AI CAT 1 ANSWER

What do you mean by control strategies ? What are requirements of good control strategies ?

Control Strategy in Artificial Intelligence scenario is a technique or strategy, tells us about which rule has to be applied next while searching for the solution of a problem within problem space. It helps us to decide which rule has to apply next without getting stuck at any point. These rules decide the way we approach the problem and how quickly it is solved and even whether a problem is finally solved.

Control Strategy helps to find the solution when there is more than one rule or fewer rules for finding the solution at each point in problem space. A good Control strategy has two main characteristics:

Control Strategy should cause Motion

Each rule or strategy applied should cause the motion because if there will be no motion than such control strategy will never lead to a solution. Motion states about the change of state and if a state will not change then there be no movement from an initial state and we would never solve the problem.

Control strategy should be Systematic

Though the strategy applied should create the motion but if do not follow some systematic strategy than we are likely to reach the same state number of times before reaching the solution which increases the number of steps. Taking care of only first strategy we may go through particular useless sequences of operators several times. Control Strategy should be systematic implies a need for global motion (over the course of several steps) as well as for local motion (over the course of single step).

OR

The first requirement for a good control strategy is that it should cause motion

The second requirement for a good control strategy is that it should be systematic

Finally, it must be efficient in order to find a good answer

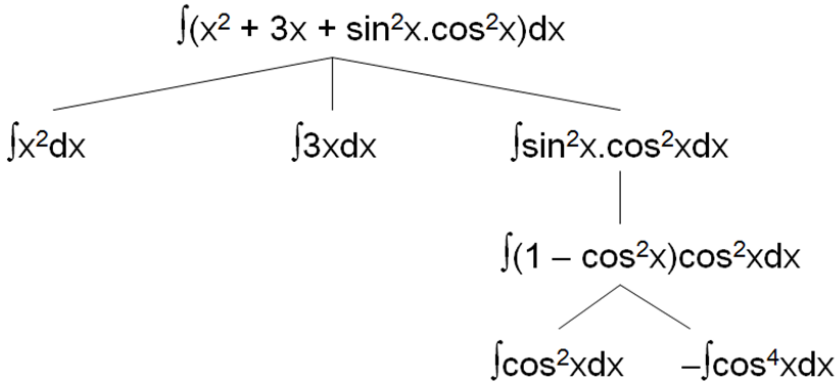
Write & explain various AI problem characteristics.

1. Is the problem decomposable into small sub-problems which are easy to solve?

Can the problem be broken down into smaller problems to be solved independently?

The decomposable problem can be solved easily.

Example: In this case, the problem is divided into smaller problems. The smaller problems are solved independently. Finally, the result is merged to get the final result.

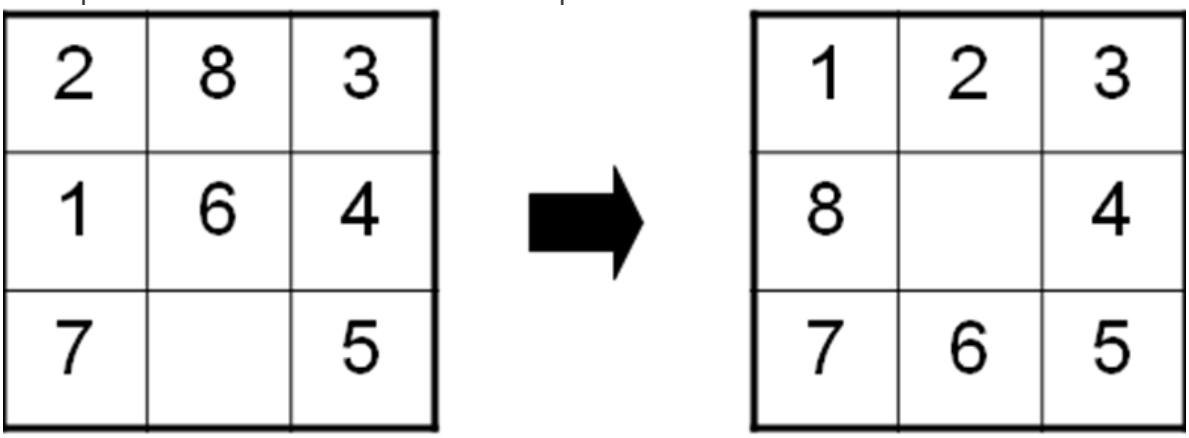


2. Can solution steps be ignored or undone?

In the Theorem Proving problem, a lemma that has been proved can be ignored for the next steps.

Such problems are called **Ignorable** problems.
In the 8-Puzzle, Moves can be undone and backtracked.

Such problems are called **Recoverable** problems.



In Playing Chess, moves can be retracted.

Such problems are called **Irrecoverable** problems.
Ignorable problems can be solved using a simple control structure that never backtracks. **Recoverable** problems can be solved using backtracking. **Irrecoverable** problems can be solved by recoverable style methods via planning.

3. Is the universe of the problem is predictable?

In Playing Bridge, We cannot know exactly where all the cards are or what the other players will do on their turns.

Uncertain outcome!

For **certain-outcome problems**, planning can be used to generate a sequence of operators that is guaranteed to lead to a solution.
For **uncertain-outcome problems**, a sequence of generated operators can only have a good probability of leading to a solution. Plan revision is made as the plan is carried out and the necessary feedback is provided.

4. Is a good solution to the problem is absolute or relative?

The Travelling Salesman Problem, we have to try all paths to find the shortest one.

Any path problem can be solved using heuristics that suggest good paths to explore.

For best-path problems, a much more exhaustive search will be performed.

5. Is the solution to the problem a state or a path

The Water Jug Problem, the path that leads to the goal must be reported.

A path-solution problem can be reformulated as a state-solution problem by describing a state as a partial path to a solution. The question is whether that is natural or not.

6. What is the role of knowledge in solving a problem using artificial intelligence?

Playing Chess
Consider again the problem of playing chess. Suppose you had unlimited computing power available. How much knowledge would be required by a perfect program? The answer to this question is very little—just the rules for determining legal moves and some simple control mechanism that implements an appropriate search procedure.
Additional knowledge about such things as good strategy and tactics could of course help considerably to constrain the search and speed up the execution of the program. Knowledge is important only to constrain the search for a solution.

Reading Newspaper

Now consider the problem of scanning daily newspapers to decide which are supporting the Democrats and which are supporting the Republicans in some upcoming election. Again assuming unlimited computing power, how much knowledge would be required by a computer trying to solve this problem? This time the answer is a great deal.

