Module-1

Chapter 1: Introduction to Artificial Intelligence

1. Definition of Artificial Intelligence

- Artificial Intelligence (AI) is the science and engineering of making intelligent machines, especially intelligent computer programs, **John McCarthy (2004).**
- The concept of AI was first explored in **1950** by **Alan Turing**, a British mathematician and computer scientist, who proposed the **Turing Test.**
- **From a layman's view,** artificial intelligence (AI) simply means the intelligence demonstrated by machines that helps them to mimic the actions of humans. AI simulates natural intelligence in machines that are programmed to learn from experiences, adjust to new inputs, and perform human-like tasks.
- **For researchers**, AI refers to a set of algorithms that help a machine make decisions and act without being explicitly told what to do each time.
- A few languages that are popularly used to code AI applications are R, Python, and Java.
- AI applications perform specialized tasks by processing large amounts of data and recognizing patterns in them.
- Most AI examples, from chess-playing computers to self-driving cars, heavily depend on deep learning and natural language processing techniques.
- Some applications of AI are:
    1) **Healthcare** – Disease diagnosis, medical imaging, and drug discovery.
    2) **Education** – Personalized learning, AI tutors, and automated grading.
    3) **Business & Finance** – Fraud detection, customer chatbots, and stock predictions.
    4) **Agriculture** – Crop disease detection, precision farming, and yield prediction.
    5) **Transportation** – Self-driving cars, traffic management, and logistics optimization.

2. How Does AI Work?

AI works best when it is trained with a large amount of labeled data. By studying this data, it finds patterns and uses them to make predictions.

Example: A chatbot trained with many chat examples can learn to talk with people.

AI programming mainly focuses on three skills:

1. **Learning** – gaining knowledge from data.

   AI programs need **data** to work. They collect and study this data to understand it. To make the data useful, AI creates **rules called algorithms**. An algorithm is simply a **step-by-step set of instructions** that tells the computer how to solve a problem or perform a task.

   Example: If an AI is trained to recognize fruits, the algorithm will guide it to look at features like color, size, and shape to decide whether an image shows an apple, banana, or orange.

2. **Reasoning** – making decisions using rules and logic.

   The success of an AI program depends on selecting the **right algorithm** because different problems need different methods. If the correct algorithm is chosen, the AI can process data properly and give the **best possible results**.

   Example: To recognize faces, an image-processing algorithm works better than a text-processing one.

3. **Self-correction** – improving performance over time.

   AI programs are built to **keep improving themselves**. They update and refine their algorithms so that the results become **more accurate over time**.

   Example: A voice assistant like Alexa gets better at understanding your speech the more you use it.

Case Study: Why is AI important?

AI helps businesses understand their operations better and find new opportunities. In many cases, AI can perform tasks faster and more accurately than humans.

Example: **Uber** uses AI and machine learning to study ride patterns. It can predict when and where more people will need taxis and alert drivers to be ready in those areas. This makes the service faster and more efficient.

3. Advantages and Disadvantages of AI

Advantages:

1. Performs well on tasks that uses detailed data.
2. Takes less time to perform tasks that needs to process huge volumes of data.
3. Generates consistent and accurate results.
4. Can be used 24 X 7.
5. Optimizes tasks by better utilizing resources.
6. Automates complex processes.
7. Minimizes downtime by predicting maintenance needs.
8. Enables companies to produce new products having better quality and speed.

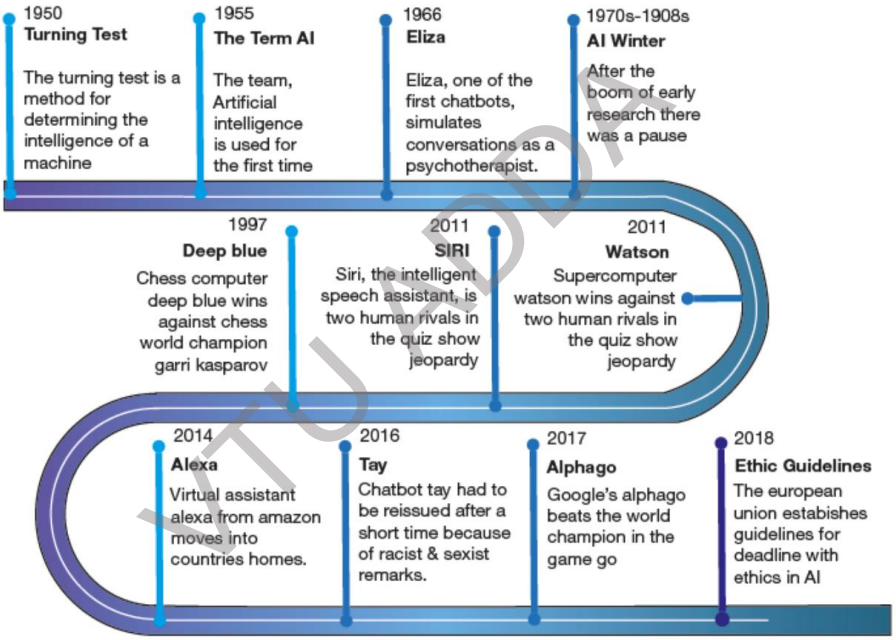
Disadvantages

1. Involves more cost.
2. Technical expertise required to develop and use AI applications.
3. Lack of trained professionals.
4. Incomplete or inaccurate data may result in disastrous results.
5. Lacks the capability to generalize tasks.

4. History of AI

1. **1943** – McCullough & Pitts proposed the first neural network model.

2. **1950** – Alan Turing introduced the Turing Test; Minsky & Edmonds built the first neural network computer; Shannon published work on chess.
3. **1956** – John McCarthy coined the term *Artificial Intelligence* at the Dartmouth Conference.
4. **1958–1959** – Lisp programming language developed; Newell & Simon created the General Problem Solver; Arthur Samuel coined *Machine Learning*.
5. **1963–1969** – Stanford AI Lab established; ELIZA chatbot created; first medical expert system developed.
6. **1974–1993** – Two "AI Winters" slowed research due to lack of funding and results.
7. **1997–2016** – IBM's Deep Blue beat Kasparov (1997); Stanley self-driving car won DARPA challenge (2005); Siri launched (2011); AlphaGo defeated Lee Sedol (2016).
8. **2018–2020** – Google BERT improved NLP (2018); Waymo launched self-driving taxi; Baidu's LinearFold AI helped COVID-19 vaccine research (2020).



5. Types of AI

AI is mainly categorized into two types:

1. **Based on Capabilities**
2. **Based on Functionalities**

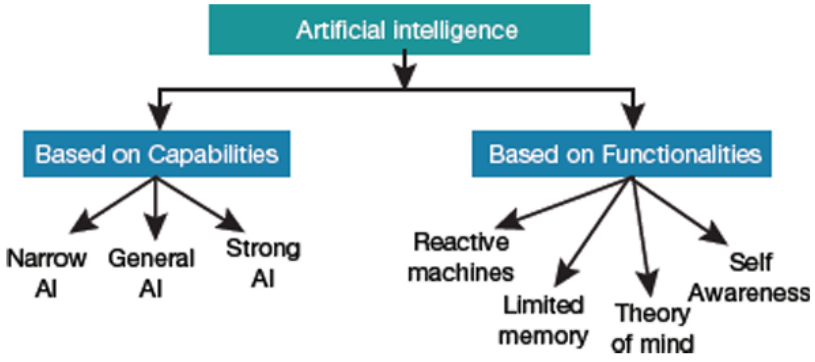


Figure 2: Types of AI

1. Based on Capabilities

Capabilities mean the level of intelligence of AI – how powerful it is compared to humans, and whether it can match or surpass human abilities.

1. Weak AI (Narrow AI)

- Weak AI, also known as narrow AI, is designed to do one specific task.
- Examples: Siri and Alexa are examples. When you tell Alexa to play a song, it does so because it's trained to understand that specific command.
- Other examples include weather forecasting, predicting stock prices, and Google search.
- How it works: These systems are great at doing one thing really well, but they don't work outside their specific task. For example, Alexa can't drive a car; it's just built for voice commands.
- Why it's important: Weak AI has helped make many tasks easier and more efficient, and it is the most common type of AI in use today.

2. Strong AI (Artificial General Intelligence - AGI)

- Strong AI, also called Artificial General Intelligence (AGI) or Superintelligence (ASI), tries to mimic human thinking.
- Strong AI is a type of AI that tries to work like the human brain. It can do tasks even if it was not trained for them.
- Uses thinking skills and fuzzy logic (not just yes/no).
- Can perform tasks like:
  - Seeing and understanding images (visual perception).
  - Understanding and speaking languages.
  - Making decisions like humans.
  - Translating between languages.
- Future potential: Experts believe that Strong AI might one day surpass human intelligence, but it's not expected to happen anytime soon.

Weak AI	Strong AI
It supports a narrow range of applications with a limited scope.	It supports a wider range of application with a wide scope.
This application is good at specific tasks.	This application has an incredible human-level intelligence.
It uses supervised and unsupervised learning to process data.	It uses clustering and association techniques to process data.
Example: Siri, Alexa.	Example: Advanced robotics

2. Based on Functionalities

This means AI is classified according to how it works, behaves, and interacts with its environment – i.e., the way it functions rather than how powerful it is.

