

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR**  
**B.Tech (CSE)– III-I Sem** **L T P C 3 0 0 3**  
**(19A05502T) ARTIFICIAL INTELLIGENCE**  
**(Common to CSE & IT)**

**Course Objectives:**

This course is designed to:

- Define Artificial Intelligence and establish the cultural background for study
- Understand various learning algorithms
- Explore the searching and optimization techniques for problem solving
- Provide basic knowledge on Natural Language Processing and Robotics

**Unit – I: Introduction:** What is AI, Foundations of AI, History of AI, The State of Art.

**Intelligent Agents:** Agents and Environments, Good Behaviour: The Concept of Rationality, The Nature of Environments, The Structure of Agents.

**Learning Outcomes:**

At the end of the unit, students will be able to:

- Recognize the importance of Artificial Intelligence (L1)
- Identify how intelligent agent is related to its environment (L2)
- Build an Intelligent agent (L3)

**Unit – II: Solving Problems by searching:** Problem Solving Agents, Example problems, Searching for Solutions, Uninformed Search Strategies, Informed search strategies, Heuristic Functions, Beyond Classical Search: Local Search Algorithms and Optimization Problems, Local Search in Continuous Spaces, Searching with Nondeterministic Actions, Searching with partial observations, online search agents and unknown environments.

**Learning Outcomes:**

At the end of the unit, students will be able to:

- Explain how an agent can formulate an appropriate view of the problem it faces. (L2)
- Solve the problems by systematically generating new states (L2)
- Derive new representations about the world using process of inference (L5)

**Unit – III: Reinforcement Learning:** Introduction, Passive Reinforcement Learning, Active Reinforcement Learning, Generalization in Reinforcement Learning, Policy Search, applications of RL

**Natural Language Processing:** Language Models, Text Classification, Information Retrieval, Information Extraction.

**Learning Outcomes:**

At the end of the unit, students will be able to:

- Examine how an agent can learn from success and failure, reward and punishment. (L5)
- Develop programs that make queries to a database, extract information from texts, and retrieve relevant documents from a collection using Natural Language Processing. (L6)

**Unit-IV: Natural Language for Communication:** Phrase structure grammars, Syntactic

Analysis, Augmented Grammars and semantic Interpretation, Machine Translation, Speech Recognition

**Perception:** Image Formation, Early Image Processing Operations, Object Recognition by appearance, Reconstructing the 3D World, Object Recognition from Structural information, Using Vision.

**Learning Outcomes:**

At the end of the unit, students will be able to:

- Develop programs that translate from one language to another, or recognize spoken words. (L6)
- Explain the techniques that provide robust object recognition in restricted context.(L2)

**Unit-V: Robotics:** Introduction, Robot Hardware, Robotic Perception, Planning to move, planning uncertain movements, Moving, Robotic software architectures, application domains

**Philosophical foundations:** Weak AI, Strong AI, Ethics and Risks of AI, Agent Components, Agent Architectures, Are we going in the right direction, What if AI does succeed.

**Learning Outcomes:**

At the end of the unit, students will be able to:

- Explain the role of Robot in various applications. (L2)
- List the main philosophical issues in AI. (L1)

**Course outcomes:**

Upon completion of the course, the students should be able to:

- Apply searching techniques for solving a problem (L3)
- Design Intelligent Agents (L6)
- Develop Natural Language Interface for Machines (L6)
- Design mini robots (L6)
- Summarize past, present and future of Artificial Intelligence (L5)

**Textbook:**

1. Stuart J.Russell, Peter Norvig, “Artificial Intelligence A Modern Approach” , 3rd Edition, Pearson Education, 2019.

**References:**

1. Nilsson, Nils J., and Nils Johan Nilsson. Artificial intelligence: a new synthesis. Morgan Kaufmann, 1998.
2. Johnson, Benny G., Fred Phillips, and Linda G. Chase. "An intelligent tutoring system for the accounting cycle: Enhancing textbook homework with artificial intelligence." Journal of Accounting Education 27.1 (2009): 30-39.

**Unit – I: Introduction:** What is AI, Foundations of AI, History of AI, The State of Art.  
**Intelligent Agents:** Agents and Environments, Good Behaviour: The Concept of Rationality, The Nature of Environments, The Structure of Agents.