It would have to know such things as:

- The names of the candidates in each party.
- The fact that if the major thing you want to see done is have taxes lowered, you are probably supporting the Republicans.
- The fact that if the major thing you want to see done is improved education for minority students, you are probably supporting the Democrats.
- The fact that if you are opposed to big government, you are probably supporting the Republicans.
- And so on …

Knowledge is required even to be able to recognize a solution.

7. Does the task of solving a problem require human interaction?

Sometimes it is useful to program computers to solve problems in ways that the majority of people would not be able to understand.

This is fine if the level of the interaction between the computer and its human users is problem-in solution-out.

But increasingly we are building programs that require intermediate interaction with people, both to provide additional input to the program and to provide additional reassurance to the user.

The **solitary problem**, in which there is no intermediate communication and no demand for an explanation of the reasoning process.
The **conversational problem,** in which intermediate communication is to provide either additional assistance to the computer or additional information to the user.

What is production system ? Give the classification of production system. Explain it with examples.

A production system is based on a set of rules about behavior. These rules are a basic representation found helpful in expert systems, automated planning, and action selection. It also provides some form of artificial intelligence.

Production system or production rule system is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behavior but it also includes the mechanism necessary to follow those rules as the system responds to states of the world.

List various task domains of AI.

Artificial Intelligence tasks are divided into three groups, Mundane Tasks, Formal Tasks and Expert Tasks.

Mundane Tasks:

- Perception
- Vision
- Speech
- Natural Languages
- Understanding
- Generation
- Translation
- Common sense reasoning
- Robot Control

Formal Tasks

- Games: chess, checkers, etc
- Mathematics: Geometry, logic, Proving properties of programs

Expert Tasks:

- Engineering (Design, Fault finding, Manufacturing planning)
- Scientific Analysis
- Medical Diagnosis
- Financial Analysis

What is AI ? What are its related fields?

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving

Artificial Intelligence is the capability of a machine or computer device to emulate human intelligence (cognitive process), acquire from experiences, adapt to the latest information and operate humans-like-activities.

AI now consists many sub-fields, using a variety of techniques, such as:

Neural Networks – e.g. brain modelling, time series prediction, classification

Evolutionary Computation – e.g. genetic algorithms, genetic programming

Vision – e.g. object recognition, image understanding

Robotics – e.g. intelligent control, autonomous exploration

Expert Systems – e.g. decision support systems, teaching systems

Speech Processing– e.g. speech recognition and production

Natural Language Processing – e.g. machine translation

Planning – e.g. scheduling, game playing

Machine Learning – e.g. decision tree learning, version space learning

Most of these have both engineering and scientific aspects.

What are intelligent agents? Draw and explain generic architecture of intelligence agents.

An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.

Following are the main four rules for an AI agent:

- **Rule 1:** An AI agent must have the ability to perceive the environment.
- **Rule 2:** The observation must be used to make decisions.
- **Rule 3:** Decision should result in an action.
- **Rule 4:** The action taken by an AI agent must be a rational action.

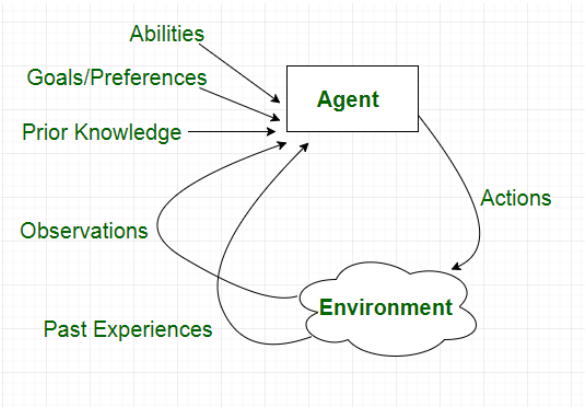
An intelligent agent is **a program that can make decisions or perform a service based on its environment, user input and experiences**.

To understand the structure of Intelligent Agents, we should be familiar with *Architecture* and *Agent* programs. **Architecture** is the machinery that the agent executes on. It is a device with sensors and actuators, for example, a robotic car, a camera, a PC. **Agent program** is an implementation of an agent function. An **agent function** is a map from the percept sequence(history of all that an agent has perceived to date) to an action.

Agent = Architecture + Agent Program

Examples of Agent:

- A **software agent** has Keystrokes, file contents, received network packages which act as sensors and displays on the screen, files, sent network packets acting as actuators.
- A Human-agent has eyes, ears, and other organs which act as sensors, and hands, legs, mouth, and other body parts acting as actuators.
- A **Robotic agent** has Cameras and infrared range finders which act as sensors and various motors acting as actuators.



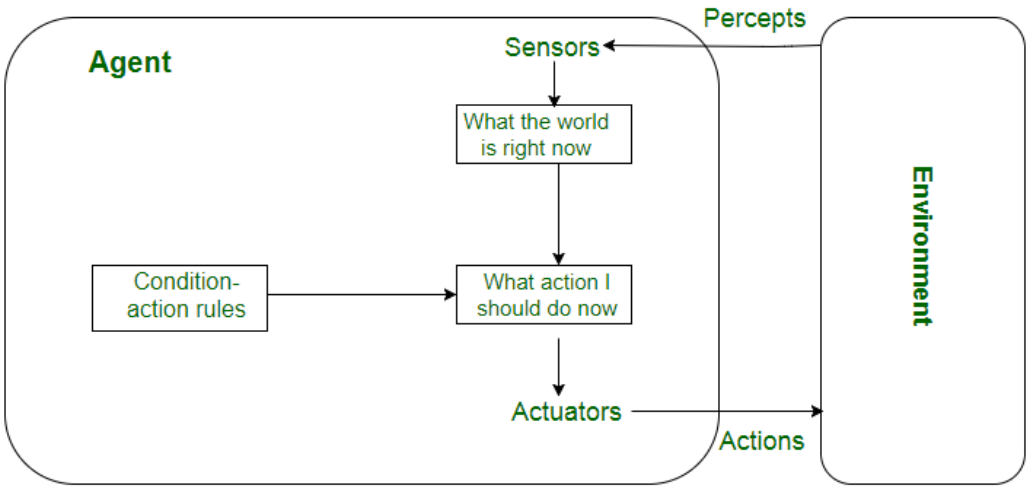
Types of Agents
Agents can be grouped into five classes based on their degree of perceived intelligence and capability :

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

Simple reflex agents

Simple reflex agents ignore the rest of the percept history and act only on the basis of the **current percept**. Percept history is the history of all that an agent has perceived to date. The agent function is based on the **condition-action rule**. A condition-action rule is a rule that maps a state i.e, condition to an action. If the condition is true, then the action is taken, else not. This agent function only succeeds when the environment is fully observable. For simple reflex agents operating in partially observable environments, infinite loops are often unavoidable. It may be possible to escape from infinite loops if the agent can randomize its actions.
Problems with Simple reflex agents are :

- Very limited intelligence.
- No knowledge of non-perceptual parts of the state.
- Usually too big to generate and store.
- If there occurs any change in the environment, then the collection of rules need to be updated.