1. **Reactive Machines**
   - Reactive Machines are the simplest type of AI that react to situations based on immediate input, but they have no memory or ability to learn from past experiences.
   - Examples: IBM's Deep Blue (chess-playing computer) is a reactive machine. It makes decisions based on the current state of the game but doesn't remember past games.

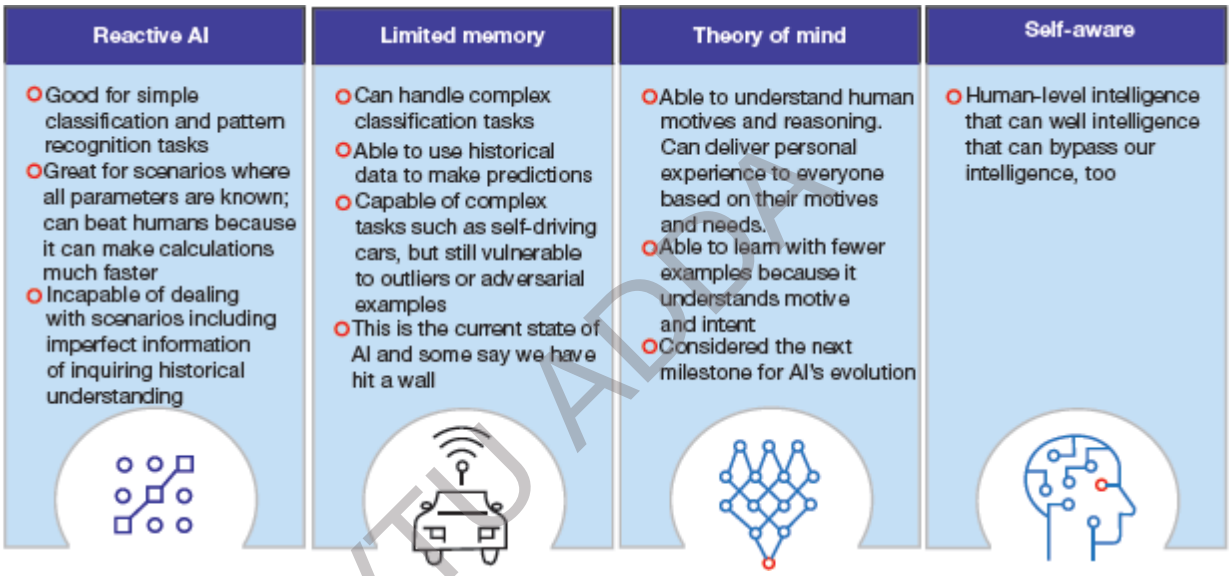
2. **Limited Memory**
   - What it is: Limited memory AI systems can remember data for a short time and use it to make decisions, but they don't keep data permanently.
   - **Examples:** Autonomous vehicles use limited memory to track information like speed of nearby cars, distance between cars, and speed limits to navigate safely. AlphaGo, the AI that defeated the world champion in the game Go, also used limited memory to play and improve during the game.
   - These systems learn and improve continuously by analyzing new data and adjusting based on feedback.
   - **Key Models:**
   - **Reinforcement Learning:** AI learns by trial and error, improving over time.
   - **Long Short-Term Memory (LSTM):** AI uses past data to predict the next step, but it focuses more on recent data.
   - **Evolutionary GANs (E-GAN):** The AI evolves over time, using data and feedback to make better decisions and predict outcomes.

3. **Theory of Mind**
   - The Theory of Mind in AI aims to create machines that can understand thoughts, emotions, and memories just like humans.

- **How it works:** AI would need to understand feelings and emotions that influence decisions. These machines would make choices by considering both reason and emotional context.
- **Current Status:** This is still theoretical, meaning it's an idea for the future, but it could become a reality soon.

4. Self-Awareness

- Self-Aware AI is the most advanced form of AI with human-like consciousness and awareness of its own existence.
- Can understand its own state, deduce others' feelings/needs, and interpret *what* is said as well as how it is said.
- Could respond with empathy and adapt naturally to human emotions and context.
- Such systems do not exist yet; they are still a future possibility.



Chapter 2: Problem-Solving Techniques

1. Defining Intelligence

Categories of Intelligence

1. **Linguistic Intelligence**
   Ability to speak, recognize, and use phonology (sounds), syntax (grammar), and semantics (meaning). Seen in narrators, writers, poets, and orators.
2. **Musical Intelligence**
   Ability to create, understand, and communicate using pitch, rhythm, and sound. Used by musicians, singers, and composers.
3. **Logical–Mathematical Intelligence**
   Ability to use and understand complex, abstract ideas. Applied by scientists, engineers, and mathematicians.
4. **Spatial Intelligence**
   Ability to perceive and manipulate visual/spatial information (create, ransform, rotate images in mind). Used by architects, astronauts, designers, and physicists.
5. **Bodily–Kinaesthetic Intelligence**

Ability to use the body (whole or parts) to solve problems or manipulate objects. Common in dancers, athletes, surgeons, and players.

6. **Intrapersonal Intelligence**
   Ability to understand and reflect on one's own feelings, intentions, and motivations. Example: spiritual leaders like Gautam Buddha.
7. **Interpersonal Intelligence**
   Ability to recognize and differentiate others' emotions, beliefs, and intentions.
   Used by teachers, counsellors, interviewers, and mass communicators.
   A machine/system is said to be artificially intelligent if it can exhibit at least one, or up to all, of these intelligences.

2. Components of Intelligence

Intelligence is the invisible ability to learn, understand, and solve problems, composed of the following:

1) **Reasoning**
2) **Learning**
3) **Problem solving**
4) **Perception**
5) **Linguistic intelligence**

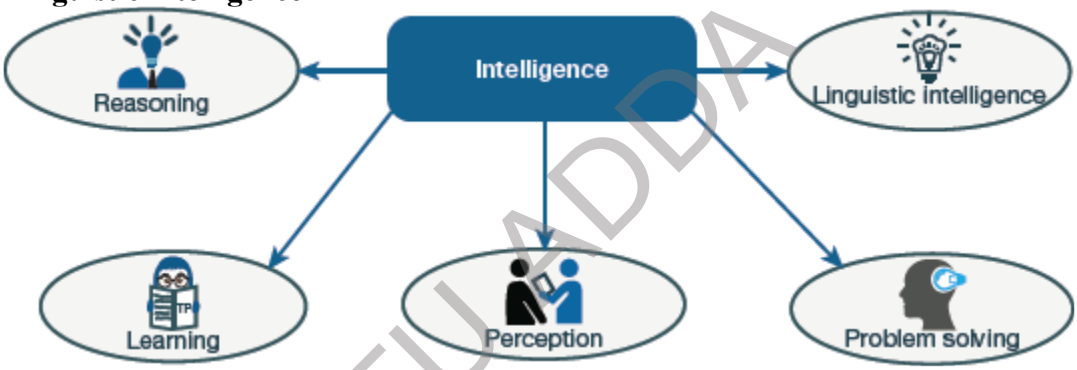


Figure 3: Components of Intelligence.

1. Reasoning

- Reasoning is the process of analyzing information to make decisions, predictions, and conclusions based on evidence.
- There are two types of reasoning: inductive and deductive.

a) Inductive Reasoning: (Observe first, then generalize.)
   - Making generalizations based on specific observations or examples.
   - Starts with specific facts or observations and moves to a general conclusion.
   - Example: You eat three different apples and they are all sweet → You conclude *"All apples must be sweet."*

b) Deductive Reasoning: (Rule first, then apply it.)
   - Drawing a specific conclusion based on general principles or facts.
   - Starts with a general statement or premise and moves to a specific conclusion.
   - Example: All fruits have seeds. An apple is a fruit → Therefore, an apple has seeds.

2. Learning

Learning is the process of gaining knowledge or skills by studying, practicing, or experiencing something. It helps humans, animals, and even AI systems to improve their understanding of different subjects.

There are different types of learning:

1. Auditory Learning:
   Learning by hearing and listening.
   Example: Listening to recorded lectures to understand a concept.
2. Episodic Learning:
   Learning by remembering events or experiences in a specific order.
   Example: Recalling what happened in a sequence, like remembering steps in a recipe.
3. Motor Learning:
   Learning through physical movement of muscles.
   Example: Learning how to pick up objects correctly.
4. Observational Learning:
   Learning by watching and imitating others.
   Example: Children learn by copying their parents' actions.
5. Perceptual Learning:
   Learning by recognizing things that have been seen before.
   Example: Identifying objects and situations based on prior experiences.
6. Relational Learning:
   Learning by recognizing patterns in relationships between things.
   Example: Adjusting the amount of spices in a dish after remembering how much was used last time.
7. Spatial Learning:
   Learning through visual stimuli like images, maps, and colors.
   Example: Creating a mental map of a route before actually driving.
8. Stimulus-Response Learning:
   Learning by reacting to a specific stimulus.
   Example: Shouting when touching a hot pan, because it causes pain.

3. Problem Solving

Problem-solving is the process of finding a solution to a challenge or issue. It involves:

- Identifying the problem: Understanding the situation and recognizing the obstacles (either known or unknown).
- Making decisions: Choosing the best approach or method to overcome the obstacles and reach the goal.

