanner

2. Results → Conditions that will be true after events in script occur.

3. Props → or the "things" that support the content of the script.

4. Roles → Person involved in the events.

5. Track → Specific variation on more general pattern in the script.
Different tracks may share many components of the same script but not all.

6. Scene → The sequence of events that occur. Events are represented in conceptual dependency form.

Scanned with CamScanner

Example!

Script: play in Theater

Track: play in theater

Props:
→ Tickets
→ Seat
→ play

Roles:
person (who wants to see a play) – P.
Ticket distributor – TD
Ticket checker – TC

Entry Conditions:
→ P wants to see a play
→ P has a money

Results
→ P saw a play
→ P has less money
→ P is happy (optional if he liked the play)

Various scene

Scene 1: Going to theater
P PTRANS P into theater
P ATTEND eyes to Ticket counter

Scene 2: Buying ticket
P PTRANS P to ticket counter
P MTRANS to TD (need a ticket)
TD ATRANS ticket to P

Scene 3: Going inside hall of theater and sitting on a seat

P PTRANS P into Hall of theater
TC ATTEND eyes on ticket pass by P
TC MTRANS to P (showed seat)
P PTRANS P to seat
P MOVES P to sitting position

Scene 4: watching a play
P ATTEND eyes on play
P MBUSLD (good moments) from play

Scene 5: Exiting
P PTRANS P out of Hall and theater

Example: 2

Script: Robbery Track: Successful snatch

Props:
G = Gun
L = loot
B = Bag
C = Get away car

Roles:
R = Robber
M = Cashier
O = Bank manager
P = policeman

Entry Condition:
R is power
R is destitute

Results:
R has more money
O is angry
M is in the state of shock
P is shot

Scene 1: Getting a gun
R PTRANS R into Gunshop
R MBUILD R choice of G
R MTRANS choice
R ATRANS ~~by~~ buys G
(goto scene 2)

Scene 2: Holding Up the bank
R PTRANS R into bank
R ATTEND eyes M, O, and P
R MOVE R to M position
R GRASP G
R MOVE G to point to M

Scanned with CamScanner

R MTRANS " Give me the money or ELSE " to M
P MTRANS " Hold it hands Up " to R
R PROPEL shoots G
P INGEST bullet from G
M ATRANS L to M
M ATRANS L puts in Bag, B
M PTRANS exit
O ATRANS raises the alarm.
go to scene 3

Scene 3: The getaway

M PTRANS C

Advantages

→ Capable of predicting implicit events
→ Single coherent interpretation may be buildup from a collection of observations.

Disadvantages

→ More specific & less general than frames
→ Not suitable to represent all kind of knowledge.

Scanned with CamScanner

Rule Based Representation System

→ A rule based system (production system) is a knowledge based system (KBS) in which the knowledge is stored as rules.

→ An Expert system is a rule based system in which the rules come from human experts in a particular domain.

production consists of two parts: A sensory pre-condition and an action.

Sensory pre-condition = "IF" statement
Action = THEN

→ IF the production pre-condition matches the current state of the world, then the production is said to be triggered.

→ A production system also contains a database, sometimes called working memory, which maintains data about current state or knowledge, and a rule interpreter.

→ The rule interpreter must provide a mechanism for prioritizing productions when more than one is triggered.

→ The underlying idea of production systems is to represent knowledge in the form of condition-action pairs called production rules.

Scanned with CamScanner

Types of production rules

(a) Situation - action rules

If it is raining then open the umbrella.

(b) Inference rules

If Cesar is a man then Cesar is a person.

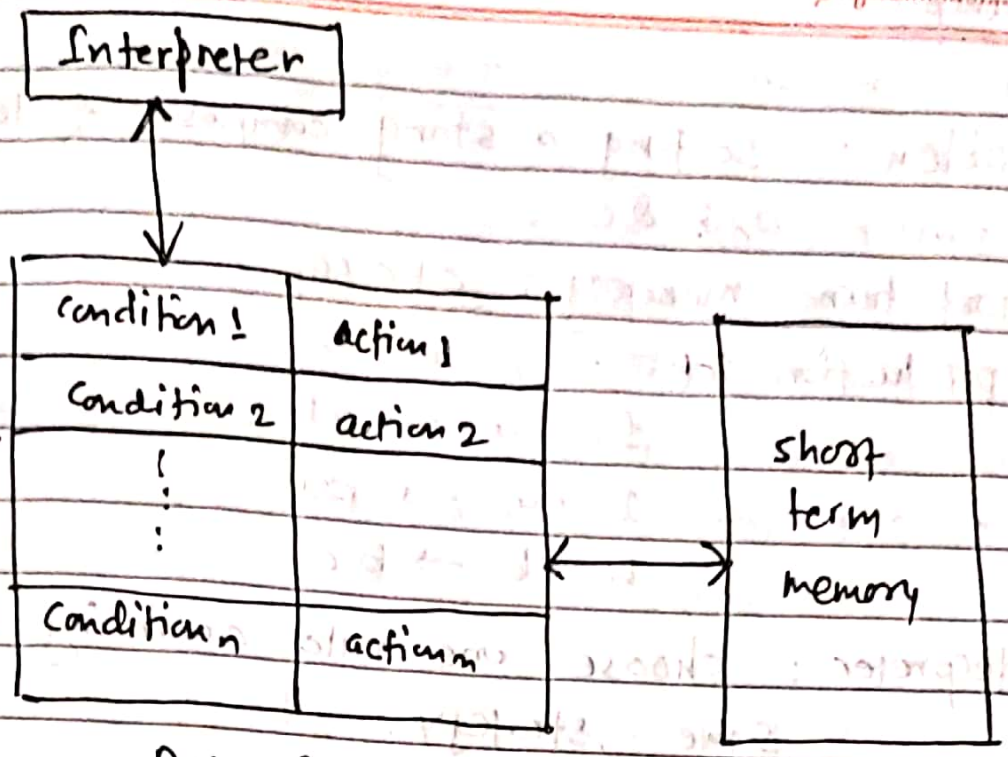
Architecture of production System

Short term memory → contains the description of the current state

set of production rule → set of condition action pair and defines a single chunk of problem solving knowledge.

Interpreter → A mechanism ~~that~~ to examine the short term memory to determine which rule to fire.

Scanned with CamScanner



Rule Base

→ The execution of a production system can be defined as a series of recognize act cycles.

Match ⇒ memory contains matched condition against condition of production rules. this produces a subset of production called conflict set

Conflict resolution → one of the production in conflict set is then selected.
Apply the rule.

This is also called Retrieval, Refinement, and Execution cycle of interpreter.

Scanned with CamScanner

Example :

problem : Sorting a string composed of letters a, b & c

Short term memory : cbaca

production set :

1. ba → ab
2. ca → ac
2. cb → bc

Interpreter : choose one rule according to some strategy.

Iteration	memory	conflict set	Rule fired
0	cbaca	1, 2, 3	1
1	cabca	2,	2
2.	acbca	2, 3	2
3	acbac	1, 3	1
4.	aabac	2	2
5	aacbc	3	3
6.	aabcc	0	Halt

Scanned with CamScanner

The water jug problem

problem → There are two jugs, a 4-gallon one and a 3-gallon one. Neither jug has any measuring marker on it. There is a pump that can be used to fill the jugs with water. How can you get exactly n (0, 1, 2, 3, 4) gallons of water into one of the two jugs?

Solution paradigm

- build a simple production system for solving this problem
- represent the problem by using the state space paradigm.

State = (x, y); where
x = no. of gallons in the 4-gallon jug.
y = no. of gallons in the 3-gallon jug

- The initial state represents the initial content of the two jugs
- The left hand side of a production rule indicates the state in which the rule is applicable and the right hand side indicates the state resulting after the application of the rule.

Scanned with CamScanner

For instance,
(x,y) such that $x<4 \rightarrow (4, y)$ represents the production.
If the 4 gallon jug is not full then fill it from the pump.

The rule base contains the following production rules:

1. (x,y) such that $x<4 \rightarrow (4, y)$: Fill the 4-gallon jug from pump

2. (x,y) such that $y<3 \rightarrow (x, 3)$: Fill the 3-gallon jug from pump

3. (x,y) such that $x>0 \rightarrow (0, y)$: Empty the 4-gallon jug on the ground

4. (x,y) such that $y>0 \rightarrow (x, 0)$: Empty the 3-gallon jug on the ground

5. (x,y) such that $x+y \geq 4, x>0, y>0 \rightarrow (4, y-(4-x))$: Completely fill the 4-gallon jug from the 3-gallon jug.

Scanned with CamScanner

6. (x,y) such that $x+y \geq 3, x>0, y<3 \rightarrow (x-(3-y), 3)$:
Completely fill the 3-gallon jug from the 4-gallon jug

7. (x,y) such that $x+y \leq 4, y>0 \rightarrow (x+y, 0)$
: Empty the 3-gallon jug into the 4-gallon jug

8. (x,y) such that $x+y \leq 3, x>0 \rightarrow (0, x+y)$
: Empty the 4-gallon jug into the 3-gallon jug

The short term memory contains the current state (x,y)

— let us consider the initial situation $(0,0)$ and the goal situation $(4,2)$

Short term memory $(0,0)$

Iteration	memory	Conflictset	Rule fired
1	$(0,0)$	1, 2	2
2.	$(0,3)$	1, 4, 7	7
3.	$(3,0)$	1, 2, 3, 6, 8	2
3.	$(3,3)$	1, 3, 4, 5	5
4.	$(4,2)$	0	Halt

Scanned with CamScanner

The sequences of the applied rule:

Fill the 3-gallon jug from pump
Empty the 3-gallon jug into 4-gallon jug
Fill the 3-gallon jug from pump
Fill the 4-gallon jug from the 3-gallon jug

Scanned with CamScanner

Logic Based Knowledge representation

Formal logical connective

In logic, a logical connective (also called a logical operator) is a symbol or word used to connect two or more sentences (of either a formal or a natural language) in a grammatically valid way, such that the compound sentence produced has a truth value dependent on the respective truth values of the original sentences.