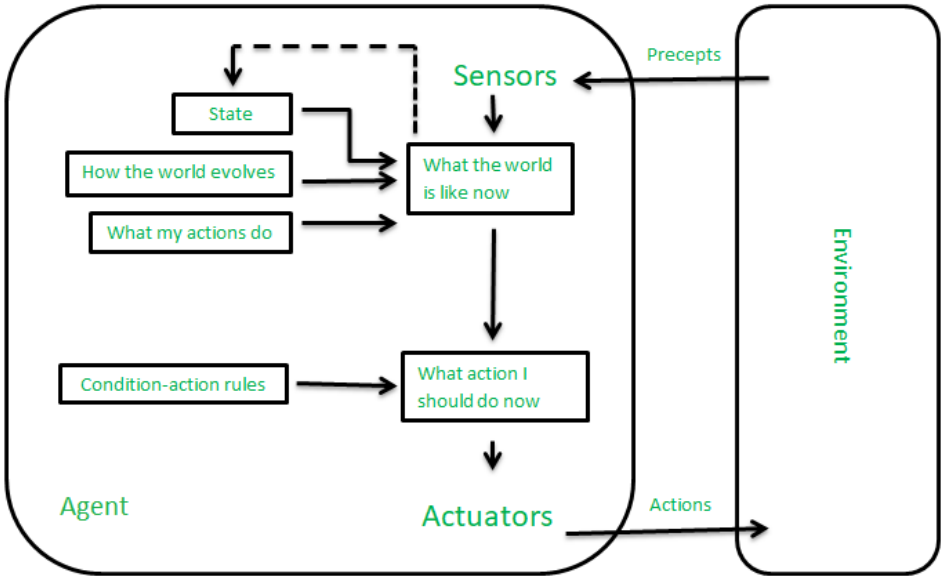


Model-based reflex agents

It works by finding a rule whose condition matches the current situation. A model-based agent can handle **partially observable environments** by the use of a model about the world. The agent has to keep track of the **internal state** which is adjusted by each

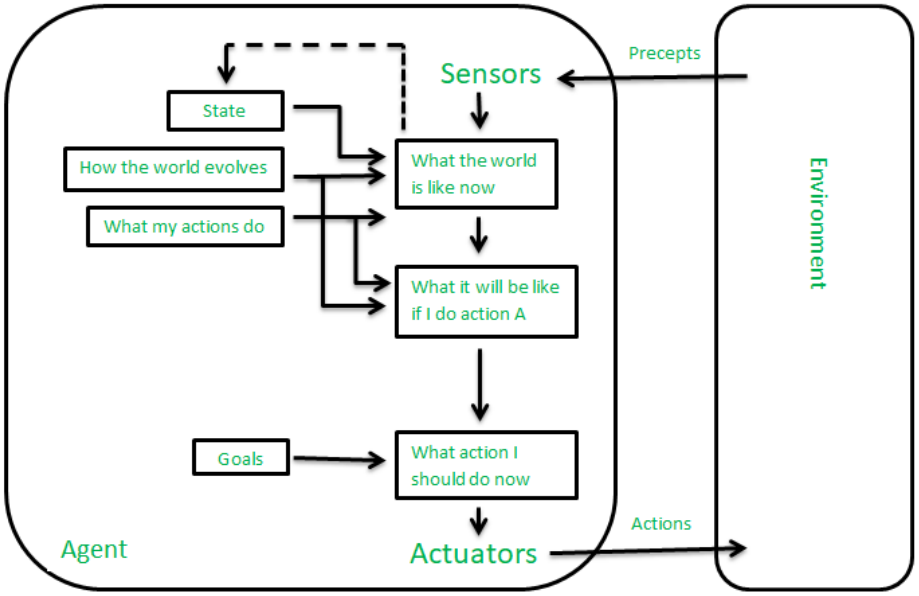
percept and that depends on the percept history. The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen.
Updating the state requires information about :

- how the world evolves independently from the agent, and
- how the agent's actions affect the world.



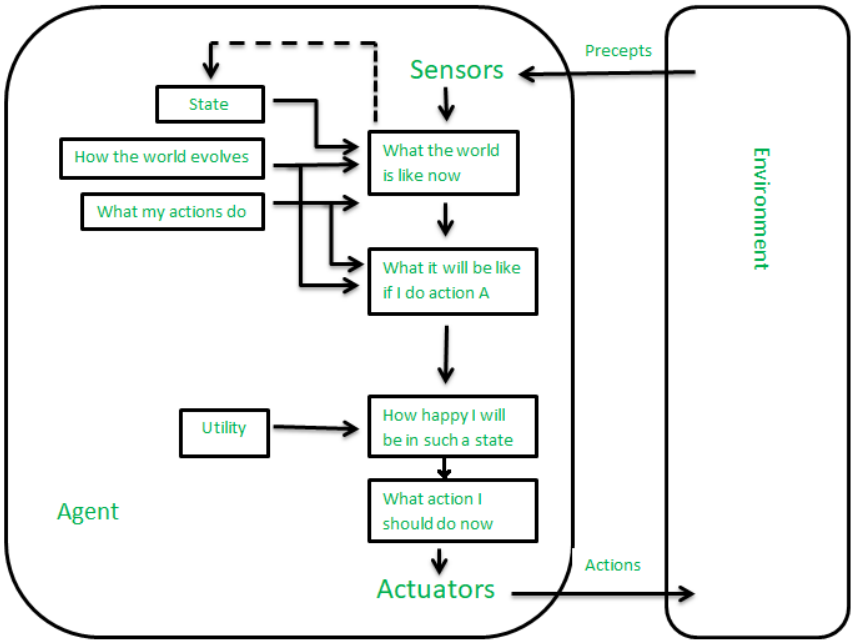
Goal-based agents

These kinds of agents take decisions based on how far they are currently from their **goal**(description of desirable situations). Their every action is intended to reduce its distance from the 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 and planning. The goal-based agent's behavior can easily be changed.