4. Perception

Perception is the process of:

- Acquiring information through the senses (like sight, hearing, etc.).
- Interpreting that information to understand what's happening around us.
- Selecting important details and organizing them to form a clear picture.
- Humans use sensory organs (like eyes, ears) to perceive the world.
- AI systems use sensors (like cameras, microphones) to gather data and understand their environment.

5. **Linguistic Intelligence**

It is used in in interpersonal communication and defines one's ability to use, comprehend, speak and write the verbal and written language.

3. Differences Between Human and Machine Intelligence

Feature	Human Intelligence	Machine Intelligence
Perception	Perceive information by recognizing patterns	Perceive information by analyzing data using rules

Memory & Recall	Store and recall information based on patterns	Use searching algorithms to store and retrieve data
Handling Missing/Distorted Info	Can deduce missing or distorted information	Cannot accurately deduce missing or distorted information
Learning Ability	Can learn from experience and adapt to new situations	Learns only from predefined data and algorithms

4. Agent and Environment

- AI is about studying rational agents – anything that makes decisions to achieve the best result.
- A rational agent can be a person, company, machine, or software that acts intelligently.
- These agents make decisions by considering past and current information (percepts) to choose the best action.
- Agents operate in an environment and may interact with other agents there.
- So, an AI system = an agent + its environment**.**

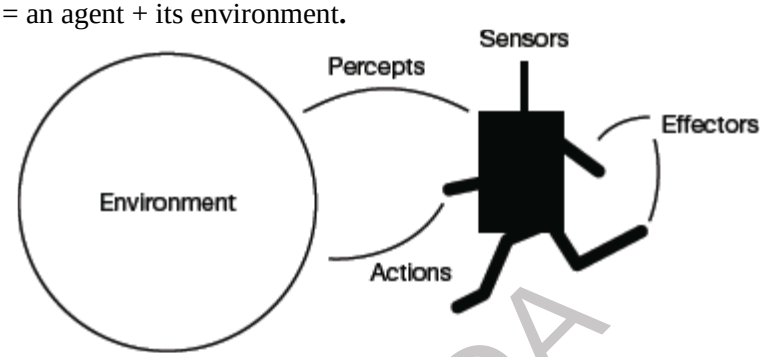


Figure: AI Agent.

AI agents act in their environment, which may include other agents. They perceive their environment using sensors. They act upon the environment using effectors.
The different types of agents in an AI system include the following:
1. H**uman agent:** A human agent has sensory organs (like eyes, ears, nose, tongue, and skin) that act as sensors, and other organs such as hands, legs, and mouth as effectors for taking actions.
2. R**obotic agent:** A robotic agent uses cameras and infrared range finders as sensors, and various motors and actuators as effectors.
3. S**oftware agent:** Such an agent uses bit strings as its programs and actions.

Key Terminology

1. Performance Measure of Agent: It helps determine how successful an agent is based on its actions.
2. Behaviour of Agent: The action performed by an agent after receiving a percept (input).
3. Percept: Perceptual input received by an agent at a specific moment in time.
4. Percept Sequence: A list of all percepts an agent has received up until now.
5. Agent Function: A map that connects the percept sequence to an action performed by the agent.

Rationality

- Rationality is the ability to make responsible and sensible decisions.
- A rational agent makes decisions that maximize its performance based on:
  a) Performance measure (how successful the agent is).
  b) Percept sequence (the inputs it has received).
  c) Prior knowledge (what the agent already knows about the environment).
  d) Possible actions (what the agent can do).
- A rational agent always performs the right action to maximize its performance. Problem Solved by Agent is characterized by (PEAS) Performance measure, Environment, Actuators, and Sensors are used to define a problem that an agent will solve.

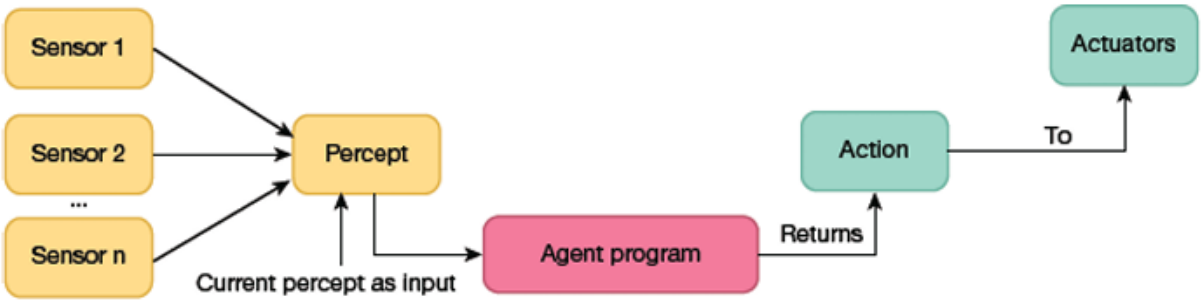


Figure: Rational Agent

Structure of Intelligent Agents

The structure of an intelligent agent can be given as,

Agent = Architecture + Agent Program

where architecture is the machinery on which an agent works, and agent program is an implementation of an agent function.

Types of Agents

1. Simple Reflex Agents
2. Model-Based Reflex Agents
3. Goal-Based Agents
4. Utility-Based Agents
5. Learning Agent

1. Simple Reflex Agents

- Simple reflex agents choose actions based only on the current percept (the data they receive at a specific moment).
- They are rational only if they make the correct decision based on the current percept.
- Working: They use a condition-action rule that maps a state (condition) to an action.
  If the condition is true, the agent performs the action; otherwise, it does nothing.
- Limitations: They require the environment to be fully observable
  If the environment is partially observable, the agent might get stuck in infinite loops.
  In such cases, the agent can only escape the loop if it randomizes its actions.
- Other Issues:
  Simple reflex agents have very limited intelligence.
  They don't know anything about states other than the current one.
  If the environment changes, the rules they follow might need to be updated.

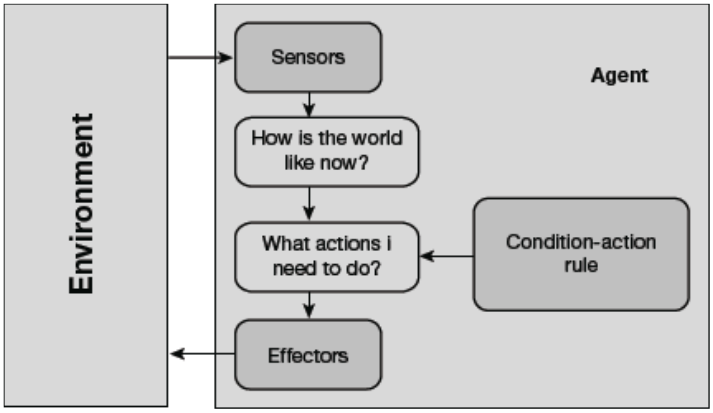


Figure: Simple reflex agent.

2. **Model-Based Reflex Agents**
   - Model-based reflex agents use a model of the world to choose their actions, which requires them to maintain an internal state.
   - Internal State: The internal state represents aspects of the current situation that are not directly observable but can be inferred based on the history of percepts.
   - Working the agent updates its internal state by understanding two things:
     a) How the world evolves (what happens over time in the environment).
     b) How the agent's actions affect the world (the consequences of the agent's actions).

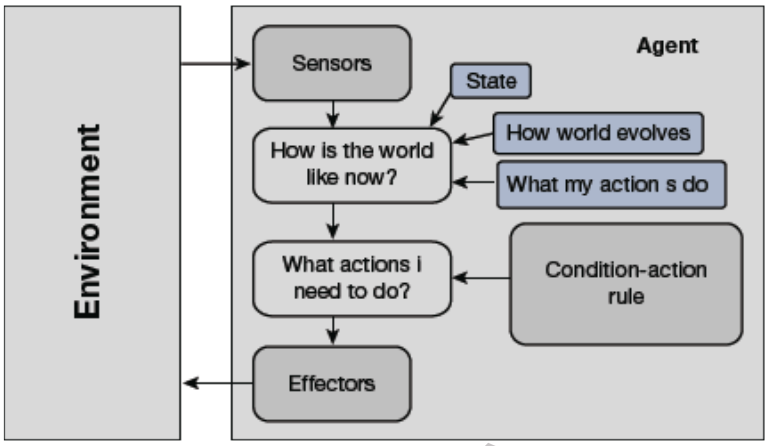


Figure: Model-Based Reflex Agent.

3. **Goal-Based Agents**

A goal is a description of desirable situations. Goal-based agents choose their actions to achieve goals. This provides more flexibility than a reflex agent, as the knowledge supporting a decision is explicitly modelled and permits any sort of modifications.

