Hasan Rafee
CAM; CI91029
48th CSE...

~~Chapter~~

FOL → Inference / Reasoning

→ material implication
¬ not
∨ or
∧ and
∀ for all
∃ there exists

Well-formed formula (wff) in propositional logic or FOL represent assertion knowledge.

wff → Rules (Assertion given in implication form)
wff → Facts (assertions that represent domain specific knowledge)

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions.

A proposition is a declarative statement which is either true or false.

Example: 5+5=10 (True proposition)
The sun rises from west (False proposition)

Basic facts about PL:

(i) It is also called Boolean logic as it works on 0 and 1.

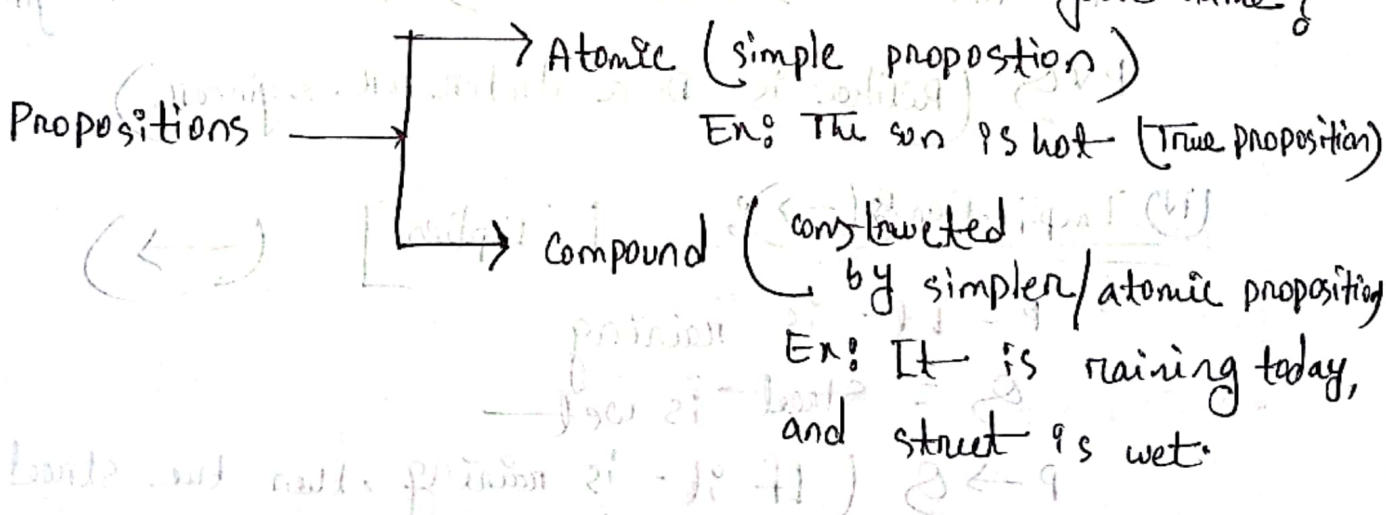
(ii) We can use symbolic variables to represent the logic such A, B, C, P, Q, R, etc.

(iii) Propositions can be either true or false, but it can't be both.

(iv) A proposition formula which is always true is called tautology and it is also called a valid sentence.

(v) A proposition formula which is always false is called "Contradiction".

(vi) statements which are questions, commands or opinions are not propositions. Ex: What is your name?



Logical Connectives:

Logical connectives are used to connect two simpler propositions or representing a sentence logically. There are mainly five connectors:

(i) Negation ($\neg$): [Not] ($\neg$ or $\sim$)

$\neg P$ it is called negation of P. P can be true or false.

(ii) Conjunction ($\wedge$): [And] ($\wedge$)

$P \wedge Q$ (He is intelligent and hardworking)

P = He is intelligent

Q = He is hardworking

(iii) Disjunction ($\vee$): [Or] ($\vee$)

P = Ritika is Doctor Q = Ritika is Engineer

$P \vee Q$ (Ritika is a doctor or engineer)

(iv) Implication ($\rightarrow$): [Implies] ($\rightarrow$)

P = It is raining

Q = Street is wet

$P \rightarrow Q$ (If it is raining, then the street is wet)

v) Biconditional (↔):

~~Biconditional~~ → If and only if ↔

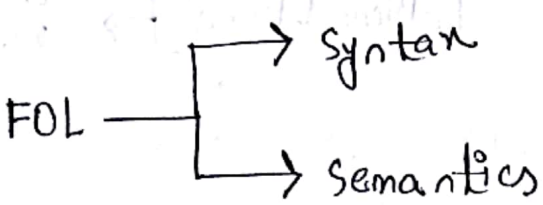
P = I am breathing
Q = I am alive
P ↔ Q (If I am breathing, then I am alive)

First Order Logic

FOL is a way of knowledge representation in artificial intelligence which is an extension to propositional logic.
FOL is also known as Predicate logic or First-order predicate logic.

FOL does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:

1. Objects: A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, - - -
2. Relations: It can be unary relation such as: red, round, is adjacent, or any relation such as: the sister of, brother of, has color, comes between
3. Function: Father of, best friend, third inning of, end of, ...



Syntax:

The syntax of FOL determines which collection of symbols is a logical expression in FOL.

Following are the basic elements of FOL syntax:

(i) Constant → 1, 2, A, Name, city name, cat, ---.

(ii) Variables → x, y, a, b, ---

~~(iii) Brother, Father, >, ---~~

~~(iv) F~~

(iii) Predicates → Bor Brother, Father, >, ---

(iv) Function → sqrt, LeftLegOf, ---

(v) Connectives → $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$

(vi) Equality → ==

(vii) Quantifier → $\forall$, $\exists$

(#) Atomic sentences:

Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.

We can represent atomic sentences as predicate (term1, term2, ---, termn).

Ex: X and Y are brothers: => Brothers (X, Y)

Chinky is a cat: => cat (Chinky)

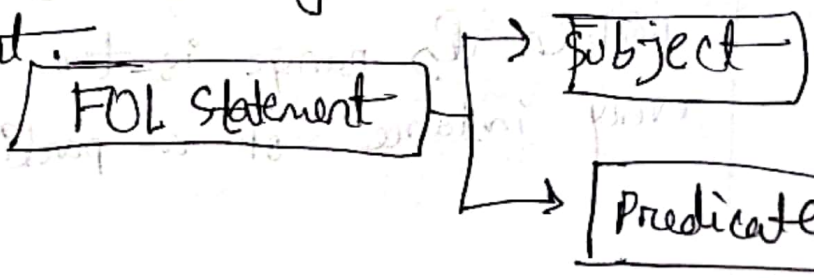
(#) Complex sentences:

Complex sentences are made by combining atomic sentences using connectives.