Each logical connective can be expressed as a function, called truth function. For this reason, logical connectives are sometimes called truth-functional connectives.

Commonly used logical connective includes

Negation (not) ($\neg$ or $\sim$)
Conjunction (and) ($\wedge$, &, or $\cdot$)
Disjunction (or) ($\vee$, or v)
material implication (if...then) ($\rightarrow$, $\Rightarrow$, or $\supset$)

Biconditional (if and only if) $\Rightarrow$ (iff) (xnor) ($\leftrightarrow$, $\equiv$, or $=$)

Scanned with CamScanner

For example, the meaning of the statements it is raining and I am indoors is transformed when the two are combined with logical connectives:

$\boxed{\text{It is raining} = P}$ $\boxed{\text{I am indoors} = Q}$

It is raining and I am indoors $(P \wedge Q)$

If it is raining, then I am indoors $(P \rightarrow Q)$

It is raining if I am indoors $(Q \rightarrow P)$

It is not raining if and only if I am indoors $(P \leftrightarrow Q)$

It is not raining $(\neg P)$

Logic

→ Logic is a formal language for representing knowledge such that the conclusion can be drawn.

→ Logic makes statements about the world which are true (or false) if the state of affairs it represents is the case (or not the case).

→ Logic is concise, unambiguous, expressive, context insensitive, effective for inferences.

→ It has syntax, ~~semat~~ semantic and proof theory.

Scanned with CamScanner

Date:
Page:

Syntax → Describes the possible configurations that constitute the sentences

Semantics → Determines what fact in the world, the sentence refers to i.e. interpretation.

proof theory (Inference method) → set of rules for generating new centences that are necessarily true given that the old sentences are true.

- Here we study two kinds of logic:
  propositional logic
  predicate logic / first-order logic

Few concepts

Entailment ⇒ Entailment means that one thing follows from other

$KB \models \alpha$

→ Knowledge Base entails sentence α if and only if α is true in all worlds where KB is true.

→ Entailment is the relationship between sentences that is based on semantics.

Scanned with CamScanner

Date:
Page:

Example:

p = All dogs are purple

Q = My dog is purple

$p \models q$

$p \models q$ iff $p = T$ & $q = T$

Propositional Logic

→ Propositional logic represents knowledge/information in terms of propositions. propositions are facts and non-facts that can be true or false.

- propositions are expressed using ordinary declarative sentences.
- It is the simplest logic.

- The complex sentences are constructed from simpler sentences using logical connectives. There are five connectives in common use:

$\neg$ (negation)
$\wedge$ (conjunction)
$\vee$ (disjunction)
$\Rightarrow$ implication
$\Leftrightarrow$ biconditional

Scanned with CamScanner

Date:
Page:

The order of precedence in propositional logic is from (highest to lowest) : $\neg, \wedge, \vee, \Rightarrow \Leftrightarrow$

propositional logic is defined as

If S is a sentence, $\neg S$ is a sentence (negation)
If S1 and S2 are sentences, $S1 \wedge S2$ is a sentence (conjunction)
If S1 & S2 are sentences, $S1 \vee S2$ is a sentence (disjunction)
If S1 & S2 are sentences, $S1 \Rightarrow S2$ is a sentence (implication)
If S1 & S2 " " $S1 \Leftrightarrow S2$ " " (biconditional)

Formal grammer for propositional logic can be given as below

Sentences → automaticSentence | ComplexSentence
Automatic Sentence → True | False | Symbol
Symbol → P | Q | R - - -

Complex Sentence → ¬ Sentence
| (Sentence ∧ Sentence)
| (Sentence ∨ Sentence)
| (Sentence ⇒ Sentence)
| (Sentence ⇔ Sentence)

Scanned with CamScanner

Propositional logic: Semantics

Each model specifies true / false for each proposition symbol.

Rules for evaluating truth with respect to a model

$\neg S$ is true if, S is false
$S1 \wedge S2$ is true if $S1$ is true and $S2$ is true
$S1 \vee S2$ is true if $S1$ is true or $S2$ is true
$S_1 \Rightarrow S_2$ is true if, $S1$ is false or S_2 is true
$S_1 \Leftrightarrow S_2$ is true if, $S1 \Rightarrow S2$ is true and $S_2 \Rightarrow S_1$ is true.

Truth Table showing the evaluation of semantics of complex sentences.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
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

Scanned with CamScanner

Date:
Page:

Logical Equivalence

Two sentences α and β are logically equivalent ($\alpha \equiv \beta$) iff true they are true inn same set of models or Two sentences α and β are logically equivalent ($\alpha \equiv \beta$) iff $\alpha \models \beta$ and $\beta \models \alpha$.

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

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

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

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ " " $\vee$

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

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ Contraposition

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

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ 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

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

Scanned with CamScanner

Validity

A sentence is said to be valid if it is true under all interpretations.
Valid sentences are also called tautologies and equivalent to true logically.

Example.

$$A \wedge \neg A, \quad (A \wedge (A \rightarrow B)) \rightarrow B$$

Satisfiability

A sentence is said to be satisfiable if it is true in some models or under some interpretation.

Example: $A \vee B$, $B \rightarrow (B \rightarrow A)$

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

Scanned with CamScanner

Date:
Page:

A	B	$B \to A$	$B \to (B \to A)$
T	T	T	T
T	F	T	T
F	T	F	F
F	F	T	T

Unsatisfiability

A sentence is unsatisfiable if it is true in no models or under any interpretation

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

A	$\neg A$	$A \wedge \neg A$
T	F	F
F	T	F

A	B	$A \vee B$	$\neg A$	$\neg B$	$\neg A \wedge \neg B$	$(A \vee B) \wedge (\neg A \wedge \neg B)$
T	T	T	F	F	F	F
T	F	T	F	T	F	F
F	T	T	T	F	F	F
F	F	F	T	T	T	F

Scanned with CamScanner

Examples.

① State whether the following sentences are valid, satisfiable or unsatisfiable.

a. Rain $\vee$ $\neg$ Rain
b. Rain $\rightarrow$ mud
c. (Rain $\vee$ mud) $\wedge$ ($\neg$ Rain $\wedge$ $\neg$ mud)

① let Rain = R
mud = M

ⓐ

R	$\neg R$	$R \vee \neg R$
T	F	T
F	T	T

$\therefore$ given statement is valid as well as satisfiable and is not unsatisfiable.

ⓑ

R	M	$R \rightarrow M$
T	T	T
T	F	F
F	T	T
F	F	T

$\therefore$ The given statement is not valid, is satisfiable and not unsatisfiable.

Scanned with CamScanner

Date:
Page:

(c)

R	M	R∨M	¬R	¬M	¬R ∧ ¬M	(R∨M) ∧ (¬R ∧ ¬M)
T	T	T	F	F	F	F
T	F	T	F	T	F	F
F	T	T	T	F	F	F
F	F	F	T	T	T	F

∴ Above sentence is:

Not valid, not satisfiable
unsatisfiable

(e) State whether the following sentences are valid, satisfiable or unsatisfiable.

(a) Smoke → Smoke
(b) Smoke → fire
(c) (Smoke → fire) → (¬Smoke → ¬fire)
(d) Smoke ∧ fire ∧ fire.

(a) let Smoke = S, Fire = F

S	S → S
T	T
F	T

The sentence is,

- valid
- satisfiable
- Not unsatisfiable

Scanned with CamScanner

(b)

S	F	$S \to F$
T	T	T
T	F	F
F	T	T
F	F	T

Above sentence is

- Not valid
- Satisfiable
- Not unsatisfiable

(c)

S	F	$\neg S$	$\neg F$	$S \to F$	$\neg S \to \neg F$	$(S \to F) \to (\neg S \to \neg F)$
T	T	F	F	T	T	T
T	F	F	T	F	T	T
F	T	T	F	T	F	F
F	F	T	T	T	T	T

Above sentence is

- Not valid
- Satisfiable
- Not unsatisfiable

Scanned with CamScanner

Date:

S	F	¬F	S ∧ F	S ∧ F ∧ ¬F
T	T	F	T	F
T	F	T	F	F
F	T	F	F	F
F	F	T	F	F

Above sentence is
- Not valid
- Not satisfiable
- unsatisfiable

Inference rules in proposition logic

→ Inference rules are the standard patterns of inferences that can be applied to derive ~~conclu~~ conclusions from the given facts.

→ An inference theorem or rule is a mechanism of producing new sentences from other sentences.

→ In other words, the inference rules produce a new sentences based on syntactic form of logical assertions (facts)

→ Given below are the inference theorems:

Scanned with CamScanner

Date:
Page:

Modus ponens

This rule says if there is an expression $p \to Q$ and there is another expression 'p' such that both true under any interpretation then modus ponen rule allows to infer that Q is true for that interpretation.

let us consider,

p : It is raining
Q : Ground is wet

Given,

$p \to Q$: If it is raining then ground will be wet.
p : It is raining.
$\Rightarrow$ Q : Ground is wet.

$$\frac{p \Rightarrow Q, \; p}{Q}$$

Modus Tollen

If there is an expression $p \to Q$ and there is another expression 'Q' such that $p \to Q$ is true, and Q is false then we infer $\neg p$ is logically true.

Scanned with CamScanner

$P \rightarrow Q$: It is raining then ground will be wet.
$\neg Q$: Ground is not wet
$\Rightarrow$ It is not raining : $\neg P$

AND Elimination

This allows to infer the truth of the conjuncts from the truth of the conjunctive sentences.

Example

If the conjunctive sentence $P \wedge Q$ is true then both the conjuncts P and Q are also true.

AND Introduction

It is the inverse of AND elimination. It infers the truth of the conjunctive sentences from the truth of its conjuncts.

Example:

If the conjuncts P and Q both are true then the conjunctive sentences $P \wedge Q$ is also true.

Scanned with CamScanner

Universal Instantiation ($\forall$)

This states that for any universally quantified variable in a true sentence is replaced by any appropriate term from the domain; then the result is a true sentence.

Example:

$\forall x$: plays (x, cricket)
$\Rightarrow$ plays (Ram, cricket)

~~Given~~.

