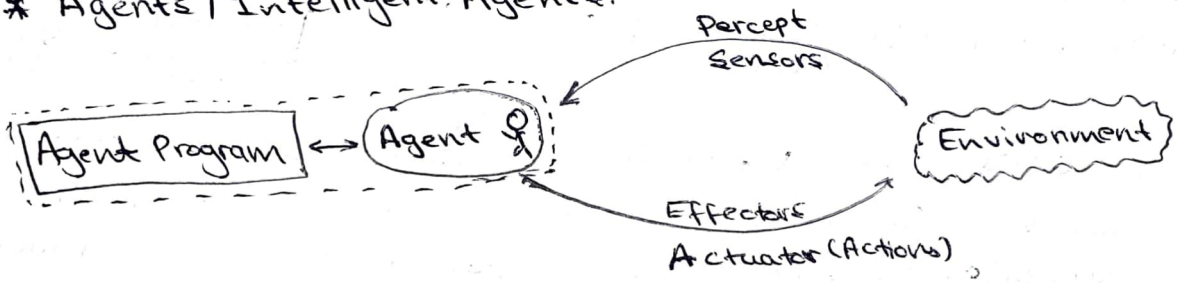
CS 401 : Artifical Intelligence:

Syllabus:

- Approach to AI : Heuristic Search (A*, AO*, Best First) DFS, BFS, Hill Climbing), Game Playing
- Knowledge Representation : Approaches, Predicate Logic, Reasoning
- Planning : Overview, Hierarchial, Goal Stack
- NLP : Syntactic, Semantic
- Multiagent System : Types, Properties
- Fuzzy Sets : Crisp, Fuzzy Set, α-Cut, Operations.
- ANN and Genetic Algo : Single, Multilayer, Feed forward, Recurrent, ML

* Agents | Intelligent Agents.



Agent → Percept → Decision → Actions $\xrightarrow{\text{change}}$ Environment

"Goals of Agent" : High Performance
Optimised Result
Rational Action

P → Performance
E → Environment
A → Actions
S → Sensors.

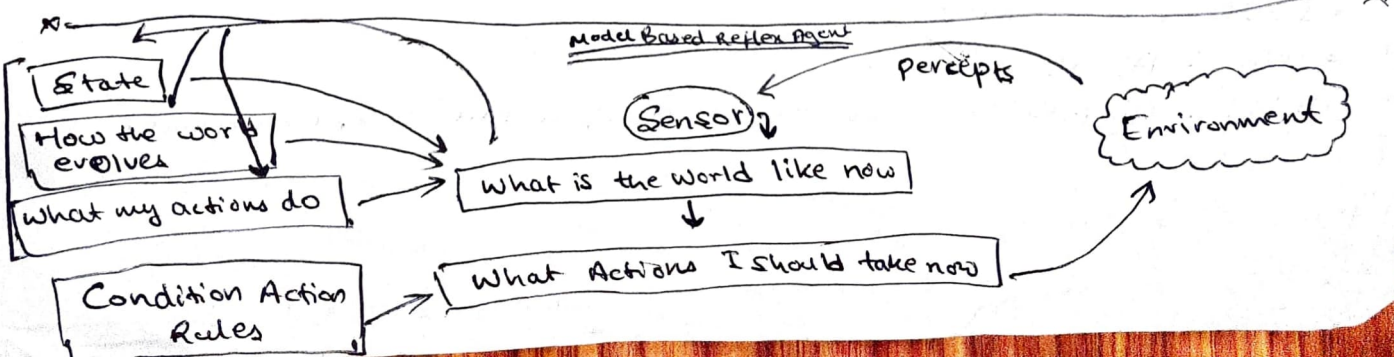
four imp. factors to model an agent

Types of Agents:

- Simple Reflex - Model Based - Goal Based - Utility Based - Learning

- Simple Reflex Agents:
  - → Act only on the basis of current perception
  - → Ignore the rest of percept history
  - → Based on if-else Rules.
  - → Environment : Fully observable.