---

**Artificial Intelligence:**

**Definition:**

→It is the science and engineering of making intelligent machines, especially intelligent computer programs.

Artificial means →man made

Intelligence means →thinking power

→It is a branch of computer Science by which we create intelligent machine which can behave like a human, think like human and able to make decision.

→With AI you don't need to preprogram the machine to do some work

→You have to create a machine with programmed algorithm which can work with own intelligence.

→With the help of AI,we can create such software or device which can solve real world problems very easily and accuracy such as health issues,marketing,traffic issues.

→With the help of AI,you can create your personal virtual assistant such as google assist,siri etc.

→With the help of AI,you can built such robots which can work in environment where survival of human can be rough.

**GOALS of AI**

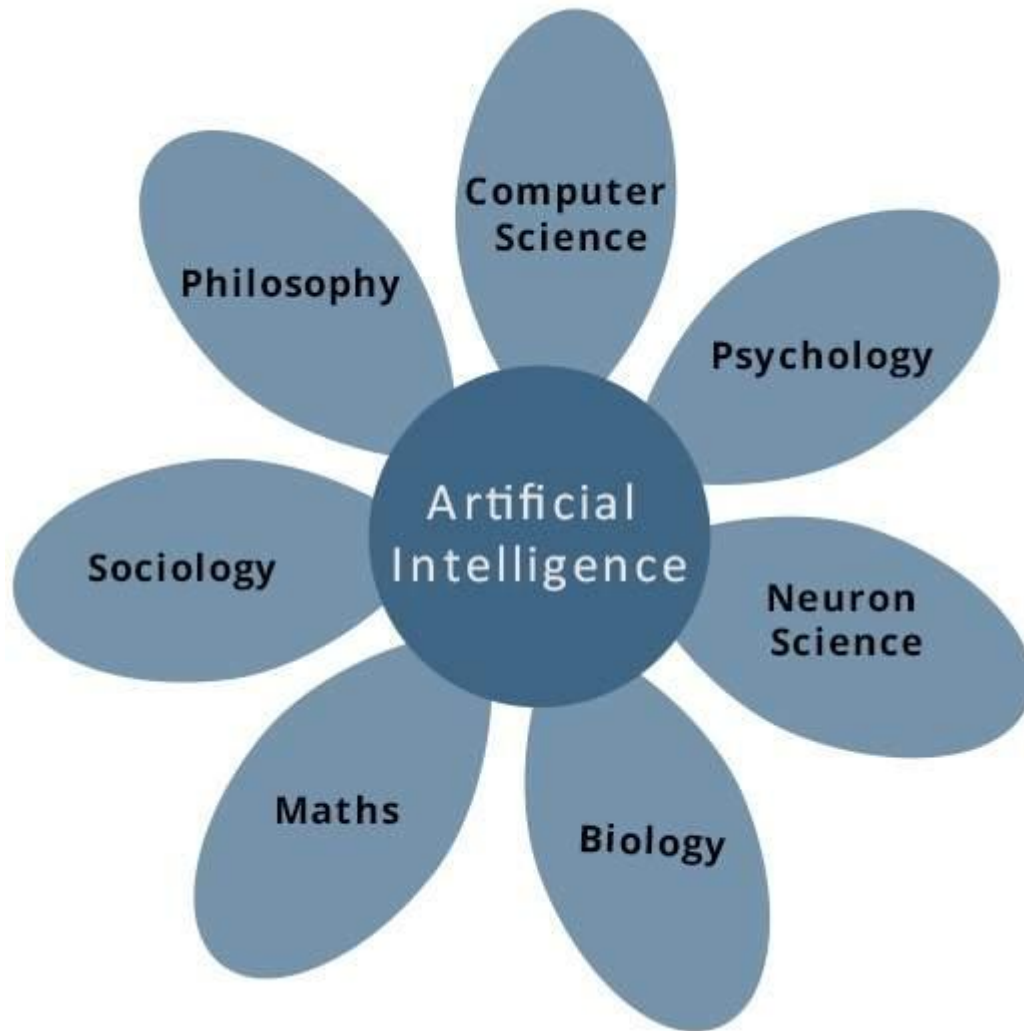
1. Replicate human intelligence
2. Solve knowledge intensive tasks.
3. An intelligent connection of perception and actions
4. Building a machine which can perform tasks that requires human intelligence such as a) providing a theorem b) playing chess c) plan some surgical operations d) driving a car in traffic.