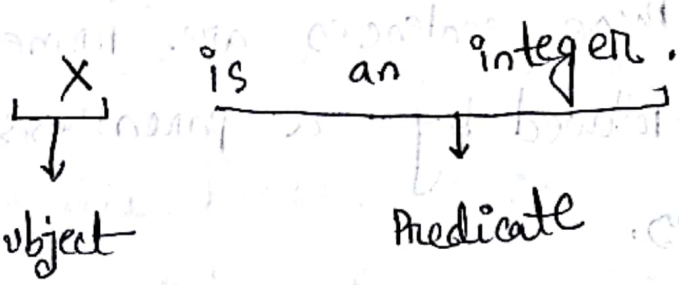
FOL statements can be divided into two part.

(i) **Subject:** The main part of the statement

(ii) **Predicate:** Can be defined as a relation, which binds two atoms together in a statement.

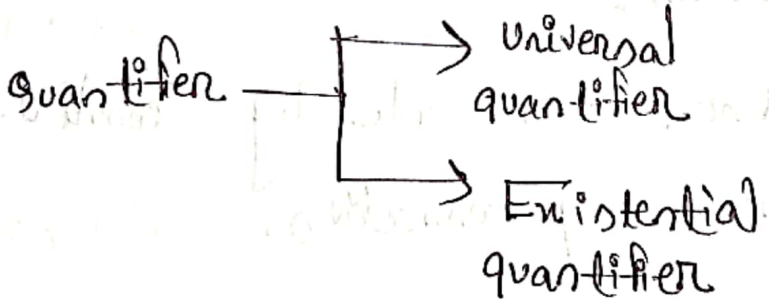


Example:
A statement → "X is an integer."



#) Quantifiers in FOL:

A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.



ⓐ Universal quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within it's range is true for everything or every instance of a particular thing.

The universal quantifier is represented by a symbol "$\forall$".

In universal quantifier we use implication "$\rightarrow$".

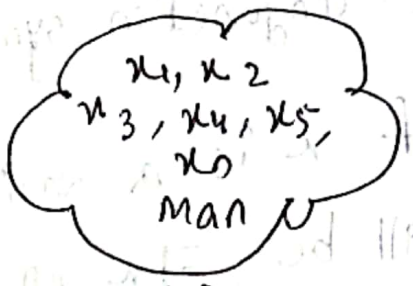
Example

If x is a variable, then $\forall x$ is read as:

(i) For all x

(ii) For each x

(iii) For every x



Universe of Discourse

Example:

All ~~no~~ men drink coffee.

Let a variable x which refers to a ~~cat~~ man so all x can be represented in Universe of Discourse (UOD) as below:

- x_1 drinks coffee
- $\wedge$
- x_2 drinks
- $\wedge$
- x_3 drinks ~~or~~ coffee
- $\wedge$
- $\wedge$
- $\wedge$
- x_n drinks ~~milk~~ coffee

So, in shorthand notation, we can write it as:

$\forall x \; man(x) \rightarrow drink(x, coffee)$

It will be read as:

There are all x where x is a man who drink coffee.

④ Existential quantifiers:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

It is denoted by the logical operator $\exists$.

[In Existential quantifier we always use AND or Conjunction symbol ($\wedge$).]

If x is a variable, then existential quantifier will be $\exists x$ or $\exists(x)$. It will be read as:

- There exists a 'x'.
- For some 'x'
- For at least one 'x'

Universe of Discourse

Example:

Some boys are intelligent.

x1 is intelligent
∨
x2 is intelligent
∨
x3 is intelligent
∨
.
.
∨
xn is intelligent

In short-hand notation, we can write it as:

$\exists x$: boys $(x) \wedge$ intelligent (x)

It will be read as:

There are some x where x is a boy who is intelligent.

Points to remember:

- The main connective for universal quantifier $\forall$ is Implication $\rightarrow$.
- The main connective for existential quantifier $\exists$ is and $\wedge$.

Properties of Quantifiers:

(i) In universal quantifier, $\forall x \forall y$ is similar to $\forall y \forall x$.

(ii) In existential quantifier, $\exists x \exists y$ is similar to $\exists y \exists x$.

(iii) $\exists x \forall y$ is not similar to $\forall y \exists x$.

Examples of FOL using quantifiers:

(i) All birds fly.

⇒ Solution:

Here the predicate is "fly (bird)."

And since there are all birds who fly so it will be represented as follows.

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

(ii) Every man respects his parent.

Solution:

Here the predicate is "respects(x, y)," where x = man and y = parent

Since there is every man so we will use ∀ (universal quantifier) and it will be represented as follows:

$\forall x \; man(x) \rightarrow respects(x, parent)$

(iii) Some boys play cricket.

Solution:

Predicate is "play(x, y)"

x = boys

y = cricket

So we will use ∃ (existential quantifier) and it will be represented as:

$$\exists x \; \text{boys}(x) \rightarrow \text{play}(x, \text{cricket})$$

(iv) Not all students like both Mathematics and Science.

Solution:

Here predicate is "like(x, y)";

where x = student ; y = subject

Since there are not all students, so we will use ∀ with negation. So following representation for this:

$$\neg \forall (x) \; [\text{student}(x) \rightarrow \text{like}(x, \text{Mathematics}) \wedge \text{like}(x, \text{Science})]$$

(v) Only one student failed in Mathematics.

Solution:

Here Predicate is "failed(x, y)"; where

x = student

y = subject

একটি subject আবারও "subject" হিসেবে categorized করা যাবে

$$\exists(x)\,[\text{student}(x) \rightarrow \text{failed}(x, \text{Mathematics}) \wedge \forall(y)\,[\neg(x == y) \wedge \text{student}(y) \rightarrow \neg\,\text{failed}(x, \text{Mathematics})]]$$

(vi) All dogs are faithful.

Solution:

$$\forall x\,(\text{dog}(x) \rightarrow \text{faithful}(x))$$

(vii) Mammals drink milk

Solution:

$$\forall x\,(\text{mammal}(x) \rightarrow \text{drink}(x, \text{milk}))$$

(viii) Some birds cannot fly.

Solution:

$$\neg(\exists x\,(\text{bird}(x)) \rightarrow \text{fly}(x))$$

(ix) Fathers are male parents with children.

Solution:

$$\forall x\,(\text{Father}(x) \rightarrow \text{male}(x) \wedge \text{has\_children}(x))$$

(x) Students are people who are enrolled in courses.

Solution: $\forall x\,(\text{Student}(x) \rightarrow \text{people}(x) \wedge \text{enrolled\_in}(x, \text{courses}))$

(xi) Brothers are siblings.

Solution:

$$\forall x, y \quad \text{Brother}(x, y) \longrightarrow \text{siblings}(x, y)$$

(xii) Everyone is loyal to someone.

Solution:

$$\forall x \; \exists y \; \text{loyal}(x, y)$$

Extra:

(i) Man is mortal.