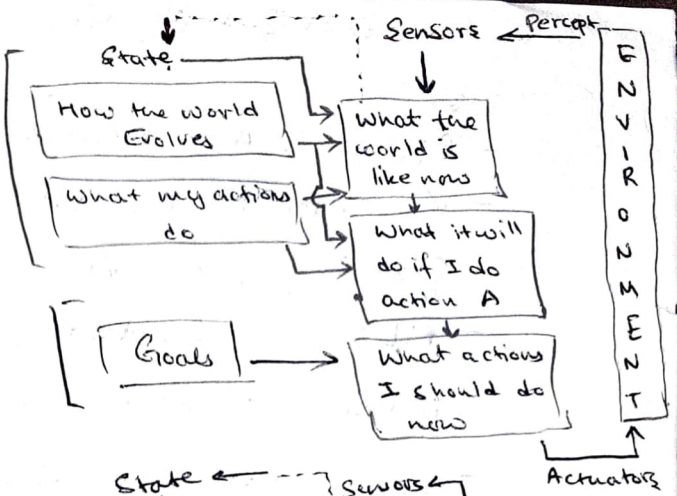
- Model-Based Reflex Agents:
  - → Environment : Partially Observable
  - → Store Percept History (Internal Model)



- Goal Based Agents:

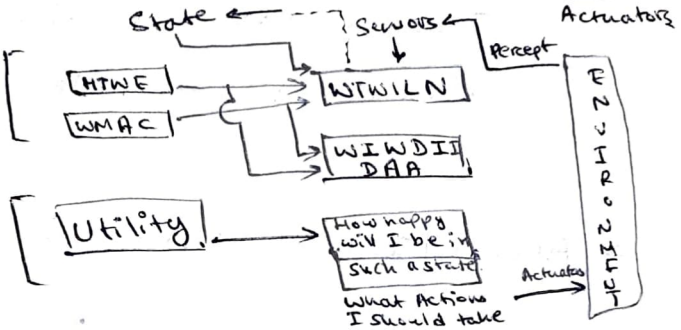
→ Expansion of Model Based Reflex Agent.
→ Desirable Situation (Goal)
→ Searching and Planning.

- Theoritical Model.



- Utility Based Agents:

→ Focus on utility NOT goal.
→ Utility function.
→ Deals with Happy and Unhappy State.
→ Expansion of Model Based Reflex Agent.



Goal vs Utility: Example: Google Map: Reaching final destination is the goal, however, GPS does not know unexpected accident / traffic scenario. In such cases, new path is determined or should it continue on the same path even if it may delay the travel? Goal based will set new path; but utility based will see if the agent is happy or not; if not happy it will set new path, if happy it will continues on the same set path.

* Artificial Intelligence:

"Can Machines Think" :- Alan Turing, 1950

- Reasoning
  - Problem Solving
  - Perception.
- Learning

- 1956: Birth of AI (Darthmouth meeting: 'Artificial Intelligence' name adopted)
- 1950: Turing's 'Computing Machinery and Intelligence'.
- 1964: Computers can understand NLP enough to solve algebra word problems.
- 1986: Rise of ML.
- 1995: AI as Science.
- 1997: Deep Blue chess program beats world chess champion.

* State Space Search : (falls under Problem Solving factor) : - Precise
- Analyse.

$S: \{ \mathcal{S}, A, Action(\mathcal{S}), Result(\mathcal{S}, a), cost(\mathcal{S}, a) \}$

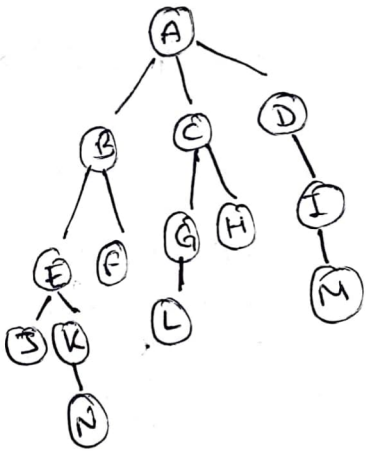
where, $\mathcal{S}$ is start / Goal state.
A is set of all possible actions.

→ Uninformed (Blind) Search. (Brute-force) :

→ Informed Search : Heuristic :

Uninformed Searching	vs,	Informed Search
→ Search without Information		→ Search with information
→ No knowledge		→ Use knowledge to find steps to solution.
→ Time Consuming		→ Quick Solution.
→ More Complexity (Time, Space)		→ Less Complexity (Time, space).
→ Blind, Brute-force		→ Heuristic
E.g.: DFS, BFS etc.		E.g.: A*, Heuristic DFS, Best first search.
→ Guaranteed Optimal Solution		→ ~~Might~~ may be optimal, may not be optimal.