Unification

It determines the substitution needed to make calculations expressions match which is essential for applying inference rule.

Given,

knows (Rabin, x), knows (Rabin, Suzi)

$\Rightarrow$ { x | Suzi }

Scanned with CamScanner

Resolution

If there is an expression of the form $E1 \vee E2$ and there is another expression of the form $\neg E2 \vee E3$, then $E1 \vee E3$ is called to be logically followed.

The expression $E1 \vee E3$ is said to be resolvent of the expressions

$$E1 \vee E2 \text{ and } \neg E2 \vee E3$$

Example: If we have some propositional sentences

- He is smart or innocent ($E_1 \vee E_2$)
- He is not innocent or intelligent ($\neg E_2 \vee E_3$)

By resolution rule, this logically infers that he is smart or intelligent.

Hypothetical Syllogism

This rule specifies that if there is an expression

$$p \rightarrow q$$

and there is another expression

$$q \rightarrow r$$

$$\Rightarrow p \rightarrow r$$

Scanned with CamScanner

For example:

1. If today is sun day, then today is raing day

2. If today is rainy day then it is wet day.

By transiting this logically infers that If today is sunday then it is wet day.

Disjunctive sylloligsm

If there is an expression $p \vee Q$ and there is another expression $\neg p$ then we infer that Q is logically true.

$$p \vee Q$$
$$\neg p$$
$$\Rightarrow Q$$

For example: (p) (Q)

- Ram is student or he is smart
- Ram is not student ($\neg p$)

$\Rightarrow$ Ram is smart.

Scanned with CamScanner

Addition Rule

If there is an expression P and another expression Q, then by using addition rule, either we can use OR or AND rule to the propositions.

Example:

If it is raining and snowing are the two propositions P and Q then by addition rule it infers that $P \vee Q$ is logically true i.e.

It is raining or snowing.

Conjunctive Syllolizsm

If there is an expression $P \wedge Q$ and there is another expression $\neg P$ then we infer that Q is logically true.

- $P \wedge Q$
- $\neg P$

$\Rightarrow Q$

Scanned with CamScanner

Normal Form

Approach that is applied to inference the use of syntactic operation on sentences is often expressed in standard form known as normal form.

Benifits

→ It makes consistency between ~~normal~~ ~~form~~ expression.

→ It makes identical expression easier to mark.

→ It makes reasoning and simplifying easier

There are following types of normal forms in AI

1. CNF
2. DNF
3. HNF

Scanned with CamScanner

CNF (Conjunctive Normal Form)

A sentence that is expressed as a conjunction of disjunctions of literals is said to be in CNF.

A formula 'p' is said to be in CNF if it has the form:

$$p = P_1 \wedge P_2 \wedge P_3 \wedge \dots P_n \quad n >= 1$$

where each $P_1, P_2, P_3 \dots P_n$ is disjunction of an atom or negation of an atom (literals)

Example:

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

$\Rightarrow$ It is in CNF

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

where disjuncts are:

$$\left.\begin{array}{l} p \wedge q \wedge r \\ p \wedge r \\ r \end{array}\right\} \text{clauses}$$

This expression is not in CNF

Scanned with CamScanner

Date:
Page:

Steps to convert a statement in CNF

1. $p \rightarrow q \Rightarrow \neg p \vee q$
2. $p \leftrightarrow q \Rightarrow (p \rightarrow q) \wedge (q \rightarrow p)$

3. Apply demorgan's law and reduce the negation symbol so as to bring the negation before the atom.

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

use distributive law and other laws to ~~obt~~ obtain the normal form.

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

Eliminate double negation.

Scanned with CamScanner

Example:

Convert to CNF

$B \Leftrightarrow (A \vee C)$

Eliminate $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta \Rightarrow (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

$(B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$

Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$

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

move $\neg$ inwards using deMorgan's rule and double negation:

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

Apply distributivity law ($\wedge$ over $\wedge$) and flatten:

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

$\therefore$ above expression is in CNF

Scanned with CamScanner

Example 2.

If Ram is not smart then he is not intelligent and if he is not topper then he just pass the exam. Convert it into CNF.

Soln: let,

P : Ram is smart
Q : Ram is intelligent
R : Ram is topper
S : Ram just pass the exam.

Thus,

$$(\neg P \rightarrow \neg Q) \wedge (\neg R \rightarrow S)$$

for, $\neg P \rightarrow \neg Q$	for $\neg R \rightarrow S$
$\neg(\neg P) \vee \neg Q$	$\neg(\neg R) \vee S$
$P \vee \neg Q$	$R \vee S$

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

$\therefore$ It is in CNF

Scanned with CamScanner

Date:
Page:

Resolution Method

Resolution method is an inference rule which is used in both propositional as well as FOPL in different ways.

- This method is basically used for proving the satisfiability of a sentence.
- In resolution method, we use proof by refutation technique to prove the given statement.

Algorithm

- Convert KB into CNF
- Add negation of sentences to be entailed in KB i.e. $(KB \wedge \neg\alpha)$
- Apply resolution rule to resulting clauses.
- The process continues until:

- There are no new clauses that can be added.
  - Hence KB doesnot entail α
- Two clauses resolve to entail the empty clause
  - Hence KB doesnot entail α.

Scanned with CamScanner

Example: Consider the following KB:

1. The humidity is high or the sky is cloudy.
2. If the sky is cloudy, then it will rain
3. If the humidity is high, then it is hot.
4. It is not hot.

Goal: It will rain

Use propositional logic and apply resolution method to prove that the goal is derivable from the given knowledge base.

Soln: let's construct preposition of the given sentences

let: P: Humidity is high
Q: sky is cloudy
R: It will rain
S: It is hot

Now,

1. $P \vee Q$
2. $Q \rightarrow R$
3. $P \rightarrow S$
4. $\neg S$

Scanned with CamScanner

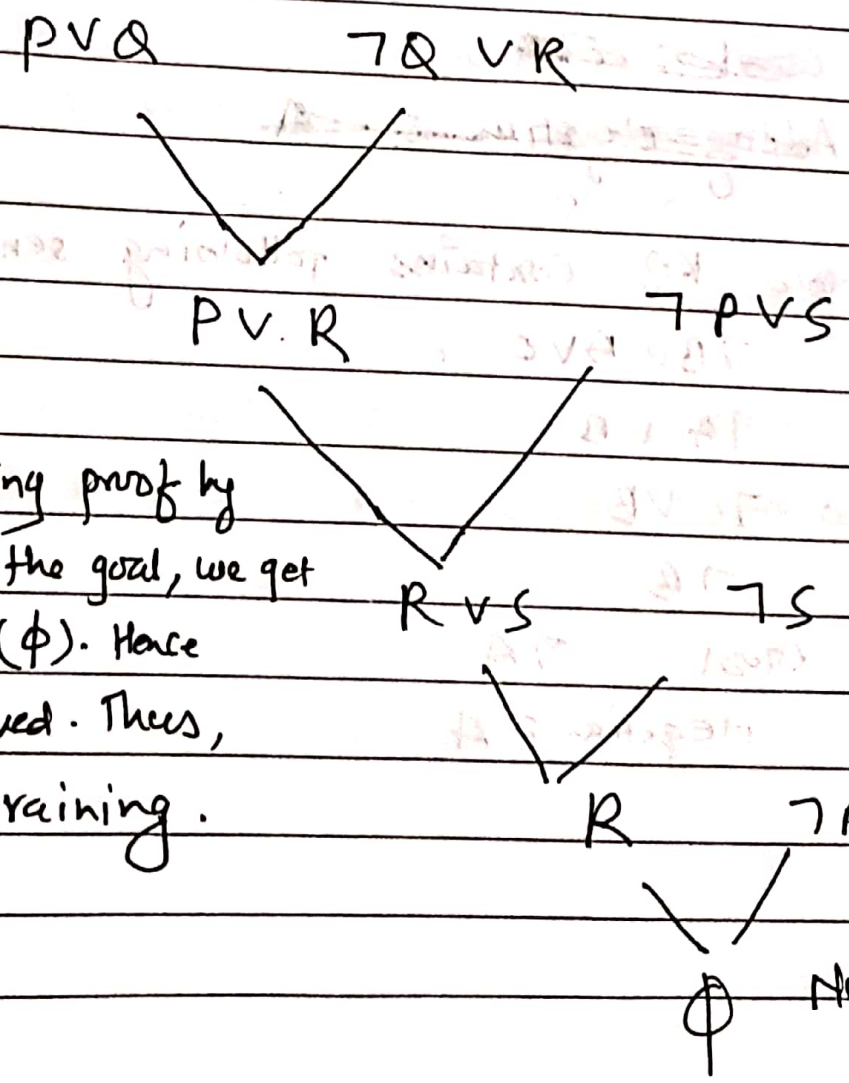
Applying resolution method,

(i) convert given axioms into CNF

(i) $P \vee Q$	(iv) $\neg S$
(ii) $Q \rightarrow R$ $\neg Q \vee R$	
(iii) $P \rightarrow S$ $\neg P \vee S$	

(ii) Negation of the goal : $\neg R$

Now,



After applying proof by refutation on the goal, we get null clause(ϕ). Hence goal achieved. Thus, It is not raining.

Scanned with CamScanner

Example 2 consider the knowledge base given as:

$KB = (B \Leftrightarrow (A \vee C)) \wedge \neg B$

prove that $\neg A$ can be inferred from above KB using resolution.