Solution:

$$\forall x \quad \text{Man}(x) \longrightarrow \text{mortal}(x)$$

(ii) All birds cannot fly.

Solution:

$$\neg\left(\forall x\left(\text{bird}(x) \longrightarrow \text{fly}(x)\right)\right)$$

(iii) No dogs purr.

Solution:

$$\neg\left[\exists x\left(\text{dog}(x) \wedge \text{purr}(x)\right)\right]$$

(iv) Some dogs bark.

Solution:

$$\exists x\left(\text{dog}(x) \wedge \text{bark}(x)\right)$$

(v) All dogs have four legs.

Solution:

$$\forall x \left[dog(x) \longrightarrow have\text{-}four\text{-}legs(x) \right]$$

(vi) There are at least two cats.

Solution:

$$\exists x \exists y \left[cat(x) \wedge cat(y) \wedge \neg(x = y) \right]$$

(vii) Every dog chases some ~~cats~~ cats.

Solution:

$$\forall x \left[dog(x) \longrightarrow \exists y (cat(y) \wedge chases(x, y)) \right]$$

Inference

Generating the conclusions from evidence and facts is termed as inference.

(ii) Inference rules:

Inference rules are the templates for generating valid arguments. Following are some terminologies related to inference rules:

(i) **Implication:** It is one of the logical connectives which can be represented as $P \rightarrow Q$. It is a Boolean expression.

(ii) **Converse:** $Q \rightarrow P$; converse of implication

(iii) **Contrapositive:** The negation of converse is termed as ~~cons~~ contrapositive and it is represented as $\neg Q \rightarrow \neg P$

(iv) **Inverse:** The negation of implication is called inverse. It can be represented as $\neg P \rightarrow \neg Q$

Truth table:

P	Q	$P \to Q$	$Q \to P$	$\neg Q \to \neg P$	$\neg P \to \neg Q$
T	T	T	T	T	T
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	T

From the above truth table, we can prove that $P \to Q$ is equivalent to $\neg Q \to \neg P$ and $Q \to P$ is equivalent to $\neg P \to \neg Q$

(11) Types of Inference rules:

(1) Modus Ponens:

The modus Ponens rule states that if P and $P \to Q$ is true, then we can infer that Q will be true. It can be represented as:

Notation for Modus Ponens: $\dfrac{P \to Q, P}{\therefore Q}$

Example:

statement-1: "If I am sleepy then I go to bed" ==> $P \rightarrow Q$

statement-2: "I am sleepy" ==> P

conclusion: "I go to bed." ==> Q.

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

Proof by Truth table:

P	Q	P → Q
0	0	0
0	1	1
1	0	0
1	1	1

(2) Modus Tollens, (3) Hypothetical Syllogism,
(4) Disjunctive Syllogism, (5) Addition, (6) simplification
(7) Resolution:

The resolution rule state that if $P \vee Q$ and $\neg P \wedge R$ is true, then $Q \vee R$ will also be true. It can be represented as

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

Proof by Truth-Table :

P	¬P	Q	R	P ∨ Q	¬P ∧ R	Q ∨ R
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1

Game Playing

Min - Max Algorithm:

It is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.

Chess, Checkers, tic-tac-toe, go and various two-players game. This algorithm computes the minmax decision for the current state. In this algorithm two players play the game, one is called Max and other is called Min.

(#) Pseudo-code for MinMax Algorithm:

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

if MaximizingPlayer then //for Maximizer Player
maxEva = -infinity
  for each child of node do
  eva = minmax(child, depth -1, false)
  maxEva = max(maxEva, eva) //gives Maximum of the values
  return maxEva
```

else // for Minimizer player

minEva = +infinity

for each child of node do

eva = minmax (child, depth -1, true)

minEva = min (minEva, eva) // gives minimum of the values

return minEva

~~Initial call:~~

Initial call:

Minmax (node, 3, true)

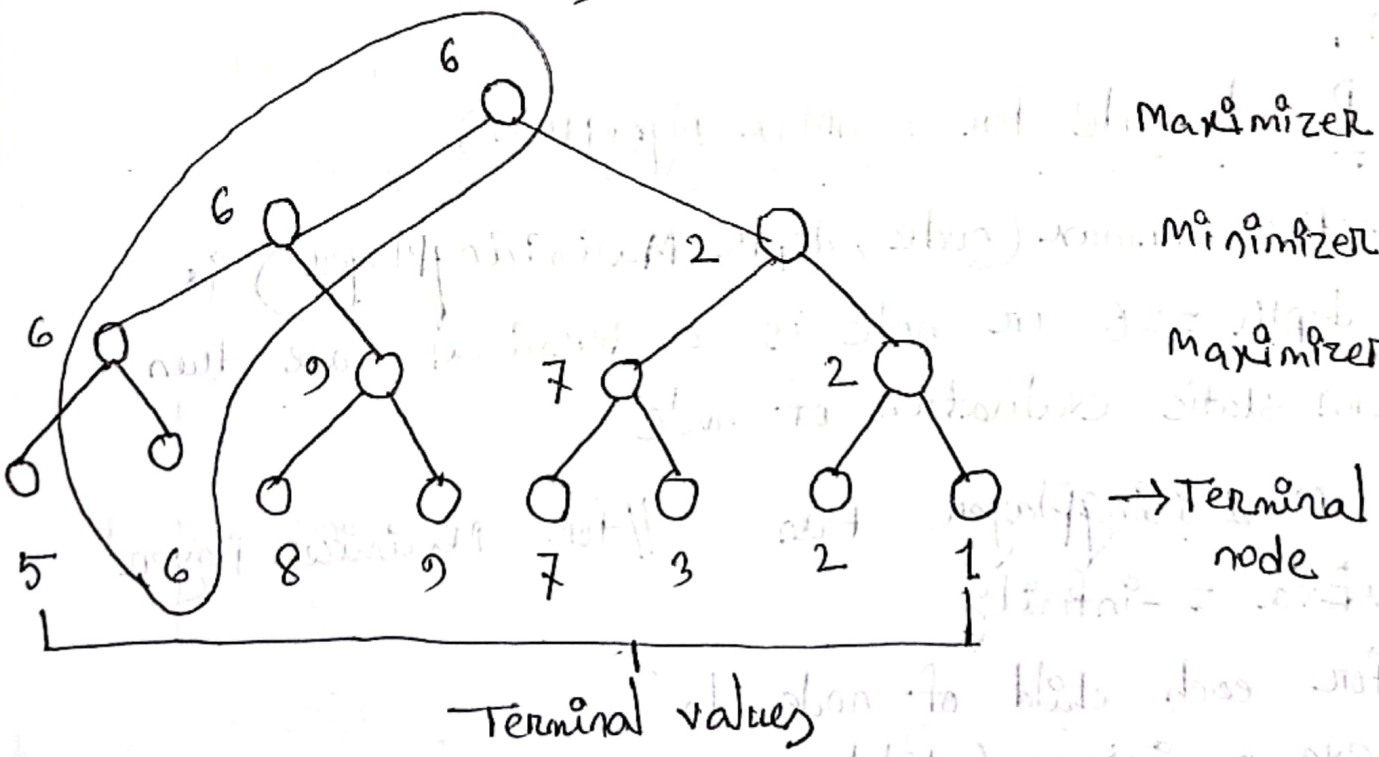


Fig: max-min Algorithm

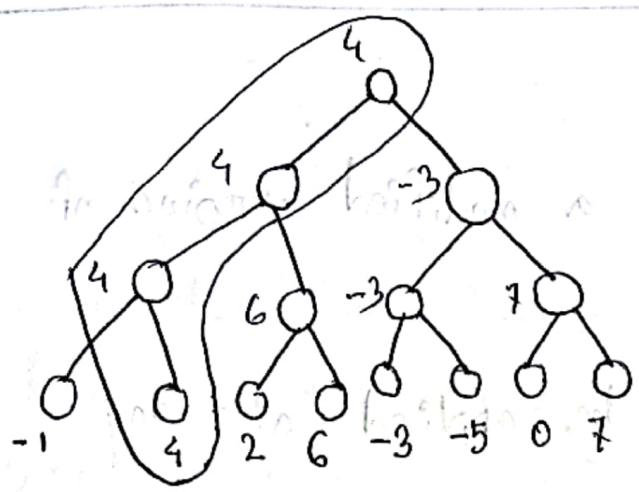


Fig: Max-Min Algorithm

(#) Properties of Max-Min algorithm:

(*) Complete — There must be a solution in the finite search tree.

(**) Optimal — It is optimal if both opponents are playing optimally.

(***) Time complexity — Time complexity is $O(b^m)$; where
b = branching factor of the game-tree
m = maximum depth of the tree.

(****) Space Complexity — Space complexity is $O(bm)$.

(#) Limitation of the max-min algorithm:

The main drawback of this algorithm is that it gets really slow for complex games such as chess, go, etc. Because these games contain huge branching factor and the player has lots of choices to decide.

(#) Alpha-Beta Pruning:

Alpha-beta pruning is a modified version of the max-min algorithm.

Alpha-beta pruning can be applied at any depth of a tree ~~, and sometimes~~

(*) Alpha = The best (highest α value) choice
The initial value of alpha is $-\infty$

(*) Beta = The best (lowest -value) choice.
The initial value of beta is $+\infty$.

(#) Condition of Alpha-beta pruning:
$\alpha >= \beta$

Key points:

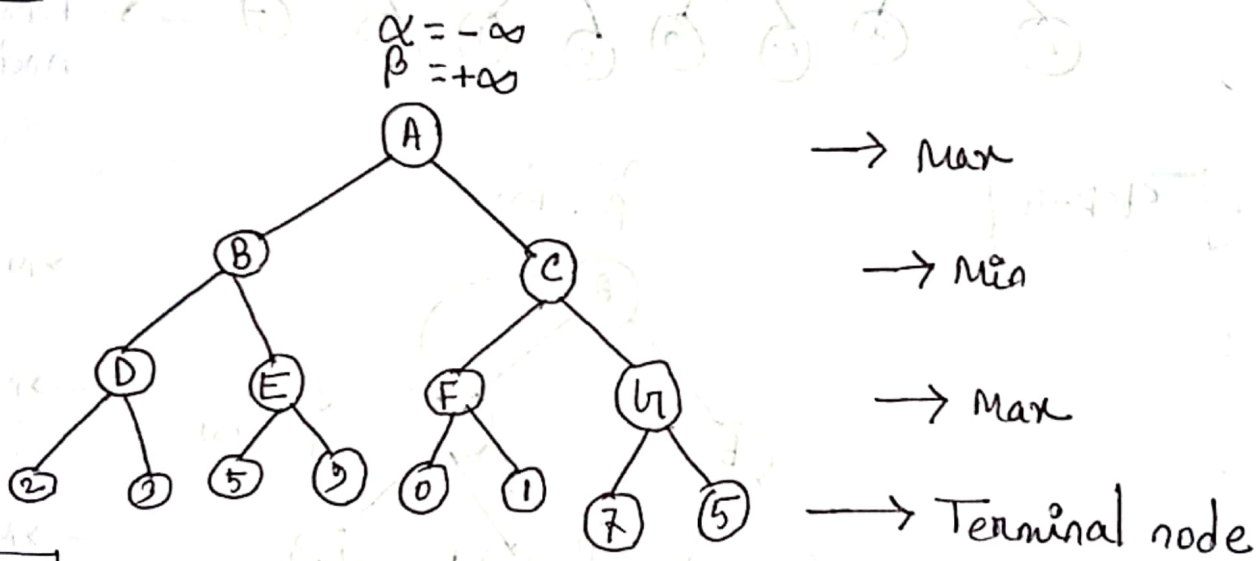
- The max player will only update the value of alpha.
- The min player " " " " of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.

• we will only pass the alpha, beta values to the child nodes.

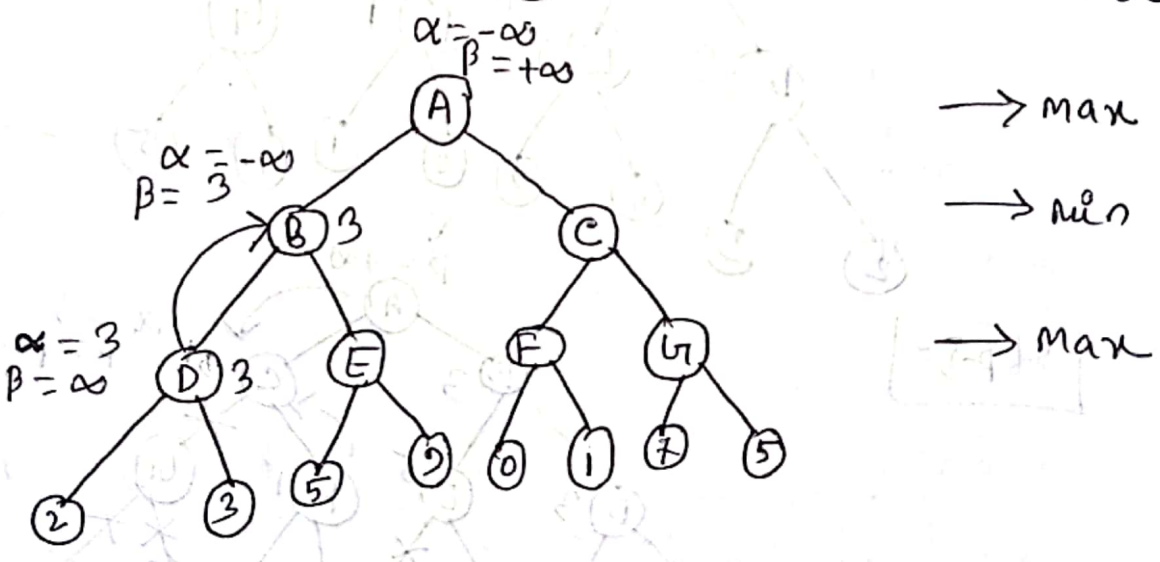
Working of Alpha-Beta Pruning:

Let's take an example of two-player search tree to understand the working of Alpha-beta pruning.

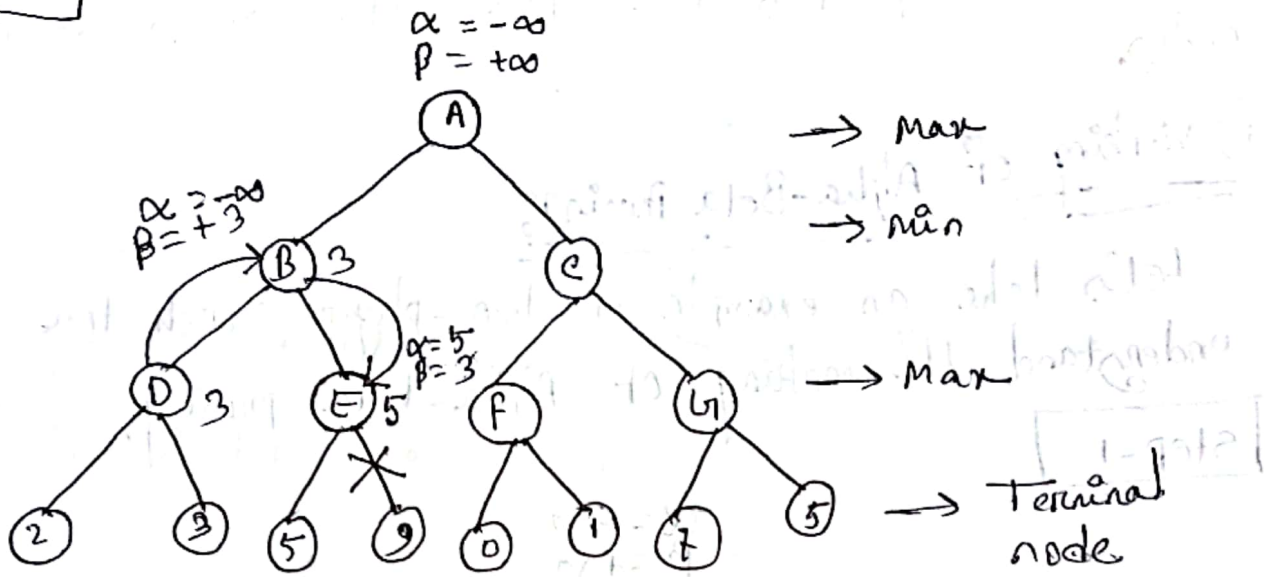
Step-1



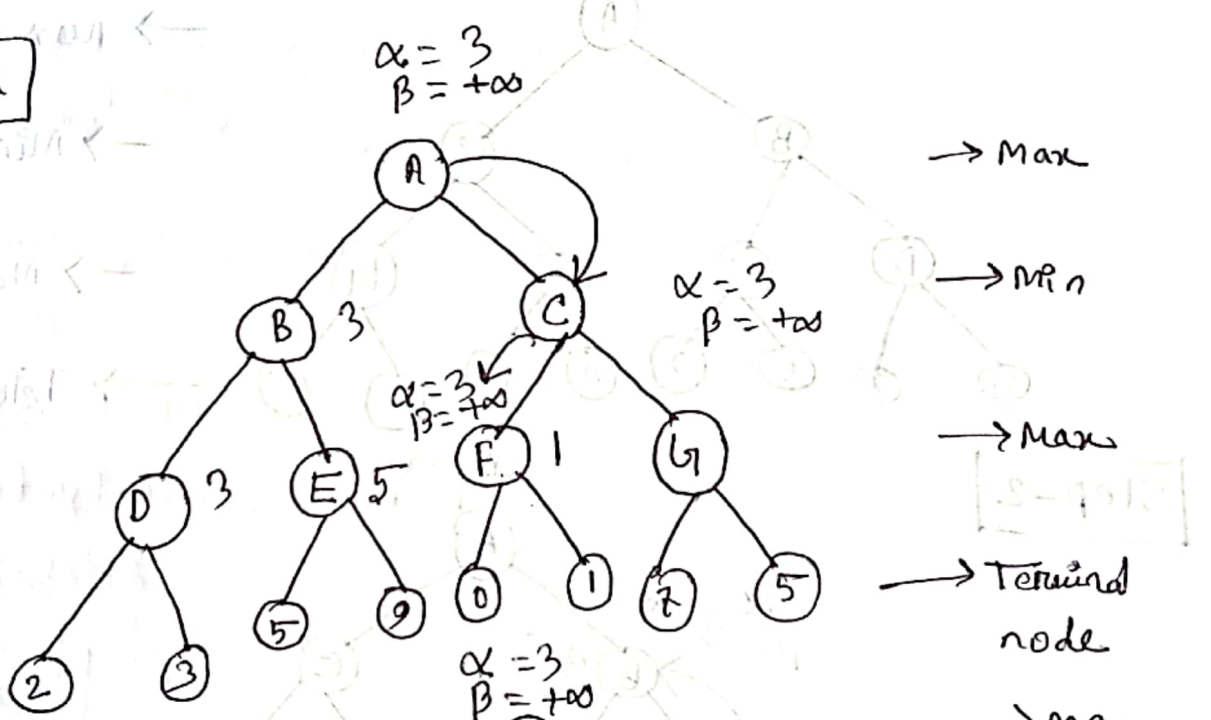
Step-2



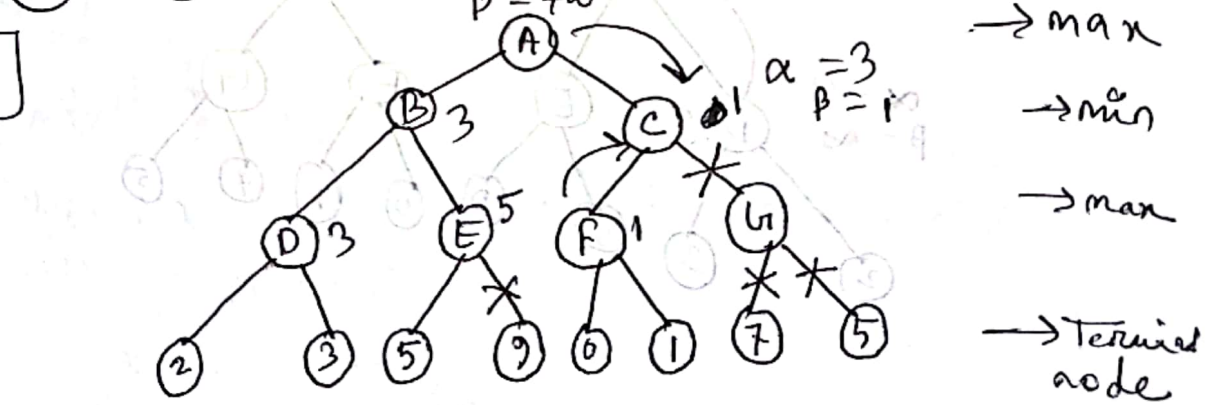
Step-3



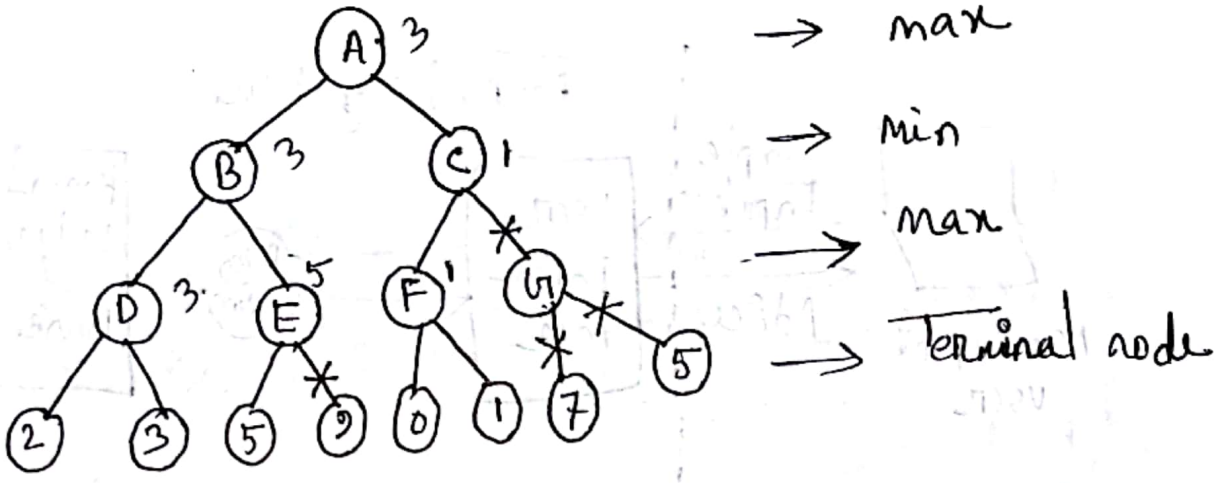
Step-4



Step-5



Step-6



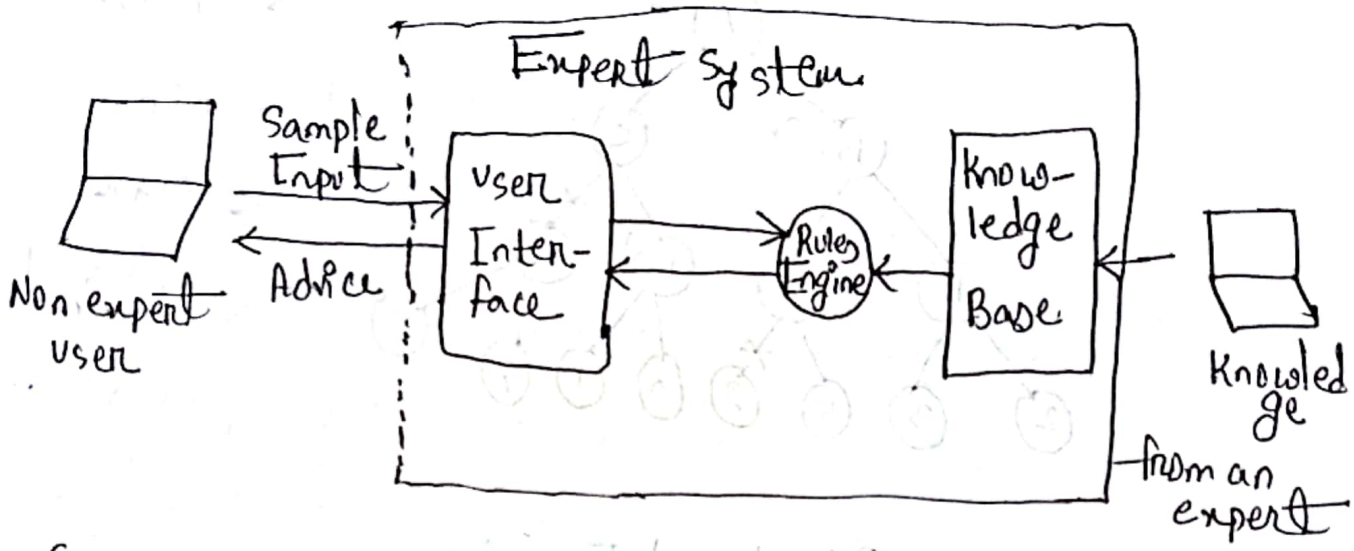
Rule-based Expert System

(#) What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert.

The ~~perforce~~ performence of an expert system is based on the ~~expert~~ expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance.

Example: suggestion of spelling errors while typing in the Google search box.



Some popular examples of the Expert System:

(i) Dendral (ii) Mycin (iii) Prides (iv) CaDeT

(4) Characteristics of Expert System:

(i) High Performence
(ii) Understandable
(iii) Reliable
(iv) Highly responsive

(5) Components:

An expert system mainly ~~cost~~ consists of three components:

(i) User interface
(ii) Interface Engine
(iii) Knowledge Base

(*) User Interface => It helps a non-expert user to communicate with the expert system to find a solution.

(*) Interface Engine => It is the brain of the expert system as it is the main processing unit of the system.

Interface Engine
- → Deterministic interface engine (It is based on facts and rules. Conclusions are assumed to be true)
- → Probabilistic Interface engine (Contains uncertainty in conclusions and based on the probability.)

