Subject: Principles of Artificial Intelligence **Subject Code: AIML 207**
Branch: AIML **Semester:03**

QUESTION BANK

1. Define A.I or what is A.I?

Artificial intelligence is the branch of computer science that deals with the automation of intelligent behavior. AI gives basis for developing human like programs that can be useful to solve real life problems and thereby become useful to mankind.

2. What is meant by a robotic agent?

A machine that looks like a human being and performs various complex acts of a human being. It can do the task efficiently and repeatedly without fault. It works based on a program feeder; it can have previously stored knowledge from the environment through its sensors. It acts with the help of actuators.

3 Define an agent?

An agent is anything (a program, a machine assembly) that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

4 Define rational agent?

A rational agent is one that does the right thing. Here right thing is one that will cause agent to be more successful. That leaves us with the problem of deciding how and when to evaluate the agent's success.

5 Give the general model of learning agent?

Learning agent model has 4 components –

1) Learning element.
2) Performance element.
3) Critic
4) Problem Generator.

6 What is the role of agent program?

Agent program is important and central part of agent system. It drives the agent, which means that it analyzes date and provides probable actions agent could take.

An agent program is internally implemented as agent function.

An agent program takes input as the current percept from the sensor and returns an action to the effectors.

7 List down the characteristics of intelligent agent?

The IA must learn and improve through interaction with the environment. The IA must adapt online and in the real time

situation.
The IA must accommodate new problem-solving rules incrementally.
The IA must have memory which must exhibit storage and retrieval capabilities.

8 Define abstraction?

In AI the abstraction is commonly used to account for the use of various levels in detail in a given representation language or the ability to change from one level to another level to another while preserving useful properties. Abstraction has been mainly studied in problem solving, theorem solving

9 State the concept of rationality.

Rationality is the capacity to generate maximally successful behavior given the available information. Rationality also indicates the capacity to compute the perfectly rational decision given the initially available information. The capacity to select the optimal combination of computation – sequence plus the action, under the constraint that the action must be selected by the computation is also rationality.
Perfect rationality constrains an agent's actions to provide the maximum expectations of success given the information available.

10 What are the functionalities of the agent function?

Agent function is a mathematical function that maps each and every possible percept sequence to a possible action.
The major functionality of the agent function is to generate the possible action to each and every percept. It helps the agent to get the list of possible actions the agent can take. Agent function can be represented in the tabular form.

11 Define basic agent program?

The basic agent program is a concrete implementation of the agent function which runs on the agent architecture. Agent program puts bound on the length of percent sequence and considers only required percept sequences. Agent program implements the functions of percept sequence and action which are external characteristics of the agent.
Agent program takes input as the current percept from the sensor and return an action to the effectors (Actuators)

12 what are the four components to define a problem? Define them.

1. initial state: state in which agent starts in.
2. A description of possible actions: description of possible actions which are available to the agent.
3. The goal test: it is the test that determines whether a given state is goal state.
4. A path cost function: it is the function that assigns a numeric cost (value) to each path. The problem-solving agent is expected to choose a cost function that reflects its own performance measure.

Descriptive questions:

1. **Explain properties of environment**.

Properties of Environment

The environment has multifold properties −

1. Fully observable vs Partially Observable
2. Static vs Dynamic
3. Discrete vs Continuous
4. Deterministic vs Stochastic
5. Single-agent vs Multi-agent
6. Episodic vs sequential
7. Known vs Unknown
8. Accessible vs Inaccessible

1. Fully observable vs Partially Observable:

- If an agent sensor can sense or access the complete state of an environment at each point of time then it is **a fully observable** environment, else it is **partially observable**.
- A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world.
- An agent with no sensors in all environments then such an environment is called as **unobservable**.

2. Deterministic vs Stochastic:

- If an agent's current state and selected action can completely determine the next state of the environment, then such environment is called a deterministic environment.
- A stochastic environment is random in nature and cannot be determined completely by an agent.
- In a deterministic, fully observable environment, agent does not need to worry about uncertainty.

3. Episodic vs Sequential:

- In an episodic environment, there is a series of one-shot actions, and only the current percept is required for the action.
- However, in Sequential environment, an agent requires memory of past actions to determine the next best actions.

4. Single-agent vs Multi-agent

- If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment.

- However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment.
- The agent design problems in the multi-agent environment are different from single the agent environment.

5. Static vs Dynamic:

- If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment it is called a static environment.
- Static environments are easy to deal with because an agent does not need to continue looking at the world while deciding for an action.
- However for a dynamic environment, agents need to keep looking at the world at each action.
- Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment.

6. Discrete vs Continuous:

- If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment it is called a continuous environment.
- A chess game comes under a discrete environment as there is a finite number of moves that can be performed.
- A self-driving car is an example of a continuous environment.

7. Known vs Unknown

- Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.
- In a known environment, the results for all actions are known to the agent. While in unknown environment, agent needs to learn how it works in order to perform an action.
- It is quite possible that a known environment to be partially observable and an Unknown environment to be fully observable.