Utility-based agents

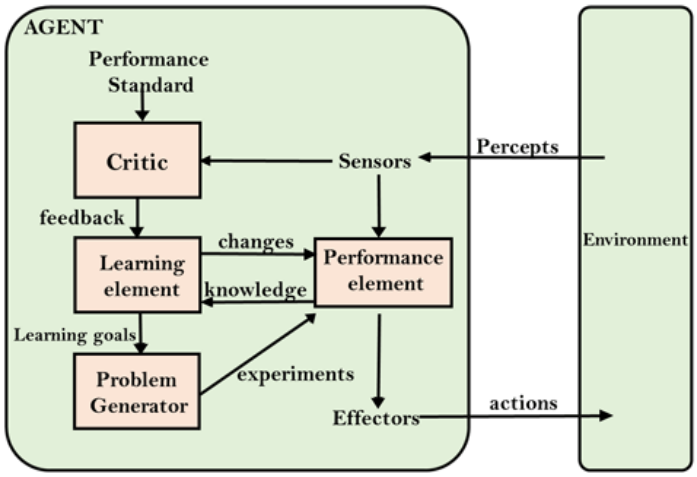
The agents which are developed having their end uses as building blocks are called utility-based agents. When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used. They choose actions based on a **preference (utility)** for each state. Sometimes achieving the desired goal is not enough. We may look for a quicker, safer, cheaper trip to reach a destination. Agent happiness should be taken into consideration. Utility 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.



Learning Agent :

A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning.
A learning agent has mainly four conceptual components, which are:

1. **Learning element:** It is responsible for making improvements by learning from the environment
2. **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.
3. **Performance element:** It is responsible for selecting external action
4. **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.



B)Explain the importance of defining the problem as a state space search. Give example.

A state-space defined as a set of all possible states of a problem. A State Space Search representation allows for the formal definition of a problem that makes the move from the initial state to the goal state.

"It is the question which is to be solved. For solving the problem it needs to be precisely defined. The definition means, defining the start state, goal state, other valid states and transitions".

A state space representation allows for the formal definition of a problem which makes the movement from initial state to the goal state quite easily. So we can say that various problems like planning, learning, theorem proving etc. are all essentially search problems only.

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

For Example:

The eight tile puzzle problem formulation

The eight tile puzzle consist of a 3 by 3 (3*3) square frame board which holds 8 movable tiles numbered 1 to 8. One square is empty, allowing the adjacent tiles to be shifted. The objective of the puzzle is to find a sequence of tile movements that leads from a starting configuration to a goal configuration.

For Example:

The eight tile puzzle problem formulation

The eight tile puzzle consist of a 3 by 3 (3*3) square frame board which holds 8 movable tiles numbered 1 to 8. One square is empty, allowing the adjacent tiles to be shifted. The objective of the puzzle is to find a sequence of tile movements that leads from a starting configuration to a goal configuration.

The states of 8 tile puzzle are the different permutations of the tiles within frame.

Lets do a standard formulation of this problem now.

States: It specifies the location of each of the 8 tiles and the blank in one of the nice squares.

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

Goal : Many goal configurations are possible one such is shown in the figure

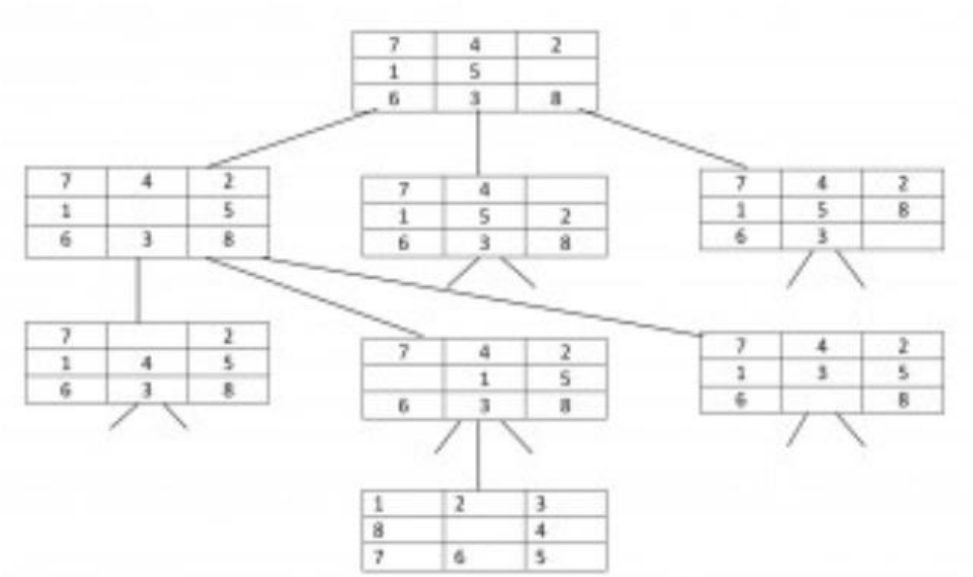
Legal moves (or state) : They generate legal states that result from trying the four actions-

Legal moves (or state) : They generate legal states that result from trying the four action

- Blank moves left
- Blank moves right
- Blank moves up
- Blank moves down

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

The tree diagram showing the search space is shown in figure.



B)Give production system rules or water jug problem. Also give the sequence of rules to solve water jug problem

There are two jugs of **volume A litre** and **B litre.** Neither has any **measuring mark** on it.There is a pump that can be used to fill the jugs with water.How can you get exactly **x litre** of water into the **A litre jug.**Assuming that we have unlimited supply of water.

Note: Let's assume we have **A=4 litre and B= 3 litre jugs**. And **we want exactly 2 Litre water into jug A (i.e 4 litre jug)** how we will do this.

Solution:

The state space for this problem can be described as the set of ordered **pairs of integers (x,y)**

Where,

x represents the quantity of water in the 4-gallon jug x= 0,1,2,3,4

y represents the quantity of water in 3-gallon jug y=0,1,2,3

Start State: (0,0)

Goal State: (2,0)

Generate production rules for the water jug problem

We basically perform three operations to achieve the goal.