Interface engine uses two modes to derive solutions
- → Forward chaining (starts from the known facts and rules)
- → Backward chaining (starts from the goal and works backward)

(*) Knowledge Base => It is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert system.

Components of knowledge Base:

(i) Factual knowledge: The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

(ii) Heuristic knowledge: The knowledge which is based on practice, the ability to guess, evaluation and experiences.

(iii) Knowledge Representation:

It is used to formalize the knowledge stored in the knowledge base using the if-else rules.

(iv) Knowledge Acquisitions:

It is the process of extracting, organizing and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts and store that knowledge into the knowledge base.

(#) Participants in the development of Expert System:

(*) Expert: The success of an ES much depends on the knowledge provided by human experts. These experts are those ~~who~~ persons who are specialized in that specific domain.

(*) Knowledge Engineer: Knowledge engineer is the person who gathers the knowledge from the domain experts and then codifies that knowledge to the system according to the formalism.

(*) End-User: This is a particular person or a group of people who may not be experts and working on expert system needs the solution or advice for his queries, which are complex.

Why we need Expert system?

(i) No memory limitations: It can store as much data as required and can memorize it at the time of it's application. But, for human experts, there are some limitations to memorize all things at every time.

(ii) High Efficiency: If the knowledge base is updated with the correct knowledge, then it provides a highly efficient output, which may not be possible for a human.

(iii) Expertise in a domain: There are lots of experts in each domain. They all have different skills, experiences. So it is not easy to get a final output for the query. But if we put the knowledge gained from human experts into the expert system, then it provides an efficient output by mixing all the facts and knowledge.

(iv) Not affected by emotions: These systems are not affected by human emotions such as fatigue (थकान), anger, depression, anxiety, etc. Hence the performance remains constant.

(v) High ~~secu~~ security: These systems provide high security to resolve any query.

(vi) ~~Conds~~ Considers all the facts: To respond to any query, it checks and considers all the available facts and provides the result accordingly. But it is possible that a human expert may not consider some facts due to any reason.

(vii) Regular updates improve the performance:
If there is any issue in the result provided by the expert systems, we can improve the performance of the system by updating the knowledge base.

Capabilities of the Expert System:

(i) Advising: It is capable of advising the human being for the query of any domain from the particular ES.

(ii) Provide decision-making capabilities: It provides the capability of decision making in any domain, such as for making any financial decision, decising in medical science, etc.

(iii) Demonstrate a device: It is capable of demonstrating any new products such as it's features, specifications, how to use that product, etc.

(iv) Problem-solving: It has problem-solving capabilities.