8. Accessible vs Inaccessible

- If an agent can obtain complete and accurate information about the state's environment, then such an environment is called an Accessible environment else it is called inaccessible.
- An empty room whose state can be defined by its temperature is an example of an accessible environment.
- Information about an event on earth is an example of Inaccessible environs

2. **What is an agent? Explain the basic kinds of agent programs.**

AGENT

Introduction

An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents.

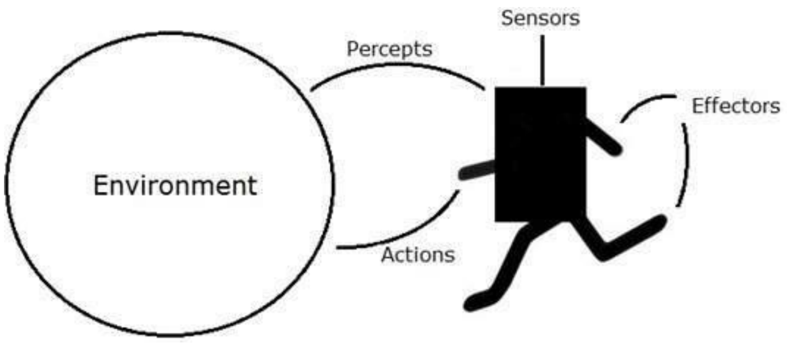
An **agent** is anything that can perceive its **environment** through **sensors** and acts upon that environment through **actuators.**

Sensor: Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.

Actuators: Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.

Effectors: Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.

- A **human agent** has sensory organs such as eyes, ears, nose, tongue and skin parallel to the sensors, and other organs such as hands, legs, mouth, for effectors.
- A **robotic agent** replaces cameras and infrared range finders for the sensors, and various motors and actuators for effectors.
- A **software** agent has encoded bit strings as its programs and actions.



Agent Terminology

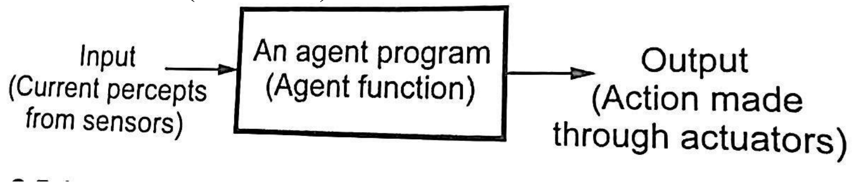
- **Performance Measure of Agent** − It is the criteria, which determines how successful an agent is.
- **Behavior of Agent** − It is the action that agent performs after any given sequence of percepts.
- **Percept** − It is agent's perceptual inputs at a given instance.
- **Percept Sequence** − It is the history of all that an agent has perceived till date.
- **Agent Function** − It is a map from the precept sequence to an action.

AI Agent Action Process:

Following diagram illustrated the agent action process, a specified by architecture.

Role of an Agent Program

- An agent program is internally implemented as agent function.
- An agent program take input as the current percept from the sensor and return an action to the effectors(Actuators)



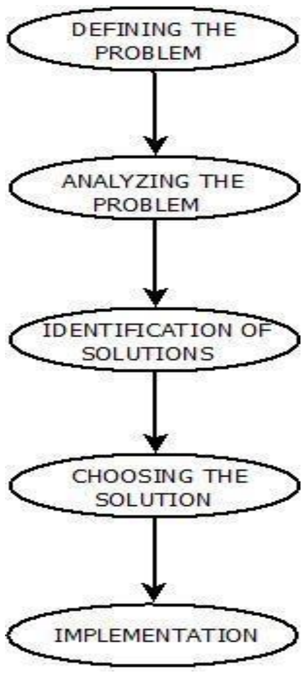
Agent Environment in AI

An environment is everything in the world which surrounds the agent, but it is not a part of an agent itself. An environment can be described as a situation in which an agent is present.

The environment is where agent lives, operate and provide the agent with something to sense and act upon it. An environment is mostly said to be non-feministic.

3. How a problem is formally defined? List down the components of it?

Problems are the issues which comes across any system. A solution is needed to solve that particular problem.
The process of solving a problem consists of five steps. These are:



Problem Solving in Artificial Intelligence

1. Defining The Problem: The definition of the problem must be included precisely. It should contain the possible initial as well as final situations which should result in acceptable solution.
2. Analyzing The Problem: Analyzing the problem and its requirement must be done as few features can immensely impact the resulting solution.
3. Identification Of Solutions: This phase generates a reasonable amount of solutions to the given problem in a particular range.
4. Choosing a Solution: From all the identified solutions, the best solution is chosen basis on the results produced by respective solutions.
5. Implementation: After choosing the best solution, its implementation is done.

Components of Planning System

The important components of planning are

Choosing the best rule to apply next based on the best variable available heuristic information.
Applying the chosen rule to compute the new problem state that arises from its application.
Detecting when a solution has been found.
Detecting dead ends so that they can be abandoned and the system‘s effort directed in correct direction.Repairing an almost correct solution.