1. **Fill water jug.**
2. **Empty water jug**
3. and **Transfer water jug**

Rule	***State***	***Process***
1	(X,Y | X<4)	(4,Y) {Fill 4-gallon jug}
2	(X,Y |Y<3)	(X,3) {Fill 3-gallon jug}
3	(X,Y |X>0)	(0,Y) {Empty 4-gallon jug}
4	(X,Y | Y>0)	(X,0) {Empty 3-gallon jug}
5	(X,Y | X+Y>=4 ^ Y>0)	(4,Y-(4-X)) {Pour water from 3-gallon jug into 4-gallon jug until 4-gallon jug is full}
6	(X,Y | X+Y>=3 ^X>0)	(X-(3-Y),3) {Pour water from 4-gallon jug into 3-gallon jug until 3-gallon jug is full}

7	(X,Y | X+Y<=4 ^Y>0)	(X+Y,0) {Pour all water from 3-gallon jug into 4-gallon jug}
8	(X,Y | X+Y <=3^ X>0)	(0,X+Y) {Pour all water from 4-gallon jug into 3-gallon jug}
9	(0,2)	(2,0) {Pour 2 gallon water from 3 gallon jug into 4 gallon jug}

Initialization:

Start State: (0,0)

Apply Rule 2:

Fill 3-gallon jug
Now the state is (x,3)

Iteration 1:

Current State: (x,3)

Apply Rule 7:

Pour all water from 3-gallon jug into 4-gallon jug

Now the state is (3,0)

Iteration 2:

Current State : (3,0)

Apply Rule 2:

Fill 3-gallon jug
Now the state is (3,3)

Iteration 3:

Current State:(3,3)

Apply Rule 5:

Pour water from 3-gallon jug into 4-gallon jug until 4-gallon jug is full

Now the state is (4,2)

Iteration 4:

Current State : (4,2)

Apply Rule 3:

Empty 4-gallon jug
Now state is (0,2)

Iteration 5:

Current State : (0,2)

Apply Rule 9:

Pour 2 gallon water from 3 gallon jug into 4 gallon jug

Now the state is (2,0)-- Goal Achieved.

Water Jug Solution using DFS (Depth First Search)

Explain generate & test search technique.

Generate and Test Search is a heuristic search technique based on Depth First Search with Backtracking which guarantees to find a solution if done systematically and there exists a solution. In this technique, all the solutions are generated and tested for the best solution. It ensures that the best solution is checked against all possible generated solutions.

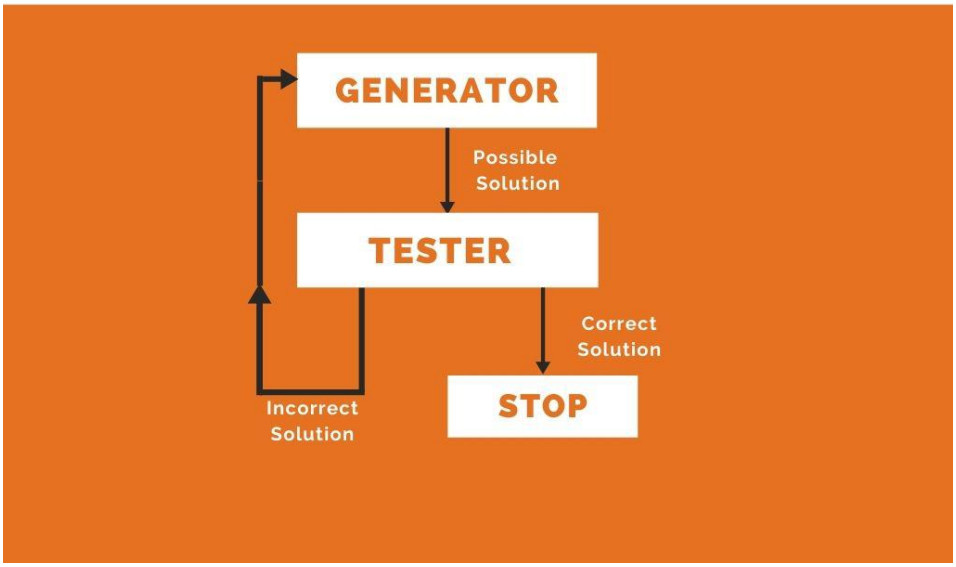
It is also known as British Museum Search Algorithm as it's like looking for an exhibit at random or finding an object in the British Museum by wandering randomly.

The evaluation is carried out by the heuristic function as all the solutions are generated systematically in generate and test algorithm but if there are some paths which are most unlikely to lead us to result then they are not considered. The heuristic does this by ranking all the alternatives and is often effective in doing so. Systematic Generate and Test may prove to be ineffective while solving complex problems. But there is a technique to improve in complex cases as well by combining generate and test search with other techniques so as to reduce the search space. For example in Artificial Intelligence Program DENDRAL we make use of two techniques, the first one is Constraint Satisfaction Techniques followed by Generate and Test Procedure to work on reduced search space i.e. yield an effective result by working on a lesser number of lists generated in the very first step.

Algorithm

1. *Generate a possible solution. For example, generating a particular point in the problem space or generating a path for a start state.*
2. *Test to see if this is a actual solution by comparing the chosen point or the endpoint of the chosen path to the set of acceptable goal states*
3. *If a solution is found, quit. Otherwise go to Step 1*

DIAGRAMMATIC REPRESENTATION

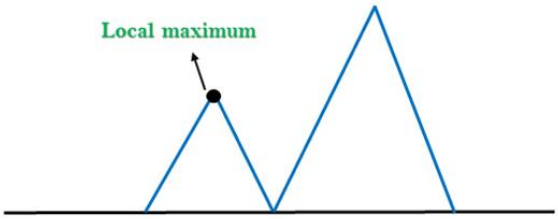


What are the problems of hill climbing? How they are overcome?