**What Contributes to AI?**

Artificial intelligence is a science and technology based on disciplines such as Computer Science, Biology, Psychology, Linguistics, Mathematics, and Engineering. A major thrust of AI is in the development of computer functions associated with human intelligence, such as reasoning, learning, and problem solving.

Out of the following areas, one or multiple areas can contribute to build an intelligent system.

Out of the following areas, one or multiple areas can contribute to build an intelligent system.



## Programming Without and With AI

The programming without and with AI is different in following ways –

Programming Without AI	Programming With AI
A computer program without AI can answer the specific questions it is meant to solve.	A computer program with AI can answer the generic questions

	it is meant to solve.
Modification in the program leads to change in its structure.	AI programs can absorb new modifications by putting highly independent pieces of information together. Hence you can modify even a minute piece of information of program without affecting its structure.
Modification is not quick and easy. It may lead to affecting the program adversely.	Quick and Easy program modification.

## Applications of AI

AI has been dominant in various fields such as –

- **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,
  - A spying aeroplane 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.

- **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.

## History of AI

Here is the history of AI during 20th century –

Year	Milestone / Innovation
1923	Karel Čapek play named “Rossum's Universal Robots” (RUR) opens in London, first use of the word "robot" in English.
1943	Foundations for neural networks laid.
1945	Isaac Asimov, a Columbia University alumni, coined the term Robotics.
1950	Alan Turing introduced Turing Test for evaluation of intelligence and published Computing Machinery and Intelligence. Claude Shannon published Detailed Analysis of Chess Playing as a search.
1956	John McCarthy coined the term Artificial Intelligence. Demonstration of the first running AI program at Carnegie Mellon University.
1958	John McCarthy invents LISP programming language for AI.
1964	Danny Bobrow's dissertation at MIT showed that computers can understand natural language well enough to solve algebra word problems correctly.
1965	Joseph Weizenbaum at MIT built ELIZA, an interactive problem that carries

	on a dialogue in English.
1969	Scientists at Stanford Research Institute Developed Shakey, a robot, equipped with locomotion, perception, and problem solving.
1973	The Assembly Robotics group at Edinburgh University built Freddy, the Famous Scottish Robot, capable of using vision to locate and assemble models.
1979	The first computer-controlled autonomous vehicle, Stanford Cart, was built.
1985	Harold Cohen created and demonstrated the drawing program, Aaron.
1990	Major advances in all areas of AI – Significant demonstrations in machine learning Case-based reasoning Multi-agent planning Scheduling Data mining, Web Crawler natural language understanding and translation Vision, Virtual Reality Games
1997	The Deep Blue Chess Program beats the then world chess champion, Garry Kasparov.
2000	Interactive robot pets become commercially available. MIT displays Kismet, a robot with a face that expresses emotions. The robot Nomad explores remote regions of Antarctica and locates meteorites.

## What is Intelligence?

The ability of a system to calculate, reason, perceive relationships and analogies, learn from experience, store and retrieve information from memory, solve problems, comprehend complex ideas, use natural language fluently, classify, generalize, and adapt new situations.

## Types of Intelligence

As described by Howard Gardner, an American developmental psychologist, the Intelligence comes in multifold –

Intelligence	Description	Example
Linguistic intelligence	The ability to speak, recognize, and use mechanisms of phonology (speech sounds), syntax (grammar), and semantics (meaning).	Narrators, Orators
Musical intelligence	The ability to create, communicate with, and understand meanings made of sound, understanding of pitch, rhythm.	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.	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.	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.	Players, Dancers
Intra-personal intelligence	The ability to distinguish among one's own feelings, intentions, and motivations.	Gautam Buddha



Interpersonal intelligence	The ability to recognize and make distinctions among other people's feelings, beliefs, and intentions.	Mass Communicators, Interviewers
----------------------------	--	----------------------------------

You can say a machine or a system is artificially intelligent when it is equipped with at least one and at most all intelligences in it.