• BFS (Breadth first Search) : → Uninformed, FIFO (Queue), Shallowest Node, → Complete, → Optimal, → Time Complexity $O(V+E)$ $O(b^d)$

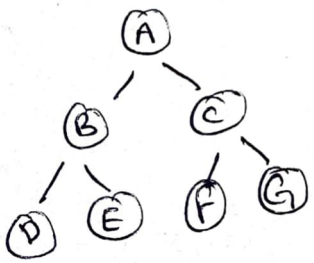


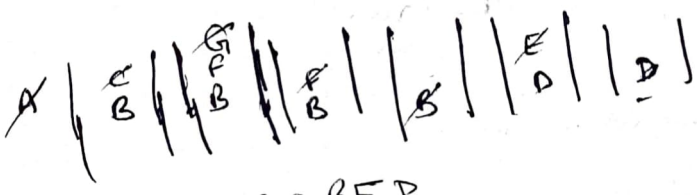
A
→ BCD
→ CDEF
→ DEFGH
→ EFGHI
→ FGHIJK
→ GHIJK
→ HIJKL
→ IJKL
→ JKLM
→ KLM
→ LMN
→ MN
→ N

where, b is branch factor
d is depth.

V → Nodes No.
E → Edges No.

• DFS (Depth first Search) : → Uninformed, Stack (LIFO), Deepest Node, Incomplete, → Non-optimal → Time Complexity $O(V+E)$ $O(b^d)$





A C G F B E D

b → branching factor
d → depth
V → Num. of Nodes
E → Num. of edges.

* Bidirectional Search:
  - can be both BFS and DFS; better to use BFS.
- Uniformed - Complete in BFS - Incomplete in DFS.
- Time Complexity $2(b^{d/2})$

* Generate and Test (DFS with backtracking)
- Heuristic Technique - Continuous Process - Complete
- Non Redundant - Informed.

* Best First Search
- Informed - Heuristic - Priority queue (sort every time ASC)
- Greedy Method. - Takes only straight line distance
- Good Solution (Not optimal) (May or may not) - Does not consider distance between two travelled nodes; i.e, $f(n) = h(n)$
- Time complexity: $O(b^m)$ or, $O(b^d)$ - Complete.

* Beam First Search
- Takes care of space complexity (constant)
- Priority queue (sorts ASC and stores only β best nodes)
- Like Best First Search, takes only straight line distance; $f(n) = h(n)$
- Time complexity, $O(b^d)$
- Incomplete
- Good solution, but may/may not provide optimal soln)

β is beam width i.e, β can be 1, 2, 3, ... n.

* Hill Climbing Algorithm:
- It is beam first search but $\beta = 1$.
- Greedy Approach. - No backtrack.
- Local Search Algorithm.

- Problems
  - Local Maxima
  - Plateau/Flat Maximum
  - Ridge.

* A* Algorithm:
- Informed. - Time Complexity: $O(b^d)$
- Priority queue? - Global consideration. Sort every time. like Best first search.
- Admissible. (optimal soln when underestimation)

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

→ $f(n) = g(n) + h(n)$

$g(n)$: Actual cost from Start Node to n

$h(n)$: Estimation cost from n to Goal Node; i.e, Straight line distance.

* AO* (And/Or) Algorithm:
- Problem decomposition (breakdown into smaller pieces.
- Does not explore all the solution paths once it got a soln (incomplete)

* Knowledge Representation and Reasoning

→ Logic : Propositional logic and Predicate logic
→ Rules : if then
→ Semantic Net : Google graph / Meaning graph
→ Frame : Slots and fillers / Object Attribute.
→ Script :

* Propositional Logic (Either True or False; not Both).

→ Syntax → Atomic
→ Complex.
→ Semantic

- Negation: ¬ , ~ , ⁊
- Disjunction/or : ∨
- conjuction/And: ∧
- if then : → imply, implications.
- iff : ↔
-

Example: You can access internet from campus only if you are CSE student or you are not freshman.

Let, P: access internet from campus
Q: CSE Student
R: Freshman.