(v) Explaining a problem: It is also capable of providing a detailed description of an input problem.

(vi) Interpreting the input: It is capable of interpreting the input given by the user.

(vii) Predicting results: It can be used for the prediction of a result.

(viii) Diagnosis: An ES designed for the medical field is capable of diagnosing a disease without using multiple components as it already contains various inbuilt medical tools.

Advantages of Expert System:

(i) These systems are highly reproducible.

(ii) They can be used for risky places where the human presence is not safe.

(iii) Error possibilities are less if the KB contains correct knowledge.

(iv) The performance remains steady as it is not affected by emotions, tension or fatigue.

(v) They provide a very high speed to respond to a particular query.

(#) Limitations of Expert Systems

(i) The response of the ES may get wrong if the knowledge base contains the wrong information.

(ii) Like human being, it can't produce a creative output for different scenarios.

(iii) Its maintenance and development costs are very high.

(iv) Knowledge acquisition for designing is much difficult.

(v) For each domain, we require a specific ES, which is one of the big limitations.

(vi) It can't learn from itself and hence requires manual updates.

Applications of Expert System:

(i) In designing and manufacturing domain: It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.

(ii) In the knowledge domain: These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.

(iii) In the finance domain: In the finance industries, it is used to detect any type of possible fraud, suspicious activity and advise bankers that if they should provide loans for business or not.

(iv) In the diagnosis and troubleshooting of devices: In medical diagnosis, the ES system is used and it was the first area where these systems were used.

(v) Planning and Scheduling: The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

(#) When a rule becomes fact?

=> When antecedent part becomes null, then a rule becomes fact.

Planning

(#) What is a plan?

Ans: A plan is considered a sequence of actions, and each action has it's preconditions that must be satisfied before it can act and some effects that can be positive or negative.

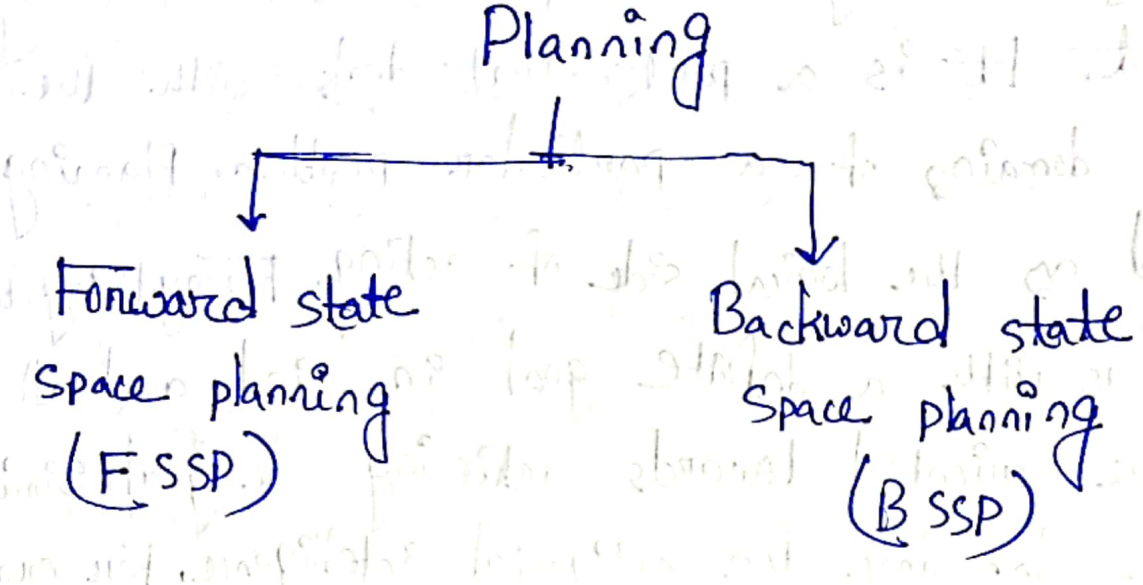
(#) What is planning in AI?

Ans: Planning in artificial intelligence is about decision making actions performed by robots or computer programs to achieve a specific goal.

(#) What is the role of planning in artificial intelligence?

Ans: Planning is very much important to make any AI project. It is a part that deals with the tasks and domains of a particular problem. Planning is considered as the logical side of acting. Everything we humans do is with a definite goal in mind and all our actions are oriented towards achieving our goal. Similarly, planning is also done for artificial intelligence. For example, planning is required to reach a particular destination. It is necessary to find the best route in planning, but the tasks to be done at a particular time and why they are done are also very important. That is why planning is considered the logical side of acting. In other words, planning is about deciding the tasks to be performed by the artificial intelligence system and the system's functioning under domain-independent conditions...

There are two types of planning in AI.