1. Choosing rules to apply:

First isolate a set of differences between the desired goal state and the current state. Detect rules that are relevant to reduce the differences.
If several rules are found, a variety of heuristic information can be exploited to choose among them.This technique is based on the means end analysis method.

2. Applying rules:

Applying the rules is easy.

Each rule specifies the problem state that would result from its application.

We must be able to deal with rules that specify only a small part of the complete problem state.Different ways to do this are Describe, for each action, each of the changes it makes the state description.A state was described by a set of predicates representing the facts that were true in that state. Each state is represented as a predicate.The manipulation of the state description is done using a resolution theorem prover.

3. Detecting a solution

A planning system has succeeded in finding a solution to a problem when it has found a sequence of operators that transforms the initial problem state into the goal state.Any of the corresponding reasoning mechanisms could be used to discover when a solution has been found.

4. Detecting dead ends

As a planning system is searching for a sequence of operators to solve a particular problem, it must be able to detect when it is exploring a path that can never lead to a solution. The same reasoning mechanisms that can be used to detect a solution can often be used for detecting a dead end.

If the search process is reasoning forward from the initial state, it can prune any path that leads to a state from which the goal state cannot be reached.

If the search process is reasoning backward from the goal state, it can also terminate a path either because it is sure that the initial state cannot be reached or little progress is being made.

In reasoning backward, each goal is decomposed into sub goals. Each of them may lead to a set of additional sub goals. Sometimes it is easy to detect that there is now way that the entire sub goals in a given set can be satisfied at once. Other paths can be pruned because they lead nowhere.

5. Repairing an almost correct solution

Solve the sub problems separately and then combine the solution to yield a correct solution. But it leads to wasted effort.

The other way is to look at the situations that result when the sequence of operations corresponding to the proposed solution is executed and to compare that situation to the desired goal. The difference between the initial state and goal state is small. Now the problem solving can be called again and asked to find a way of eliminating a new difference. The first solution can then be combined with second one to form a solution to the original problem.

When information as possible is available, complete the specification in such a way that no conflicts arise. This approach is called least commitment strategy. It can be applied in a variety of ways.

- To defer deciding on the order in which operations can be performed.

- Choose one order in which to satisfy a set of preconditions, we could leave the order unspecified until the very end. Then we could look at the effects of each of the sub solutions to determine the dependencies that exist among them. At that point, an ordering can be chosen.

Two-mark questions:

1. How will you measure the problem-solving performance?

Problem solving performance is measured with 4 factors.

1) Completeness - Does the algorithm (solving procedure) surely finds solution if really the solution exists.
2) Optimality – If multiple solutions exits then do the algorithm returns optimal amongst them.
3) Time requirement.
4) Space requirement.

2. What is the application of BFS?

It is simple search strategy, which is complete i.e. it surely gives solution if solution exists. If the depth of search tree is small then BFS is the best choice. It is useful in tree as well as in graph search.

3. State on which basis search algorithms are chosen?

Search algorithms are chosen depending on two components.

1) How is the state space – That is, state space is tree structured or graph?
   Critical factor for state space is what is branching factor and depth level of that tree or graph.
2) What is the performance of the search strategy? A complete, optimal search strategy with better time and space requirement is critical factor in performance of search strategy.

4. Evaluate performance of problem-solving method based on depth-first search algorithm?

DFS algorithm performance measurement is done with four ways –

1) Completeness – It is complete (guarantees solution)
2) Optimality – it is not optimal.
3) Time complexity – It's time complexity is O (b).
4) Space complexity – its space complexity is O (b d+1).

5. What are the four components to define a problem? Define them?

The four components to define a problem are,

1) Initial state – it is the state in which agent starts in.
2) A description of possible actions – it is the description of possible actions which are available to the agent.
3) The goal test – it is the test that determines whether a given state is goal (final) state.
4) A path cost function – it is the function that assigns a numeric cost (value) to each path. The problem-solving agent is expected to choose a cost function that reflects its own performance measure.

6. State on what basis search algorithms are chosen?
Refer Question 3

7. Define the bi-directed search?
As the name suggests bi-directional that is two directional searches are made in this searching technique. One is the forward search which starts from initial state and the other is the backward search which starts from goal state. The two searches stop when both the search meet in the middle.

8. List the criteria to measure the performance of search strategies?
Refer Question 3

9. Why problem formulation must follow goal formulation?
Goal based agent is the one which solves the problem. Therefore, while formulating problem one need to only consider what is the goal to be achieved so that problem formulation is done accordingly. Hence problem formulation must follow goal formulation.

10. mention how the search strategies are evaluated?
Search strategies are evaluated on following four criteria

1. completeness: the search strategy always finds a solution, if one exits?
2. Time complexity: how much time the search strategy takes to complete?
3. Space complexity: how much memory consumption search strategy has?
4. Optimality: the search strategy finds a highest solution.

11. define admissible and consistent heuristics?
Admissible heuristics: a heuristic is admissible if the estimated cost is never more than actual cost from the current node to the goal node.
Consistent heuristics:
A heuristic is consistent if the cost from the current node to a successor node plus the estimated cost from the successor node to the goal is less than or equal to the estimated cost from the current node to the goal.