Then, Answer: $P \rightarrow Q \vee \neg R$

Syntax of First order Logic:

Constant: 1, 2, A, Name, Location ...
Variables: x, y, z, a, b ...
Predicates: Brother, father, greater than ...
Functions: sqrt, ...
Connectives: ¬ , ⇒ , ∧ , ∨ , ⇔ , → , ↔
Equity : =
Quantifiers : ∀ , ∃

* Main Connective for ∀ is '→' E.g: $\forall x\, man(x) \rightarrow respect(x, parent)$
Main Connective for ∃ is '∧' E.g: $\exists x: boy(x) \wedge intelligent(x)$

* Resolution in First Order Logic:

→ Resolution is a theorem proving technique that proofs by contradiction

→ "Unification" is key concept in proofs by Resolution.
(by substitution)

→ Resolution is single inference rule which can effectively operate on CNF (conjunctive normal form) or Clausal form.

Clause: Disjunction (∧) of literals

CNF : Sentence Represented as a Conjuction (∨) of Clauses

- Steps for Resolution:
  1. Conversion of facts into FOL
  2. Convert FOL into CNF
  3. Negate the statement which needs to be proved.
  4. Draw Resolution Graph (tree (Unification).

Example:

<a> John likes all kind of food.
<b> Apple and vegetable are food.
<c> Anything anyone eats and not killed is food.
<d> Anil eats peanuts and still alive.
<e> Harry eats everything Anil eats.
PROVE by Resolution that:
<f> John likes peanuts.

Step I:

<a> $\forall x : food(x) \rightarrow likes(John, x)$

<b> $food(Apple) \wedge food(vegetable)$.

<c> $\forall x \forall y : eats(x,y) \wedge \neg killed(x) \rightarrow food(y)$

<d> $eats(Anil, peanuts) \wedge alive(Anil)$

<e> $\forall x : eats(Anil, x) \rightarrow eats(Harry, x)$

Added Predicates:

<f> $\forall x : \neg killed(x) \rightarrow alive(x)$

<g> $\forall x : alive(x) \rightarrow \neg killed(x)$

To prove by resolution.

<h> $likes(John, peanuts)$

Step II: Conversion of FOL into CNF:

Eliminate all '→' implication and Rewrite

<a> $\forall x \neg food(x) \vee likes(John, x)$
<b> $food(Apple) \wedge food(vegetable)$
<c> $\forall x \forall y \neg [eats(x,y) \wedge \neg killed(x)] \vee food(y)$
<d> $eats(Anil, peanuts) \wedge alive(Anil)$
<e> $\forall x \neg eats(Anil, x) \vee eats(Harry, x)$
<f> $\forall x \neg [\neg killed(x)] \vee alive(x)$

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

(h) likes (John, peanuts).

$\Rightarrow$

Move negations(¬) inwards and Rename variables or Standardise variables

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

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

(c) $\forall y \forall z \neg eats(y, z) \vee killed(y) \vee food(z)$

(d) eats (Anil, peanuts) $\wedge$ alive (Anil)

(e) $\forall w \neg eats(Anil, w) \vee eats(Harry, w)$

(f) $\forall g \neg killed(g) \vee alive(g)$

(g) $\forall k \neg alive(k) \vee \neg killed(k)$

(h) likes (John, peanuts).

$\Rightarrow$ Eliminate existential instantion quantifier (Skolemize, $\exists$).
(but in the above example, we don't have any $\exists$), so skip this step

$\Rightarrow$

Drop universal quantifier ($\forall$) and write disjunction ($\vee$) in different point.

(a) $\neg food(x) \vee likes(John, x)$

(b) food (Apple)

(c) food (vegetables)

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

(e) eats (Anil, peanuts)

(f) alive (Anil)

