Artificial Intelligence

BY

DR. ANUPAM GHOSH

Email: anupam.ghosh@rediffmail.com

LinkedIn Profile: https://www.linkedin.com/in/anupam-ghosh-1504273b/

Research Profile: https://www.researchgate.net/profile/Anupam_Ghosh14

DATE: .7TH SEPT, 2024

What is Artificial Intelligence?

- The power of a machine to copy intelligent human behaviour

<u>Artificial Intelligence Tests: Turing Test</u>

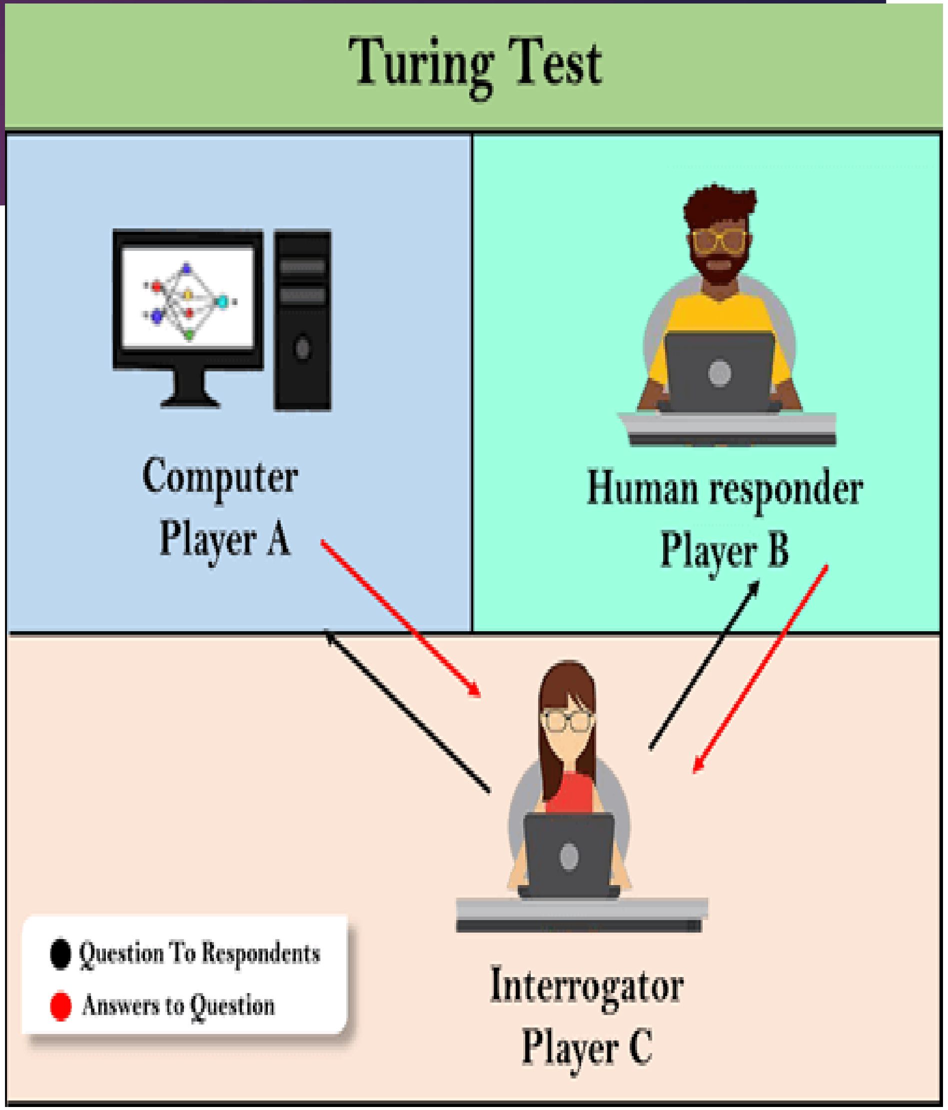
- Developed by Alan Turing
- Involves an interpreter, a human, and a computer.
- The computer and human have separate conversations with the interpreter.
- If the interpreter can't guess which is the computer or if the interpreter gets it wrong then the computer has Artificial Intelligence.

Turing Test in AI

- In 1950, Alan Turing introduced a test to check whether a machine can think like a human or not, this test is known as the Turing Test. In this test, Turing proposed that the computer can be said to be an intelligent if it can mimic human response under specific conditions.
- Consider, Player A is a computer, Player B is human, and Player C is an interrogator. Interrogator is aware that one of them is machine, but he needs to identify this on the basis of questions and their responses.

The questions and answers can be like:

- **Interrogator:** Are you a computer?
- **Player A (Computer):** No
- **Interrogator:** Multiply two large numbers such as (256896489*456725896)
- **Player A:** Long pause and give the wrong answer.
- In this game, if an interrogator would not be able to identify which is a machine and which is human, then the computer passes the test successfully, and the machine is said to be intelligent and can think like a human.



Features required for a machine to pass the Turing test:

- **Natural language processing:** NLP is required to communicate with Interrogator in general human language like English.
- **Knowledge representation:** To store and retrieve information during the test.
- **Automated reasoning:** To use the previously stored information for answering the questions.
- **Machine learning:** To adapt new changes and can detect generalized patterns.
- **Vision (For total Turing test):** To recognize the interrogator actions and other objects during a test.
- **Motor Control (For total Turing test):** To act upon objects if requested.

Intelligence - some definitions

- Intelligence:- ability to adapt oneself adequately to relatively new situations in life. (R. Pintner).
- Intelligence:- having learned or the ability to learn to adjust oneself to the environment. (Colvin)
- Intelligence:- the ability to carry out abstract thinking. (Terman)
- Intelligence:- innate general cognitive ability (Burt)
- Intelligence:- appropriate and adaptable behaviour in given circumstances. (Psihologija, group of authors, SK, Zagreb, 1992)
- Intelligence:- manifests itself only relative to specific social and cultural contexts. (J. Weizenbaum, 1975)

- A branch of computer science: Mathematical Sciences -> Computer Science -> Artificial Intelligence
- The branches of Artificial Intelligence (according to Association of Computing Machinery, ACM):

(1) General AI (cognitive modelling, philosophical foundations)

(2) Expert systems and applications

(3) Automated programming

(4) Deduction and theorem proving

(5) Formalisms and methods for knowledge representation

(6) Machine learning

(7) Understanding and processing of natural and artificial languages

(8) Problem solving, control methods, and state space search

(9) Robotics

(10) Computer vision, pattern recognition, and scene analysis

(11) Distributed artificial intelligence

Goals of AI

- To Create Expert Systems: The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users

- To Implement Human Intelligence in Machines: Creating systems that understand, think, learn, and behave like humans

Applications of AI

- **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.
- **Natural Language Processing**: It is possible to interact with the computer that understands natural language spoken by humans.
- **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.
- **Vision Systems**: These systems understand, interpret, and comprehend visual input on the computer. For example,
  - 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.
- **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.
- **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.
- **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.

Types of Intelligence

- **Linguistic intelligence**: The ability to speak, recognize, and use mechanisms of phonology (speech sounds), syntax (grammar), and semantics (meaning). Ex: Narrators, Orators
- **Musical intelligence**: The ability to create, communicate with, and understand meanings made of sound, understanding of pitch, rhythm. Ex: Musicians, Singers, Composers
- **Logical-mathematical intelligence**: The ability of use and understand relationships in the absence of action or objects. Understanding complex and abstract ideas. Ex: Mathematicians, Scientists
- **Spatial intelligence**: The ability to perceive visual or spatial information, change it, and re-create visual images without reference to the objects, construct 3D images, and to move and rotate them. Ex: Map readers, Astronauts, Physicists
- **Bodily-Kinesthetic intelligence**: The ability to use complete or part of the body to solve problems or fashion products, control over fine and coarse motor skills, and manipulate the objects. Ex: Players, Dancers
- **Intra-personal intelligence**: The ability to distinguish among one's own feelings, intentions, and motivations.
- **Interpersonal intelligence**: The ability to recognize and make distinctions among other people's feelings, beliefs, and intentions. Ex: Mass Communicators, Interviewers

What’s involved in Intelligence?

- Ability to interact with the real world:

  To perceive, understand, and act

  e.g., speech recognition and understanding and synthesis, image understanding, ability to take actions, have an effect