Problems in Hill Climbing Algorithm:

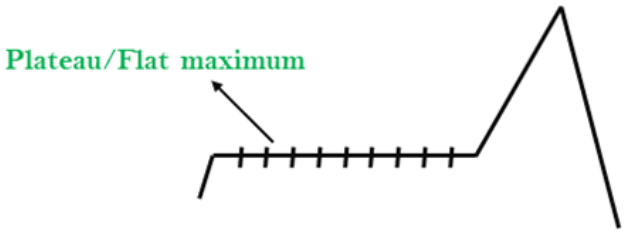
1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



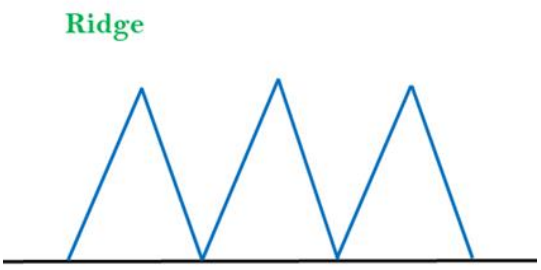
2. Plateau: A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Solution: The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



3. Ridges: A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Solution: With the use of bidirectional search, or by moving in different directions, we can improve this problem.



What is Hill Climbing Algorithm? What are its limitations? How these limitations are solved?

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and value.
- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

The standard version of hill climb has some limitations and often gets stuck in the following scenario:

- Local Maxima: Hill-climbing algorithm reaching on the vicinity a local maximum value, gets drawn towards the peak and gets stuck there, *having no other place to go*.
- Ridges: These are *sequences of local maxima*, making it difficult for the algorithm to navigate.
- Plateaux: This is a *flat state-space region*. As there is no uphill to go, algorithm often gets lost in the plateau.

To resolve these issues many variants of hill climb algorithms have been developed. These are most commonly used:

- **Stochastic Hill Climbing** selects at random from the uphill moves. The probability of selection varies with the steepness of the uphill move.
- **First-Choice Climbing** implements the above one by generating successors randomly until a better one is found.
- **Random-restart hill climbing** searches from randomly generated initial moves until the goal state is reached.

Explain best-first search with example

Ques-36) What is Best first search Algo.? Give its stepwise illustration with respect to suitable example.

→ (1) Thee best first search uses the concept of a priority queue and heuristic search.

(2) It is a search algorithm that works on a specific rule.

(3) The main aim is to reach the goal from the initial state via the shortest path.

(4) The BFS in AI is used for finding the shortest path from a given starting node to a goal node in a graph.

(5) The algo. works by expanding the nodes of the graph in order of increasing the distance from the starting node until the goal node is reached.

* Steps :→

Step 1 : place the starting node into the OPEN list

Step 2 : If the open list is empty, Stop and return failure.

Page:
Date:

Step 3 : Remove the node n, from the open list which has the lowest value of h(n) and places it in the closed list.

Step 4 : Expand the node n and generate the successors of node n.

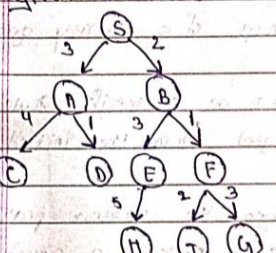
Step 5 : check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node the return success and terminate the search.

Step 6 : For each successor node, algo. checks for evaluation function f(n) and then check if the node has been in either open or closed list. If the node has not been in both list, then add it to the open list.

Step 7 : Return to step 2.

* Example :

Consider the below search problem and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n) = h(n), which is given in below table.



node	H(n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

Explain constraint satisfaction with some example.

What are various approaches to knowledge representation? Explain in detail.

→ There are mainly four approaches to knowledge representation which are given below:

① Simple relational knowledge :→

- Is the simplest way of storing facts which uses the relational methods, and each about a set of the object in set out systematically in columns.
- This approach has little opportunity for inference.
- Example:

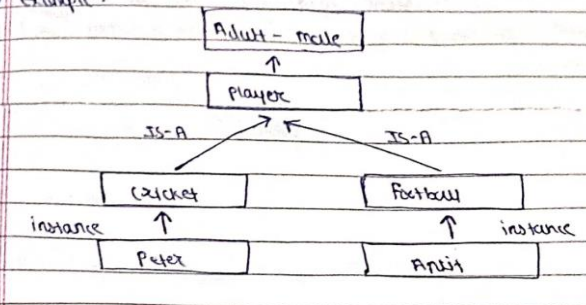
Player	weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

② Inheritable knowledge :→

- In this, all data must be stored into a hierarchy of classes.
- All classes should be arranged in generalized form.
- In this, we apply inheritance property.
- Every individual frame can represent the collection of attributes and its value.
- objects and values are represented in boxed nodes

Page:
Date:

- We use arrows which point from object to their values.
- Example:



③ Inferential knowledge :→

- Inferential knowledge approach represent knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guranteed correctness.
- Example:

Lets suppose there are two statements:

a) marcus is a man

b) All men are mortal

man (marcus)

$\forall x = man(x) \dashrightarrow mortal(x)$s

④ Procedural knowledge :→

- It is uses small program and codes which describes how to do specific things, and how to proceed.
- One imp. rule is used which is If-Then rule.
- We can easily represent heuristic or domain-specific knowledge using this approach.
- But it is not necessary that we can represent all case in this approach

short notes on i)Procedural & declarative knowledge ii) Forward & backward reasoning

S.NO	Procedural Knowledge	Declarative Knowledge
1.	It is also known as Interpretive knowledge.	It is also known as Descriptive knowledge.
2.	Procedural Knowledge means how a particular thing can be accomplished.	While Declarative Knowledge means basic knowledge about something.
3.	Procedural Knowledge is generally not used means it is not more popular.	Declarative Knowledge is more popular.
4.	Procedural Knowledge can't be easily communicate.	Declarative Knowledge can be easily communicate.
5.	Procedural Knowledge is generally process oriented in nature.	Declarative Knowledge is data oriented in nature.
6.	In Procedural Knowledge debugging and validation is not easy.	In Declarative Knowledge debugging and validation is easy.
7.	Procedural Knowledge is less effective in competitive programming.	Declarative Knowledge is more effective in competitive programming.

Procedural Knowledge:
Procedural Knowledge also known as Interpretive knowledge, is the type of knowledge in which it clarifies how a particular thing can be accomplished. It is not so popular because it is generally not used.
It emphasize **how to do** something to solve a given problem.
Let's see it with an example:

```
var a=[1, 2, 3, 4, 5];

var b=[];

for(var i=0;i<a.length;i++)

{

  b.push(a[i]);

}

console.log(b);
```

Output is:

```
[1, 2, 3, 4, 5]
```

Declarative Knowledge:
Declarative Knowledge also known as Descriptive knowledge, is the type of knowledge which tells the basic knowledge about something and it is more popular than Procedural Knowledge.
It emphasize **what to do** something to solve a given problem.
Let's see it with an example:

```
var a=[1, 2, 3, 4, 5];

var b=a.map(function(number)

{

 return number*1});

console.log(b);
```

Output is:

```
[1, 2, 3, 4, 5]
```

Forward Reasoning

- It is a data-driven task.
- It begins with new data.
- The object is to find a conclusion that would follow.
- It uses an opportunistic type of approach.
- It flows from incipient to the consequence.
- The inference engine searches the knowledge base with the given information depending on the constraints.
- The precedence of these constraints have to match the current state.
- The first step is that the system is given one or more constraints.
- The rules are searched for in the knowledge base for every constraint.
- The rule that fulfils the condition is selected.
- Every rule can produce new condition from the conclusion which is obtained from the invoked one.
- New conditions can be added, and are processed again.
- The step ends if no new conditions exist.
- It may be slow,
- It follows top-down reasoning.