Soln: At first, convert KB into CNF

$B \rightarrow (A \vee C) \wedge ((A \vee B) \rightarrow B) \wedge \neg B$

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

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

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

~~Goal: ¬A~~

~~Adding negation: A~~

Now KB contains following sentences all in CNF

$\neg B \vee A \vee C$

$\neg A \vee B$

$\neg C \vee B$

$\neg B$

Goal : $\neg A$

Negation : A

Scanned with CamScanner

Now, using Resolution Algorithm,

$\neg B \vee A \vee C$ $\quad$ $\neg A \vee B$

$\neg B \vee C \vee B$ $\quad$ $\neg A \vee B$

$\neg A \vee C \vee B$ $\quad$ $\neg C \vee B$

$\neg A \vee B$ $\quad$ $\neg B$

$\neg A$ $\quad$ A

ϕ Null

After applying proof by refutation, we get null clause (ϕ). Hence goal achieved. //

Scanned with CamScanner

Horn clause

A Horn clause is disjunction of literals with at most one positive literal

$$(A \wedge B) \rightarrow C$$
$$\neg(A \wedge B) \vee C$$
$$\neg A \vee \neg B \vee C$$

$\therefore$ expression is in Horn clause because there is only one positive literal.

properties

- → Can be written as implication
- → Inference through forward chaining and backward chaining
- → Deciding entailment can be done in a time linear size of the knowledge base.

Scanned with CamScanner

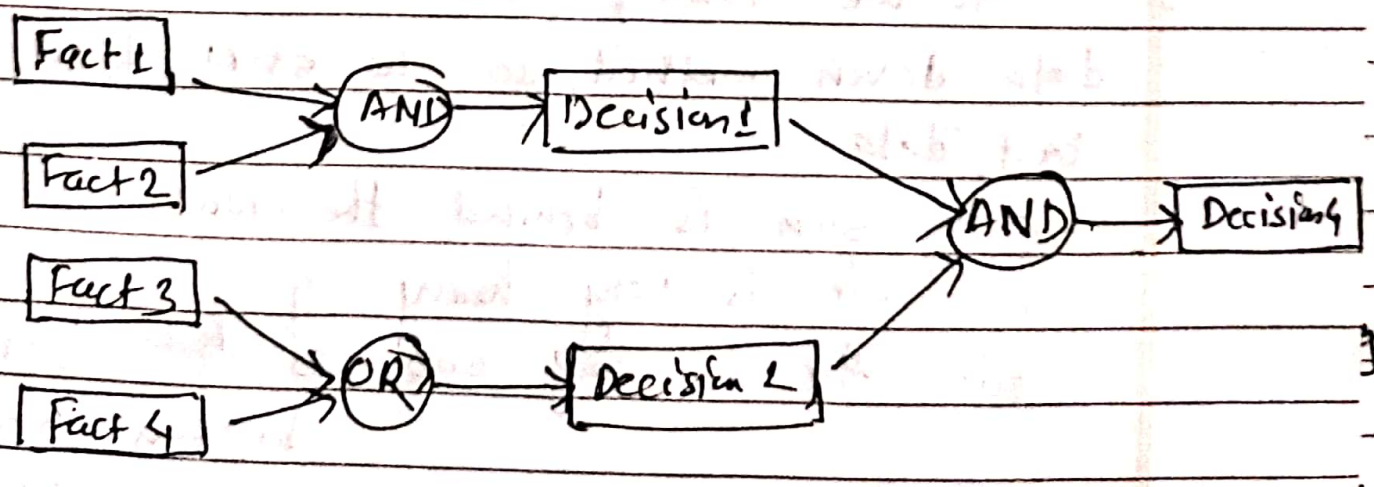
Forward chaining

This is the form of reasoning that starts with what is known and works towards the solution. It is known as bottom-up approach.

→ It starts with available data and uses the inference rules to extract more data until an optimal goal is reached.

→ An inference engine using forward chaining searches the inference rules until it finds one where the "If" clause is known to be true. It can ~~cont~~ conclude or infer "THEN" clause resulting in the addition of new inference to its data set.

→ also known as data driven reasoning



Scanned with CamScanner

Example:

Rule 1:

If we suspect temperature is less than 20°
AND there is humidity in the air
Then there is chance of rain.

Rule 2

If sun is behind the clouds
AND air is very cool
Then we suspect temperature is less than 20°

Rule 3

If air is very heavy
Then there is humidity in the air.

facts:

sun is behind the cloud
air is very heavy and cool

Using forward chaining try to conclude that there are chances of rain.

Sol:

As we know, forward chaining is a data driven method, so we start from given fact/data.

(I) sun is behind the cloud
(II) air is very heavy
(III) Air is very cool

} fact 2 broken into two parts.

Scanned with CamScanner

from (I) & (III) fact we get new fact
" we suspect temperature is less than 20.

Now, we have following facts present:
(IV) Air is very heavy
(IV) we suspect temperature is less than 20°

Fact (II) matches with rule 3. Thus our new fact will become
(V) There is humidity in the air

Now, fact (IV) & (V) match with rule 1. Thus our new fact will be

(VI) "There are chances of rain" which is our goal state.

Example 2

Using given KB, prove that "I am happy" can be inferred from the K.B using ~~backward~~ forward chaining.

1. I feel sleepy → I am happy
2. I am at home ∧ It is snowing → Heating on
3. I am at home ∧ Heating on → I feel sleepy.

Scanned with CamScanner

4. Today is holiday ∧ I feel sleepy → I am at home.

5. Today is holiday ∧ It is snowing → I am at home.

6. Today is holiday

7. It is snowing

Soln: from the given fact rule 5 matches. We get new fact

8. I am at & home

from fact 7. and 8 rule 2 matches. We get new fact

9. Heating on.

from fact 8 and 9 rule 3 matches. We get new fact

10. I feel sleepy

from fact 10. rule 1 matches. We get new fact.

11. I am happy

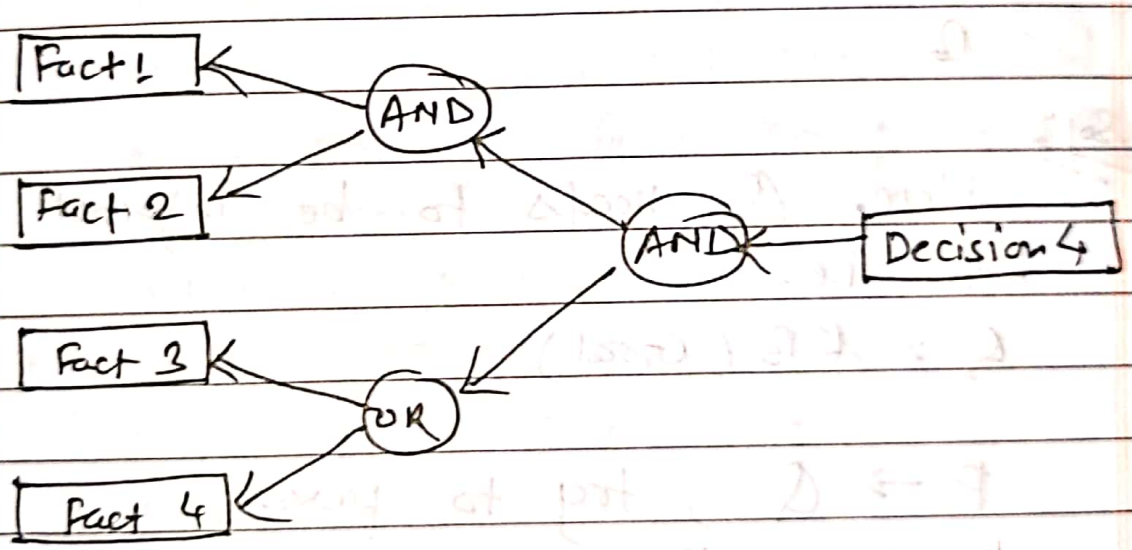
which is our goal state.

Scanned with CamScanner

Backward chaining

→ This is a form of reasoning that starts with the conclusion and works backward. The goal is ~~back~~ broken into many sub-goals which can be solved more easily.

→ It is also called top down approach



→ Backward chaining starts with a list of goals and works backward to see if there is data available that will support any of these goals. An inference engine using backward chaining would search the inference rule that matches a ~~desig~~ desired goal. If the "IF" clause of the inference rule is not known to be true then it is added to the list of goals.

Scanned with CamScanner

Example 1

using backward chaining prove that Q can be inferred from the given K.B.

$P \rightarrow Q$
$L \wedge M \rightarrow P$
$B \wedge L \rightarrow M$
$A \wedge P \rightarrow L$
$A \wedge B \rightarrow L$
A
B

Soln:

Here, Q needs to be inferred from K.B. i.e.

Q : KB (Goal)

$P \rightarrow Q$, try to prove P.
$L \wedge M \rightarrow P$, try to prove L and M
$B \wedge L \rightarrow M$, try to prove B
$A \wedge P \rightarrow L$, try to prove A
$A \wedge B \rightarrow L$

Here, A and B are known, so $A \wedge B$ are also known and it is true.

Now,

$A \wedge B \rightarrow L$, L is known
$B \wedge L \rightarrow M$, M is known

Scanned with CamScanner

$L \wedge M \rightarrow P$, P is known

$P \rightarrow Q$, Q is known //

Example 2 using given KB, prove that "I am happy" can be inferred from the KB using backward chaining.

1. I feel sleepy → I am happy
2. I am at home ∧ It is snowing → Heating on
3. I am at home ∧ Heating on → I feel sleepy
4. Today is holiday ∧ I feel sleepy → I am at home.
5. Today is holiday ∧ It is snowing → I am at home
6. Today is holiday
7. It is snowing.

let, I feel sleepy → p
I am happy → q
I am at home → r
It is snowing → s
Heating on → t
today is holiday → u

Now,
$p \rightarrow q$
$r \wedge s \rightarrow t$
$r \wedge t \rightarrow p$
$u \wedge p \rightarrow r$
$u \wedge s \rightarrow r$
u
s

Scanned with CamScanner

Now,

$p \to q$: try to prove p

$r \wedge t \to p$: try to prove r & t

$u \wedge s \to r$ try to prove u & s

$r \wedge s \to t$ try to prove r & s

Here, u and s are known, so,
$u \wedge s$ is also known, and is true.

$u \wedge s \to r$, r is known
$r \wedge s \to t$, t is known
$r \wedge t \to p$, p is known
$p \to q$, q is known //

Scanned with CamScanner

predicate logic / FOPL / FOPC

FOPL models the world in terms of objects.

objects : people, house, numbers, colors, baseball games, wars etc

~~Relations~~ :

properties : Color, price, age, gender, name etc

Relations : part-of, brother-of, has-color, occurs etc.

functions : subset of relation.
- father-of, bestfriend, beginning-of etc.