(g) $\neg eats(Anil, w) \vee eats(Harry, w)$

(h) $\neg killed(g) \vee alive(g)$

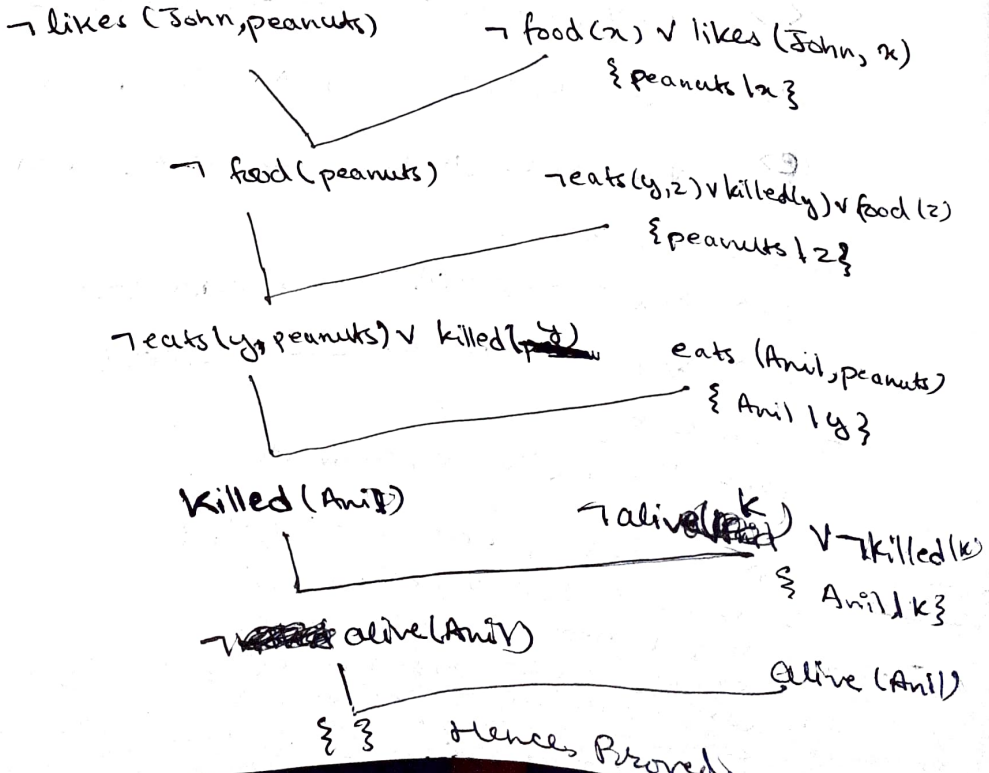
(i) $\neg alive(k) \vee \neg killed(k)$

(j) likes (John, peanuts)

$\Rightarrow$ Negate statement to be proved:

(k) $\neg$ likes (John, peanuts)

$\Rightarrow$ Draw Resolution Graph:



Example: (a) If it is Sunny and warm day you will enjoy.

(b) If it is raining, you will get wet.

(c) It is a warm day.

(d) It is raining.

(e) It is Sunny.

Goal: You will enjoy.

→ ⇒ (a) Sunny ∧ Warm → Enjoy.

(b) Raining → wet

(c) Warm.

(d) Raining

(e) Sunny.

(f) Enjoy.

⇒ (a) ¬(Sunny ∧ Warm) ∨ Enjoy → (a) ¬Sunny ∨ ¬Warm ∨ Enjoy

(b) ¬Raining ∨ Wet → (b) ¬Raining ∨ wet.

(c) Warm → (c) Warm

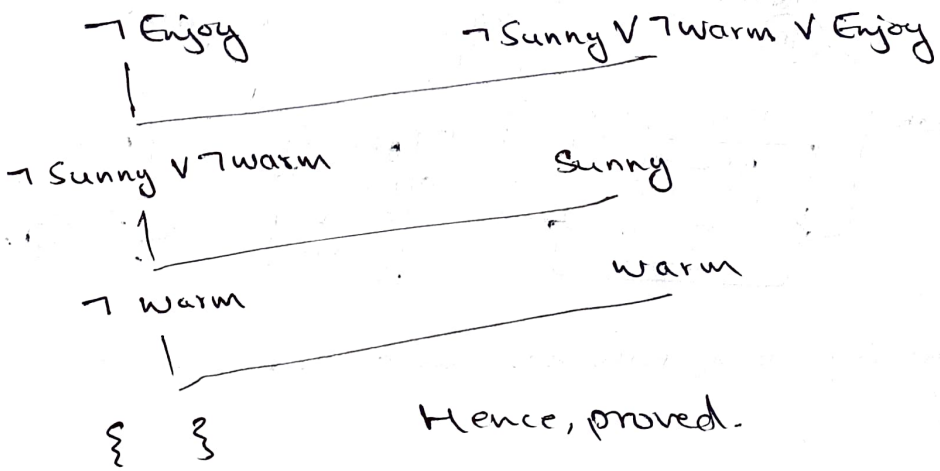
(d) Raining → (d) Raining

(e) Sunny → (e) Sunny

(f) Enjoy → (f) Enjoy

⇒ Assume : ¬Enjoy

⇒ Resolution Graph:



Hence, proved.