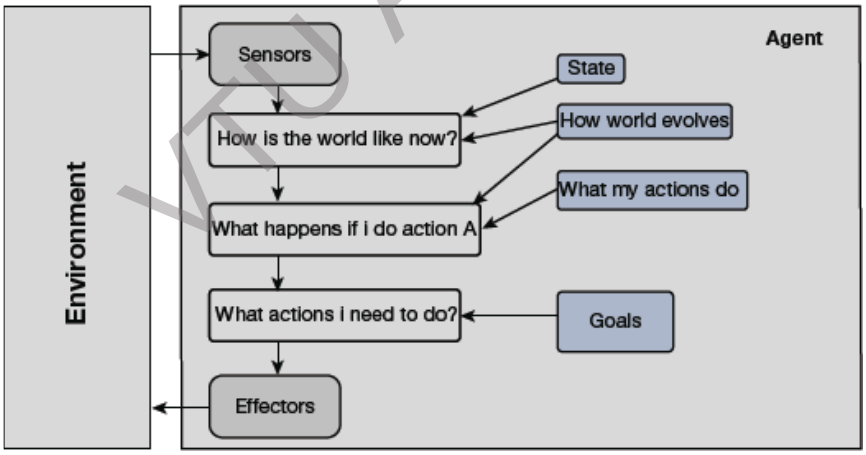


Figure: **Goal-Based Agents**

4. **Utility-Based Agents**
   - Sometimes, goals can clash with each other or be hard to reach. In such cases, **utility-based agents** are used to focus on the goals that matter most.
   - These agents make decisions based on how useful (or important) each possible outcome is. But simply reaching a goal is not always enough — the agent's **happiness or satisfaction** also matters.
   - Here, utility represents how happy the agent feels with the result. So, a utility-based agent picks the action that gives it the **highest level of happiness or satisfaction**.

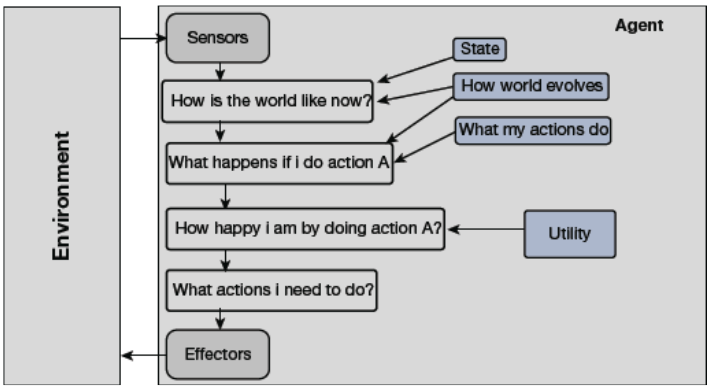


Figure: Utility-Based Agents

5. Learning Agent

A learning agent is an agent that can improve itself by learning from past experiences. At first, it starts with only basic knowledge, but over time, it adapts and gets better automatically through learning.

It is made up of four main parts:

1. Learning element – helps the agent improve by learning from its environment.
2. Critic – gives feedback to the learning element, telling the agent how well it is performing compared to a set standard.
3. Performance element – decides which external action the agent should take.
4. Problem generator – suggests new actions that give the agent fresh and useful experiences.

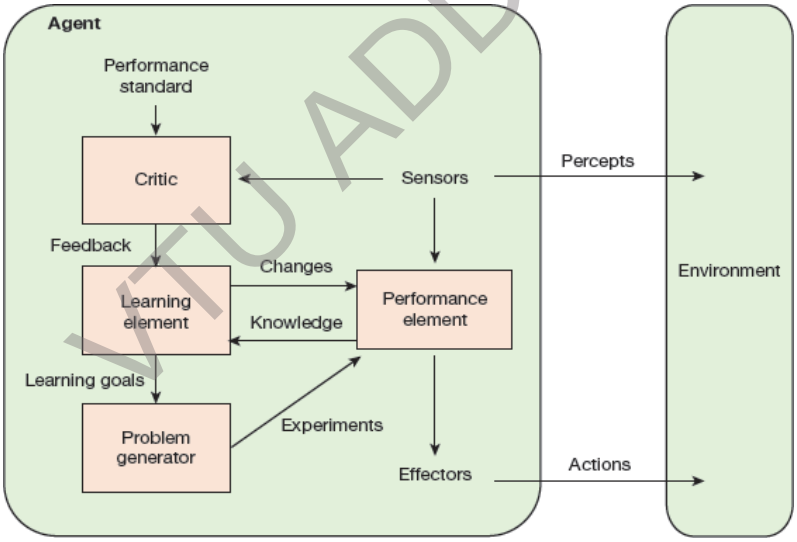


Figure: Learning agent.

The Nature of Environments

- Some AI programs work only with limited inputs and outputs (like keyboard, files, or screen text), while software agents or softbots can operate in much larger, complex environments and make real-time decisions.
- A famous example of such an artificial environment is the Turing Test, where a software agent is tested to see if it can perform and respond like a human.
- Turing test: The Turing Test is used to check if a system shows intelligent behavior. In this test, there are two people and one machine (software agent).
- One person acts as the tester, while the other person and the machine sit in separate rooms. The tester doesn't know which one is the machine. They both answer the tester's questions through typing.

- If the tester cannot tell whether the answers are from a human or a machine, then the machine is considered intelligent.

Types of Environments

1. **Discrete vs. Continuous**
   - **Discrete Environment**: Has a limited number of possible states and actions. The agent can easily count and track them.
     Example: A chess-playing agent works in a discrete environment because the number of moves and positions is finite.
   - **Continuous Environment**: Has an infinite number of states and actions, making it much more complex.
     Example: A self-driving car operates in a continuous environment since speed, position, and traffic situations change continuously.
2. **Known vs. Unknown**
   - **Known Environment**: The agent already knows the outcomes of its actions. Rules are fixed and clear.
     Example: In a chess game, the agent knows what will happen if it moves a piece.
   - **Unknown Environment**: The agent does not know the results of its actions in advance. It must explore, learn, and adapt.
     Example: A robot learning to walk in a new place doesn't know how each step will affect it.
3. **Observable vs. Partially Observable**
   - **Fully Observable Environment**: The agent has complete information about the environment at all times.
     Example: In chess, the agent can see all pieces and positions.
   - **Partially Observable Environment**: The agent only gets limited or noisy information, so it cannot see the whole environment.
     Example: In the Kriegspiel version of chess, players don't see the opponent's pieces directly.
4. **Static vs. Dynamic**
   - **Static Environment**: Does not change while the agent is deciding or acting. Easy to handle.
     Example: A crossword puzzle doesn't change once given.
   - **Dynamic Environment**: Changes over time, even while the agent is thinking or acting. Much harder to handle.
     Example: Self-driving cars must respond to moving traffic and pedestrians.
   - **Semi-Dynamic**: The environment doesn't change itself, but the agent's performance score may change with time.
     Example: A timed chess game—chessboard is static, but the score (time left) decreases.
5. **Single-Agent vs. Multi-Agent**
   - **Single-Agent Environment**: Only one agent is acting, and its performance doesn't depend on others.
     Example: A vacuum cleaner working alone in a room.
   - **Multi-Agent Environment**: Many agents operate together. Actions of one agent affect the others.
     Competitive: Agents work against each other (e.g., chess).

Cooperative: Agents work together for a common goal (e.g., two self-driving taxis avoiding collisions).

6. **Accessible vs. Inaccessible**
   - **Accessible Environment**: Agent's sensors can gather complete and accurate information about the environment.
     Example: A room where a robot can sense exact temperature and light.
   - **Inaccessible Environment**: The agent cannot fully sense the environment and must make decisions with incomplete information.
     Example: Predicting the weather—it's impossible to measure all factors in real-time.

7. **Deterministic vs. Non-Deterministic (Stochastic)**
   - **Deterministic Environment**: The next state of the environment is completely predictable from the current state + action.
     Example: In tic-tac-toe, the board's next state is fully determined by the move.
   - **Non-Deterministic Environment**: The same action may lead to different results (uncertainty exists).
     Example: In Ludo, rolling a dice creates uncertainty.
   - **Stochastic Environment**: The uncertainty is expressed in terms of probability.
     Example: A weather prediction model may say "70% chance of rain."

8. **Episodic vs. Sequential**
   - **Episodic Environment**: Each action is independent; the outcome of one episode does not affect future ones.
     Example: Image classification—each image is separate and does not affect the next.
   - **Sequential (Non-Episodic) Environment**: Current actions affect future outcomes, so the agent needs memory of past events.
     Example: Driving a car or playing chess, where one move affects future possibilities.

5. Search

Searching Algorithm

Artificial Intelligence is about building rational agents that can make smart decisions. To achieve their tasks, these agents often use search algorithms in the background.

For example, single-player games like Sudoku, crossword puzzles, or tile games use search techniques to figure out the next move or reach the solution.

Components of a Search Problem

1. State Space: The set of all possible states the agent can reach.
2. Start State: The initial state where the search begins.
3. Goal Test: A function that checks if the current state is the goal state.
4. Solution: A sequence of actions (plan) that transforms the start state to the goal state, achieved using search algorithms.

Types of Searching Algorithms

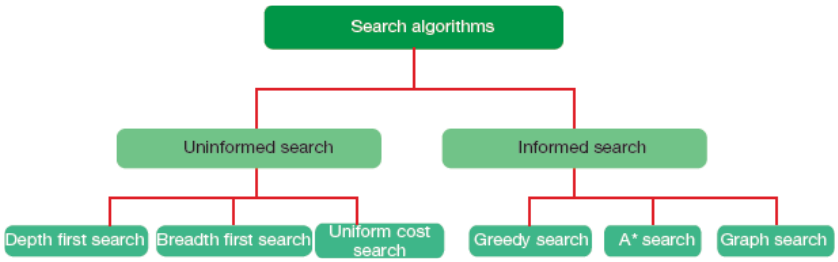


Figure: Different Types of Search Algorithms

Properties of Search Algorithms

1. Completeness: A search algorithm is said to be complete if it returns at least one solution for a particular input.
2. Optimality: A search algorithm is said to give an optimal (best) solution if it has the lowest path cost.
3. Time and space complexity: Time complexity is the time taken by an algorithm to complete a given task, and space complexity is the maximum storage space required to perform the search operation. A good search algorithm takes less time and space to do its work.