12. List some of the uninformed search techniques?
The uninformed search strategies are those that do not take into account the location of the goal. These algorithms ignore where they are going until they find a goal and report success. The three most widely used uninformed search strategies are
1.depth-first search-it expands the deepest unexpanded node 2. breadth-first search-it expands shallowest unexpanded node
3.lowest -cost-first search (uniform cost search)- it expands the lowest cost node

13. When is the class of problem said to be intractable?

The problems whose algorithm takes an unreasonably large amount of resources (time and / or space) are called intractable.
For example – TSP
Given set of 'N' points, one should find shortest tour which connects all of them. Algorithm will consider all N! Orderings, i.e. consider n = 16 ∴ 16! > 2^{50} which is impractical for any computer?

14. What is the power of heuristic search? or

Why does one go for heuristics search?

Heuristic search uses problem specific knowledge while searching in state space. This helps to improve average search performance. They use evaluation functions which denote relative desirability (goodness) of a expanding node set. This makes the search more efficient and faster. One should go for heuristic search because it has power to solve large, hard problems in affordable times.

15. What are the advantages of heuristic function?

Heuristics function ranks alternative paths in various search algorithms, at each branching step, based on the available information, so that a better path is chosen.
The main advantage of heuristic function is that it guides for which state to explore now, while searching. It makes use of problem specific knowledge like constraints to check the goodness of a state to be explained. This drastically reduces the required searching time.

16. State the reason when hill climbing often gets stuck?

Local maxima are the state where hill climbing algorithm is sure to get struck. Local maxima are the peak that is higher than each of its neighbor states, but lower than the global maximum. So we have missed the better state here. All the search procedure turns out to be wasted here. It is like a dead end.

17. When a heuristic function h is said to be admissible? Give an admissible heuristic function for TSP?

Admissible heuristic function is that function which never over estimates the cost to reach the goal state. It means that h(n) gives true cost to reach the goal state 'n'.
The admissible heuristic for TSP is

a. Minimum spanning tree.
b. Minimum assignment problem.

18. What do you mean by local maxima with respect to search technique?

Local maximum is the peak that is higher than each of its neighbor states, but lowers than the global maximum i.e. a local maximum is a tiny hill on the surface whose peak is not as high as the main peak (which is a optimal solution). Hill climbing fails to find optimum solution when it encounters local maxima. Any small move, from here also makes things worse (temporarily). At local maxima all the search procedure turns out to be wasted here. It is like a dead end.

19. How can we avoid ridge and plateau in hill climbing?

Ridge and plateau in hill climbing can be avoided using methods like backtracking, making big jumps. Backtracking and making big jumps help to avoid plateau, whereas, application of multiple rules helps to avoid the problem of ridges.

20. What is CSP?

CSP are problems whose state and goal test conform to a standard structure and very simple representation. CSPs are defined using set of variables and a set of constraints on those variables. The variables have some allowed values from specified domain. For example – Graph coloring problem.

21. How can minimax also be extended for game of chance?

In a game of chance, we can add extra level of chance nodes in game search tree. These nodes have successors which are the outcomes of random element.

The minimax algorithm uses probability P attached with chance node d_i based on this value. Successor function $S(N,d_i)$ give moves from position N for outcome d_i

Descriptive questions:

1. Discuss any 2 uninformed search methods with examples.

Breadth First Search (BFS)

Breadth first search is a general technique of traversing a graph. Breadth first search may use more memory but will always find the shortest path first. In this type of search the state space is represented in form of a tree. The solution is obtained by traversing through the tree. The nodes of the tree represent the start value or starting state, various intermediate states and the final state. In this search a queue data structure is used and it is level by level traversal. Breadth first search expands nodes in order of their distance from the root. It is a path finding algorithm that is capable of always finding the solution if one exists. The solution which is found is always the optional solution. This task is completed in a very memory intensive manner. Each node in the search tree is expanded in a breadth wise at each level.

Concept:

Step 1: Traverse the root node

Step 2: Traverse all neighbours of root node.

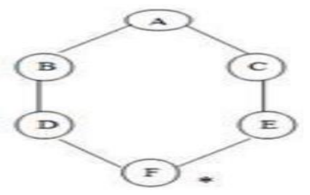
Step 3: Traverse all neighbours of neighbours of the root node.

Step 4: This process will continue until we are getting the goal node.

Algorithm:

Step 1: Place the root node inside the queue.

Step 2: If the queue is empty then stops and return failure.
Step 3: If the FRONT node of the queue is a goal node then stop and return success.
Step 4: Remove the FRONT node from the queue. Process it and find all its neighbours that are in readystate then place them inside the queue in any order.
Step 5: Go to Step 3.
Step 6: Exit.
Implementation:
Let us implement the above algorithm of BFS by taking the following suitable example.



