Concept of Intelligence

Intelligence is defined as mental capability that involves the ability to reason, to plan, to solve problems, to think abstractly, to comprehend complex ideas, to learn quickly and to learn from experience. It is not merely book learning, a narrow academic skill, or test-taking smartness.

In simple words, intelligence is nothing but thinking skills and the ability to adapt to and to learn from life's everyday experiences.

The intelligence is intangible. It is composed of −

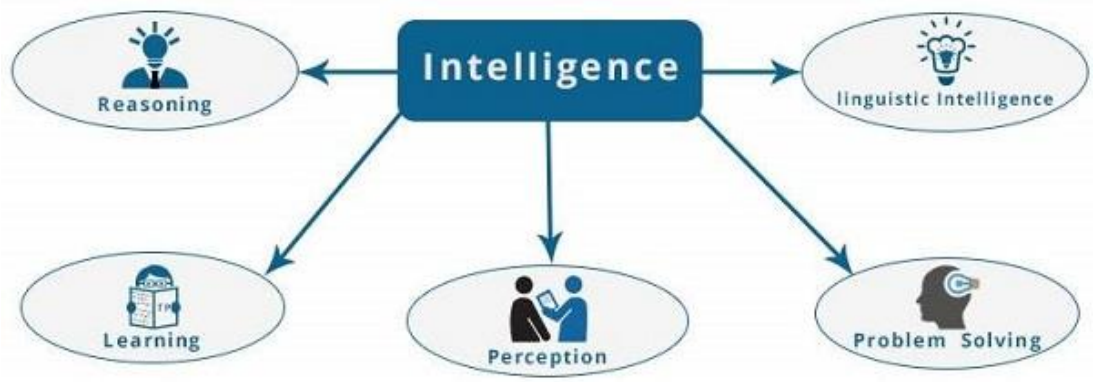
Reasoning

Learning

Problem Solving

Perception

Linguistic Intelligence



Artificial Intelligence

According to the father of Artificial Intelligence **John McCarthy**, it is "The science and engineering of making intelligent machines, especially intelligent computer programs".

Artificial Intelligence is a way of making a computer, a computer-controlled robot, or a software think intelligently, in the similar manner the intelligent humans think.

Artificial intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans. Some of the activities computers with artificial intelligence are designed for include:

- Speech recognition
- Learning
- Planning
- Problem solving

Applications of AI

AI has been dominant in various fields such as −

1. **Gaming** − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.
2. **Natural Language Processing** − It is possible to interact with the computer that understands natural language spoken by humans.
3. **Expert Systems** − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.
4. **Vision Systems** − These systems understand, interpret, and comprehend visual input on the computer. For example,
    - A spying aero plane takes photographs, which are used to figure out spatial information or map of the areas.
    - Doctors use clinical expert system to diagnose the patient.
    - Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.
5. **Speech Recognition** − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.
6. **Handwriting Recognition** − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.
7. **Intelligent Robots** − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

Turing Test in Artificial Intelligence

Alan Turing's 1950 article Computing Machinery and Intelligence discussed conditions for considering a machine to be intelligent. He argued that if the machine could successfully pretend to be human to a knowledgeable observer then you certainly should consider it intelligent. This test would satisfy most people but not all philosophers. The observer could interact with the machine and a human by teletype (to avoid requiring that the machine imitate the appearance or voice of the person), and the human would try to persuade the observer that it was human and the machine would try to fool the observer. The Turing test is a one-sided test. A machine that passes the test should certainly be considered intelligent, but a machine could still be considered intelligent without knowing enough about humans to imitate a human.

Search Techniques

Search is the process of inspecting several such sequences and choosing one that achieves the goal. Searching is the universal technique of problem solving in AI.

Artificial Intelligence is the study of building agents that act rationally. Most of the time, these agents perform some kind of search algorithm in the background in order to achieve their tasks.

- A search problem consists of:
    - A State Space. Set of all possible states where you can be.
    - A Start State. The state from where the search begins.
    - A Goal Test. A function that looks at the current state returns whether or not it is the goal state.
- The Solution to a search problem is a sequence of actions, called the plan that transforms the start state to the goal state.
- This plan is achieved through search algorithms.

State Spaces

One general formulation of intelligent action is in terms of **state space**. A **state** contains all of the information necessary to predict the effects of an action and to determine if it is a goal state. State-space searching assumes that

- the agent has perfect knowledge of the state space and can observe what state it is in (i.e., there is full observability);
- the agent has a set of actions that have known deterministic effects;
- some states are goal states, the agent wants to reach one of these goal states, and the agent can recognize a goal state; and
- a **solution** is a sequence of actions that will get the agent from its current state to a goal state.