Knowledge representation in FOPL

The objects from real world are represented by constraint symbols.

- The properties of objects may be represented by predicates applied to these objects.

For example : Ram is student
Ram(student)

Scanned with CamScanner

The relationship between the objects are represented by the predicates with more arguments.

The ball's color is Red.

color (Ball, red)

predicate — arguments / object

Ram likes bike

likes (Ram, bike)

Beside constraints, the arguments of the predicate may be function or variable. It is also possible to have a function as an argument.

Robin's father is Rani's father

father_of (father_of(Robin), Rani)

Scanned with CamScanner

Quantifier

- A quantifier is a symbol that permits one to declare or identify the range and scope of variables in the logical expression.
- There are two types of quantifiers in predicate logic

universal quantifier

It states that the statement within its scope are true for every value of the specific variable. It is denoted by the symbol $\forall$ or A

Now,

$\forall x\, P_x$ is read as for every value of x, P_x is true

$\forall x$ is read as for all x.

All men are mortal.

$\Rightarrow \forall x : Man(x) \rightarrow Mortal(x)$

Existential quantifier

It states that the statements within its scope are true for some variable of the specific variable.

Scanned with CamScanner

It is denoted by the symbol $\exists$ or E.

- $\exists x$ is read as there exists x and

$\exists x\, p(x)$ is read as for some value of x $p(x)$ is true.

Some people are dishonest

$\Rightarrow \exists x : \text{person}(x) \rightarrow \text{dishonest}(x)$

Nested quantifier

If one quantifier that appears within the scope of another quantifier then it is called as nested quantifier.

Everybody loves somebody

$\Rightarrow \forall x \exists y : \text{loves}(x, y)$

There is someone who is loved by everyone

$\Rightarrow \exists y \forall x : \text{loves}(x, y)$

Scanned with CamScanner

Consecutive quantifiers of same type:

Brothers are siblings

$\Rightarrow \forall x \forall y : \text{Brother}(x, y) \rightarrow \text{Sibling}(x, y)$

- can also be written as one quantifier with several variables.

$\forall x, y \ \text{Sibling}(x, y) \rightarrow \text{Sibling}(y, x)$

Bound Variables

A variable in a formula is bounded if and only if its occurance is within the scope of the quantifier

$\forall x : A(x) \rightarrow B(x)$

where, x is bounded variable

Free Variable

A variable in a formula is free if and only if the occurance of it is outside the slope of quantifier.

$\forall x : A(x, y)$

Bounded variable

free variable

Scanned with CamScanner

Translate the following sentences into FOPL

1. All purple mushrooms are poisonous

$\forall x$: Mushroom(x) $\wedge$ color (x, purple) $\rightarrow$ poisonous (x)

2. Some students like every lecturers.

$\exists x \forall y$: student (x) $\wedge$ lecturer (y) $\rightarrow$ like(x,y)

3. No students like every lecturers.

$\neg \forall x \forall y$: student (x) $\wedge$ lecturer(y) $\rightarrow$ like (x,y)

4. ~~No~~ yellow frogs are ~~edible~~ not edible

$\forall x$: frog (x) $\wedge$ color (x, yellow) $\rightarrow$ $\neg$ edible (x)

5. Every gardner like sun.

$\forall x$: gardner (x) $\rightarrow$ like (x, sun)

6. Everyone is younger than his father.

$\forall x$: person (x) $\rightarrow$ youngerthan (x, father)

Scanned with CamScanner

7. Everyone likes someone

$\forall x \exists y$: likes (x, y)

8. John is not tall

$\neg$ tall (john)

9. All students are men

$\forall x$: student $(x) \rightarrow$ man (x)

10. Cow is mammal

mammal (cow)

11. Some of the females are mothers.

$\exists x$: female $(x) \longrightarrow$ mother (x)

12. Some people like cricket

$\exists x$: person $(x) \longrightarrow$ likes $(x,$ cricket$)$

13. Nobody likes ~~taxes~~ taxes.

$\neg \exists x \rightarrow$ likes $(x,$ taxes$)$

Scanned with CamScanner

Inferences in FOPL

1. Universal Instantiation

It is also called universal elimination and it is valid rule of inference.
To do so, substitute the ground term for the variables.
Remove universally quantified sentences after universal instantiation.

Rule: $\forall x : P(x)$

$\therefore P(c)$

where, c is some arbitrary element of universe.

2. Existential Instantiation

It is also called existential elimination and it is called valid rule of the inference
To do so substitute the ground term (term without variable).
- Remove existentially quantified sentences after existential instantiation.

Rule: $\exists x : P(x)$ where, c is arbitrary element of the universe
$\therefore P(c)$

Scanned with CamScanner

Unification

- The goal of the unification is to make two expression look like identical by using substitution
- It means the meaning of the sentence should not be changed, but it should be expressed in multiple ways.

- The UNIFY algorithm in unification takes two sentences as input and then returns a unifier if one exists.

- substitution means replacing one variable with another term.
- It takes two literal as input and makes them identical using substitution.
- It returns fail if the expression do not match with each other.

UNIFY (P, q) = θ where SUBST (θ, P) = SUBST (θ, q)

Example 1

- lets say there are two different expressions, p(x, y) and p(a, f(z))
- We need to make both above statements identical to each other.

Scanned with CamScanner

perform the substitution.

$p(x, y)$ — (I)

$p(a, f(z))$ — (II)

Substitute x with a, and y with f(z) in the first expression, and it will be represented as

a/x and $f(z)/y$

- With both the substitution, the first expression will be identical to the second expression and the substitution set will be:

$[a/x, f(z)/y]$

Example 2

Given knows(Ram, X) is a predicate.
Whom does Ram know?

- The unify algorithm will search all the related sentences in the KB, which could unify with knows(Ram, X)

unify(knows(Ram, x), knows(Ram, shyam)) = {x/shyam}

unify(knows(Ram, x), knows(y, akash)) = {x/akash, y/Ram}

unify(knows(Ram, x), knows(z, Raman)) = fails.
Unifier is empty.

- The last one fails because we have used the same variable name x.

Scanned with CamScanner

Resolution of FOPL

Resolution is a procedure which gains its efficiency from the fact that it operates on the statements that have been converted to very convenient standard form.
Resolution produces proofs by refutation i.e. in order to prove a statement or so that it is valid, resolution attempts to show that the negation of a statement produces a contradiction with the known statements.

Resolution is used to find the truth or falsity of the logical sentences.

Algorithm for resolution

let 'f' be the given set of sentences and 'p' be the sentence to be proved then the following steps should be performed.

1. Convert the sentences of 'f' into clausal form (CF)

2. Negate p and convert the result into clausal form. Add it to the set of clauses obtained in step 1.

Scanned with CamScanner

3. Repeat until a contradiction is obtained or no further progress can be made.

a. Select two clauses and call them parent clauses and one of them should be ¬p.

b. Resolve them together and the resulting clauses (resolvant) will be the disjunction of the literals of both the parents clauses with the following exception

- If there are any pair of literals 'L' and '¬L' such that one of the parent clauses contain 'L' and other contains '¬L', then select such pair and eliminate both 'L' & '¬L' from resolvant.

- If the resolvant is empty clause then the contradiction is found.
- If it is not then add it to the set of clauses available to the procedure.

Scanned with CamScanner

Algorithm for clausal form

1. Renames all the variables so that all the quantifiers bind to distinct value.

2. Eliminate ($\rightarrow$) operator

$$a \rightarrow b = \neg a \vee b$$

3. Reduce '$\neg$' scope by using

$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$

4. Eliminate '$\exists$' quantifier, use skolem constant
5. Eliminate '$\forall$' quantifier
6. Use distributive law and distribute '$\vee$' as:

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

Example:

Write the following sentences into predicate form

(i) All the animals that can fly are birds.

(ii) Pigeon is an animal and it can fly.

Convert them into clausal form and use resolution procedure to prove that pigeon is a bird.

Scanned with CamScanner

Solⁿ: To prove

pigeon is a bird

$\Rightarrow$ bird (pigeon)

pigeon is not a bird

$\neg$ bird (pigeon)

Step 1: Converting into FOPL

1. All the animals that can fly are birds

$\forall x$: animal $(x) \wedge$ canfly $(x) \rightarrow$ bird (x)

2. Pigeon is an animal and it can fly

animal (pigeon) $\wedge$ canfly (pigeon)

Step 2: Converting into clausal form.

1. $\forall x$: animal $(x) \wedge$ canfly $(x) \rightarrow$ bird (x)

~~$\neg$ (animal $(x) \wedge$ canfly (x)) $\vee$ bird (x)~~

$\neg$ animal $(x_1) \vee \neg$ canfly $(x_1) \vee$ bird (x_1)

2. animal (pigeon) $\wedge$ canfly (pigeon)

Scanned with CamScanner

Step 2: Using Resolution Algorithm

$\neg$bird (pigeon) $\qquad$ $\neg$animal(x_1) $\vee$ $\neg$canfly(x_1) $\vee$ bird(x_1)

$\neg$animal(pigeon) $\vee$ $\neg$canfly (pigeon) $\qquad$ animal (pigeon) $\wedge$ canfly (pigeon)

$\neg$(animal (pigeon $\wedge$ canfly (pigeon)) $\qquad$ animal(pigeon) $\wedge$ canfly (pigeon)

ϕ Null clause.

Conclusion: Since we have a null statement which shows a contradiction meaning i.e. $\neg$bird (pigeon) is false and which is equivalent to bird (pigeon)

Hence "pigeon is a bird" is proved //

* Do more examples from book.

Scanned with CamScanner

Handling uncertain knowledge

Symbolic versus statistical reasoning

Symbolic methods basically represent uncertainty beliefs as being

True
False or
Neither true nor False

Some methods also had problem with
- incomplete knowledge
- ~~Contrad~~ contradiction in the knowledge.

Statistical methods provide a method for representing beliefs that are not certain (uncertai ~~beg~~ but for which there may be some supporting (or contradictory) evidence.