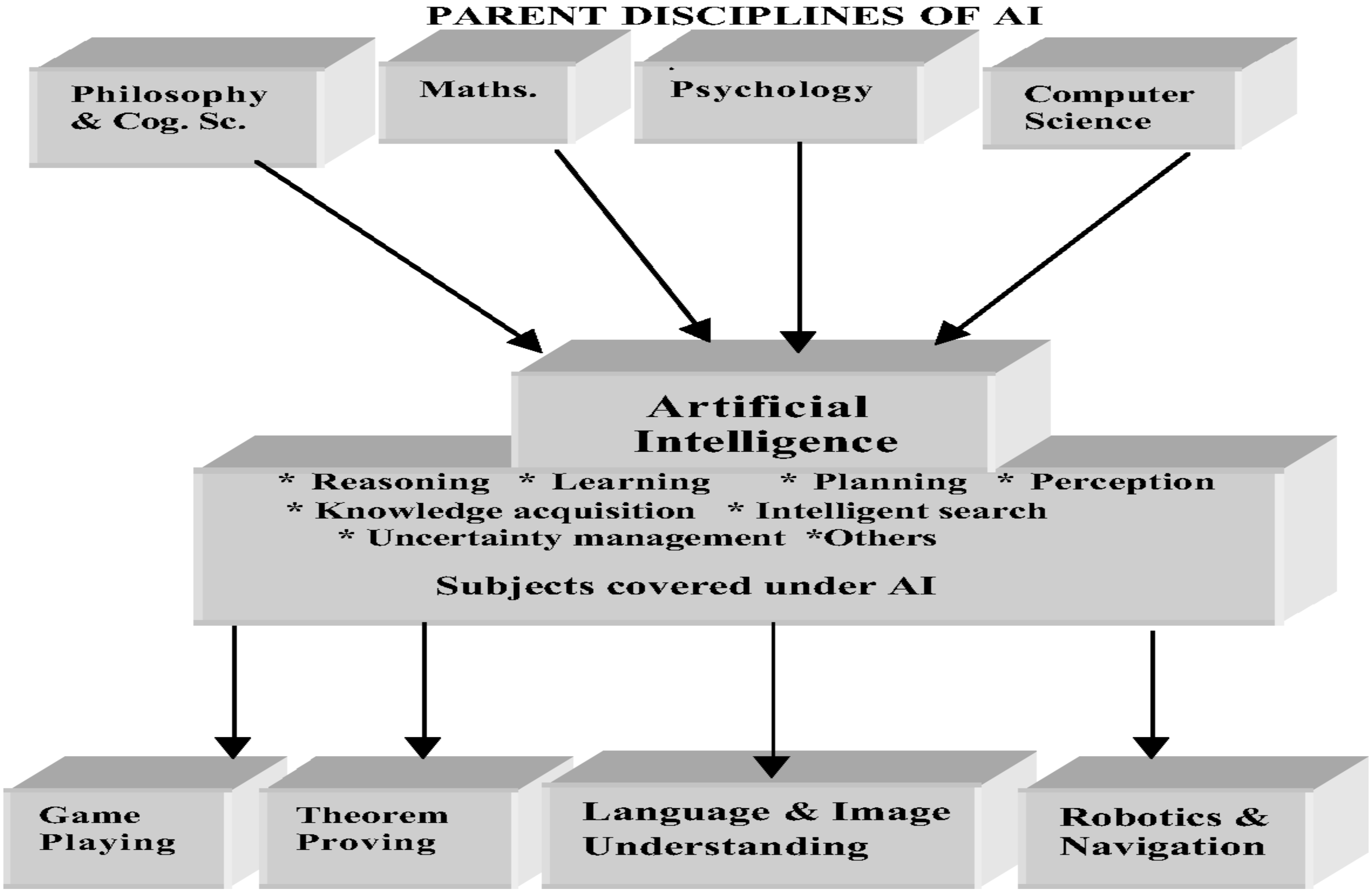
- Reasoning and Planning:

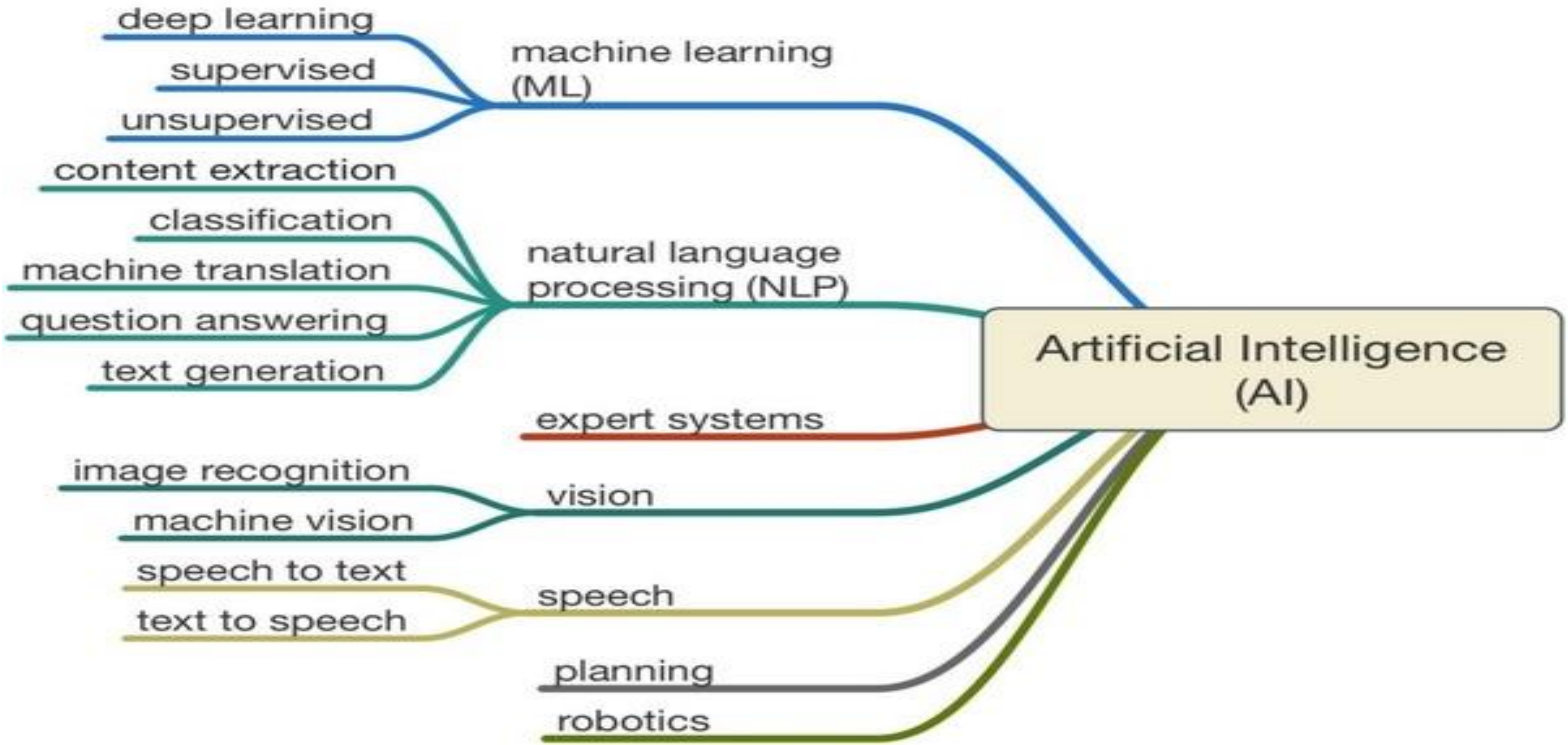
  Modelling the external world, given input solving new problems, planning, and making decisions, ability to deal with unexpected problems, uncertainties

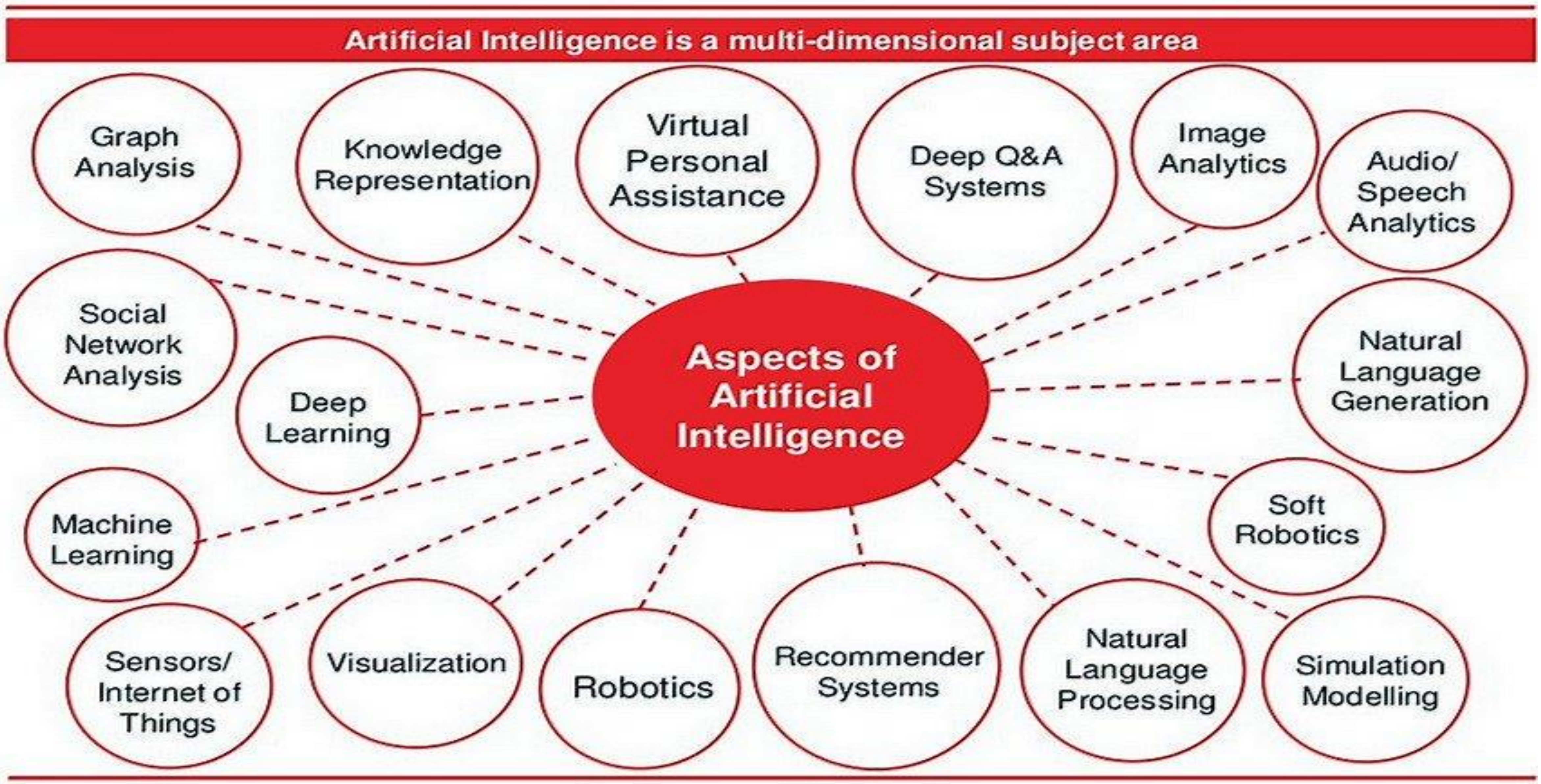
- Learning and Adaptation:

  we are continuously learning and adapting our internal models are always being “updated”

  e.g., a baby learning to categorize and recognize animals


PARENT DISCIPLINES OF AI
Philosophy & Cog. Sc.
Maths.
Psychology
Computer Science
Artificial Intelligence
* Reasoning * Learning * Planning * Perception
* Knowledge acquisition * Intelligent search
* Uncertainty management *Others
Subjects covered under AI
Game Playing
Theorem Proving
Language & Image Understanding
Robotics & Navigation


deep learning
supervised
unsupervised
machine learning (ML)
content extraction
classification
machine translation
question answering
text generation
natural language processing (NLP)
expert systems
image recognition
machine vision
vision
speech to text
text to speech
speech
planning
robotics
Artificial Intelligence (AI)