(#) FSSP:

It says that given an initial state S in any domain, we perform some necessary actions and obtain a new state S' (contains some new items), called a progression. It continues until we reach the target position. Action should be taken in this matter.

(*) Advantage: The algorithm is sound
(*) Disadvantage: Large branching factor

#) BSSP:

In this, we move from the target state g to the sub-goal g, tracing the previous action to achieve that goal. This process is called regression (going back to the previous goal or sub-goal). These sub-goals should also be checked for consistency. The action should be relevant in this case.

(*) Advantage: Small branching factor (much smaller than FSSP)

(*) Disadvantage: Not sound algorithm (sometimes inconsistency can be found)

#Block-world planning problem:

The block-world problem is known as the Sussmann anomaly. When two sub-goals, b1 and b2, are given, a non-interleaved planner either produces a plan for b1 that is combined with a plan for b2 or vice-versa.

In the block-world problem, three blocks labeled 'A', 'B' and 'C' are allowed to rest on a flat surface. The given condition is that only one block can be moved at a time to achieve the target. The start position and target position are shown in the following diagram:

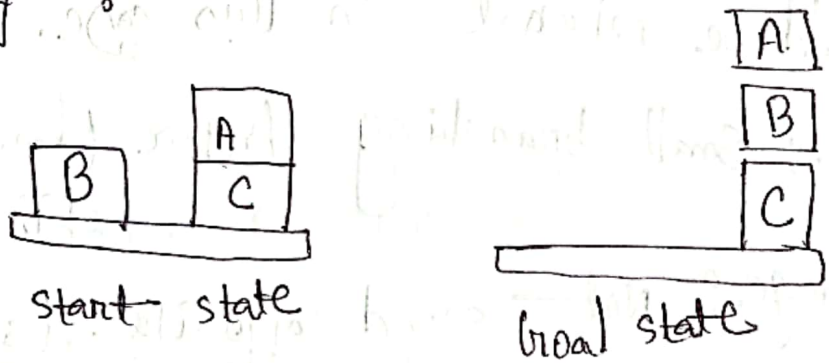


Fig: word planning problem

Components of the planning system:

The plan includes the following important steps:

(i) Choose the best rule to apply the next rule based on the best available guess.

(ii) Apply the chosen rule to calculate the new problem condition.

(iii) Find out when a solution has been found.

(iv) Detect dead ends so they can be discarded and direct system effort in more useful directions.

(v) Find out when a near-perfect solution is found.

(#) Target stack plan:

(i) It is one of the most important planning algorithms used by STRIPS.

(ii) Stacks are used in algorithms to capture the action and complete the target. A knowledge base is used to hold the ~~curren~~ current situation and actions.

(iii) A target stack is similar to a node in a search tree, where branches are created with a choice of action.

(#) The important steps of the algorithm are mentioned below:

(i) Start by pushing the original target onto the stack. Repeat this until the pile is empty. If the stack top is a mixed target, push its unsatisfied sub-targets onto the stack.

(ii) If the stack top is a single unsatisfied target, replace it with action and push the action precondition to the stack to satisfy the condition.

(iii) If the stack top is an action, pop it off the stack, execute it and replace the knowledge base with the action effect.

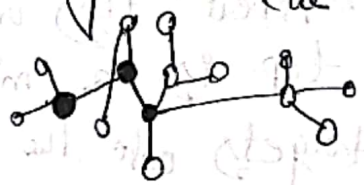
(iv) If the stack top is a satisfactory target, pop it off the stack.

Non-linear Planning:

This planning is used to set a goal stack and is included in the search space of all possible sub-goal orderings. It handles the goal interactions by the interleaving method.

Advantages:

It may be an optimal solution concering planning length (depending on the search strategy used).



(*) Disadvantages:

It takes a larger search space since all possible goal orderings are considered.

(#) Complex algorithm to understand:

Algorithm:

1. Choose a goal 'g' from the goal set
2. If 'g' does not match the state, then
   - Choose an operator 'O' whose add-list matches goal g