6. Uninformed Search Algorithms

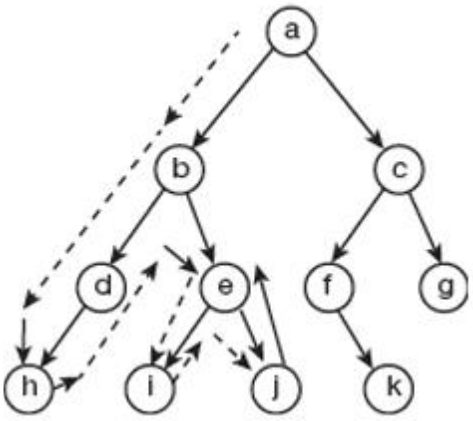
An uninformed (blind) search does not use extra information about the goal. It only knows how to move between nodes and explores blindly, without checking if the path is efficient. There may be many possible paths to reach the goal, differing in order or length of actions.

Key Concepts:

a.Problem Graph: Represents the problem, from the start node (S) to the goal node (G).
b.Strategy: The path taken in the search to reach the goal.
c.Fringe: A data structure that stores all possible states (nodes) that can be reached from the current state.
d.Tree: The path representation that the algorithm follows while searching for the goal node.
e.Solution Plan: The sequence of nodes (states) from start node (S) to goal node (G).
f.Path/Step Cost: Integer values that represent the cost to move from one node to another.

Depth First Search (DFS)

- Depth First Search (DFS) is a simple search algorithm used to explore a tree or graph by starting from the root node and exploring as far as possible along each branch before backtracking.
- Steps of DFS:
  1.Start from the root node: Begin searching from the root node (node A).
  2.Explore each branch: Move from node A to its child node (B), then to the next child (D), and continue exploring until you reach the leaf node (the last node of that branch).
  3.Backtrack: If the key you're looking for isn't found at the leaf node, backtrack to the last node with unexplored branches and explore them.
  4.Repeat the process: Continue exploring each branch by backtracking and then moving to the next unexplored branch, until the entire tree is searched or the goal is found.



Advantages of Depth First Search (DFS)

1. Less Memory Usage: DFS stores only the nodes along the path from the root node to the current node, requiring less memory.
2. Faster to Reach Goal: It often takes less time to find a goal compared to Breadth First Search (BFS), especially when the solution is deep in the tree.

Disadvantages of Depth First Search (DFS)

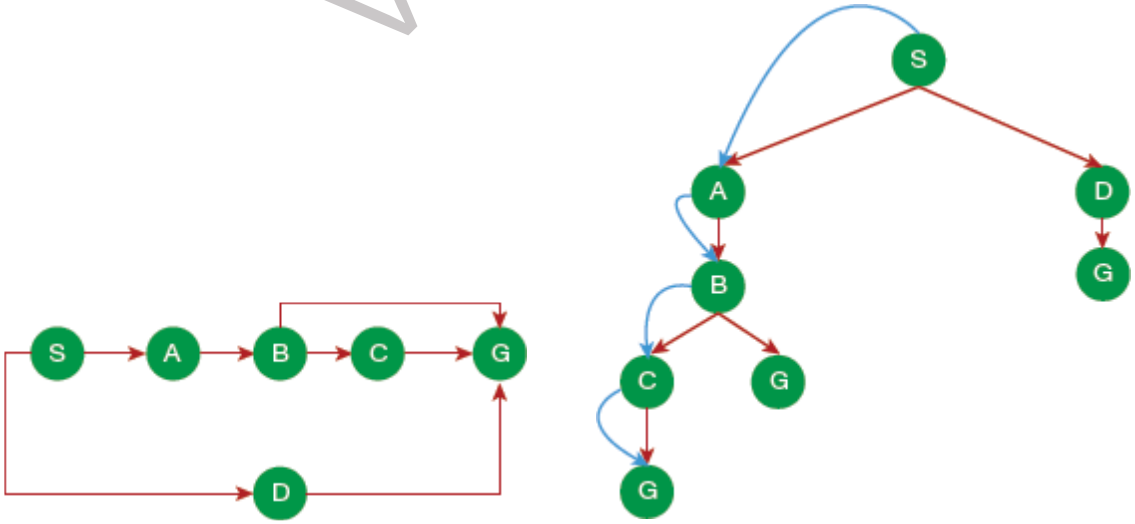
1. Recurring States: Sometimes, many states repeat. In such cases, there's no guarantee of finding the solution.
2. Infinite Loops: DFS may get stuck in an infinite loop when it keeps going deeper. This can be avoided by setting an appropriate cut-off depth,
3. Complexity: The algorithm's complexity depends on the number of paths it needs to explore.
4. Duplicate Nodes: DFS cannot check for duplicate nodes, potentially leading to inefficiency in the search.

Time complexity of the algorithm can be given as, $O(b^m)$. It is equivalent to the number of nodes traversed in DFS. Correspondingly, space complexity is equivalent to how large the fringe can get. It is given as O(bm), where

b is the maximum branching factor in a tree.

d defines the depth of the least-cost solution.

m specifies the maximum depth of the state space. It may be infinity.

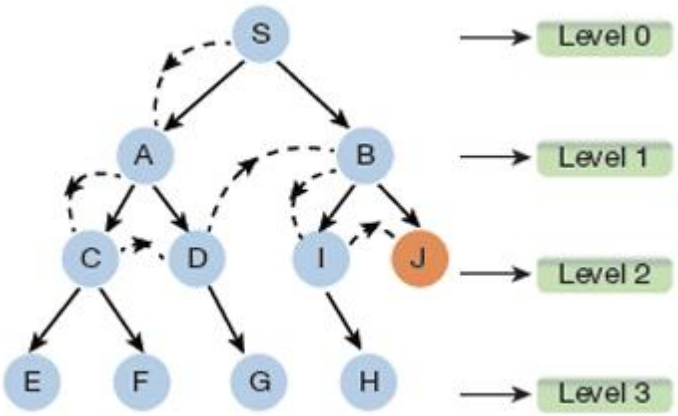


The DFS path can be given as, S -> A -> B -> C -> G

Depth-Limited Search (DLS)

- Depth-Limited Search (DLS) is similar to Depth First Search (DFS), but with a predetermined depth limit to avoid infinite paths. When the search reaches the specified depth limit, nodes at that depth are treated as leaf nodes (i.e., nodes with no successors).
- Termination Conditions of DLS:

1.No Solution: If the problem has no solution, it's called standard error failure.
2.No Solution within Limit: If the solution is not found within the given depth limit, it's called cut-off failure.
3.Solution Found: If the solution is found within the depth limit, the algorithm stops.

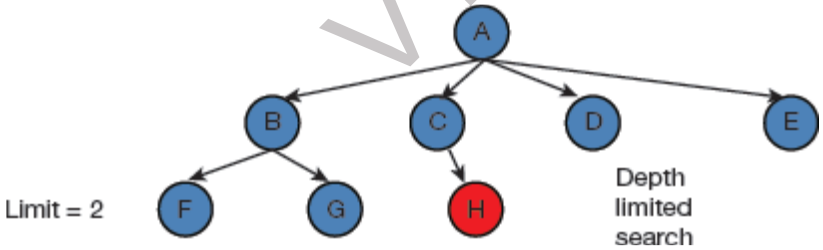


- Advantages

1. It is a memory-efficient algorithm as it consumes less space.
2. The algorithm takes less time to execute.
3. It terminates in finite time.

- Disadvantages

1. It is an incomplete algorithm as we may not be able to get the
solution every time we do the search (even if the solution exists, due to the limit constraint).
2. If multiple solutions of a problem exist, then DLS may not be able to find the optimal solution. In other words, the algorithm is not optimal even if ℓ>d.



Traversal Path:

A → B → (children of B checked) → backtrack

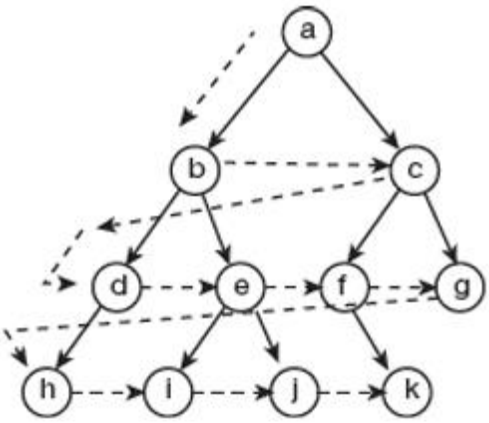
A → C → H → goal found

Breadth First Search (BFS)

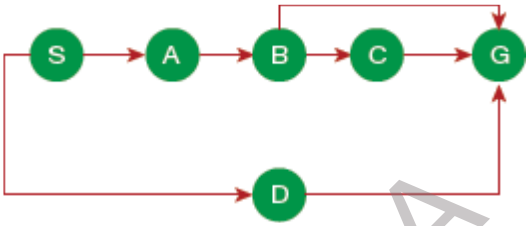
- Breadth First Search (BFS) is an algorithm that traverses the tree breadthwise, meaning it explores all nodes at the current level before moving to the next level.
- How It Works:

1. Start at the root node: The search begins at the root node (node A).
2. Traverse level by level: First, visit the immediate children of the root (nodes B and C).
3. Move to the next level: After visiting all nodes at the current level, the search moves to the next level, visiting nodes D, E, F, and G.