Artificial Intelligence is a multi-dimensional subject area
Aspects of Artificial Intelligence
Graph Analysis
Knowledge Representation
Virtual Personal Assistance
Deep Q&A Systems
Image Analytics
Audio/ Speech Analytics
Social Network Analysis
Deep Learning
Natural Language Generation
Machine Learning
Soft Robotics
Sensors/ Internet of Things
Visualization
Robotics
Recommender Systems
Natural Language Processing
Simulation Modelling

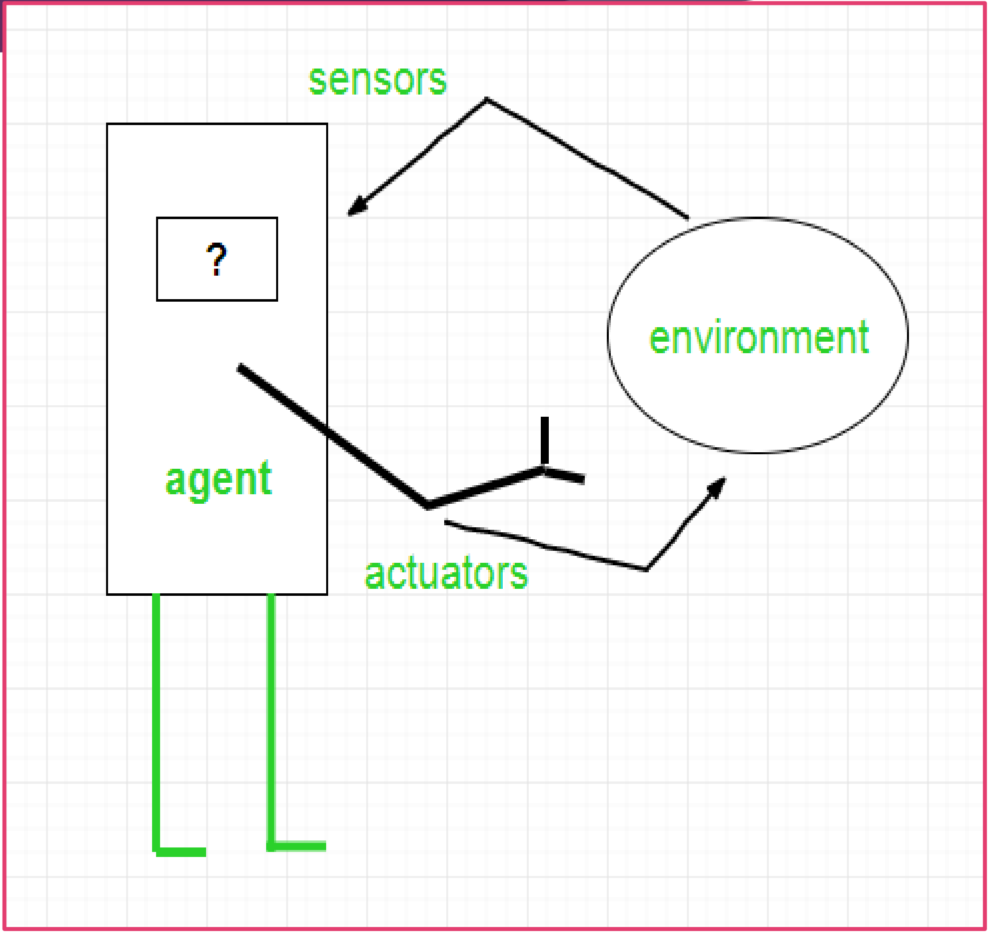
Types of Agents and Learning

Agents in Artificial Intelligence

- Artificial intelligence is defined as the study of rational agents. A rational agent could be anything that makes decisions, as a person, firm, machine, or software. It carries out an action with the best outcome after considering past and current percepts(agent's perceptual inputs at a given instance). 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 be viewed as :

- perceiving its environment through sensors and
- acting upon that environment through actuators

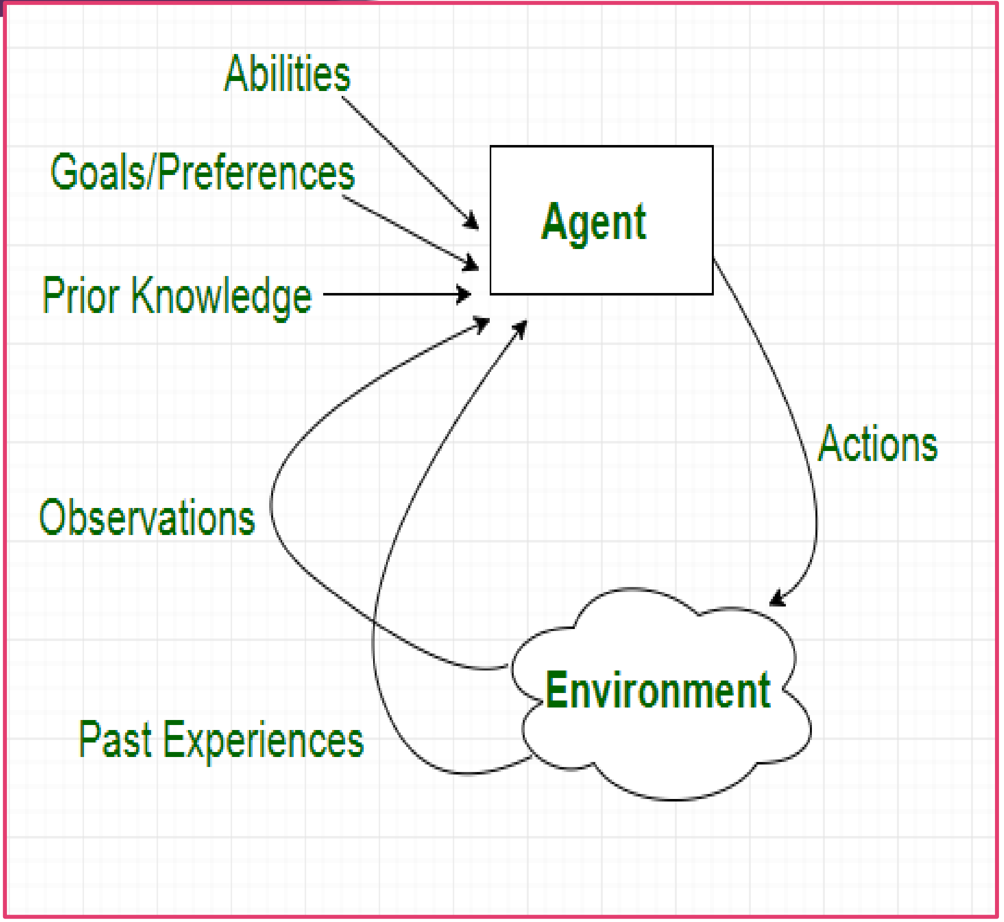


Agent = Architecture + Agent Program

- 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.

<u>Examples of Agent:</u>

- 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 :

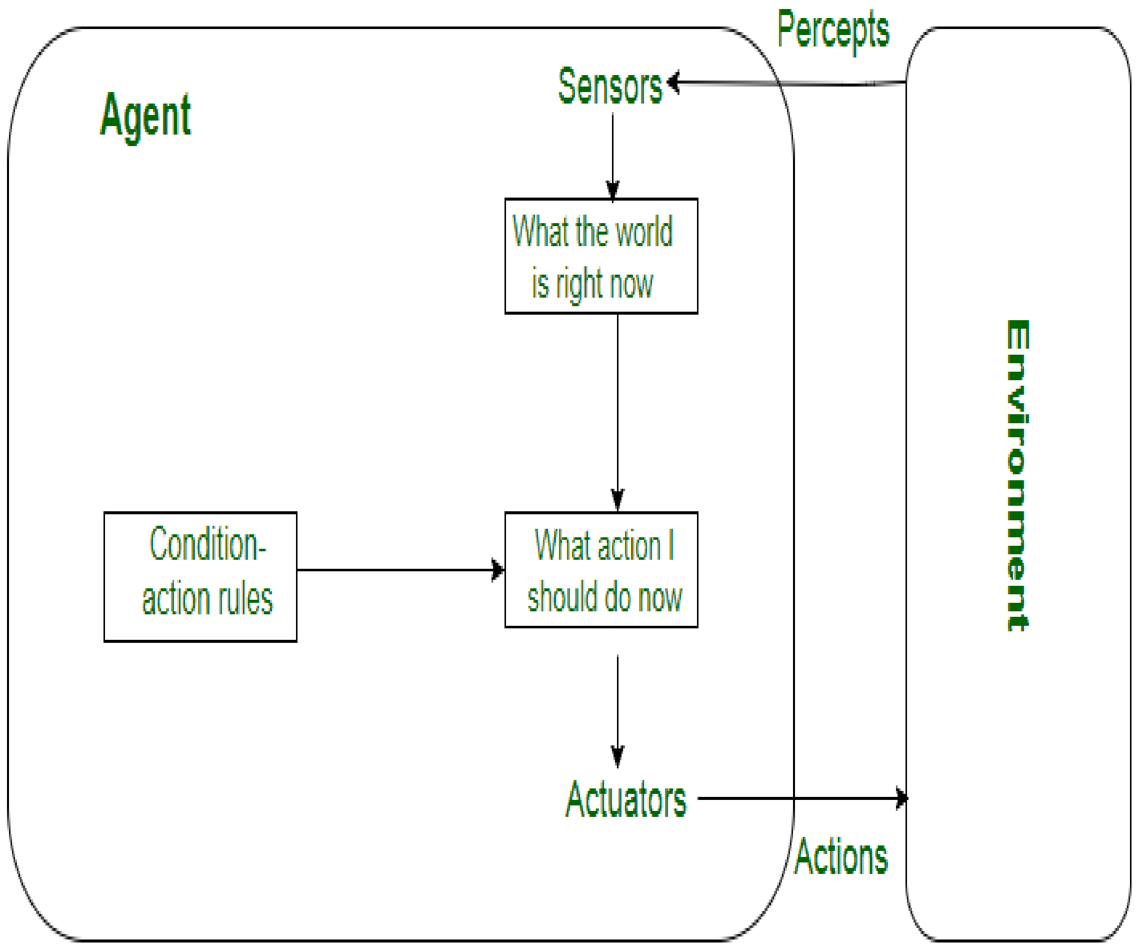
- Simple Reflex Agents
- Model-Based Reflex Agents
- Goal-Based Agents
- Utility-Based Agents
- Learning Agent