Consider the graph in which let us take A as the starting node and F as the goal node (*)
Step 1: Place the root node inside the queue i.e. A
Step 2: Now the queue is not empty and also the FRONT node i.e. A is not our goal node. So move to step 3.
Step 3: So remove the FRONT node from the queue i.e. A and find the neighbour of A i.e. B and C B C A
Step 4: Now b is the FRONT node of the queue .So process B and finds the neighbours of B i.e. D. C D B
Step 5: Now find out the neighbours of C
i.e. E D B E
Step 6: Next find out the neighbours of D as D is the FRONT node of the queue E F D
Step 7: Now E is the front node of the queue. So the neighbour of E is F which is our goal node. F E
Step 8: Finally F is our goal node which is the FRONT of the queue. So exit. F
Advantages:
- In this procedure at any way it will find the goal.
- It does not follow a single unfruitful path for a long time. It finds the minimal solution in case of multiple paths.

Disadvantages:
- BFS consumes large memory space. Its time complexity is more.
- It has long pathways, when all paths to a destination are on

approximately the same search depth.

Depth First Search (DFS)

DFS is also an important type of uniform search. DFS visits all the vertices in the graph. This type of algorithm always chooses to go deeper into the graph. After DFS visited all the reachable vertices from a particular sources vertices it chooses one of the remaining undiscovered vertices and continues the search. DFS reminds the space limitation of breath first search by always generating next a child of the deepest unexpanded nodded. The data structure stack or last in
first out (LIFO) is used for DFS. One interesting property of DFS is that, the discover and finish time of each vertex from a parenthesis structure. If we use one open parenthesis when a vertex is finished then the result is properly nested set of parenthesis.

Concept:

Step 1: Traverse the root node.

Step 2: Traverse any neighbour of the root node.

Step 3: Traverse any neighbour of neighbour of the root node.

Step 4: This process will continue until we are getting the goal node.

Algorithm:

Step 1: PUSH the starting node into the stack.

Step 2: If the stack is empty then stop and return failure.

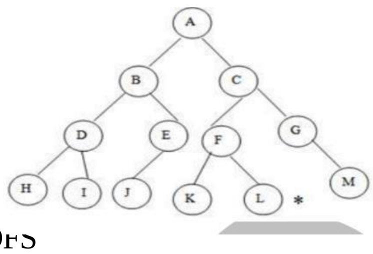
Step 3: If the top node of the stack is the goal node, then stop and
return success.

Step 4: Else POP the top node from the stack and process it. Find all
its neighbours that are in ready stateand PUSH them into the stack in
any order. Step 5: Go to step 3.

Step 6: Exit.

Implementation:

Let us take an example for implementing the above DFS algorithm.



Examples of DFS

Consider A as the root node and L as the goal node in the graph
figure Step 1: PUSH the starting node into the stack i.e.A

Step 2: Now the stack is not empty and A is not our goal node. Hence
move to next step.

Step 3: POP the top node from the stack i.e. A and find the neighbours of A
i.e. B and
C. B C A

Step 4: Now C is top node of the stack. Find its neighbours i.e. F
and G. B F G C

Step 5: Now G is the top node of the stack. Find its neighbour
i.e. M B F M G
Step 6: Now M is the top node and find its neighbour, but there is no
neighbours of M in the graph soPOP it from the stack. B F M
Step 7: Now F is the top node and its neighbours are K and L. so PUSH
them on to the stack. B K L F
Step 8: Now L is the top node of the stack, which is our goal
node. B K L
Advantages:

- DFSconsumes very less memory space.
- It will reach at the goal node in a less time period than BFS if it traverses in a right path.
-

Miscellaneous Problem-Solving Techniques:

1. **Enumerate the Classical "Water jug Problem". Describe the state space for this problem and also give the solution.**

Solution:

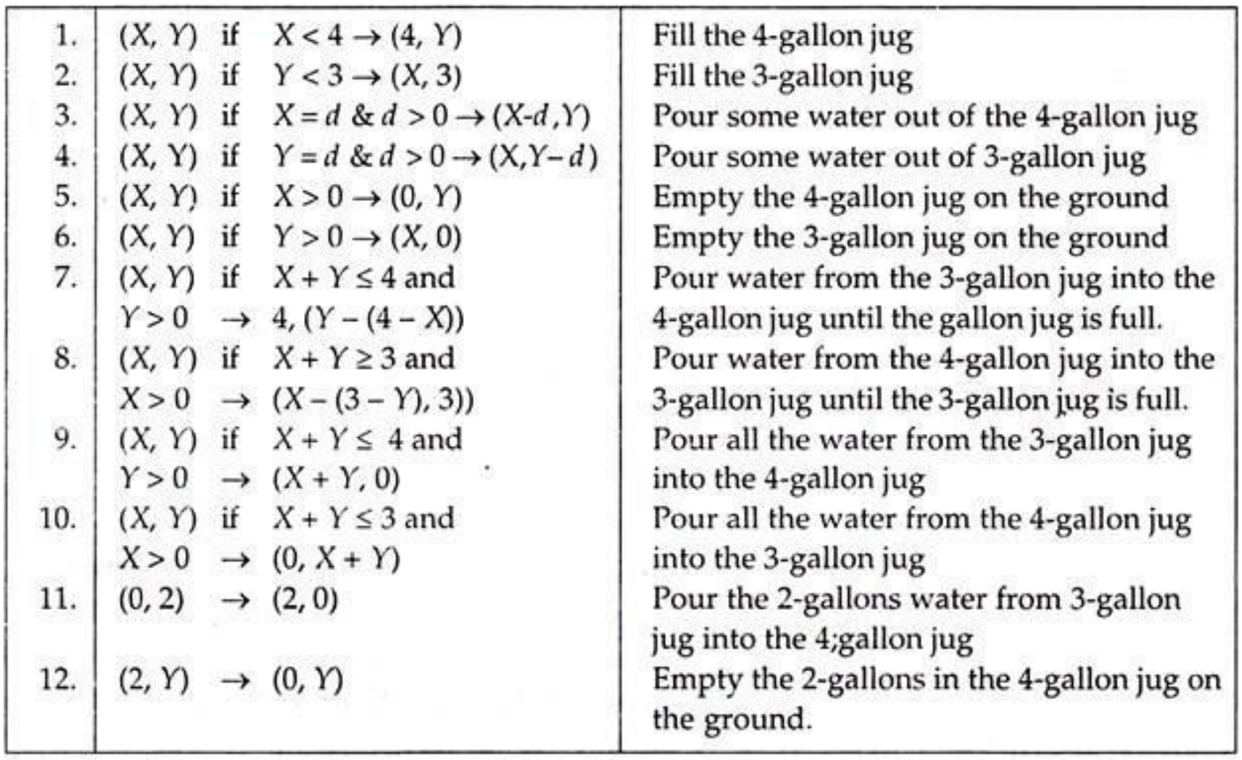
In order to show the generality of state space representation let us take a problem Water Jug Problem which is stated as:

We are given two jugs, a 4-gallon one and 3-gallon one. Neither has any measuring marked on it. There is a pump, which can be used to fill the jugs with water. How can we get exactly 2 gallons of water into 4-gallon jug?

The state space for this problem can be described as the set of ordered pairs of integers (X, Y) such that X = 0, 1, 2, 3 or 4 and Y = 0, 1, 2 or 3; X is the number of gallons of water in the 4-gallon jug and Y the quantity of water in the 3-gallon jug.

The start state is (0, 0) and the goal state is (2, n) for any value of n, as the problem does not specify how many gallons need to be filled in the 3-gallon jug (0, 1, 2, 3). So the problem has one initial state and many goal states. Some problems may have many initial states and one or many goal states.

The operators to be used to solve the problem can be described as shown in Fig. 2.3:



1.	(X, Y) if $X < 4 \rightarrow (4, Y)$	Fill the 4-gallon jug
2.	(X, Y) if $Y < 3 \rightarrow (X, 3)$	Fill the 3-gallon jug
3.	(X, Y) if $X = d \;\&\; d > 0 \rightarrow (X\text{-}d, Y)$	Pour some water out of the 4-gallon jug
4.	(X, Y) if $Y = d \;\&\; d > 0 \rightarrow (X, Y - d)$	Pour some water out of 3-gallon jug
5.	(X, Y) if $X > 0 \rightarrow (0, Y)$	Empty the 4-gallon jug on the ground
6.	(X, Y) if $Y > 0 \rightarrow (X, 0)$	Empty the 3-gallon jug on the ground
7.	(X, Y) if $X + Y \leq 4$ and $Y > 0 \rightarrow 4, (Y - (4 - X))$	Pour water from the 3-gallon jug into the 4-gallon jug until the gallon jug is full.
8.	(X, Y) if $X + Y \geq 3$ and $X > 0 \rightarrow (X - (3 - Y), 3))$	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full.
9.	(X, Y) if $X + Y \leq 4$ and $Y > 0 \rightarrow (X + Y, 0)$	Pour all the water from the 3-gallon jug into the 4-gallon jug
10.	(X, Y) if $X + Y \leq 3$ and $X > 0 \rightarrow (0, X + Y)$	Pour all the water from the 4-gallon jug into the 3-gallon jug
11.	$(0, 2) \rightarrow (2, 0)$	Pour the 2-gallons water from 3-gallon jug into the 4;gallon jug
12.	$(2, Y) \rightarrow (0, Y)$	Empty the 2-gallons in the 4-gallon jug on the ground.

Fig. 2.3. *Production rules (operators) for the water jug problem.*

As in chess playing they are represented as rules whose left side are matched against the current state and their right sides describe the new state which results from applying the rule.

In order to describe the operators completely here are some assumptions, not mentioned, in the problem state.