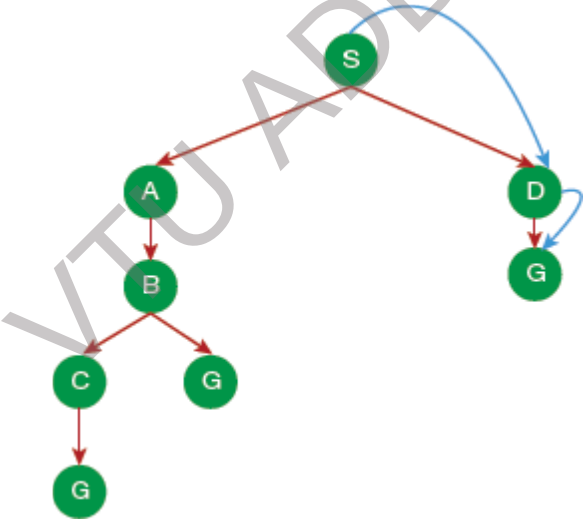
4. Continue level-wise traversal: The algorithm continues this process, exploring all neighbor nodes (children) at each level before moving deeper.



Find the BFS traversal from node S to node G.



Solution: The BFS creates a tree and traverses it using the principle 'shallowest node first'. So, at note S, node D will be traversed followed by node G. Thus, **Path is** S -> D -> G



Uniform Cost Search (UCS)

UCS finds the cheapest path from a start node to a goal node when the step costs are different.

How It Works:

1.It calculates the total cost to reach each node from the start.

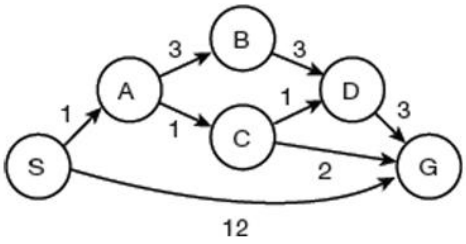
2.Always expands the node with the lowest cumulative cost next.

3.It does not follow depth-first or breadth-first order.

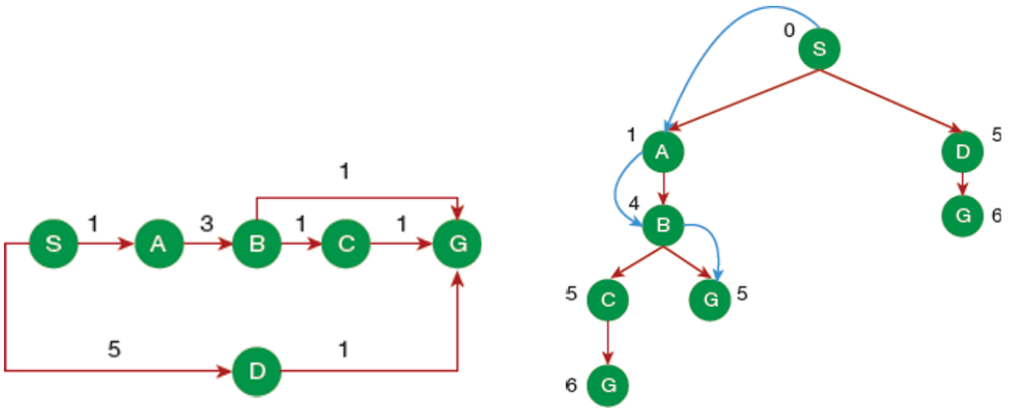
Cost of a Node: cost(node) = sum of costs from start to this node cost(start) = 0

Key Feature:

UCS always finds the optimal (cheapest) path. Works like BFS if all step costs are the same.



Find the cheapest path from S (start) to G (goal) using Uniform Cost Search (UCS).



How UCS Works:
UCS expands nodes based on cumulative cost from the start node.
Always selects the node with the lowest total cost to explore next.
Step-by-Step Traversal:
Start at S → check child nodes A and other options.
Choose A because it has the lowest cumulative cost.
From A, explore B → cost is still lowest.
From B, reach G → goal reached with least total cost.
Path Found:
S → A → B → G
Total Cost:5
Key Idea: UCS always finds the optimal path based on actual costs, not depth or heuristic.

Iterative Deepening Depth-First

- The Iterative Deepening Search (IDDFS) combines DFS and BFS. It works by running DFS with a depth limit, starting from 1 and increasing step by step until the goal is found. This way, it uses the fast search of BFS and the low memory use of DFS. It is especially useful when the search space is large and the depth of the goal is unknown.
- Advantages

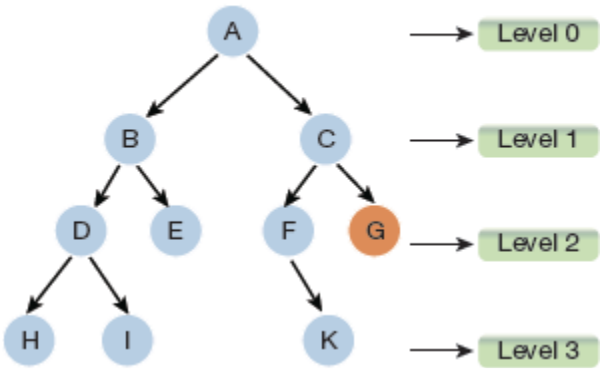
1. It combines the benefits of BFS and DFS search algorithms—fast search and memory efficiency.
2. The algorithm is complete if the branching factor is finite.
3. IDDFS is an optimal algorithm if path cost is a non-decreasing function of the depth of the node.

- Disadvantages

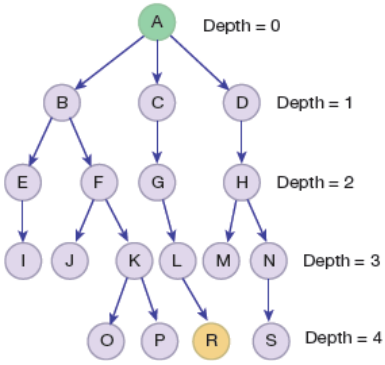
1. It repeats all the work of the previous phase.
2. It takes more time (exponential) to reach the goal node.
3. The algorithm fails when the BFS fails.

Example: Traverse the given tree using the iterative deepening depth-first search algorithm.

Iterative deepening depth first search



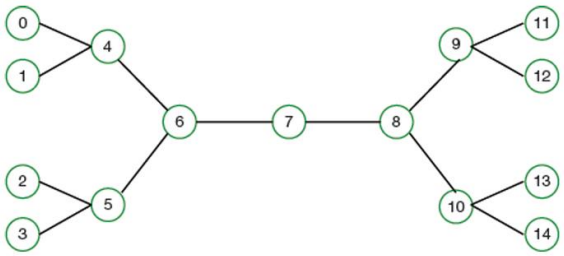
Solution: In the first iteration, node A at level 0 is explored.
In the second iteration, nodes B and C are traversed at level 1.
In the third iteration, nodes D, E, F and G are traversed.
In the fourth iteration, node H is reached.



The tree can be visited as: A B E F C G D H
Depth = {0, 1, 2, 3, 4}

Bidirectional Search

In this search, we explore from both sides at once — one search starts from the initial state and the other from the goal state.
The two searches continue until they meet at a common node.
The final solution is made by joining the path from the start node to the meeting point with the reverse path from the goal node to the meeting point.
Since each search only goes halfway, it is faster and finds the shortest path efficiently.
A heuristic (distance estimate) can also guide the search.



Bidirectional search works best when:

1. Start and goal nodes are unique and clearly defined.
2. Branching factor is the same in both directions (forward and backward).

7. Informed Search Algorithms

Unlike uninformed search, informed search uses extra knowledge about the problem, such as:

- How far the goal is,
- The cost of a path,

- Or the best way to reach the goal. This makes the search faster and more efficient, since the agent explores fewer nodes.

A heuristic function is used, which estimates how close the current state is to the goal. While it may not always give the perfect solution, it usually finds a good solution quickly.

Pure Heuristic Search

- When problems are very large with many possible states, we add problem-specific knowledge to make searching faster and more efficient. This is where heuristic search algorithms work well.
- In a heuristic search: Each node has a heuristic value h(n), which estimates how close it is to the goal.
- The algorithm uses two lists:
  - OPEN list → nodes that are discovered but not yet expanded.
  - CLOSED list → nodes that have already been expanded.
- In every step, the node with the lowest heuristic value is expanded first. Child nodes are evaluated, and only the shorter/better paths are kept, while longer ones are discarded.
- This process continues until the goal state is found.
- We will mainly study two important heuristic search algorithms:
- Best-First Search (Greedy Search)
- A* Search Algorithm

Best-First Search Algorithm/Greedy

To find the goal node quickly by always choosing the most promising path.