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**. If the condition is true, then the action is taken, else not. 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.

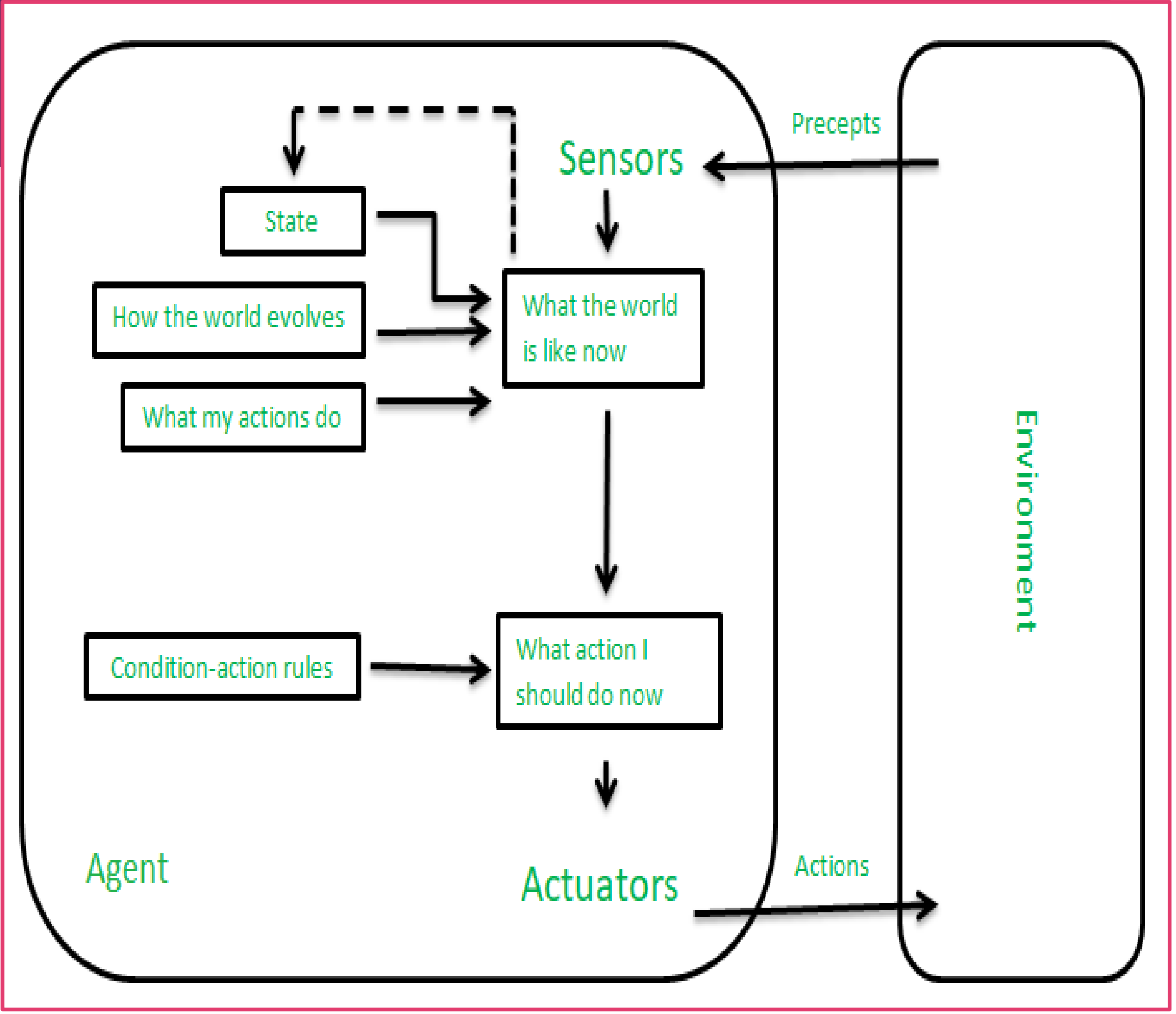


Model-based reflex agents

- 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.

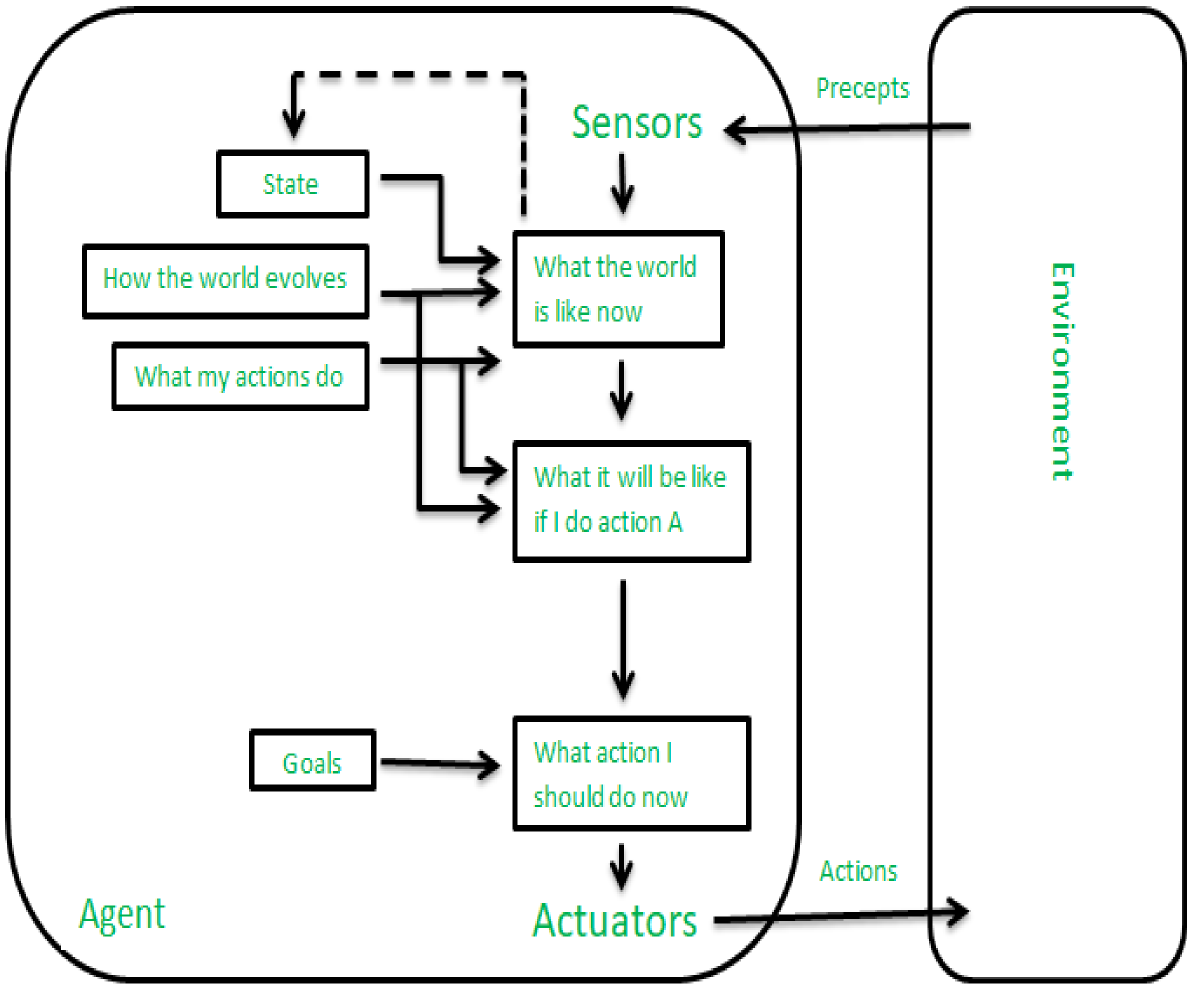
Updating the state requires information about :