A **state-space problem** consists of

- a set of states;
- a distinguished set of states called the **start states**;
- a set of actions available to the agent in each state;
- an **action function** that, given a state and an action, returns a new state;
- a set of goal states, often specified as a Boolean function, *goal(s)*, that is true when *s* is a goal state; and

- a criterion that specifies the quality of an acceptable solution. For example, any sequence of actions that gets the agent to the goal state may be acceptable, or there may be costs associated with actions and the agent may be required to find a sequence that has minimal total cost. This is called an **optimal** solution. Alternatively, it may be satisfied with any solution that is within 10% of optimal.

Production rules

The production rules operate on the global database. Each rule has a precondition that is either satisfied or not by the database. If the precondition is satisfied, the rule can be applied. Application of the rule changes the database. The control system chooses which applicable rule should be applied and ceases computation when a termination condition on the database is satisfied. If several rules are to fire at the same time, the control system resolves the conflicts.

Problem Characteristics

A problem may have different aspects of representation and explanation. In order to choose the most appropriate method for a particular problem, it is necessary to analyze the problem along several key dimensions. Some of the main key features of a problem are given below.

- Is the problem decomposable into set of sub problems?
- Can the solution step be ignored or undone?
- Is the problem universally predictable?
- Is a good solution to the problem obvious without comparison to all the possible solutions?
- Is the desire solution a state of world or a path to a state?
- Is a large amount of knowledge absolutely required to solve the problem?
- Will the solution of the problem required interaction between the computer and the person?

The above characteristics of a problem are called as 7-problem characteristics under which the solution must take place.

Production System and Its Characteristics

The production system is a model of computation that can be applied to implement search algorithms and model human problem solving. Such problem solving knowledge can be packed up in the form of little quanta called productions. A production is a rule consisting of a situation recognition part and an action part. A production is a situation-action pair in which the left side is a list of things to watch for and the right side is a list of things to do so. When productions are used in deductive systems, the situation that trigger productions are specified combination of facts. The actions are restricted to being

assertion of new facts deduced directly from the triggering combination. Production systems may be called premise conclusion pairs rather than situation action pair.

A production system consists of following components.

(a) A set of production rules, which are of the form A→B. Each rule consists of left hand side constituent that represent the current problem state and a right hand side that represent an output state. A rule is applicable if its left hand side matches with the current problem state.

(b) A database, which contains all the appropriate information for the particular task. Some part of the database may be permanent while some part of this may pertain only to the solution of the current problem.

(c) A control strategy that specifies order in which the rules will be compared to the database of rules and a way of resolving the conflicts that arise when several rules match simultaneously.

(d) A rule applier, which checks the capability of rule by matching the content state with the left hand side of the rule and finds the appropriate rule from database of rules.

The important roles played by production systems include a powerful knowledge representation scheme.

A production system not only represents knowledge but also action. It acts as a bridge between AI and expert systems. Production system provides a language in which the representation of expert knowledge is very natural. We can represent knowledge in a production system as a set of rules of the form

If (condition) THEN (condition)

along with a control system and a database. The control system serves as a rule interpreter and sequencer.

The database acts as a context buffer, which records the conditions evaluated by the rules and information on which the rules act. The production rules are also known as condition – action, antecedent –consequent, pattern – action, situation – response, feedback – result pairs.

For example,

If (you have an exam tomorrow)

THEN (study the whole night)

The production system can be classified as monotonic, non-monotonic, partially commutative and commutative.

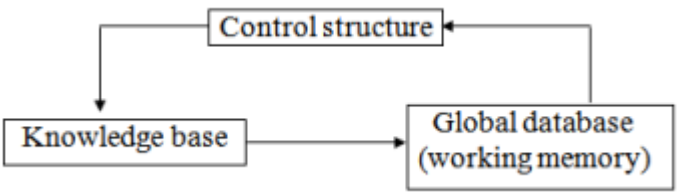


Figure Architecture of Production System

Features of Production System

Some of the main features of production system are:

Expressiveness and intuitiveness: In real world, many times situation comes like "if this happen-you will do that", "if this is so-then this should happen" and many more. The production rules essentially tell us what to do in a given situation.

1. Simplicity: The structure of each sentence in a production system is unique and uniform as they use "IF-THEN" structure. This structure provides simplicity in knowledge representation. This feature of production system improves the readability of production rules.

2. Modularity: This means production rule code the knowledge available in discrete pieces. Information can be treated as a collection of independent facts which may be added or deleted from the system with essentially no deleterious side effects.

3. Modifiability: This means the facility of modifying rules. It allows the development of production rules in a skeletal form first and then it is accurate to suit a specific application.

4. Knowledge intensive: The knowledge base of production system stores pure knowledge. This part does not contain any type of control or programming information. Each production rule is normally written as an English sentence; the problem of semantics is solved by the very structure of the representation.