Statistical method offer advantages in two broad scenarios

Genuine Randomness → We may not be able to predict any outcomes with certainty but we have knowledge about the likelihood of certain items and we can exploit this
- ~~play~~ card games.

Scanned with CamScanner

Exceptions → symbolic methods can represent this. However if the no. of exceptions is large such system tend to break down.
- common sense, expert reasoning task etc.
- statistical technique can summarise large ~~exp~~ exceptions without resorting enumeration.

Basic statistical Methods — Probability

The basic approach statistical methods ~~adt~~ adopt to deal with uncertainty is via the axioms of probability.

→ probability are (real) numbers in the range 0 to 1.

→ A probability of $P(A) = 0$ indicates total uncertainty in A, $P(A) = 1$, total certainty and values in between some degree of (un) certainty.

probabilities can be calculated in a no. of ways

① prior probability

probability = (no. of desired outcome) / total no. of outcomes.

Scanned with CamScanner

(ii) Conditional / posterior probability →

Conditional probability is the probability of one event occuring with some relationship to one or more other events.

$$P(B|A) = P(A \text{ and } B) / P(A)$$

which can be written as,

$$P(B/A) = P(A \cap B) / P(A)$$

<u>Example</u>: In a group of 100 sports car buyers, 40 bought alarm systems, 30 purchased bucket seats, and 20 purchased an alarm system and bucket seats. If a car buyer chosen at random bought an alarm system, what is the probability they also bought bucket seats?

probability of alarm system $P(A) = 0.4$

" " bucket seats $P(B) = 0.3$

" " Alarm $\cap$ bucket $P(A \cap B) = 0.2$

$$\therefore P(B|A) = P(A \cap B) / P(A)$$

$$= \frac{0.2}{0.4} = 0.5$$

The probability that a buyer bought bucket seats given that they purchase an alarm system is 50%

Scanned with CamScanner

Bayes Theorem

Bayes' theorem is a way to apply conditional probability for prediction.

Mathematically,

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

prior

posterior probility

likelihood

marginal

Example:

Find out a patient's probability of having liver disease if they are an alcoholic. "Being an alcoholic" is tested for liver disease with following

1. Let A could mean the event "patient has liver disease". past data tells you that 10% of patients entering your clinic have liver disease $P(A) = 0.10$

2. B could mean the litmus test that "patient is an alcoholic". Five percent of the clinic's patient are alcoholic.

$$P(B) = 0.05$$

3. Among those patients diagnosed with liver disease, 7% are alcoholic. The $P(B|A)$:

Scanned with CamScanner

the probability that a patient is alcoholic, given that they have liver disease, is 7%.

Now, using Baye's theorem,

$$P(A|B) = \frac{P(B/A)\,P(A)}{P(B)}$$

$$= \frac{0.07 \times 0.10}{0.05}$$

$$= 0.14$$

In other words, if the patient is an alcoholic, their chances of having liver disease is 0.14 (14%).

probability distribution

The probability distribution is the distribution table formed to obtain the occurrence of each event in the sample space.

- From probability distribution table we can find the other statics such as mean, standard deviation, and expected value.

Example: If a coin is tossed, find the probability distribution of getting tails

Scanned with CamScanner

Solⁿ:

If a coin is tossed, either 0 or 1 tail can be obtained.

probability of getting zero tail, $P(0) = 1/2$

" " " one " $P(1) = 1/2$

The probability distribution table

X	0	1
PX	1/2	1/2

Example 2. If two coins are tossed simultaneously, find the probability distribution of getting head.

Solⁿ:

If two coins are being tossed, there can be either 0 heads, 1 head, or 2 head.

Sample space, $S = \{HH, HT, TH, TT\}$

The probability distribution of getting heads

X	0	1	2
P(X)	1/4	1/4	1/4

Scanned with CamScanner

Joint probability Distribution

Joint probability is a statistical measure that calculates the likelihood of two events occuring together and at the same point in time.

- Joint probability is the probability of event Y occuring at the same time that event X occurs.

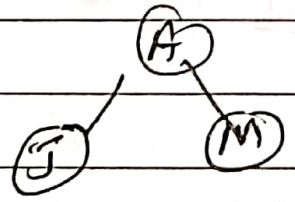
For example if we have two random variables: Weather and cavity with set of domain for:
Weather : { Rainy, sunny, cloudy, snow }
and
Cavity : {True, False} then
p(Weather, Cavity) = a 4x2 matrix of values.

Weather	Sunny	Rainy	Cloudy	Snow
Cavity = True	0.144	0.02	0.016	0.02
Cavity = False	0.576	0.08	0.064	0.08

formula

$$P(x_1, x_2, x_3 \cdots x_n) = \prod_{i=0}^{n} p(x_i / \text{parent}(x_i))$$

eg.



$$P(A, J, M) = P(A) * P(J/A) * P(M/A)$$
or
$P(A \cap J \cap M)$

Bayesian Network

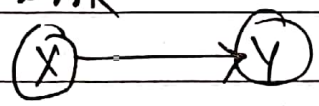
Bayesian networks are a type of probabilistic graphical model that uses Bayesian inference for probability computation.

- It is also called belief network or probabilistic network or casual network or knowledge map.
- directed acyclic graph

Bayesian Network Consists:

- Nodes : set of random variable

Directed link



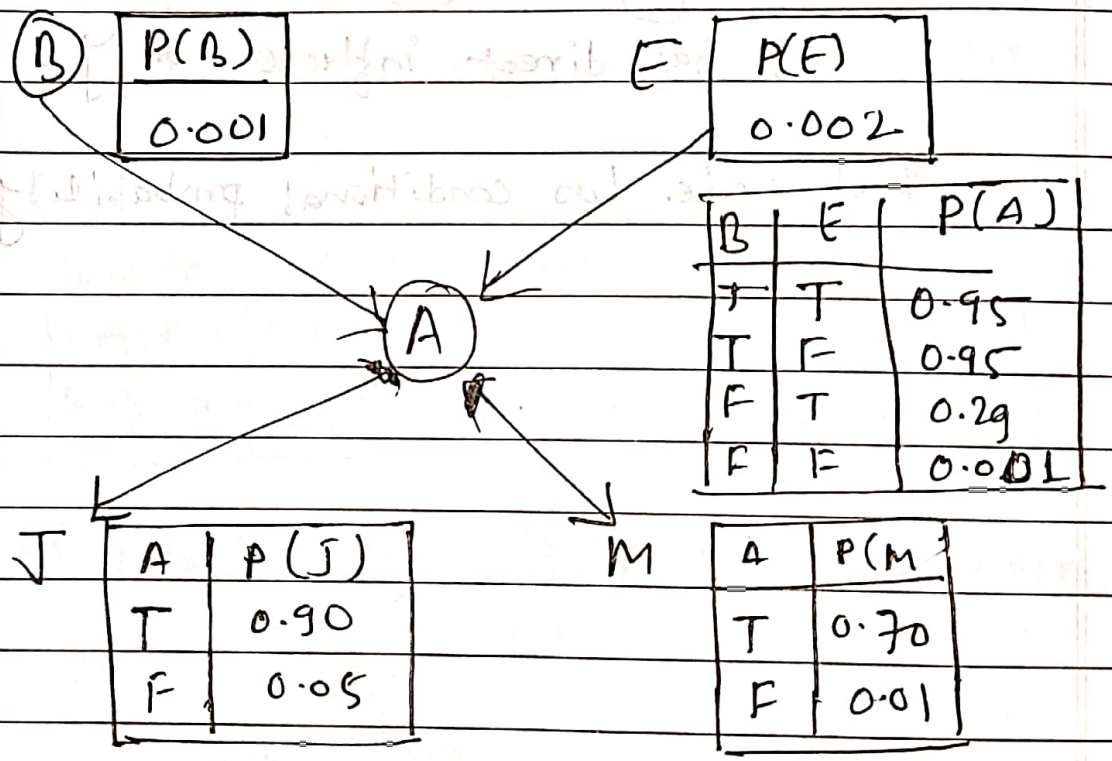
X has direct influence on Y.

- Each node has conditional probability table

Scanned with CamScanner

Example: You have a burglar alarm installed in your home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. You also have two neighbours, John and Mary, who have promised to call you at work when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the otherhand, likes rather loud music and sometimes misses the alarm ~~tog~~ together.

We would like to have estimate the probability of a burglary with given evidence who has or has not call.



Q. What is the probability that the alarm has sounded, but neither burglary nor earthquake has occured, and both John and Mary call.