1. We can fill a jug from the pump.

2. We can pour water out a jug, onto the ground.

3. We can pour water out of one jug into the other.

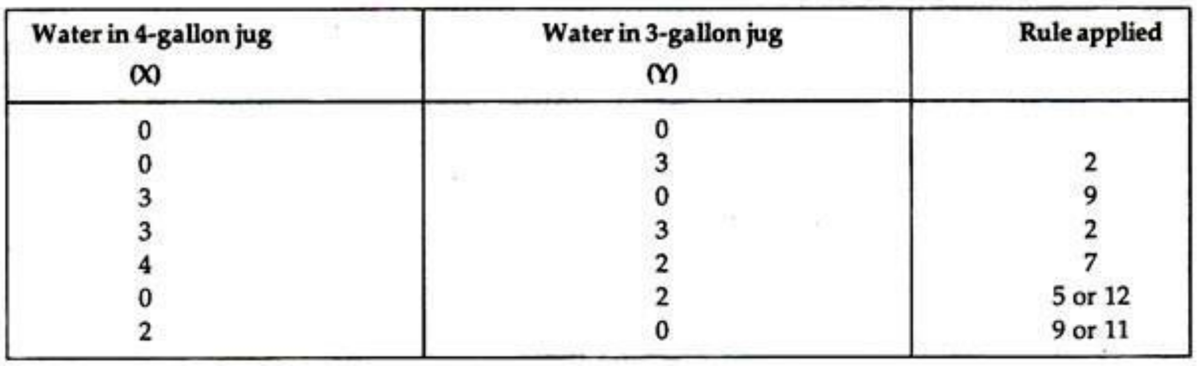
4. No other measuring devices are available.

All such additional assumptions need to be given when converting a problem statement in English to a formal representation of the problem, suitable for use by a program.

To solve the water jug problem, all we need, in addition to the problem description given above, is a control structure which loops through a simple cycle in which some rule whose left side matches the current state is chosen, the appropriate change to the state is made as described in the corresponding right side and the resulting state is checked to see if it corresponds to a goal state.

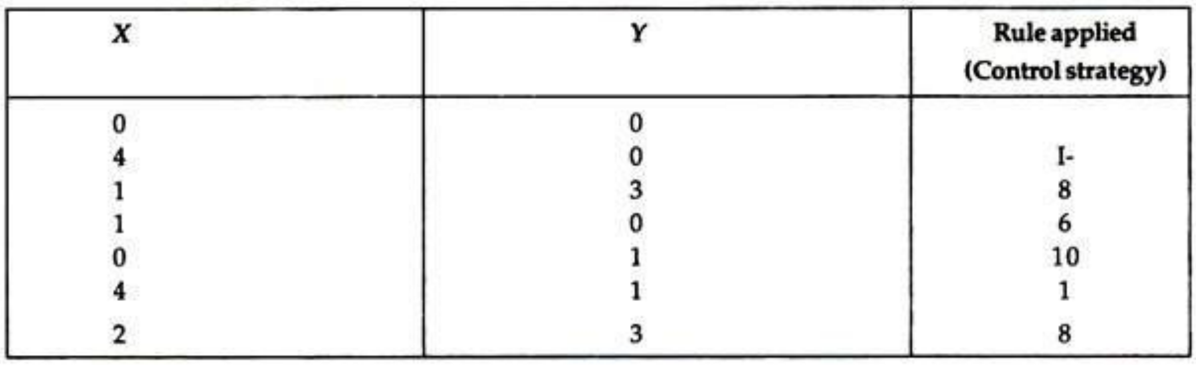
The loop continues as long as it does not lead to the goal. The speed with which the problem is solved depends upon the mechanism, control structure, which is used to select the next operation.

There are several sequences of operators which will solve the problem, two such sequences are shown in Fig. 2.4:



Water in 4-gallon jug (X)	Water in 3-gallon jug (Y)	Rule applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