   - Push 'O' on the OpStack
   - Add the preconditions of 'O' to the goal set
3. While all preconditions of the operator on top of OpenStack are met in a state
   - Pop operator O from top of opstack
   - state = apply (O, state)
   - plan = [plan ; O]

Reasoning Under Uncertainty

Uncertainty:

Considering a situation where we are not sure about whether A is true or not then we can not ~~sure about~~ express a the related statement, this situation is called uncertainty.

Causes of uncertainty:

(i) Information occurred from unreliable sources.
(ii) Experimental Errors
(iii) Equipment fault
(iv) Temperature variation
(v) Climate change

Probabilistic reasoning:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge.

#3 Need of probabilistic reasoning in AI:

(i) When there are unpredictable outcomes.

(ii) When specifications or possibilities of predicates become too large to handle.

(iii) When an unknown error occurs during an experiment.

Probability: Probability can be defined as a chance that an uncertain event will occur.

$$\text{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

$P(\neg A)$ = Probability of a not happening event

$$P(\neg A) + P(A) = 1$$

Event - Each possible outcome of a variable is called an event.

Sample space - The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new information.

(#) Bayes' theorem:

Bayes' theorem is also known as Bayes rule, Bayes' law or Bayesian reasoning; which determines the probability of an event with uncertain knowledge. It relates the conditional probability and marginal probabilities of two random events. It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

From the product rule;

$P(A \wedge B) = P(A|B)\, P(B)$.

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

The above eqn is called as Bayes' rule.

$P(A|B)$ = Posterior $P(B|A)$ = likelihood

$P(A)$ = Prior probability

$P(B)$ = Marginal probability

Ex:1:

What is the probability that a patient has diseases meningitis with a stiff neck?

Soln: A doctor is aware that disease meningitis causes a patient to have a stiff neck and it occurs 80% of the time. He is also aware of some more facts; →

The known probability that a patient has meningitis disease is 1/30,000.

The known probability that a patient has a stiff neck is 2%.

Let;

a = Patient has stiff neck

b = patient has meningitis

$P(a|b) = 0.8$

$P(b) = 1/30,000$

$P(a) = 0.2$

$$\therefore P(b|a) = \frac{P(a|b)P(b)}{P(a)} = \frac{0.8 \times \frac{1}{30,000}}{0.02}$$

$$= 0.00133$$

Hence we can assume that 1 patient out of 750 patients has meaningitis disease with a stiff neck.

Ex:2:

From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King | Face), which means the drawn face card is a king card.

Solution:

$$P(King \mid Face) = \frac{P(Face \mid king) \times P(king)}{P(Face)}$$

P(King) = Probability that the card is king $= \frac{4}{52} = \frac{1}{13}$

$P(Face) = \frac{3}{13}$

P(Face | King) = 1 [Probability of face card when we assume it is a king = 1]

Putting all the values in the eqn; →

$$P(King \mid Face) = \frac{1 \times \left(\frac{1}{13}\right)}{\left(3/13\right)}$$

$= \frac{1}{3}$; it is the probability that a face card is a king card.

(#) Application of Bayes theorem:

(i) It is used to calculate the next step of the robot when already executed step is given.

(ii) Bayes theorem is helpful in weather forecasting.

(iii) It can solve the Monty Hall problem.