$P(J \wedge M \wedge A \wedge \neg B \wedge \neg E) = ?$

$= P(J/A) * P(M/A) * P(A/\neg B, \neg E) * P(\neg B) * P(\neg E)$

$= 0.90 * 0.70 * 0.001 * 0.999 * 0.998$

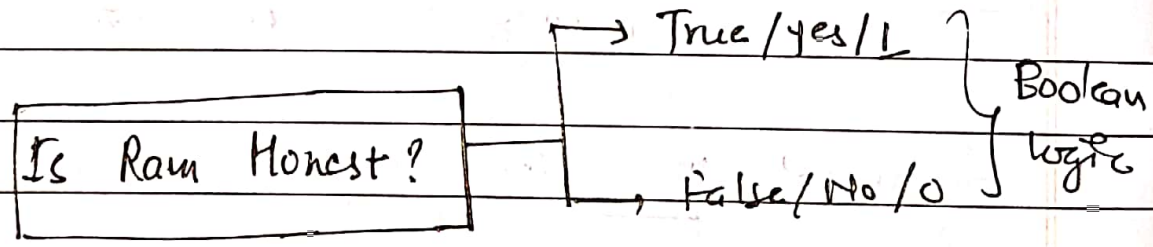
$= 0.00062$ //

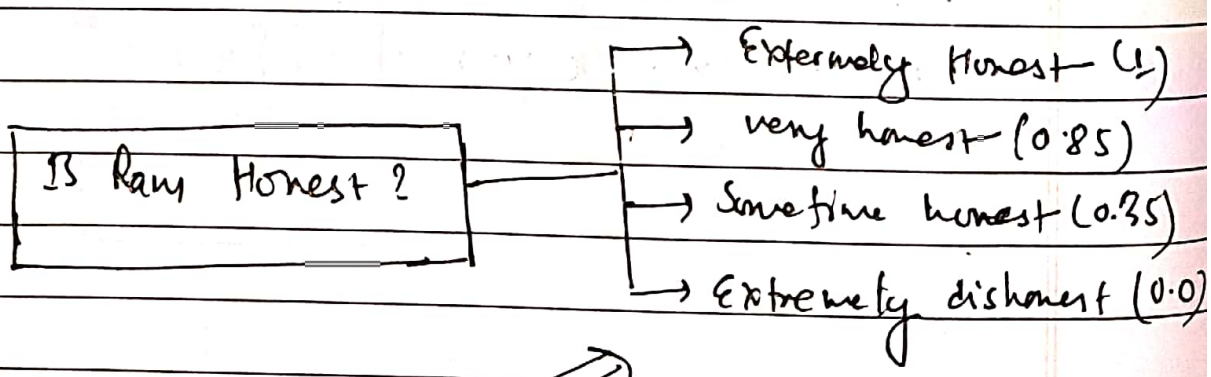
Scanned with CamScanner

Fuzzy Logic

- Fuzzy logic was introduced in 1965 by Lotfi A. Zadeh.
- Fuzzy logic is an approach to variable processing that allows for multiple values to be processed through the same variable.
- Fuzzy logic is designed to solve problems by considering all available information and making the best possible decision given the input
- The values are indicated by a number in the range from 0 to 1.
- 1 represents absolute truth
- 0 " " falseness

For example:





Fuzzy logic ⇒

Scanned with CamScanner

Set → A set is an unordered collection of different elements.

Operations on classical sets

Union

$$A \cup B = \{x \mid x \in A \text{ OR } x \in B\}$$

→ Set of elements which are in A, in B, or in both A and B.

Intersection

→ Set of elements which are in both A and B.

$$A \cap B = \{x \mid x \in A \text{ AND } x \in B\}$$

Difference / Relative complement

– set of elements which are only in A but not in B.

$$A - B = \{x \mid x \in A \text{ AND } x \notin B\}$$

Complement of a set

– set of elements which are not in set A.

$$\bar{A} = \{x \mid x \notin A\}$$

Scanned with CamScanner

Fuzzy Set

- Fuzzy sets can be considered as an extension and gross oversimplification of classicals sets.
- It can be understood in the ~~set~~ ~~of~~ content of set membership.
- It allows partial membership which means that it contains elements that have varying degree of membership in the set.

Example: If cold is a fuzzy set, exact temperature values might be mapped to the fuzzy set as follows:

$15^\circ \rightarrow 0.2$ (slightly cold)
$10^\circ \rightarrow 0.5$ (quite cold
$0^\circ \rightarrow 1$ (totally cold

Set of terms:
{ Slightly cold, quite cold, totally cold }

Formal Fuzzy logic

→ Fuzzy logic can be seen as an extension of ordinary (bivalent) logic, where main difference is that we use fuzzy sets for the membership of a variable.

- We can have fuzzy propositional logic & fuzzy predicate logic.

Scanned with CamScanner

Fuzzy set operation

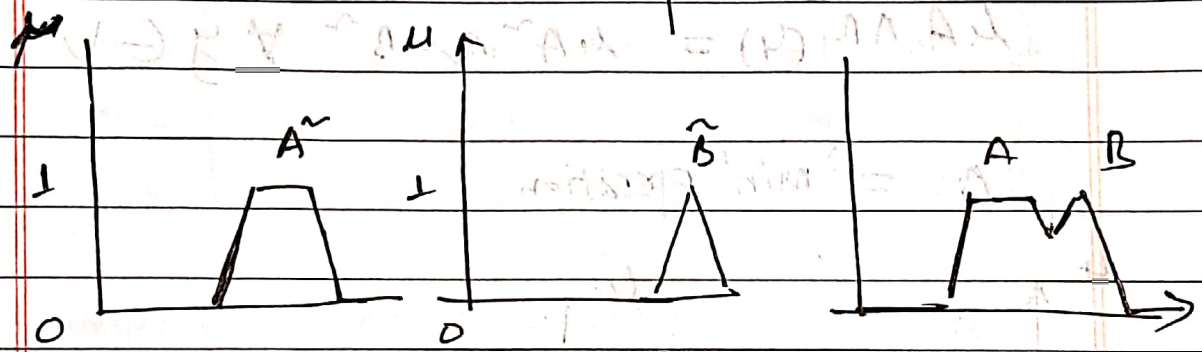
Having two fuzzy sets $\tilde{A}$ and $\tilde{B}$, the universe of information U and an element y of the universe, the following relation express the union, intersection and complement operation on fuzzy sets.

UNION / FUZZY 'OR'

$\mu_{\tilde{A} \cup \tilde{B}} = \mu_{\tilde{A}} \vee \mu_{\tilde{B}}$

$\vee \Rightarrow$ max operation

μ_A = membership fn of A



Fuzzy set $\tilde{A}$ | Fuzzy set $\tilde{B}$ | Union of two fuzzy sets.

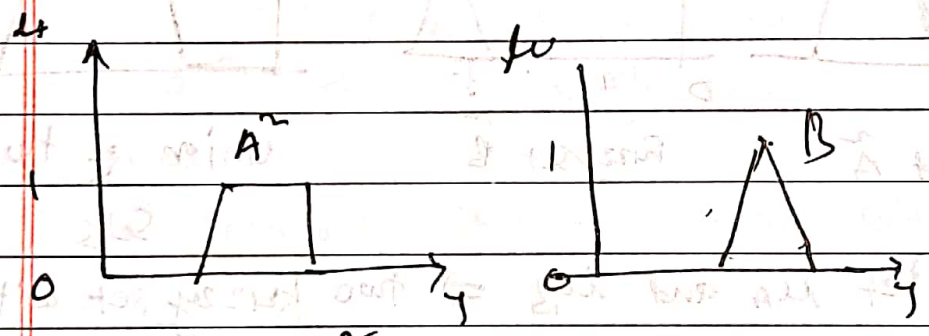
Example: let μ_A and μ_B $\Rightarrow$ two fuzzy set with given members, then the resulting union of these two set is $\mu_{A \cup B} = MAX(A, B)$

μ_A	μ_B	$\mu_{A \cup B} = MAX(A, B)$
0	0	0
0	1	1
1	0	1
1	1	1
0.2	0.5	0.5
0.7	0.2	0.7
0.6	0.6	0.6

Intersection / Fuzzy 'AND'

$$\mu_{\tilde{A} \cap \tilde{B}}(y) = \mu_{\tilde{A}} \wedge \mu_{\tilde{B}} \quad \forall y \in V$$

$\wedge$ = 'min' operation



Fuzzy set $\tilde{A}$ Fuzzy set $\tilde{B}$

Intersection of two fuzzy sets

Scanned with CamScanner

Example:

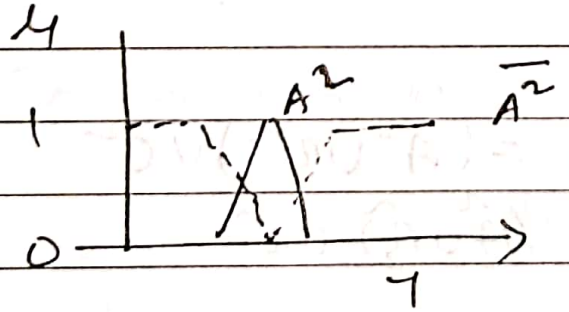
Let μ_A and μ_B represents the two fuzzy set with given members, then resulting union of these two sets is

$$\mu_{A \cap B} = \max(A, B)$$

μ_A	μ_B	$\mu_{A \cap B} = \min(A \cap B)$
0	0	0
0	1	0
1	0	0
1	1	1
0.2	0.5	0.2
0.7	0.2	0.2
0.6	0.6	0.6

Complement / Fuzzy NOT

$$\mu A^2 = 1 - \mu A^2(y) \quad y \in U$$



Complement of fuzzy set.

Scanned with CamScanner

Example: let $\mu_A = \{0, 1, 1, 0.2, 0.7, 0.6\}$ then its complement $1 - \mu_A$ is shown in table below.

μ_A	$1 - \mu_A$
0	1
0	1
1	0
1	0
0.2	0.8
0.7	0.3
0.6	0.4

Properties of Fuzzy sets

Commutative property

$$\tilde{A} \cup \tilde{B} = \tilde{B} \cup \tilde{A}$$
$$\tilde{A} \cap \tilde{B} = \tilde{B} \cap \tilde{A}$$

Associative property

$$\tilde{A} \cup (\tilde{B} \cup \tilde{C}) = (\tilde{A} \cup \tilde{B}) \cup \tilde{C}$$
$$\tilde{A} \cap (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cup \tilde{B}) \cup \tilde{C}$$

Scanned with CamScanner

Distributive property

$$\tilde{A} \cup (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cup \tilde{B}) \cap (\tilde{A} \cup \tilde{C})$$
$$\tilde{A} \cap (\tilde{B} \cup \tilde{C}) = (\tilde{A} \cap \tilde{B}) \cup (\tilde{A} \cap \tilde{C})$$

Idempotency property

$$\tilde{A} \cup \tilde{A} = \tilde{A}$$

$$\tilde{A} \cap \tilde{A} = \tilde{A}$$

Identity property

For fuzzy set $\tilde{A}$ and universal set U, this property states

$$\tilde{A} \cup \phi = \tilde{A}$$
$$\tilde{A} \cap U = \tilde{A}$$
$$\tilde{A} \cap \phi = \phi$$
$$\tilde{A} \cup U = U$$

Transitive property

If $\tilde{A} \subseteq \tilde{B} \subseteq \tilde{C}$, then $\tilde{A} \subseteq \tilde{C}$

[A is proper subset of B]

Scanned with CamScanner

De Morgan's Law

$$\overline{(\tilde{A} \cap \tilde{B})} = \overline{\tilde{A}} \cup \overline{\tilde{B}}$$

$$\overline{(\tilde{A} \cup \tilde{B})} = \overline{\tilde{A}} \cap \overline{\tilde{B}}$$

Fuzzy Logic Membership function

- Membership functions characterize fuzziness, whether the elements in fuzzy sets are discrete or continuous.
- They are represented by graphical forms.