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



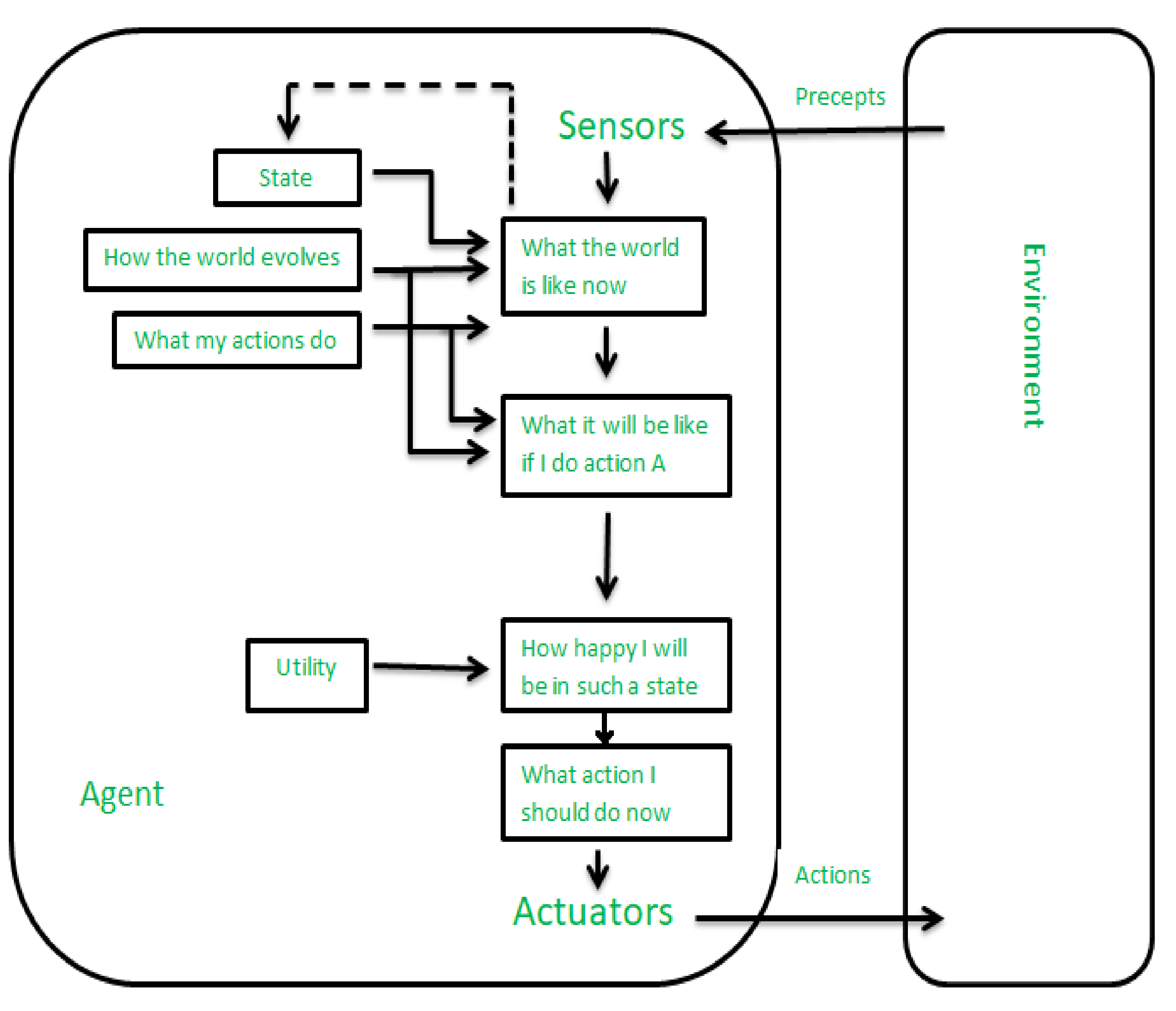
Goal-based agents

- 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.



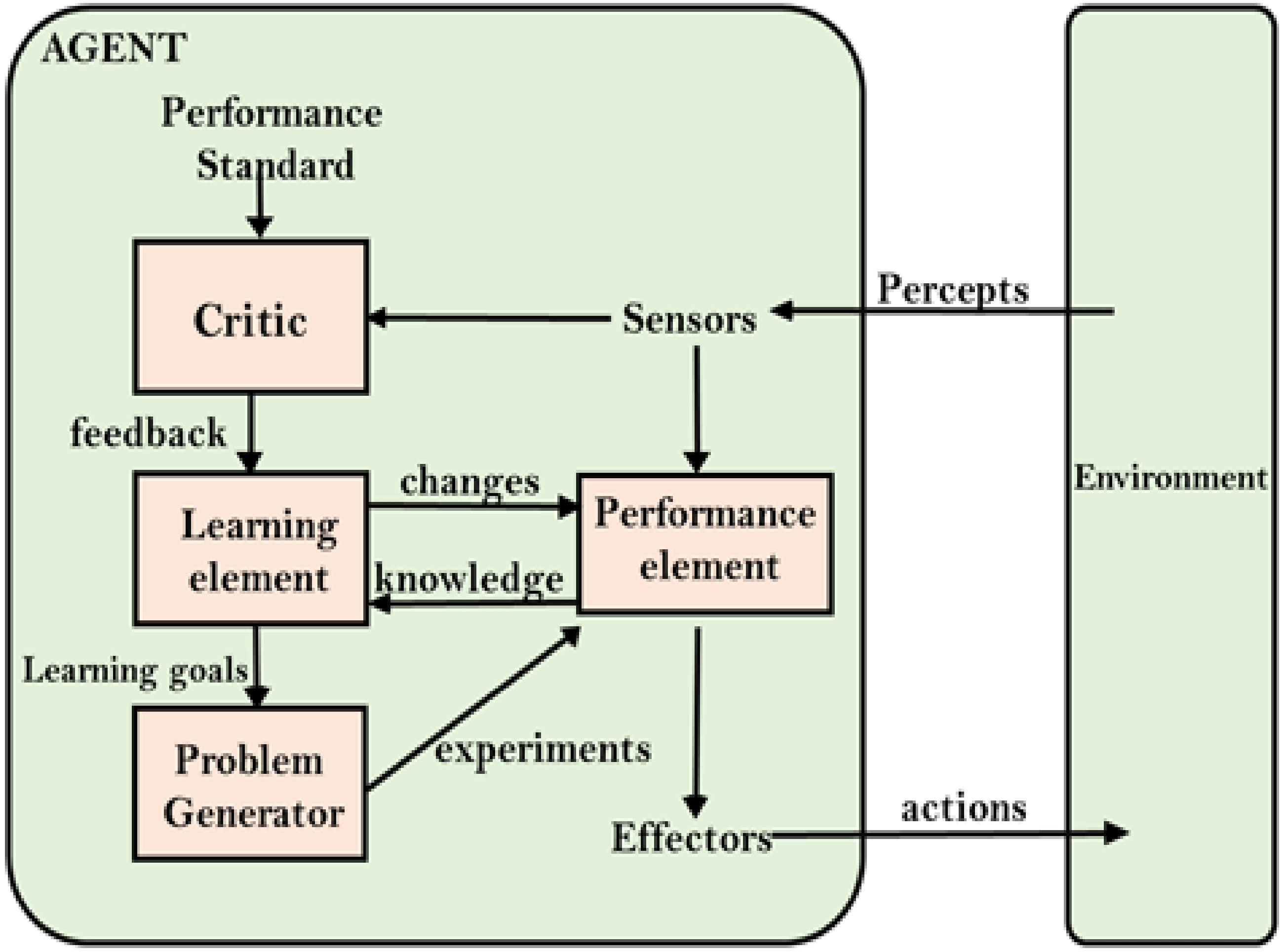
Utility-based agents

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



Example: Water Jug Problem

- Consider the following problem: A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

Solution: Phase-I

- State Representation and Initial State { we will represent a state of the problem as a tuple (x, y) where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-gallon jug. Note 0 <= x <= 4, and 0 <= y<= 3. Our initial state: (0,0)
- Goal Predicate state = (2,y) where 0 <= y <= 3.

Solution: Phase-II:--Operators

1. Fill 4-gal jug: (x,y) -> (4,y)
2. Fill 3-gal jug: (x,y) -> (x,3)
3. Empty 4-gal jug on ground: (x,y) -> (0,y)
4. Empty 3-gal jug on ground: (x,y) ->(x,0)
5. Pour water from 3-gal jug to fill 4-gal jug : (x,y) -> (4, y - (4 - x))
6. Pour water from 4-gal jug to fill 3-gal jug : (x,y) -> (x-(3-y), 3)
7. Pour all of water from 3-gal jug into 4-gal jug: (x,y) ->(x+y, 0)
8. Pour all of water from 4-gal jug into 3-gal jug: (x,y) ->(0, x+y)

Solution: Phase-III:--Rules

Gals in 4-gal jug	Gals in 3-gal jug	Rule Applied
0	0	
4	0	1. Fill 4
1	3	6. Pour 4 into 3 to fill
1	0	4. Empty 3
0	1	8. Pour all of 4 into 3
4	1	1. Fill 4
2	3	6. Pour into 3

Why Searching algorithms in AI?