Key Idea:

Uses a heuristic function h(n) to estimate which node is closest to the goal.

Combines advantages of DFS (deep exploration) and BFS (level-wise search).

How it Works:

a.Insert the start node in the OPEN list.

b.If OPEN list is empty, stop (goal not found).

c.Remove the node with lowest heuristic value h(n) from OPEN, add it to CLOSED list.

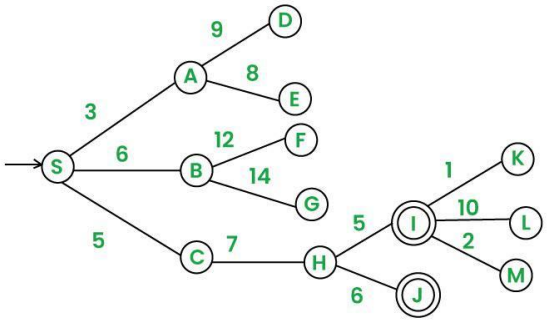
d.Expand this node and generate its successor nodes.

e.If any successor is the goal, stop and return success.

f.If a successor is not in OPEN or CLOSED, add it to OPEN list.

g.Repeat from Step b until goal is found.

Consider the tree given below and traverse it using greedy best-first search algorithm.



Best First Search (Informed Search)

- We start from source "S" and search for goal "I" using given costs and Best First search.
- OPENLIST initially contains S
  - We remove S from OPENLIST and process unvisited neighbors of S to OPENLIST.

- OPENLIST now contains {A, C, B} (C is put before B because C has lesser cost)
- We remove A from OPENLIST and process unvisited neighbors of A to OPENLIST.
  - OPENLIST now contains {C, B, E, D}
- We remove C from OPENLIST and process unvisited neighbors of C to OPENLIST.
- OPENLIST. now contains {B, H, E, D}
- We remove B from OPENLIST and process unvisited neighbors of B to OPENLIST.
  - OPENLIST now contains {H, E, D, F, G}
- We remove H from OPENLIST..
- Since our goal "I" is a neighbor of H, we return.

Chapter 03: Knowledge Representation

1. Introduction

- Artificial intelligence has always fascinated people, often shown in movies and novels as robots that can think and act like humans. But creating machines with human-like behaviour is difficult because humans have a conscience shaped by knowledge, experiences, and memories. To give AI such abilities, we need to provide it with knowledge.
- **Knowledge** of the real world is essential for intelligence and for building AI agents that behave intelligently. An intelligent agent senses its environment and uses knowledge to respond properly. Without knowledge or experience, the agent cannot understand inputs correctly or make the right decisions. Thus, knowledge and intelligence are closely connected.

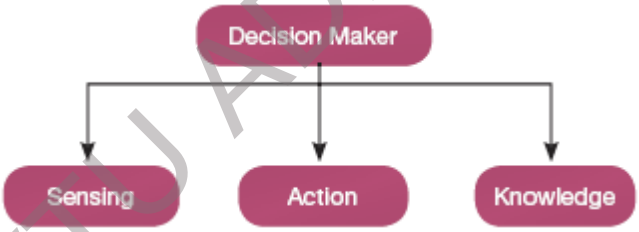


Figure: Relationship between Knowledge and Intelligence

- Humans use their knowledge to understand, reason, and act in the real world. For machines, this ability comes under **knowledge representation and reasoning (KRR)**, a branch of AI that deals with how agents think and act intelligently.
- A knowledge-based system stores real-world information in a form that computers can use to solve problems like medical diagnosis or natural language communication. Unlike simple data storage, knowledge representation helps machines learn, reason, and behave intelligently, similar to humans.

What Knowledge Needs to be Represented?

Humans don't just have knowledge of facts—they also use intuition, beliefs, judgments, intentions, and common sense. To make AI systems more intelligent, we need to represent such knowledge in a way machines can understand. In AI, the following types of knowledge are important:

1. **Objects** – Information about things in the environment.
   Example: In a self-driving car, vehicles, roads, and traffic signals are objects.
2. **Events** – Information about actions that happen in the real world.
   Example: The car applies brakes when another vehicle comes in front.
3. **Performance** – Knowledge of *how* to perform actions or tasks.

Example: Knowing the correct way to change lanes safely.

4. **Meta-knowledge** – Knowledge about knowledge itself (what we know).
5. **Facts** – True statements about the real world that must be stored for intelligent decision-making.
6. **Knowledge Base** – The central component that stores structured information (not plain English sentences) for reasoning and decision-making by the AI system.

What is Knowledge?

Knowledge is the foundation of our brain—it helps us understand things logically and is built from experiences, facts, data, and situations. A person with more knowledge can perform tasks more effectively. Generally, everyone possesses five types of knowledge, as shown in the Figure and explained below.

Types of Knowledge in AI

1. **Meta-Knowledge**
   - Knowledge about knowledge itself (knowing what we know).
2. **Heuristic Knowledge**
   - Knowledge based on experience or expert opinion.
   - Considered a *rule of thumb*—useful but not always guaranteed to work.
   - *Example: A doctor's quick guess about a disease from symptoms based on past cases.*
3. **Procedural Knowledge** (Imperative Knowledge)
   - Knowledge of *how to do something*.
   - Includes rules, strategies, and step-by-step methods.
   - *Example: Knowing how to drive a car or solve a math problem.*
4. **Declarative Knowledge** (Descriptive Knowledge)
   - Knowledge of facts and information about objects, concepts, or events.
   - Usually expressed in declarative sentences.
   - *Example: "The Earth revolves around the Sun."*
5. **Structural Knowledge**
   - Knowledge about the relationships between concepts or objects.
   - Helps in solving complex problems.
   - *Example: Knowing that a "wheel" is a part of a "car," or that a "rose" is a kind of "flower."*

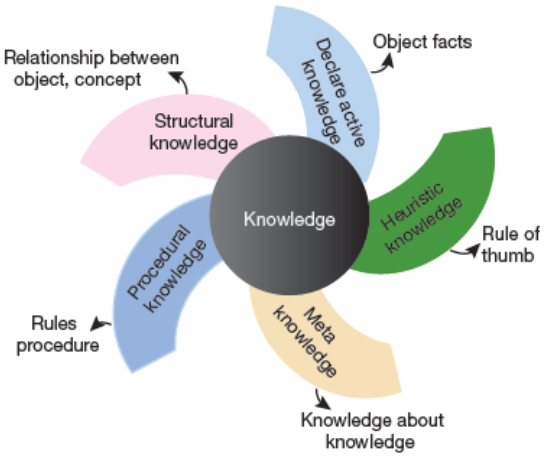


Figure: Types of knowledge

What is Logic?

Logic is an important part of knowledge because it helps us draw conclusions by selecting the right information from many statements. In AI, logic is used to represent knowledge, and it has three main parts:

1. **Syntax** – The rules that define how sentences are formed in a language. (Like grammar in English.)
2. **Semantics** – The meaning of sentences. It checks whether a sentence makes real-world sense. *Example: "Ram is riding a bike" (correct), but "Bike is riding Ram" (incorrect).*
3. **Logical Inference** – The process of reasoning, i.e., using facts and rules to deduce new conclusions or solutions.

   In short, syntax is structure, semantics is meaning, and inference is reasoning.

Cycle of Knowledge Representation in AI

An AI system has different components that work together to show intelligent behavior. These include:

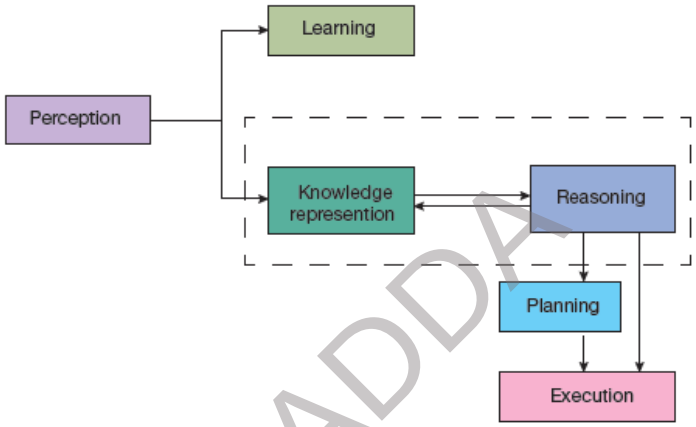


Figure: Components of an AI system.

1. **Perception** – Collects data from the environment using sensors (video, audio, text, temperature, etc.). It also detects noise, damage, and defines responses when something is sensed.
2. **Learning** – Learns from the data collected by perception. Instead of being directly programmed, the system improves itself through knowledge acquisition, inference, heuristics, and faster problem-solving.
3. **Knowledge Representation and Reasoning (KRR)** – Stores and organizes knowledge so the machine can "think" like humans. It defines how the system reasons and uses knowledge to make decisions.

Figure**:** Forms of Knowledge Representation

4. **Planning and Execution** – Uses knowledge and reasoning to create a plan (starting point, preconditions, steps, and goals). Once planned, actions are executed to achieve the desired results.