## What is Intelligence Composed of?

The intelligence is intangible. It is composed of –

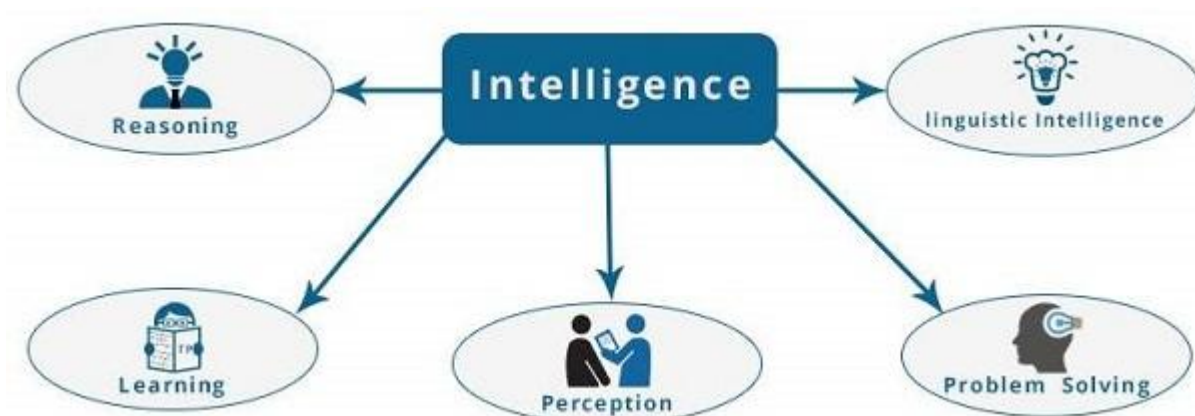
Reasoning

Learning

Problem Solving

Perception

Linguistic Intelligence



Let us go through all the components briefly –

- **Reasoning** – It is the set of processes that enables us to provide basis for judgement, making decisions, and prediction. There are broadly two types –

Inductive Reasoning	Deductive Reasoning
It conducts specific observations to makes broad general statements.	It starts with a general statement and examines the possibilities

	to reach a specific, logical conclusion.
Even if all of the premises are true in a statement, inductive reasoning allows for the conclusion to be false.	If something is true of a class of things in general, it is also true for all members of that class.
Example – "Nita is a teacher. Nita is studious. Therefore, All teachers are studious."	Example – "All women of age above 60 years are grandmothers. Shalini is 65 years. Therefore, Shalini is a grandmother."

- **Learning** – It is the activity of gaining knowledge or skill by studying, practising, being taught, or experiencing something. Learning enhances the awareness of the subjects of the study.

The ability of learning is possessed by humans, some animals, and AI-enabled systems. Learning is categorized as –

- **Auditory Learning** – It is learning by listening and hearing. For example, students listening to recorded audio lectures.
- **Episodic Learning** – To learn by remembering sequences of events that one has witnessed or experienced. This is linear and orderly.
- **Motor Learning** – It is learning by precise movement of muscles. For example, picking objects, Writing, etc.
- **Observational Learning** – To learn by watching and imitating others. For example, child tries to learn by mimicking her parent.
- **Perceptual Learning** – It is learning to recognize stimuli that one has seen before. For example, identifying and classifying objects and situations.
- **Relational Learning** – It involves learning to differentiate among various stimuli on the basis of relational properties, rather than absolute properties. For

Example, Adding ‘little less’ salt at the time of cooking potatoes that came up salty last time, when cooked with adding say a tablespoon of salt.

- **Spatial Learning** – It is learning through visual stimuli such as images, colors, maps, etc. For Example, A person can create roadmap in mind before actually following the road.
- **Stimulus-Response Learning** – It is learning to perform a particular behavior when a certain stimulus is present. For example, a dog raises its ear on hearing doorbell.
- **Problem Solving** – It is the process in which one perceives and tries to arrive at a desired solution from a present situation by taking some path, which is blocked by known or unknown hurdles.

Problem solving also includes decision making, which is the process of selecting the best suitable alternative out of multiple alternatives to reach the desired goal are available.

- **Perception** – It is the process of acquiring, interpreting, selecting, and organizing sensory information.

Perception presumes sensing. In humans, perception is aided by sensory organs. In the domain of AI, perception mechanism puts the data acquired by the sensors together in a meaningful manner.