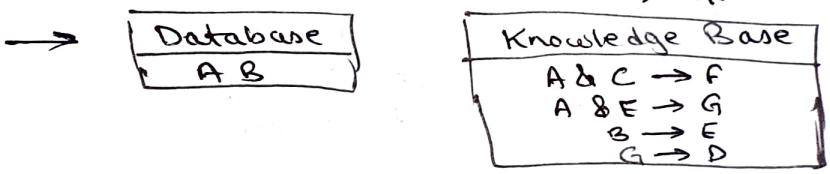
So You will enjoy.

* Forward and Backward chaining:

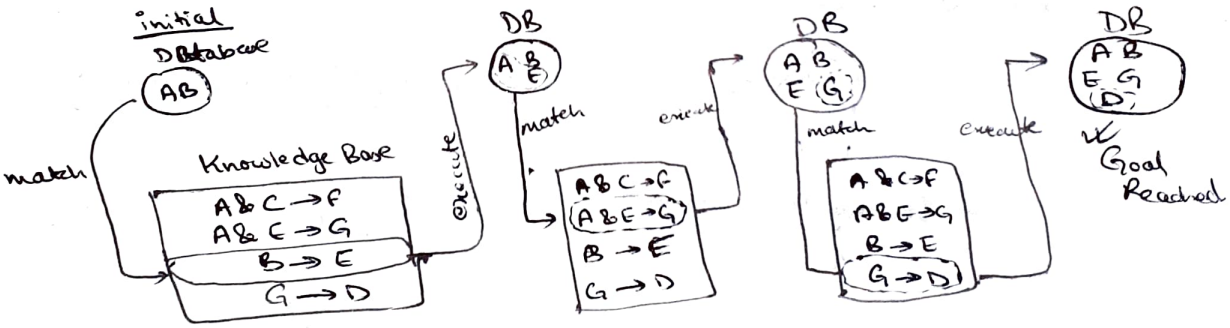
- **Forward Chaining:**
  - form of reasoning which starts with atomic sentences in the knowledge base and applies inference rules in the forward direction to extract more data until a goal is reached.
  - moves from bottom to up (top)
  - process of making conclusion based on known facts or data, by starting from initial state to reach goal state.
  - AKA data-driven.
  - commonly used in Expert system.

Example: Rule 1: If A and C, then F
R 2: If A and E then G
R 3: If B then E
R 4: If G then D.

To prove: if A and B true, then D is true.



To prove: A & B → D.



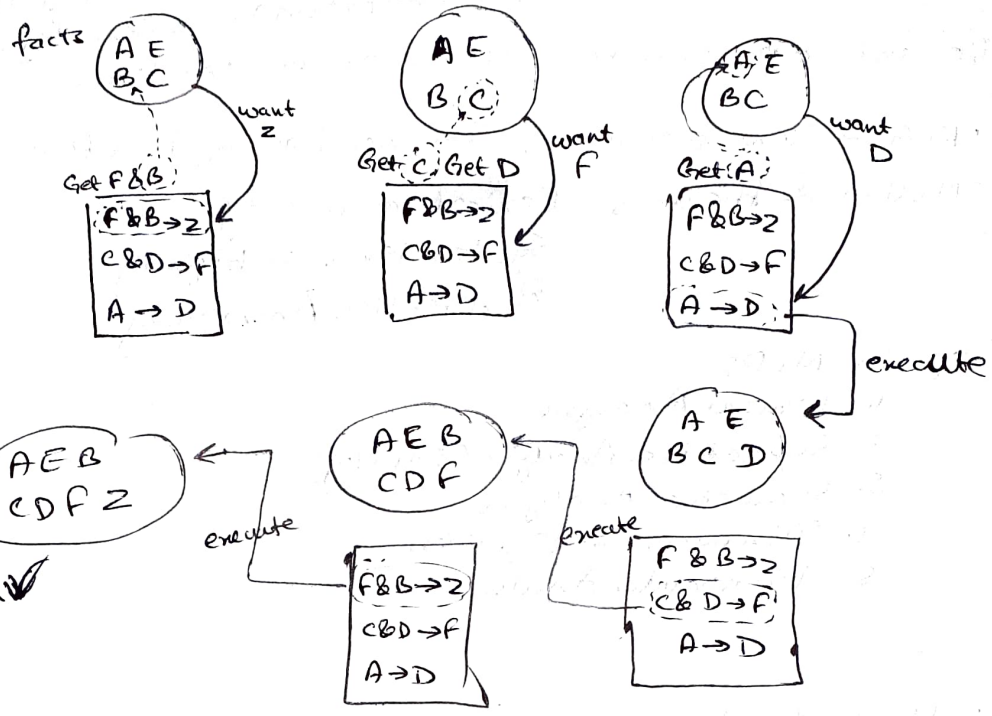
meaning, using inital A B, we reached or executed D, hence goal reached.