Backward Reasoning

- It is a goal driven task.
- It begins with conclusions that are uncertain.
- The objective is to find the facts that support the conclusions.
- It uses a conservative type of approach.
- It flows from consequence to the incipient.
- The system helps choose a goal state, and reasons in a backward direction.
- First step is that the goal state and rules are selected.
- Sub-goals are made from the selected rule, which need to be satisfied for the goal state to be true.
- The initial conditions are set such that they satisfy all the sub-goals.
- The established states are matched to the initial state provided.
- If the condition is fulfilled, the goal is the solution.
- Otherwise the goal is rejected.
- It tests less number of rules.
- It provides small amount of data.
- It follows bottom-up reasoning technique.
- It contains less number of initial goals and has large number of rules.
- It is based on the decision fetched by the initial state.
- It is also known as a decision-driven or goal-driven inference technique.
- The system selects a goal state and reasons in the backward direction.

Write the difference between:-
i) Predicate logic and propositional logic
iii) Forward and Backward reasoning.

	Propositional Logic	Predicate Logic
1	Propositional logic is the logic that deals with a collection of declarative statements which have a truth value, true or false.	Predicate logic is an expression consisting of variables with a specified domain. It consists of objects, relations and functions between the objects.
2	It is the basic and most widely used logic. Also known as Boolean logic.	It is an extension of propositional logic covering predicates and quantification.
3	A proposition has a specific truth value, either true or false.	A predicate's truth value depends on the variables' value.
4	Scope analysis is not done in propositional logic.	Predicate logic helps analyze the scope of the subject over the predicate. There are three quantifiers : Universal Quantifier (∀) depicts for all, Existential Quantifier (∃) depicting there exists some and Uniqueness Quantifier (∃!) depicting exactly one.
5	Propositions are combined with Logical Operators or Logical Connectives like Negation(¬), Disjunction(∨), Conjunction(∧), Exclusive OR(⊕), Implication(⇒), Bi-Conditional or Double Implication(⇔).	Predicate Logic adds by introducing quantifiers to the existing proposition.
6	It is a more generalized representation.	It is a more specialized representation.
7	It cannot deal with sets of entities.	It can deal with set of entities with the help of quantifiers.

Forward Chaining	Backward chaining
Forward chaining suitable for breadth first search.	Backward chaining suitable for depth search.
It begins with initial facts.	It begins with some hypothesis goal.
It may slow, because in which we tested all the rules.	It may fast as compared to Forward chaining because it test fewer rules.
It provides small amount of data in which we use to store large amount of information.	It provides small amount of data in which we store small information.
It is basically on primarily data driven.	It is basically on goal driven.
It follows Top down reasoning.	It follows bottom-up reasoning.
It contains small number of initial states but large number of conclusion.	It contains small number of initial goals and large number of rules.
It is suitable for data collection problem like planning monitoring.	It is suitable for hypothesis problem like diagnosis.
In which all data is available.	In which data must be acquired.

Explain the following.

i) Scripts

ii) Frames

② Frame :⟶

① Frame is a record like structure which consist of a collection of attributes and its value to describe an entity in the world.

② Frame are the AI data structure which divides knowledge into substructure by representing stereotypes situations.

③ It consist of collections of slot and slot values. These slots may be of any type and sizes. Slot have names and values which are called fasets.

④ Facets are features of frame which enable us to put constraint on frames.

⑤ A frame is also known as slot-filler knowledge representation in AI.

⑥ Example :

Slots	Fillers
Name	Sahil
Profession	Student
Age	20
marital status	Single

① Scripts :→

→ ① A script is a structured representation describing a stereotyped sequence of events in a particular context.

② Script are used in natural language understanding systems to organize a knowledge based in terms of the situations that the system should understand.

③ A script is a structure that prescribes a set of circumstances that could be expected to follow on from one another.

④ The components of script includes :

a) Entry condition : These are basic condition which must be fulfilled before events in the script can occur.

b) Results : condition that will be true after events in script occured.

c) props : Slots representing objects involved in events.

d) Roles : These are the actions that the individual participant performs.

e) Track : Variations on the script.

f) Scenes : The sequence of events that occur.

⑤ Example :

script for going to the bank to withdraw money.

SCRIPT : withdraw money

TRACK : Bank

PROPS : money

counter

Form

Token

Roles : P = customer

E = Employee

C = cashier

Entry conditions : P has Rs 0 or less money

The bank is open

Result : P has more money.

Page:
Date:

Scene 1 : Entering

P PTRANS P into the Bank

P ATTEND eyes to E

P MOVE P to E

Scene 2 : Filling form

P MTRANS signal to E

E ATRANS form to P

P PROPEL form for writing

P ATRANS form to P

E ATRANS form to P.

Scene 3 : withdrawing money

P ATTEND eyes to counter

P PTRANS P to queue at counter

P PTRANS token to C

C ATRANS money to P

Scene 4 : Exiting the bank

P PTRANS to out of bank.

What is Semantic Network? Explain its advantages and disadvantages.

Ques. 5b) What is Semantic Network? Explain its advantages and disadvantages.

→ * Semantic Network :→

① A semantic Network is a graphical notation for representing knowledge in pattern of interconnected nodes.

② Semantic Network became popular in AI and natural language processing only because it represent knowledge or support reasoning.

③ It consist of nodes, links and link labels. Nodes appear in form of circle or even rectangle which represent object such as physical object, concepts or situation.

④ Links appear as arrow to express the relationship betn object, and links labels specify relations.

⑤ Semantic nets also refered as associative nets. as the nodes are associated with other nodes.

⑥ Semantic Network are majorly used for:
a) Representing data
b) Revealing structure
c) Supporting conceptual edition
d) supporting navigation.