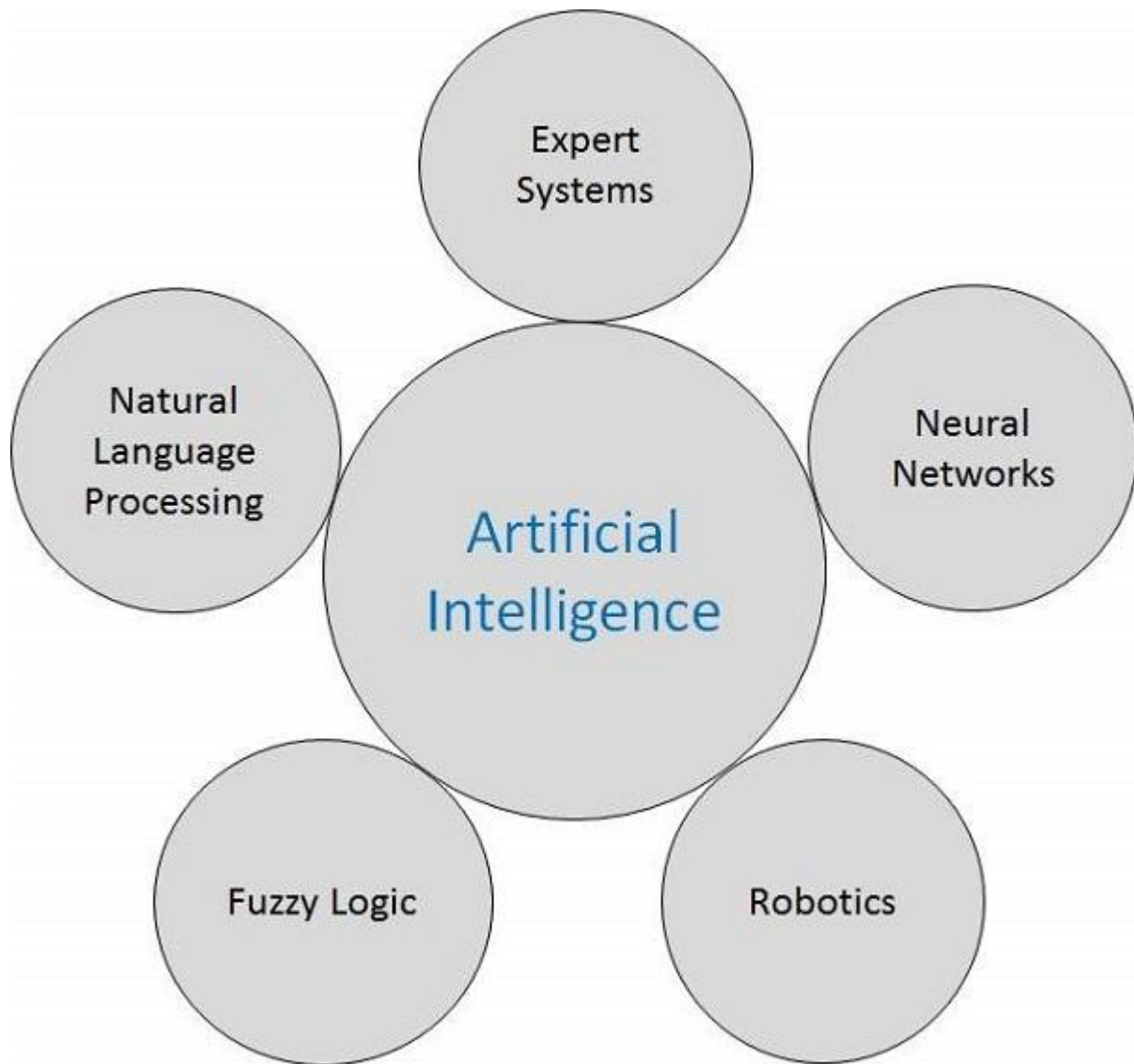
- **Linguistic Intelligence** – It is one’s ability to use, comprehend, speak, and write the verbal and written language. It is important in interpersonal communication.

### **Difference between Human and Machine Intelligence**

- Humans perceive by patterns whereas the machines perceive by set of rules and data.
- Humans store and recall information by patterns; machines do it by searching algorithms. For example, the number 40404040 is easy to remember, store, and recall as its pattern is simple.
- Humans can figure out the complete object even if some part of it is missing or distorted; whereas the machines cannot do it correctly.