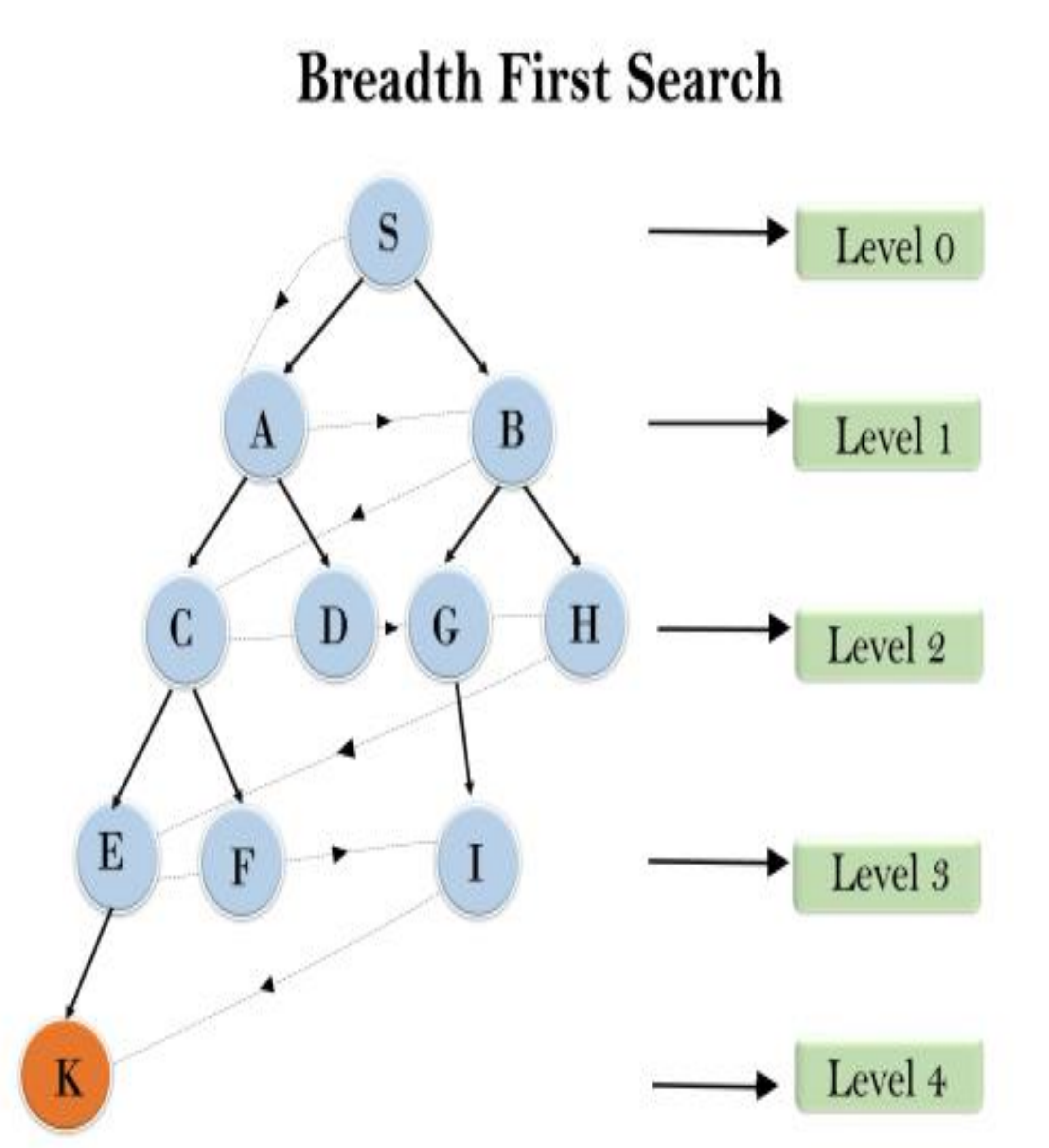
- **In 1997 'Deep Blue' an AI beat the legendary Gary Kasparov in Chess , and in 2016 'Alpha Go' defeated the champion of the game Go. The ability of artificial intelligence to mimc humans and surpass their mental capabilities has exceeded over time.**
- **Searching algorithms form the base of such programs , they are utilized in areas like course and cost optimization, action planning, information mining, mechanical technology, independent driving, computational science, programming and equipment check, hypothesis demonstrating and so on.**
- As it were, a considerable lot of the AI issues can be considered such that the objective is to reach the final goal from an initial point by means of state change rules. So the search space or options are characterized as a diagram (or a tree) and the point is to arrive at the objective from the underlying state through the shortest path.
- The searching algorithms can be classified into two types:
  - Uninformed methods : in this method no additional information is provided
  - Informed methods : also known as Heuristic method where search is carried out by using additional information to find out the next step to take.

Breadth-first Search:

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadth wise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.
- **Advantages:**
- BFS will provide a solution if any solution exists.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.
- **Disadvantages:**
- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

Example

- In the below tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:
- S---> A--->B---->C--->D---->G--->H--->E---->F---->I---->K
- **Time Complexity:** Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.
- **$T(b) = 1+b^2+b^3+.......+ b^d= O(b^d)$**
- **Space Complexity:** Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.
- **Completeness:** BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.
- **Optimality:** BFS is optimal if path cost is a non-decreasing function of the depth of the node.



Depth-first Search

- Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.

Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

Disadvantage:

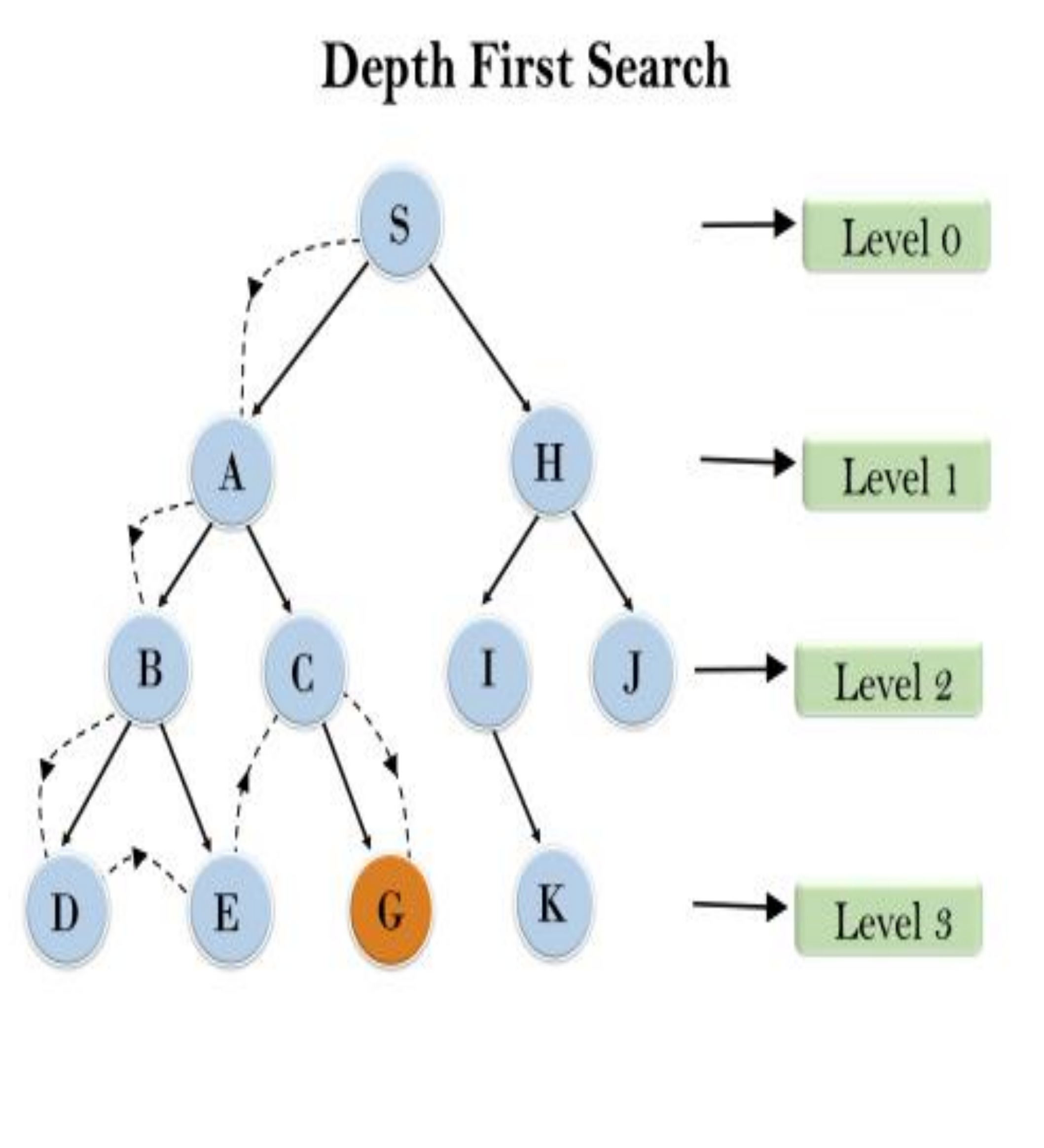
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

Example

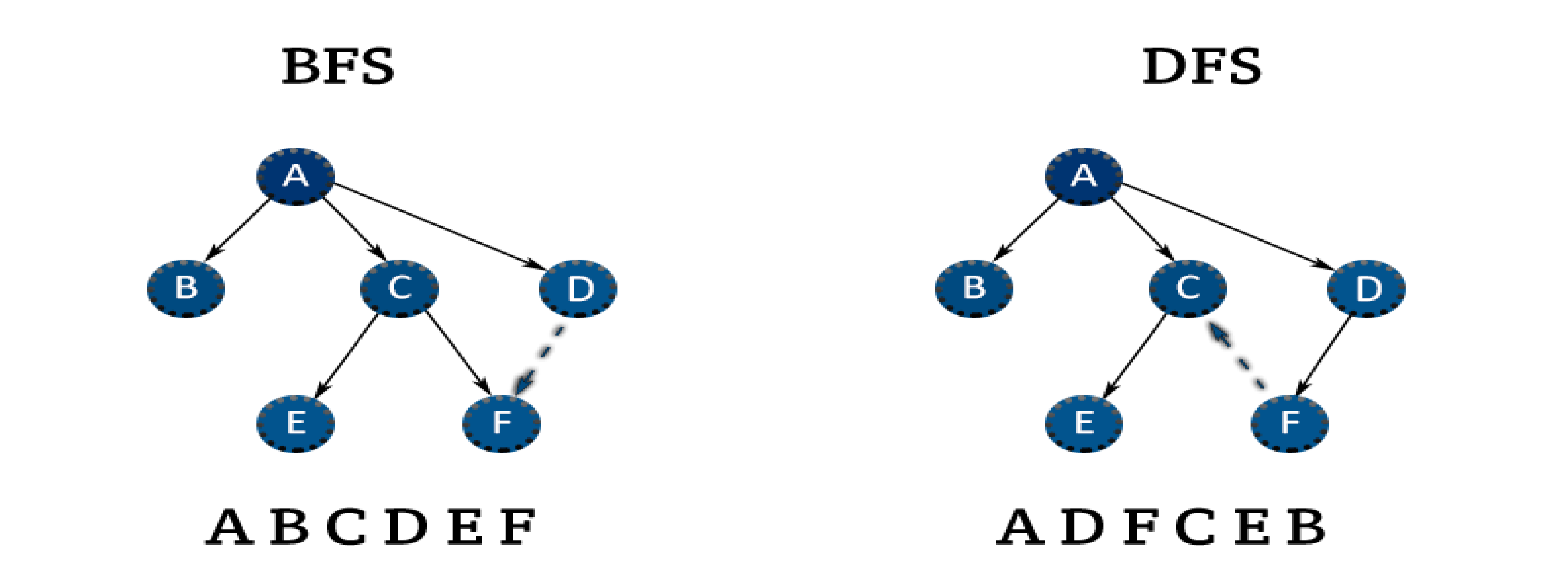
- In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:
- Root node--->Left node ----> right node.
- It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found. After backtracking it will traverse node C and then G, and here it will terminate as it found goal node.
- **Completeness:** DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.
- **Time Complexity:** Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:
- **$T(n)= 1+ n^2+ n^3 +.........+ n^m=O(n^m)$**

Where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)

- **Space Complexity:** DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.
- **Optimal:** DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.