- **Backward chaining:**
  - form of reasoning, which starts with goal state and works backward, chaining through rules to find known facts to support the goal.
  - AKA top-down approach / Moves from top to bottom.
  - Based on Modus ponens inference rule.
  - Goal is broken into sub-goal(s) to prove the facts true.
  - AKA goal-driven approach.

- Used in game theory, automated theorem proving tools, inference engines, proof assistants etc.
- Backward-chaining method uses depth-first search strategy for proof

Example:

Goal state: z



Forward Chaining	Backward Chaining
1. Starts from known facts and applies inference rule to extract more data until it reaches to the goal.	1. Starts from goal and works backward through inference rules to find required facts that support goal.
2. Bottom-up approach.	2. Top-down approach.
3. AKA data-driven technique	3. AKA goal-driven technique.
4. Applies breadth-first search.	4. Applies depth-first search strategy
5. Tests for all available rules	5. Tests for few required rules.
6. Suitable for monitoring, planning, control and interpretation.	6. Suitable for diagnostic, prescription and debugging
7. Can generate infinite number of possible conclusions.	7. generates finite number of possible conclusions.
8. Operates in forward direction.	8. Operates in backward direction
9. Aimed for 'any' conclusion	9. Aimed only for the required conclusion.

* NLP (Natural Language Processing)

— Components: NLU (NL Understanding) : Mapping i/p
NLG (NL Generation) : Generating o/p

i/p and o/p can be speech or written text.

- NLG → Text planning, Sentence Planning, Text Realisation
- NLU → Difficulties: Lexical ambiguity
Syntax level ambiguity
Referential Ambiguity

Steps in NLP:

1. Lexical Analysis
2. Syntactic Analysis (Parsing)
3. Semantic Analysis
4. Discourse Integration
5. Pragmatic Analysis.

* Applications of AI:

finance, music, transportation, Hospitals, toys and games

- Expert Systems : Computer based AI which replicates human's problem solving capabilities

cyberbullying, codemixin data, fake news detection, deepfakes, identification of hate content, fake dubbing, Chatbot

- Codemixin data : example, using hinglish to bypass hate speech detection/filtering as a form of hate speech transmission, cyber bullying etc.

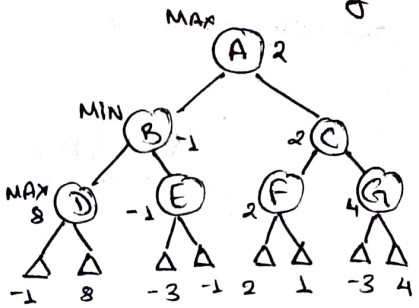
Monotonic Reasoning	Non-monotonic Reasoning
• Once the conclusion is made (or goal state is achieved), it will remain unchanged even if some other information is added to the existing info of our knowledge base.	• Varying conclusions, some conclusion may be invalidated if some other/more info(s) is/are added to our knowledge base.
• Decisions are not affected by new facts. (eg: Universal truths)	• Decisions can be changed by new facts.
• Advantage: All old proofs are valid	• Advantage: Helpful in real-world scenarios.
• Disadvantage: (i) Cant read world scenarios (ii) New knowledge from real world cant be added.	• Disadvantage: (i) Cant be used for theorem proving

* MINIMAX Algorithm :

(Not min-max ✗
Mini-max ✓)

- Minimax Algo in game playing :
  - Backtracking Algo
  - Best move strategy used
  - (Best move) → Max will try to maximise its utility
  - (worst move) → Min will try to minimise utility

- Space Complexity : $O(b^d)$
- Time complexity : $O(b^d)$
- Specialised search Algo that returns optimal sequence of moves for a player in zero sum Game.
- Recursive Backtracking Algo : uses recursion to search through Game Tree.
- Depth-first search Algo is used for exploration



PROPERTIES:

1. Complete : Definitely finds soln (if exists)
2. Optimal

LIMITATIONS :