## Artificial Intelligence - Research Areas

The domain of artificial intelligence is huge in breadth and width. While proceeding, we consider the broadly common and prospering research areas in the domain of AI –



## Speech and Voice Recognition

These both terms are common in robotics, expert systems and natural language processing. Though these terms are used interchangeably, their objectives are different.

Speech Recognition	Voice Recognition
The speech recognition aims at understanding and comprehending <b>WHAT</b> was spoken.	The objective of voice recognition is to recognize <b>WHO</b> is speaking.
It is used in hand-free computing, map, or menu navigation.	It is used to identify a person by analysing its tone, voice pitch, and accent, etc.

Machine does not need training for Speech Recognition as it is not speaker dependent.	This recognition system needs training as it is person oriented.
Speaker independent Speech Recognition systems are difficult to develop.	Speaker dependent Speech Recognition systems are comparatively easy to develop.



### Working of Speech and Voice Recognition Systems




The user input spoken at a microphone goes to sound card of the system. The converter turns the analog signal into equivalent digital signal for the speech processing. The database is used to compare the sound patterns to recognize the words. Finally, a reverse feedback is given to the database.

This source-language text becomes input to the Translation Engine, which converts it to the target language text. They are supported with interactive GUI, large database of vocabulary, etc.

### Real Life Applications of Research Areas

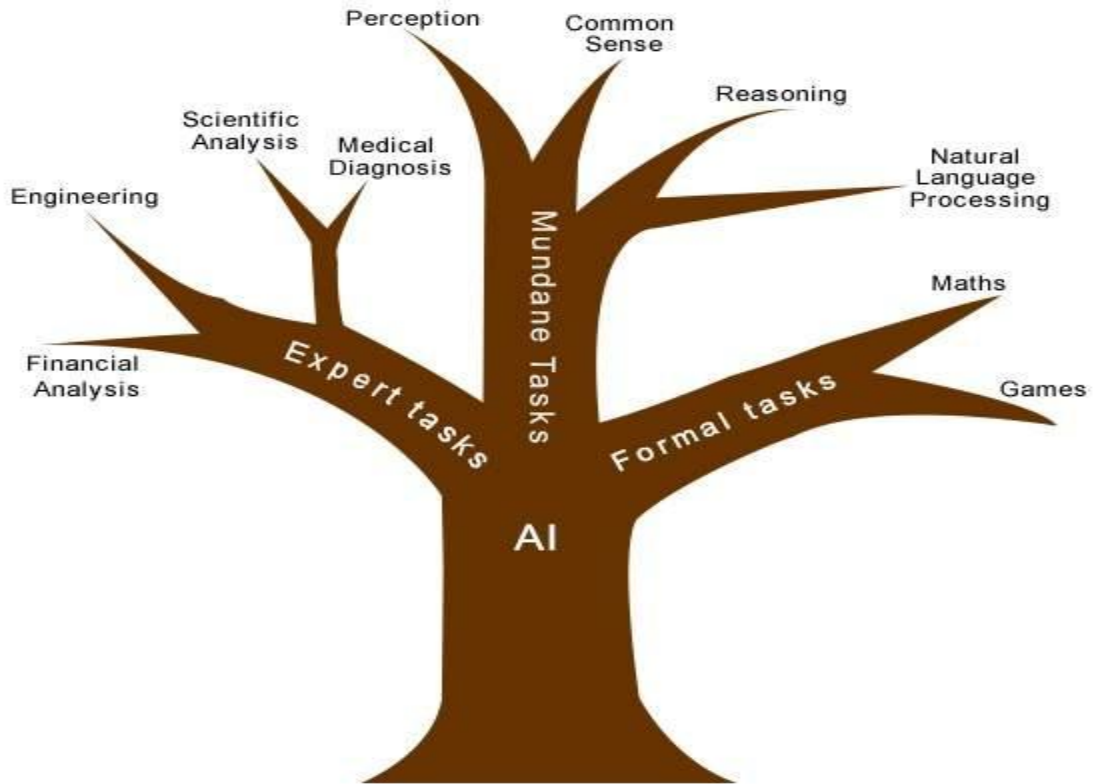
There is a large array of applications where AI is serving common people in their day-to-day lives –