BFS and DFS



Depth-Limited Search Algorithm

- A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.
- Depth-limited search can be terminated with two Conditions of failure:
- Standard failure value: It indicates that problem does not have any solution.
- Cutoff failure value: It defines no solution for the problem within a given depth limit.

Advantages:

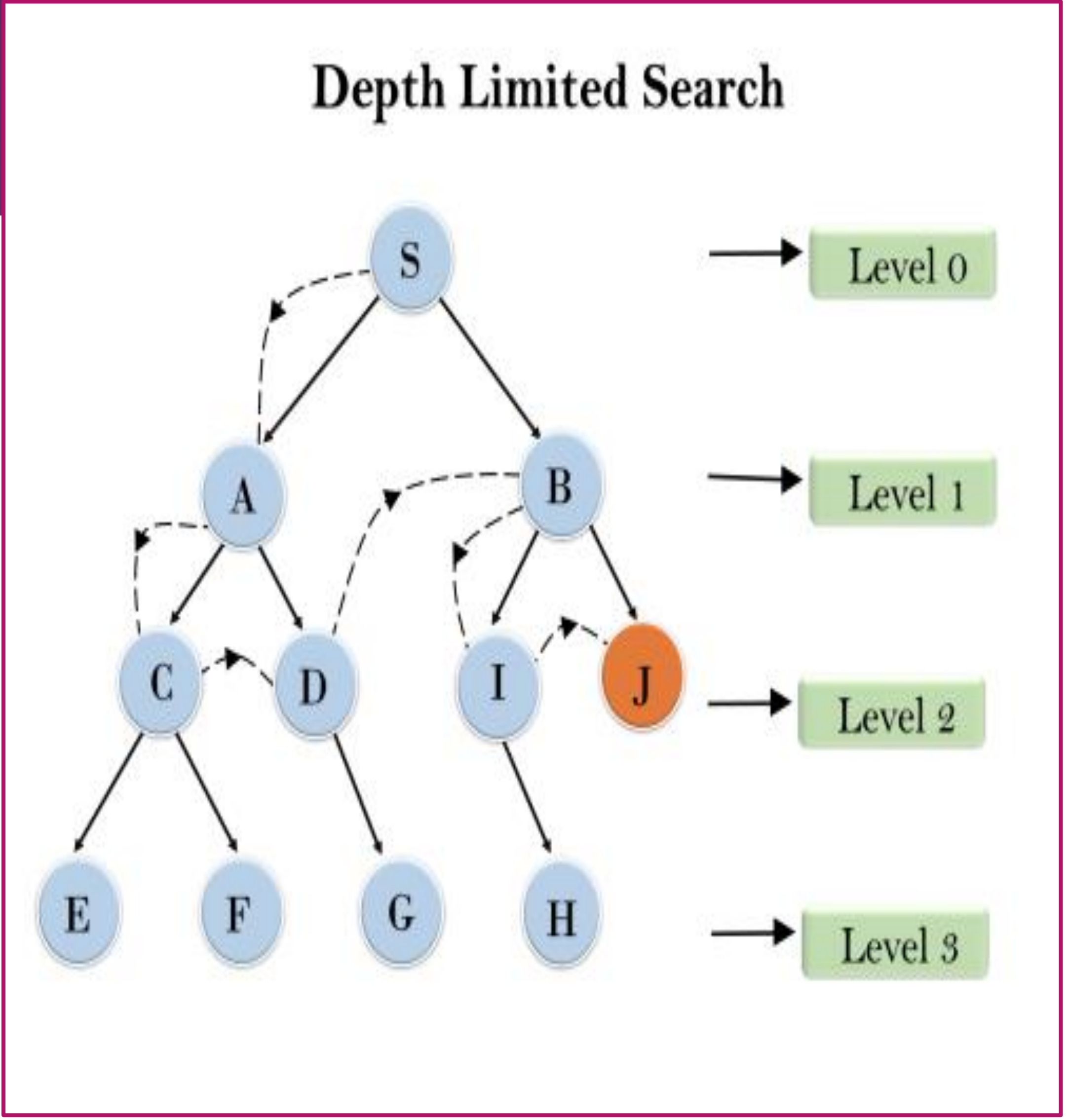
- Depth-limited search is Memory efficient.

Disadvantages:

- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.

Example:

- **Completeness:** DLS search algorithm is complete if the solution is above the depth-limit.
- **Time Complexity:** Time complexity of DLS algorithm is $\mathbf{O(b^{\ell})}$.
- **Space Complexity:** Space complexity of DLS algorithm is O$\mathbf{(b\times\ell)}$.
- **Optimal:** Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if $\ell>d$.



Uniform-cost Search Algorithm

- Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

Advantages:

- Uniform cost search is optimal because at every state the path with the least cost is chosen.

Disadvantages:

- It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

Example

Completeness:

- Uniform-cost search is complete, such as if there is a solution, UCS will find it.

Time Complexity:

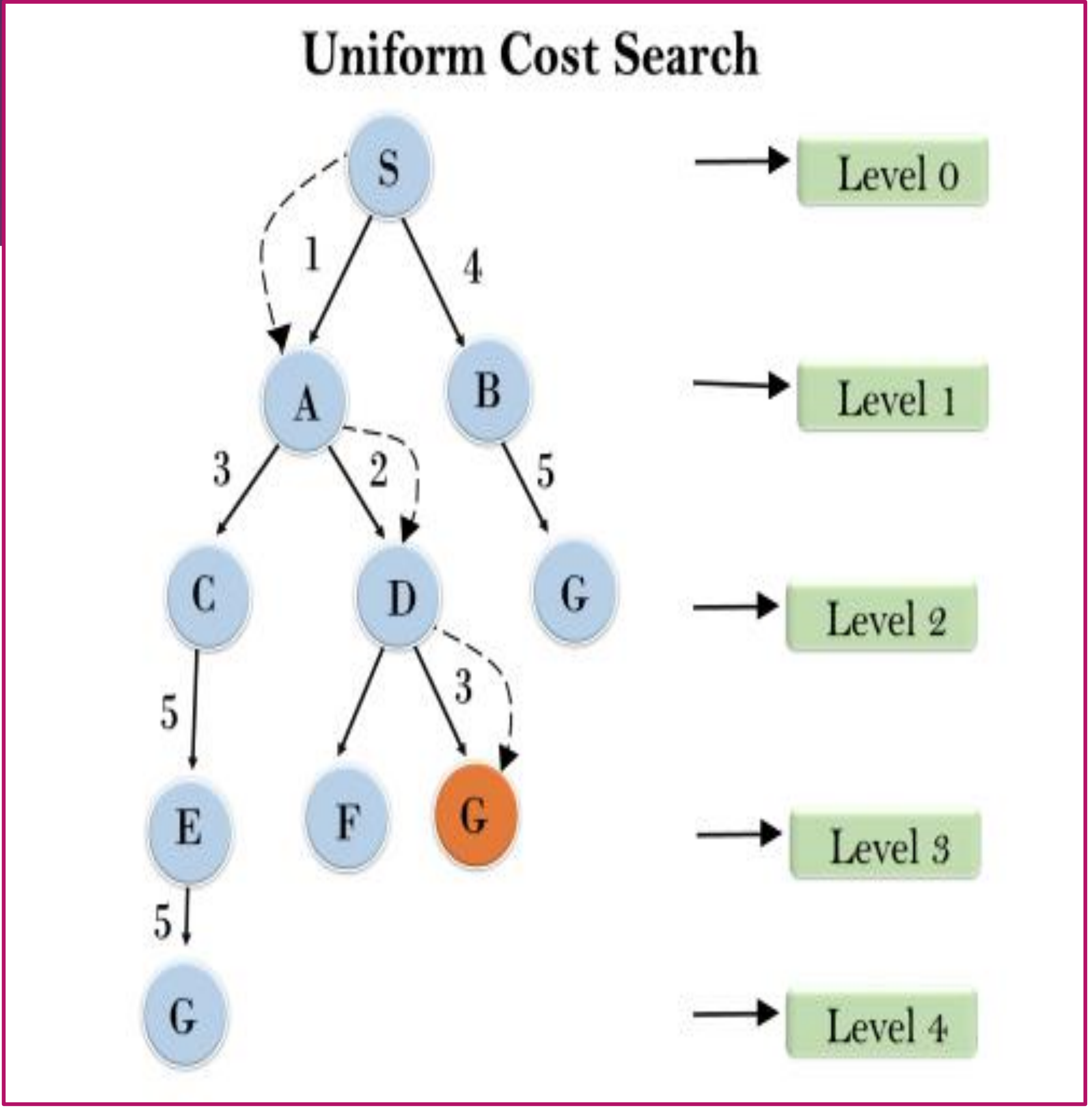
- Let C* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C*/ε+1. Here we have taken +1, as we start from state 0 and end to C*/ε.
- Hence, the worst-case time complexity of Uniform-cost search is $\mathbf{O(b^{1 + [C^*/\varepsilon]})}$/.