Disadvantages of production system

1. Opacity: This problem is generated by the combination of production rules. The opacity is generated because of less prioritization of rules. More priority to a rule has the less opacity.

2. Inefficiency: During execution of a program several rules may active. A well devised control strategy reduces this problem. As the rules of the production system are large in number and they are hardly written in hierarchical manner, it requires some forms of complex search through all the production rules for each cycle of control program.

3. Absence of learning: Rule based production systems do not store the result of the problem for future use. Hence, it does not exhibit any type of learning capabilities. So for each time for a particular problem, some new solutions may come.

4. Conflict resolution: The rules in a production system should not have any type of conflict operations. When a new rule is added to a database, it should ensure that it does not have any conflicts with the existing rules.

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.

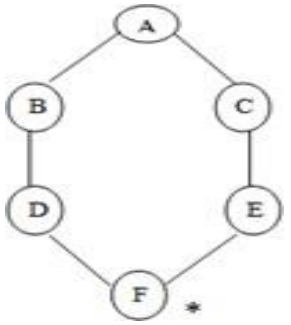
Step 4: Remove the FRONT node from the queue. Process it and find all its neighbours that are in ready state then place them inside the queue in any order

Step 5: Go to Step 3.

Step 6: Exit.

Implementation:

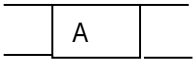
Let us implement the above algorithm of BFS by taking the following suitable example.



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

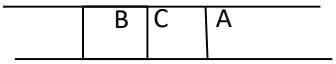


Step 2:

Now the queue is not empty and also the FRONT node i.e. A is not our goal node. So move to step 3.

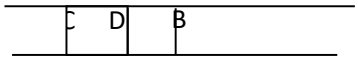
Step 3:

So remove the FRONT node from the queue i.e. A and find the neighbour of A i.e. B and C



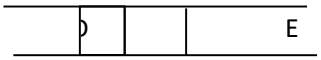
Step 4:

Now b is the FRONT node of the queue .So process B and finds the neighbours of B i.e. D.



Step 5:

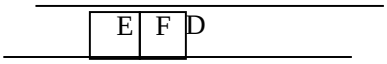
Now find out the neighbours of C i.e. E



C

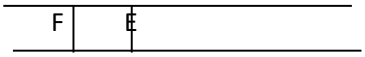
Step 6:

Next find out the neighbours of D as D is the FRONT node of the queue



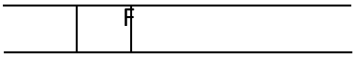
Step 7:

Now E is the front node of the queue. So the neighbour of E is F which is our goal node.



Step 8:

Finally F is our goal node which is the FRONT of the queue. So exit.



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 state and 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.

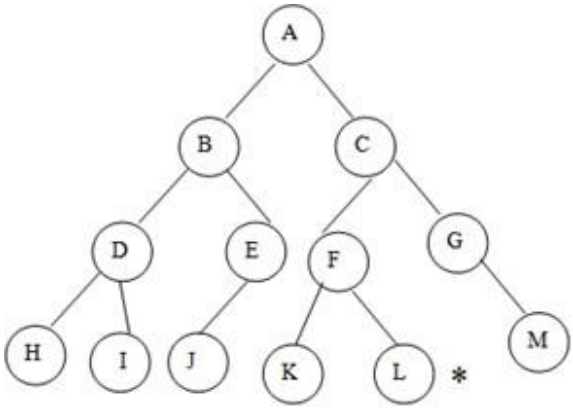


Figure Examples of DFS

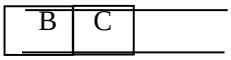
Consider A as the root node and L as the goal node in the graph figure

Step 1: PUSH the starting node into the stack i.e.

A	

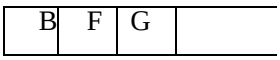
Step 2: Now the stack is not empty and A is not our goal node. Hence move to next step.

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



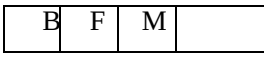
A

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



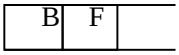
C

Step 5: Now G is the top node of the stack. Find its neighbour i.e. M



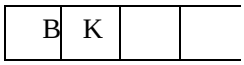
G

Step 6: Now M is the top node and find its neighbour, but there is no neighbours of M in the graph so POP it from the stack.



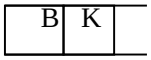
M

Step 7: Now F is the top node and its neighbours are K and L. so PUSH them on to the stack.



L F

Step 8: Now L is the top node of the stack, which is our goal node.



L

Also you can traverse the graph starting from the root A and then insert in the order C and B into the stack. Check your answer.

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.
- It may find a solution without examining much of search because we may get the desired solution in the very first go.

Disadvantages:

- It is possible that may states keep reoccurring.
- There is no guarantee of finding the goal node.
- Sometimes the states may also enter into infinite loops.