Sr.No.	Research Areas	Real Life Application
1	<p><b>Expert Systems</b></p> <p>Examples – Flight-tracking systems, Clinical systems.</p>	
2	<p><b>Natural Language Processing</b></p> <p>Examples: Google Now feature, speech recognition, Automatic voice output.</p>	

<p>3</p>	<p><b>Neural Networks</b></p> <p>Examples – Pattern recognition systems such as face recognition, character recognition, handwriting recognition.</p>	
<p>4</p>	<p><b>Robotics</b></p> <p>Examples – Industrial robots for moving, spraying, painting, precision checking, drilling, cleaning, coating, carving, etc.</p>	
<p>5</p>	<p><b>Fuzzy Logic Systems</b></p> <p>Examples – Consumer electronics, automobiles, etc.</p>	

## Task Classification of AI

The domain of AI is classified into **Formal tasks**, **Mundane tasks**, and **Expert tasks**.



Task Domains of Artificial Intelligence		
Mundane (Ordinary) Tasks	Formal Tasks	Expert Tasks
Perception <ul style="list-style-type: none"> <li>• Computer Vision</li> <li>• Speech, Voice</li> </ul>	<ul style="list-style-type: none"> <li>• Mathematics</li> <li>• Geometry</li> <li>• Logic</li> <li>• Integration and Differentiation</li> </ul>	<ul style="list-style-type: none"> <li>• Engineering</li> <li>• Fault Finding</li> <li>• Manufacturing</li> <li>• Monitoring</li> </ul>
Natural Language Processing <ul style="list-style-type: none"> <li>• Understanding</li> <li>• Language Generation</li> <li>• Language Translation</li> </ul>	Games <ul style="list-style-type: none"> <li>• Go</li> <li>• Chess (Deep Blue)</li> <li>• Ccheckers</li> </ul>	Scientific Analysis
Common Sense	Verification	Financial Analysis
Reasoning	Theorem Proving	Medical Diagnosis
Planing		Creativity
Robotics		

- |  |  |  |
|--|--|--|
| <ul style="list-style-type: none"><li>• Locomotive</li></ul> |  |  |
|--|--|--|

Humans learn **mundane (ordinary) tasks** since their birth. They learn by perception, speaking, using language, and locomotives. They learn Formal Tasks and Expert Tasks later, in that order.

For humans, the mundane tasks are easiest to learn. The same was considered true before trying to implement mundane tasks in machines. Earlier, all work of AI was concentrated in the mundane task domain.

Later, it turned out that the machine requires more knowledge, complex knowledge representation, and complicated algorithms for handling mundane tasks. This is the reason **why AI work is more prospering in the Expert Tasks domain** now, as the expert task domain needs expert knowledge without common sense, which can be easier to represent and handle.

### Applications of AI

1. Finance
2. Music
3. Transportation
4. Toys & Games
5. Hospitals
6. Expert Systems

## AI - Agents & Environments

AI Agent can have mental properties like knowledge, belief, intention etc

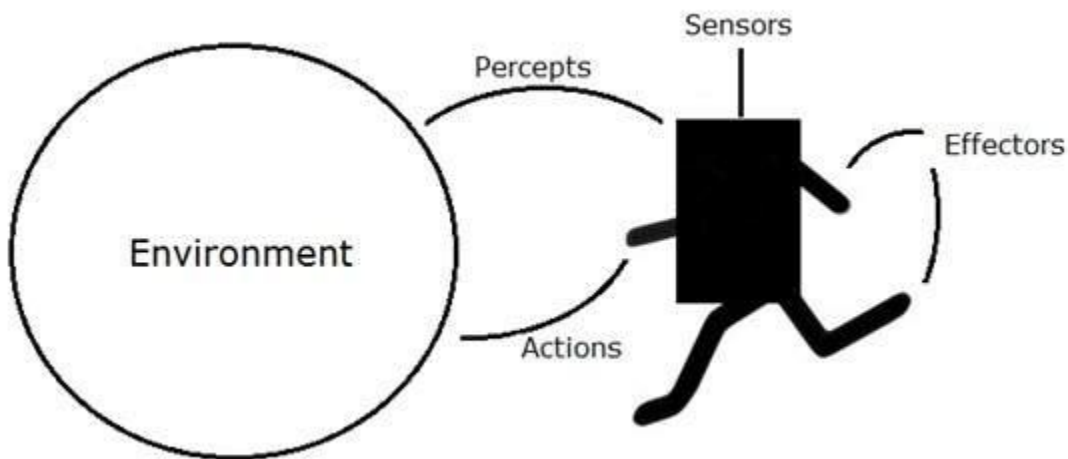
An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents.