Space Complexity:

- The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is $\mathbf{O(b^{1 + [C^*/\varepsilon]})}$.

Optimal:

- Uniform-cost search is always optimal as it only selects a path with the lowest path cost.



Iterative deepening depth-first Search

- The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.
- This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.
- This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.
- The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

Advantages:

- It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

Disadvantages:

- The main drawback of IDDFS is that it repeats all the work of the previous phase.

Example:

- 1'st Iteration-----> A
  2'nd Iteration----> A, B, C
  3'rd Iteration------>A, B, D, E, C, F, G
  4'th Iteration------>A, B, D, H, I, E, C, F, K, G
  In the fourth iteration, the algorithm will find the goal node.

Completeness:

- This algorithm is complete is ifthe branching factor is finite.

Time Complexity:

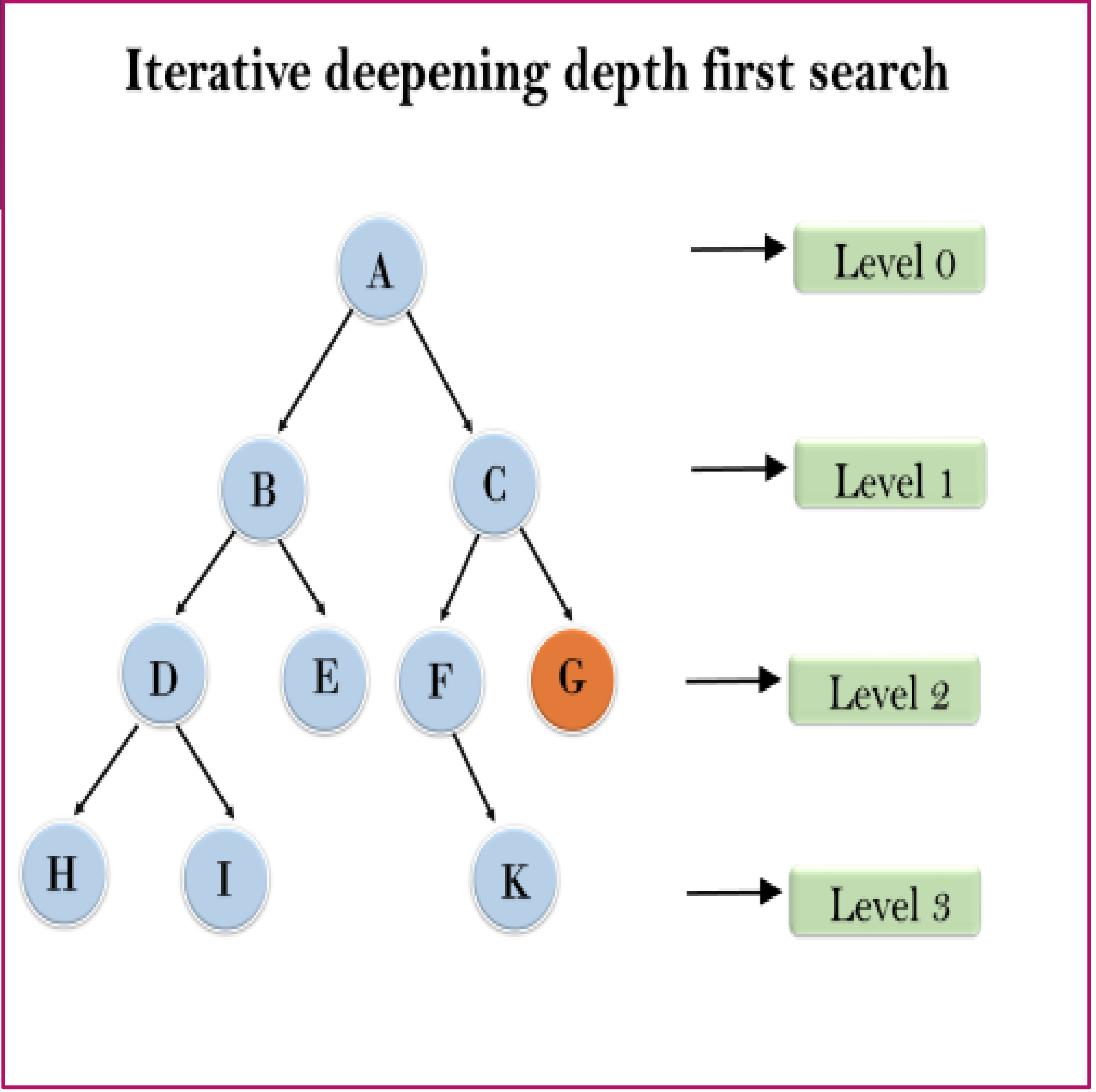
- Let's suppose b is the branching factor and depth is d then the worst-case time complexity is $\mathbf{O(b^d)}$.

Space Complexity:

- The space complexity of IDDFS will be **O(bd)**.

Optimal:

- IDDFS algorithm is optimal if path cost is a non-decreasing function of the depth of the node.



Bidirectional Search Algorithm

- Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small sub graphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.
- Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.

Advantages:

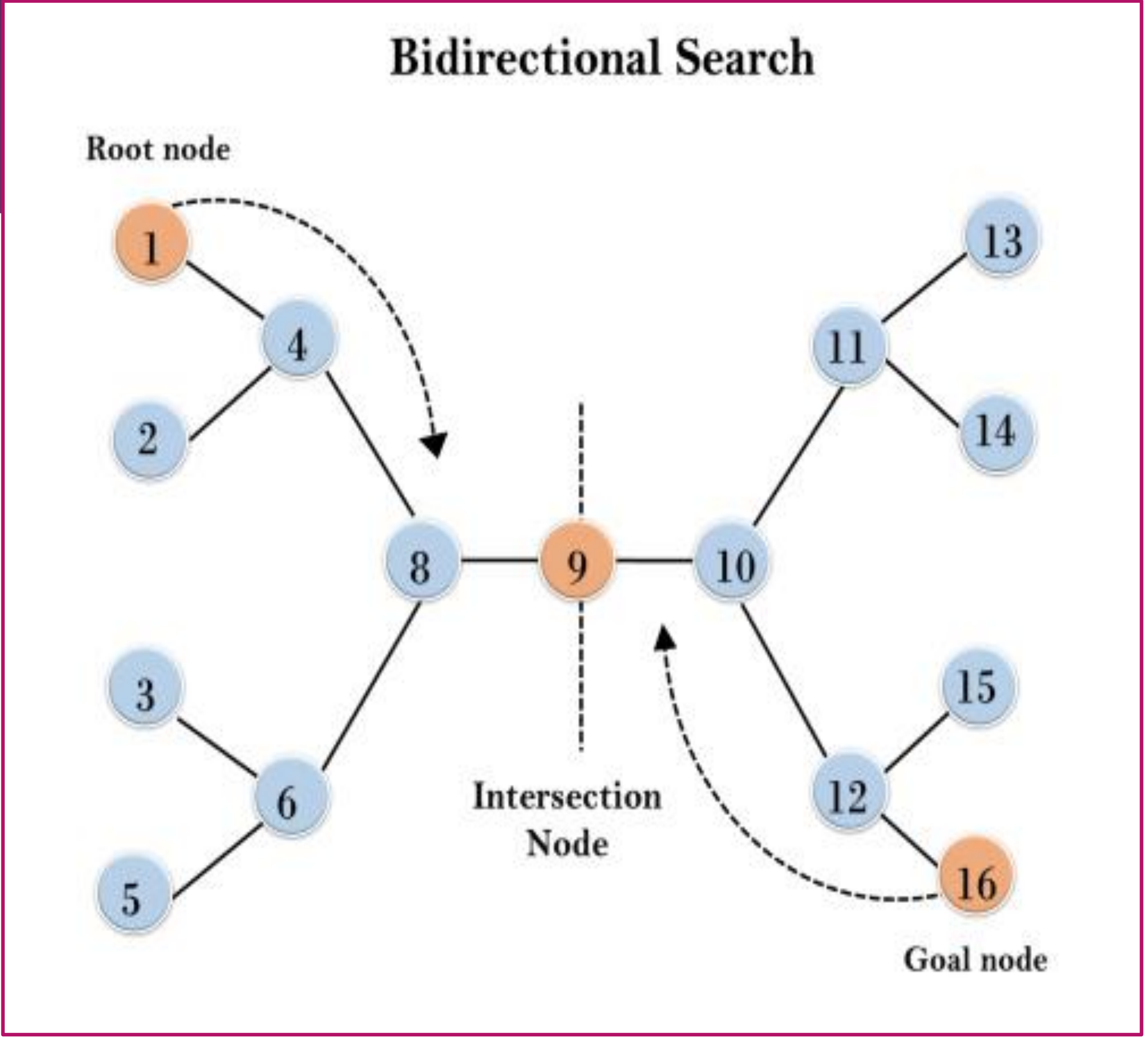
- Bidirectional search is fast.
- Bidirectional search requires less memory

Disadvantages:

- Implementation of the bidirectional search tree is difficult.
- In bidirectional search, one should know the goal state in advance.

Example

- In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.
- The algorithm terminates at node 9 where two searches meet.
- **Completeness:** Bidirectional Search is complete if we use BFS in both searches.
- **Time Complexity:** Time complexity of bidirectional search using BFS is **$O(b^d)$**.
- **Space Complexity:** Space complexity of bidirectional search is **$O(b^d)$**.
- **Optimal:** Bidirectional search is Optimal.



It will be Continued….