1. Slow for complex games such as chess ($O(35^{100})$)

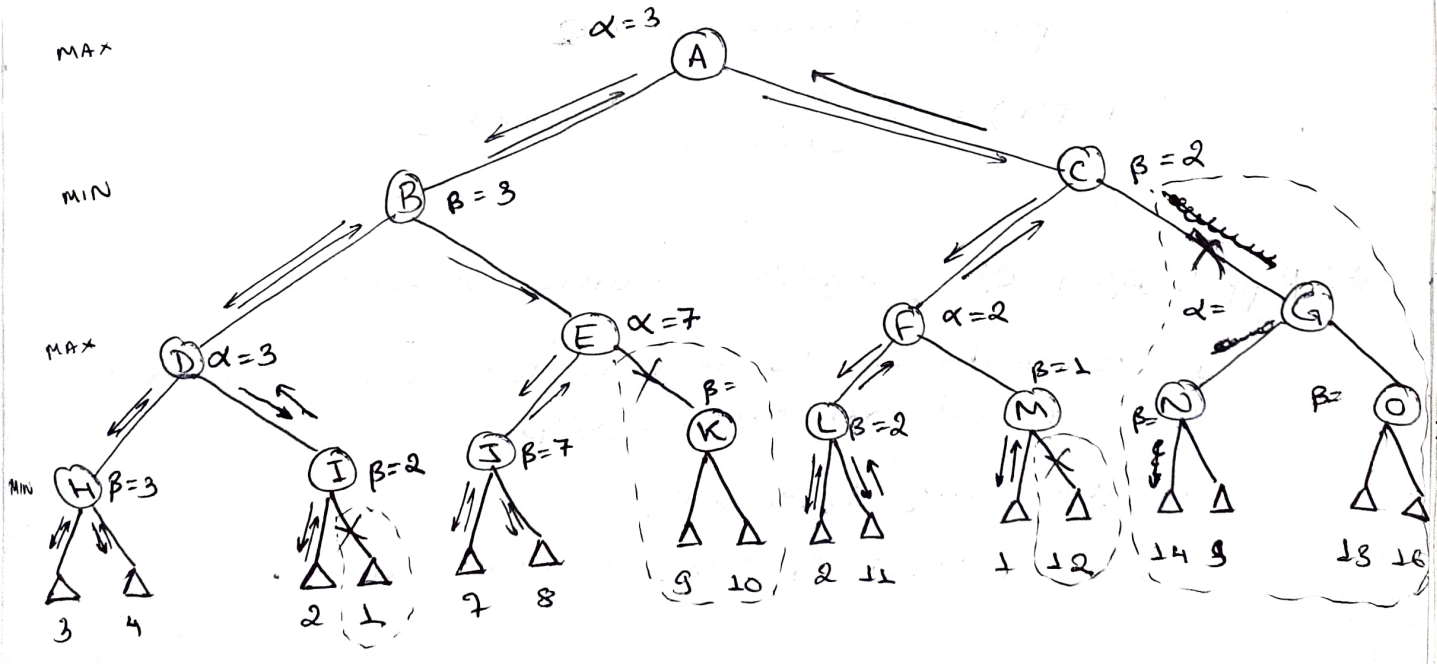
→ Initial values (worst values selected) ie, MAX = $-\infty$; MIN = ∞

* Alpha-Beta Pruning (α-β pruning).

→ Advance version of Minimax algo.

- Cut off search by exploring less no. of nodes.

α = MAX ; β = MIN.



* Bayes' Theorem:

- Describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

$$P(H|E) = \frac{\text{No. of times (H and E)}}{\text{No. of times (E)}} = \frac{P(H \cap E)}{P(E)}$$

Probability of H when E is True

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}$$ } Bayes theorem.

Likelihood, Prob. of evidence.

Posterior, (Prob. of A when B is true)

Marginal Prob. (Prob. of evidence)

Prior Prob. (Prob. of hypothesis)

Example: What is probability that person has dengue with neck pain?

Given, neck pain is caused in 80% of dengue patient

$P(\text{dengue}) = 1/30,000$

$P(\text{neck pain}) = 0.02$

Let, A → Proposition that person has neck pain.

B → Proposition that person has dengue.

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

Here, $P(A|B) = 80\% = 0.8$

$P(B) = P(\text{dengue}) = 1/30000$

$P(A) = P(\text{neck pain}) = 0.02$

$$\therefore P(B|A) = \frac{0.8 \times 1}{0.02 \times 30000} = 0.00133$$

* Application of Bayes's Theorem in AI:

1. Robot / Automatic machine: Next step is calculated based on previous step
2. weather forecasting
3. Monty Hall Problem can be solved.