Together, these components form the **cycle of knowledge representation**, enabling AI to sense, learn, reason, plan, and act intelligently.

Knowledge Representation Requirements

A good knowledge representation system must have the following properties.

1. Representational accuracy, to ensure that the system represents all kinds of required knowledge.
2. Inferential adequacy to manipulate the representational structures to produce new knowledge corresponding to the existing structure.
3. Inferential efficiency to direct the inferential knowledge mechanism to generate appropriate results.
4. Acquisitional efficiency that helps the knowledge representation system to easily acquire new knowledge using automatic methods.

Knowledge-Based Agent

In AI, the agents that mimic a human being's knowledge are known as knowledge-based intelligent agents. Such an agent uses its knowledge and reasoning to act efficiently by making appropriate decisions.

A knowledge-based agent, therefore, does the following:

1. It maintains an internal state of knowledge to represent states, actions, etc.
2. It deduces reasoning with that knowledge.
3. It updates their knowledge after observations by incorporating new precepts.
4. It takes actions accordingly.

The Architecture of Knowledge-Based Agent

A **Knowledge-Based Agent (KBA)** is an intelligent system that represents the world using a formal knowledge representation. It mainly consists of a **Knowledge Base (KB)**, which stores facts about the world as logical sentences, and an **Inference Engine (IE)**, which reasons with those facts to make decisions or derive new knowledge.

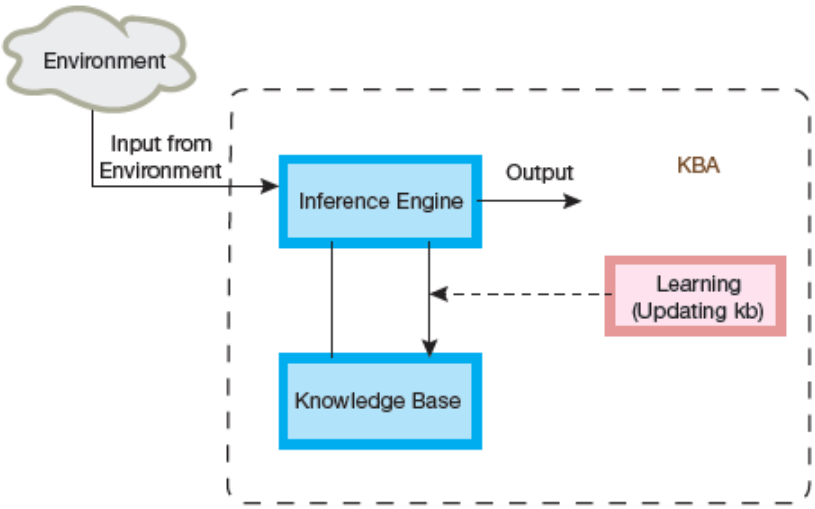


Figure: Architecture of Knowledge-Based Agent

Figure 4.5 illustrates the generalized architecture of a knowledge-based agent (KBA).

Step 1: The KBA perceives its environment and collects input.

Step 2: This input is processed by the inference engine.
Step 3: The inference engine consults the knowledge base to make decisions using the stored information.
The **learning element** of the KBA continuously updates the knowledge base by acquiring new knowledge. The knowledge base itself is a structured collection of facts about the world, expressed in a **knowledge representation language** rather than in natural English.

In AI, a "sentence" does not mean a grammatical English sentence but instead refers to a **proposition or fact** represented in this formal language. The inference system works by deriving new sentences from existing ones and by adding new facts into the knowledge base. This reasoning process uses logical rules such as **forward chaining** or **backward chaining** to deduce new information.

Operations Performed by a Knowledge-Based Agent (KBA)
Whenever a KBA is required to add or update knowledge in the knowledge base (KB), or to respond to queries, it performs a set of intelligent operations:

1. **TELL Operation**
   - Provides the knowledge base with information about the environment.
   - Adds new knowledge or updates existing knowledge in the KB.
   - Also informs the KB about which action the agent has selected and executed.
2. **ASK Operation**
   - Queries the knowledge base to determine what action should be performed.
   - The agent receives guidance or answers from the KB through this operation.
3. **PERFORM Operation**
   - Executes the action selected based on the knowledge and reasoning process.

Generic Knowledge-Based Agent
Figure illustrates the structure of a **generic knowledge-based agent program**.

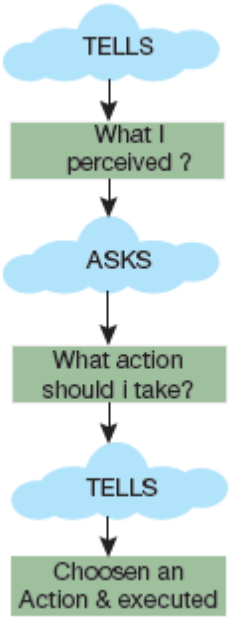


Figure: Mechanism of an agent program

The agent takes **perceptions** from the environment as input and returns an **action** as output. To function effectively, it maintains a **knowledge base (KB)** containing background knowledge about

the real world. Additionally, it keeps a **time counter (t)**, initialized to zero, to track the sequence of operations.

Each time the agent program is executed, it performs the following steps:

1. **TELL the KB about perceptions**
   - The agent informs the KB of what it has perceived at the current time.
   - The **MAKE-PERCEPT-SENTENCE** function generates a sentence that represents this information.
2. **ASK the KB for an action**
   - The agent queries the KB to determine the action it should perform at the current time.
   - The **MAKE-ACTION-QUERY** function is used to generate this query sentence.
3. **TELL the KB about the chosen action**
   - Once the agent selects an action, it updates the KB with this information.
   - The **MAKE-ACTION-SENTENCE** function generates a sentence representing the chosen action.

Various Levels of Knowledge-Based Agent

A KBA can be understood at different levels:

1. **Knowledge Level**
   - Specifies **what the agent knows** and **what its goals are**.
   - Helps determine the agent's overall behavior.
   - *Example*: If the agent must go from **A → B** and knows the shortest path, this knowledge is stored here.
2. **Logical Level**
   - Focuses on the **representation of knowledge**.
   - Sentences are encoded into logical forms for reasoning.
   - *Example*: The agent uses logical reasoning to deduce the path from **A → B**.
3. **Implementation Level**
   - The **physical realization** of knowledge and logic.
   - Involves actual execution of actions.
   - *Example*: The agent physically moves from **A → B** based on its knowledge and logic.

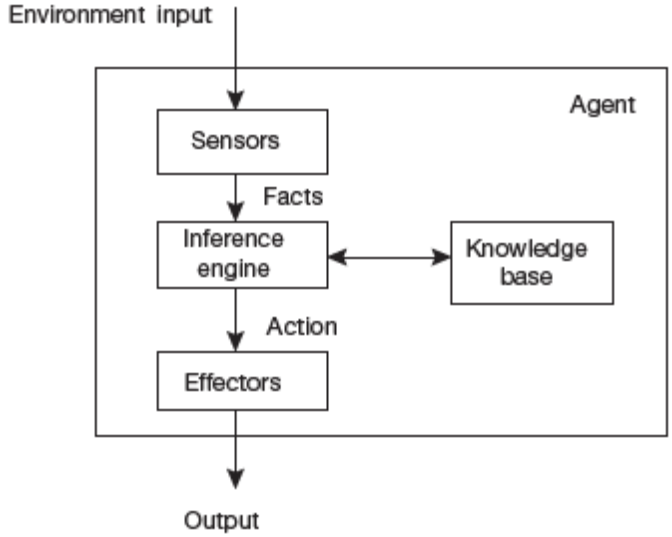


Figure: Aadjusting AC temperature

Illustration (AC example)

- **Knowledge**: AC knows "adjust temperature according to weather."
- **Logic**: Decides **when and how** to adjust.

- **Implementation**: Performs actual adjustment of temperature.

Approaches to Designing a Knowledge-Based Agent

There are mainly two approaches to build a knowledge-based agent.

1. **Declarative Approach**
   - Start with an empty KB, add facts/rules one by one.
   - Focus: *What the agent knows*.
   - Example: Medical system storing symptoms → diseases.
2. **Procedural Approach**
   - Desired behavior coded directly in programs (e.g., LISP, Prolog).
   - Focus: *How the agent acts*.
   - Example: Function directly encodes diagnosis.
3. **Hybrid Approach**
   - Combines declarative (knowledge) + procedural (execution).
   - Focus: Balance between knowledge and efficiency.
   - Example: Chatbot with stored facts + fast algorithms.

Types of Knowledge

1. **Simple Relational Knowledge**
   - Facts are stored in tables/relations (like databases).
   - Shows relationships among entities.
   - *Limitation*: Provides little inference capability.

   (Example: Student database table with names, roll numbers, etc.)
2. **Inheritable Knowledge**
   - Data is stored in a **class hierarchy** (general → specific).
   - Uses **IS-A (instance-class)** relationships.
   - *Example*:
     - Zinya → Undergraduate Student
     - Vidhi → Postgraduate Student
3. **Inferential Knowledge**
   - Represented using **formal logic**.
   - Enables deriving **new facts** from existing ones.
   - *Example*:
     - Statement 1: Diya is a student → Student(Diya)
     - Statement 2: All students are bright → ∀x Student(x) → Bright(x)
     - Therefore: Bright(Diya)