* Advantages :→

① It is simple and comprehensible.
② Efficient in space requirement.
③ Easily cluster related knowledge.
④ It is flexible and easy to visualize
⑤ It is natural representation of knowledge.

* Disadvantages :→

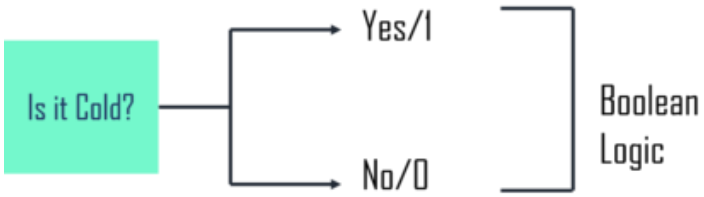
① Inheritance cause problem.
② link on objects represent only binary options.
③ Intractable for large domains.

Page:
Date:

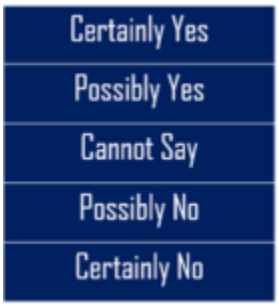
④ Don't represent performances or meta-knowledge effectively.
⑤ It's difficult to express some properties using semantic networks, like negation, disjunction etc..

Write short note on fuzzy logic.

Fuzzy Logic (FL) is a method of reasoning that resembles **human reasoning**. This approach is similar to how humans perform decision making. And it involves all intermediate possibilities between **YES** and **NO**.



The **conventional logic block** that a computer understands takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to a human being's YES or NO. The Fuzzy logic was invented by **Lotfi Zadeh** who observed that unlike computers, humans have a different range of possibilities between YES and NO, such as:



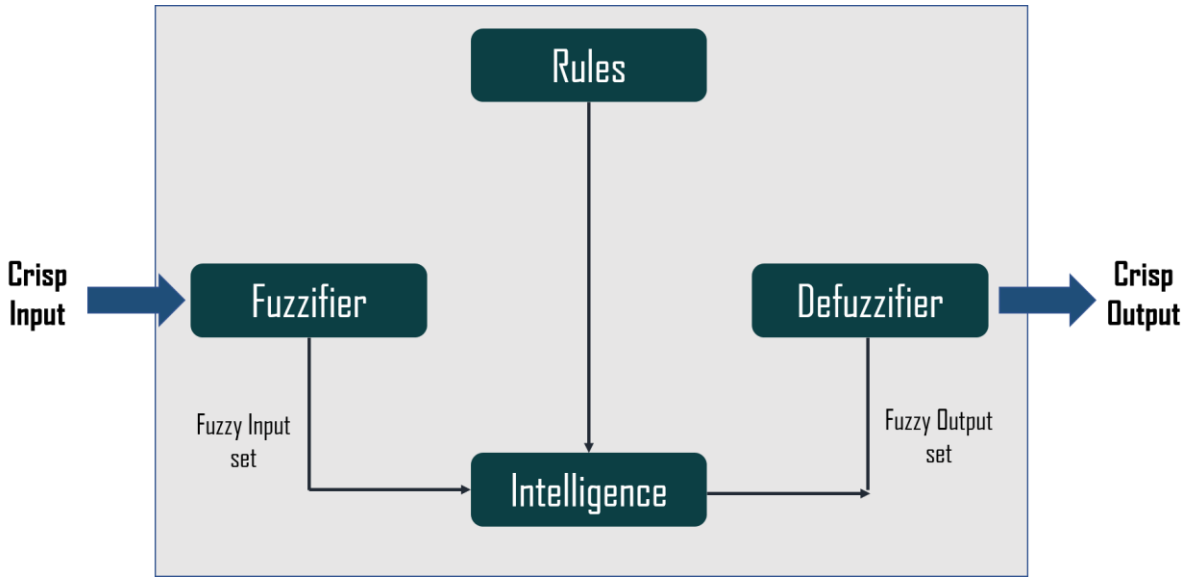
The Fuzzy logic works on the levels of possibilities of input to achieve a definite output. Now, talking about the implementation of this logic:

- It can be implemented in systems with different sizes and capabilities such as **micro-controllers, large networked** or **workstation-based systems.**
- Also, it can be implemented in **hardware, software** or a combination of **both**.

Why do we use Fuzzy Logic?

Generally, we use the fuzzy logic system for both commercial and practical purposes such as:

- It **controls machines** and **consumer products**
- If not accurate reasoning, it at least provides **acceptable reasoning**
- This helps in dealing with the **uncertainty in engineering**
- The fuzzy logic architecture consists of four main parts:



Applications of Fuzzy Logic

The Fuzzy logic is used in various fields such as automotive systems, domestic goods, environment control, etc. Some of the common applications are:

- It is used in the **aerospace field** for **altitude control** of spacecraft and satellite.
- This controls the **speed and traffic** in the **automotive systems.**
- It is used for **decision making support systems** and personal evaluation in the large company business.
- It also controls the pH, drying, chemical distillation process in the **chemical industry**.
- Fuzzy logic is used in **Natural language processing** and various intensive applications in Artificial Intelligence.
- It is extensively used in **modern control systems** such as expert systems.
- Fuzzy Logic mimics how a person would make decisions, only much faster. Thus, you can use it with Neural Networks.

Fuzzy logic provides simple reasoning similar to human reasoning. There are more such **advantages** of using this logic, such as:

- The structure of Fuzzy Logic Systems is **easy and understandable**
- Fuzzy logic is widely used for **commercial** and **practical purposes**
- It helps you to **control machines** and consumer products
- It helps you to deal with the **uncertainty in engineering**
- Mostly **robust** as no precise inputs required
- If the feedback sensor stops working, you can **program it** into the situation
- You can **easily modify** it to improve or alter system performance
- **Inexpensive sensors** can be used which helps you to keep the overall system cost and complexity low

These were the different advantages of fuzzy logic. But, it has some **disadvantages** as well:

fuzzy logic is **not always accurate**. So the results are perceived based on assumptions and may not be widely accepted

- It **cannot recognize** machine learning as-well-as neural network type patterns
- **Validation and Verification** of a fuzzy knowledge-based system needs **extensive testing** with hardware
- Setting exact, fuzzy rules and, membership functions is a **difficult task**
- At times, the fuzzy logic is **confused** with **probability theory**

Explain
Conceptual dependency.