Fig. 2.4 (*a*). *A solution to water jug problem.*

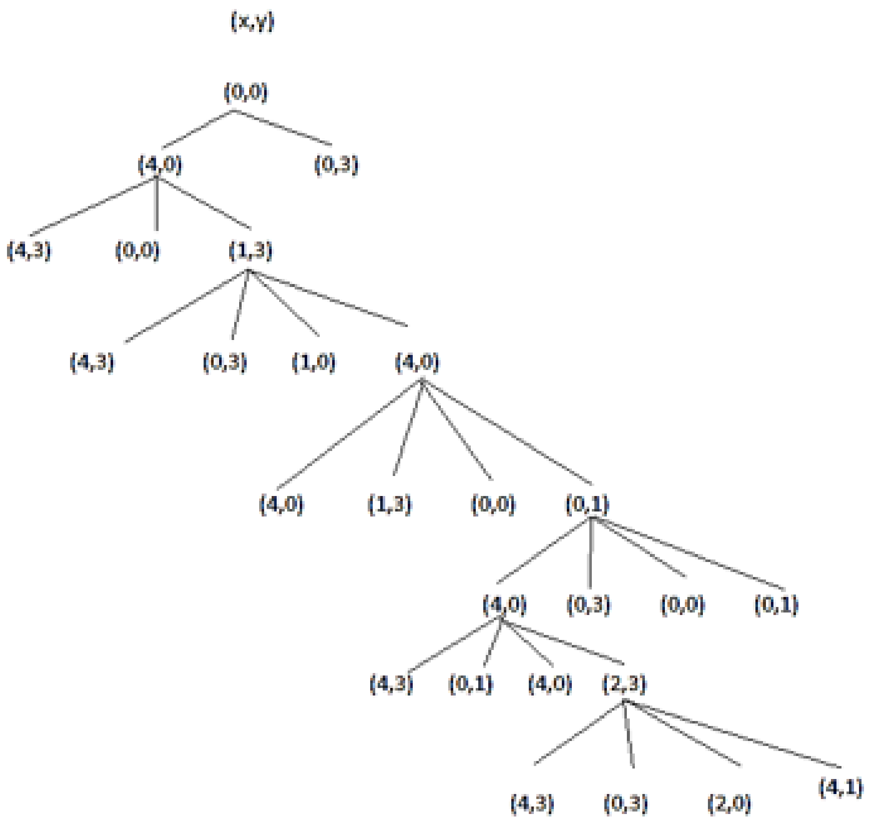


X	Y	Rule applied (Control strategy)
0	0	
4	0	I-
1	3	8
1	0	6
0	1	10
4	1	1
2	3	8

Fig. 2.4 (*b*). 2nd *solution to water jug problem.*

Thus, the state space representation forms the basis of most of the AI methods for problem solving.

State- Space Diagram of Water Jug Problem:



Issues in Water Jug Problem:

It has been shown above how an informal problem state (stated in English) has been converted into a formal one (Fig. 2.4) with the help of a water jug problem.

In doing so some issues which affect the approach towards the solution are:

1. The rules should be stated explicitly and not written because they are allowable. For example, the first rule states that "Fill the 4-gallon jug", but it should have been written as "if the 4-gallon jug is not already full, fill it" but the rule in its stated form is not wrong as there is no condition that the already filled jug cannot be filled (may be after emptying).

No doubt this is physically possible but not wise; as this won't change the problem state. In order to increase the efficiency of the problem solving program, it is imperative to encode some constraints in the left side of the rules so that the rules should lead to a solution that is the rules should be made more general.

2. Now consider the rules 3 and 4 should or should not these rules be included in the list of available operators? Emptying an unmeasured amount of water onto the ground is certainly allowed by the problem statement, but do these rules lead us near to a solution. The answer is 'no' so these can be ignored. So, such rules which are really applicable to the problem in arriving at the solution should be considered.

3. The rules 11 and 12 are special purpose rules, written to capture special purpose knowledge to solve this problem. This is clear from the above example solution. Consider

the last two steps of the solution in Fig. 2.4 (a), once the state (4, 2) is reached it is obvious what to do next? The desired 2-gallons have been produced, but they are in a wrong jug.

Move 2-gallons of water to 4-gallon jug-rule 11 (or rule 9). But before that the water already in 4-gallon jug need to be emptied, rule 12 (or rule 5). These two special purpose rules do not help in solving the problem since the operators they describe are already provided by rules 9 and 5 respectively, so could have been avoided. Sometimes special rules improve performance if preference is given to special case rules.

To formalize a problem the following steps are needed:

(i) Define the problem precisely, giving the specification of what the initial situation (s) and the final situation (s) will be.

(ii) Analyze the problem because a few important features can have immense impact on the suitability of different techniques available for solving the problem.

(iii) Represent the knowledge completely, which is necessary to solve problem in a given domain.

(iv) Choose the best technique(s) and apply it (them) to the particular problem.

Depending upon the control strategy (Fig. 2.4) to be used the performance of problem solving procedure can be improved or degraded. It is thus clear that to create a program for solving a problem simply the formal description of the problem has to be generated using the knowledge about the given problem. This process is called operationalization.

Water jug problem is a simple illustration of solving a problem through state space search. But many difficult problems such as understanding of natural Language which need to be solved by the AI techniques, the water jug problem can act as a strong basis for such tedious problems. This was done in the case of ELIZA, an early AI program.

Summarizingly any problem can be solved by the following series of step:

1. Define a state space which contains all the possible configurations of the relevant objects and even some impossible ones.

2. Specify one or more states within that space which would describe possible situations from which the problem solving process may start. These states are called the initial states.

3. Specify one or more states which would be acceptable as solutions to the problem. These states are called goal states.

4. Specify a set of rules that describe the actions (operators) available and a control strategy to decide the order of application of these rules.

The problem can be solved by collecting all knowledge of the problem, (in the present case, water jug problem) with the help of production rules and using an appropriate control strategy; move through the problem space until a path from an initial state to goal state is found.

2. Define the following problems. What types of control strategy is used in the following problem. I. The Tower of Hanoi
   II. Crypto-arithmetic
   III. The Missionaries and cannibals problems
   IV. 8-puzzle problem
   V. N-queen Problem

Solution:

I. The Tower of Hanoi:
   Tower of Hanoi (TOH) TOH is a mathematical puzzle that consists of three pegs named as origin, intermediate and destination and more than one disks. These disks are of different sizes and the smaller one sits over the larger one. In this problem we transfer all disks from origin peg to destination peg using intermediate peg for temporary storage and move only one disk at a time.