Mathematical Notation

Fuzzy set $\tilde{A}$ in the universe of information U can be defined as a set of ordered pairs and it can be represented mathematically as

$$\tilde{A} = \{ (y, \mu_{\tilde{A}}(y)) \mid y \in U \}$$

Here,

$\mu_{\tilde{A}}(\cdot)$ = membership function of $\tilde{A}$.

This ~~value~~ assumes values in the range from 0 to 1 i.e:

$$\mu_{\tilde{A}}(\cdot) \in [0;1]$$

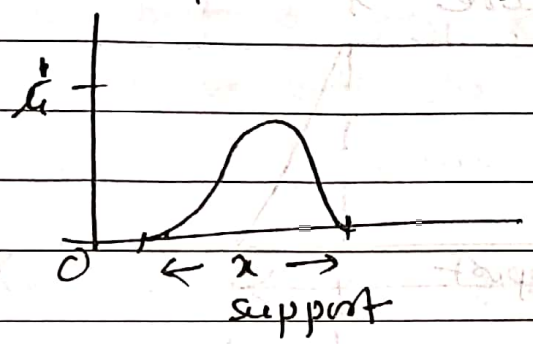
The membership fn $\mu_{\tilde{A}}(\cdot)$ maps U to the membership space M.

Scanned with CamScanner

The dot (•) in the membership f^n described above, represents the elements in a fuzzy set, whether it is discrete or continuous.

Features of Membership functions.

1. Support : set of all points such that $\{\mu_A(x) > 0\}$

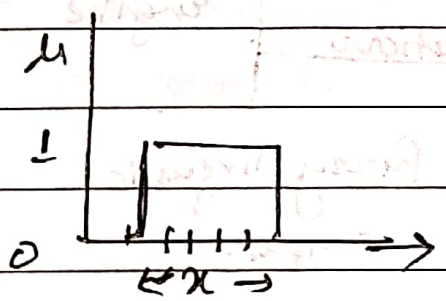


$$\text{Support}(A) = \{x \mid \mu_A(x) > 0\}$$

2. Core : region of universe that is characterize by full membership in the set

$$\mu_A(x) = 1$$

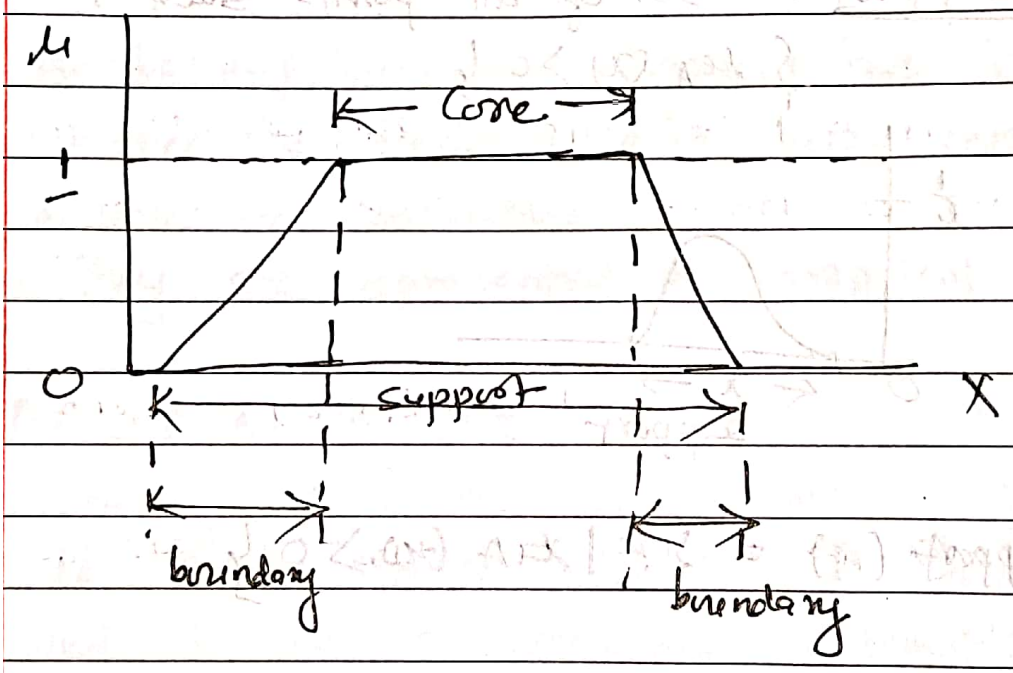
$$\text{Core}(A) = \{x \mid \mu_A(x) = 1\}$$



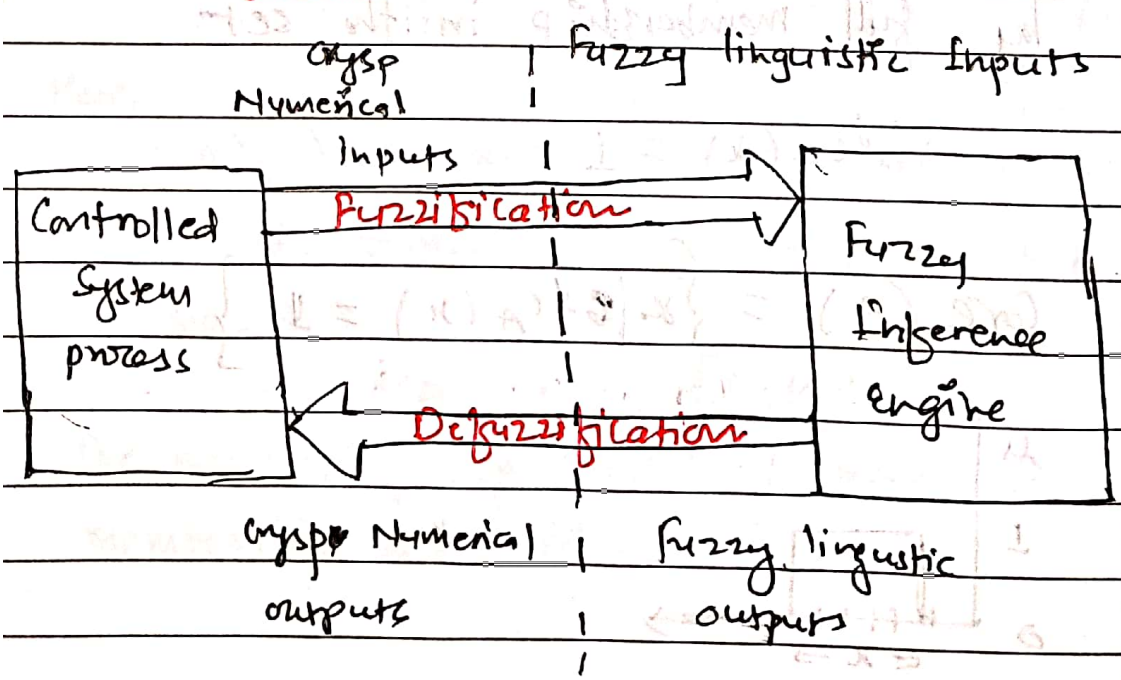
Scanned with CamScanner

3. **Boundary**: region of universe that is characterize by a nonezero but incomplete membership in the set.

$$(1 > \mu_A(x) > 0)$$



Fuzzy Inference



Scanned with CamScanner

Fuzzification

The process of translating the measured numerical values into fuzzy linguistic values is called fuzzification.

- Methods of fuzzification

① Support fuzzification method (S-fuzzification)

In this method, the fuzzified set can be expressed with the help of following relation

$$A^2 = \mu_1 Q(x_1) + \mu_2 Q(x_2) + \dots + \mu_n Q(x_n)$$

Here,

fuzz set $Q(x_i)$ = Kernel of fuzzification.

- This method is implemented by keeping μ_i constant and x_i being transformed to a fuzzy set $Q(x_i)$

② Grade fuzzification

→ similar to above.

→ x_i kept constant and μ_i is expressed as a fuzzy set.

Scanned with CamScanner

Defuzzification

Defuzzification is the process of producing a quantifiable result in fuzzy logic.
Defuzzification interprets the membership degrees in the fuzzy sets into a specific action or real value.

→ It converts the fuzzy results to crisp results.

Methods of Defuzzification

Max-Membership Method

$$\mu_{\tilde{A}}(x^*) > \mu_{\tilde{A}}(x) \text{ for all } x \in X$$

Here x^* is the defuzzified output.

Centroid Method

$$x^* = \frac{\int \mu_{\tilde{A}}(x) \cdot x \, dx}{\int \mu_{\tilde{A}}(x) \cdot dx}$$

Scanned with CamScanner

Weighted Average method

- Each membership function is weighted by its max^m membership value,

$$x^* = \frac{\sum \mu_A(\bar{x}_i) \cdot \bar{x}_i}{\sum \mu_A(\bar{x}_i)}$$

Mean-Max Membership

- Also called Middle of the maxima

$$x^* = \sum_i = 1 - n \, \bar{x}_i n$$

Advantages of Fuzzy logic system

→ This system can work with any type of input - imprecise, distorted, noisy input

→ construction of Fuzzy logic system is easy and understandable

→ provides efficient solution to complex problems in reasoning & decision making

→ Algorithm can be described with little data, so little memory is required.

Scanned with CamScanner

Disadvantages of fuzzy logic systems

→ No systematic approach to solve a given problem

→ proof of its characteristics is difficult or impossible in most cases because every time we donot get mathematical description of approach

→ accuracy is compromised due to capability to work with precise as well as imprecise data.

Application

- used in the aerospace field for altitude control of spacecraft and satellite.
- used in automotive system for speed control, traffic control
- used in chemical company for controlling the pH, drying, chemical distillation process.
- Used in Natural language processing
- used in expert systems
- used in Neural Networks.

Scanned with CamScanner