### What are Agent and Environment?

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

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





## Agent Terminology

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

## Rationality

Rationality is nothing but status of being reasonable, sensible, and having good sense of judgment.

Rationality is concerned with expected actions and results depending upon what the agent has perceived. Performing actions with the aim of obtaining useful information is an important part of rationality.

## What is Ideal Rational Agent?

An ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure, on the basis of –

- Its percept sequence
- Its built-in knowledge base

Rationality of an agent depends on the following –

- The **performance measures**, which determine the degree of success.
- Agent’s **Percept Sequence** till now.

- The agent's **prior knowledge about the environment**.
- The **actions** that the agent can carry out.

A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept sequence. The problem the agent solves is characterized by Performance Measure, Environment, Actuators, and Sensors (PEAS).

## The Structure of Intelligent Agents

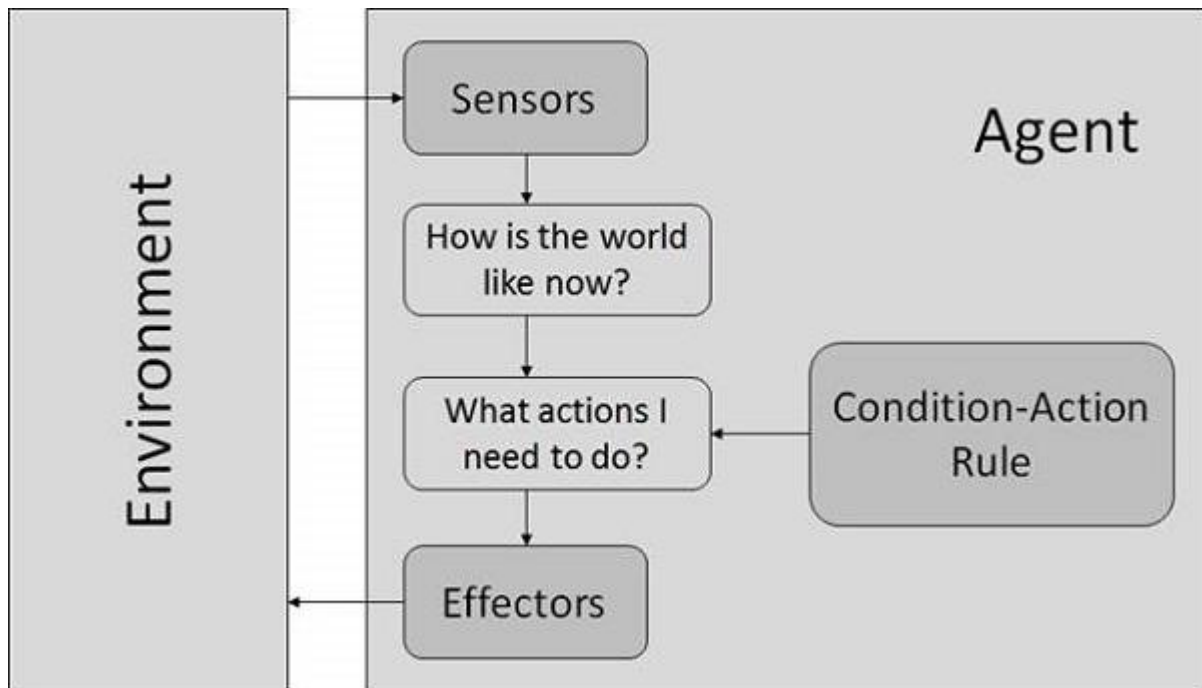
Agent's structure can be viewed as –

- Agent = Architecture + Agent Program
- Architecture = the machinery that an agent executes on.
- Agent Program = an implementation of an agent function.

### Simple Reflex Agents

- They choose actions only based on the current percept.
- They are rational only if a correct decision is made only on the basis of current percept.
- Their environment is completely observable.

**Condition-Action Rule** – It is a rule that maps a state (condition) to an action.



### Model Based Reflex Agents

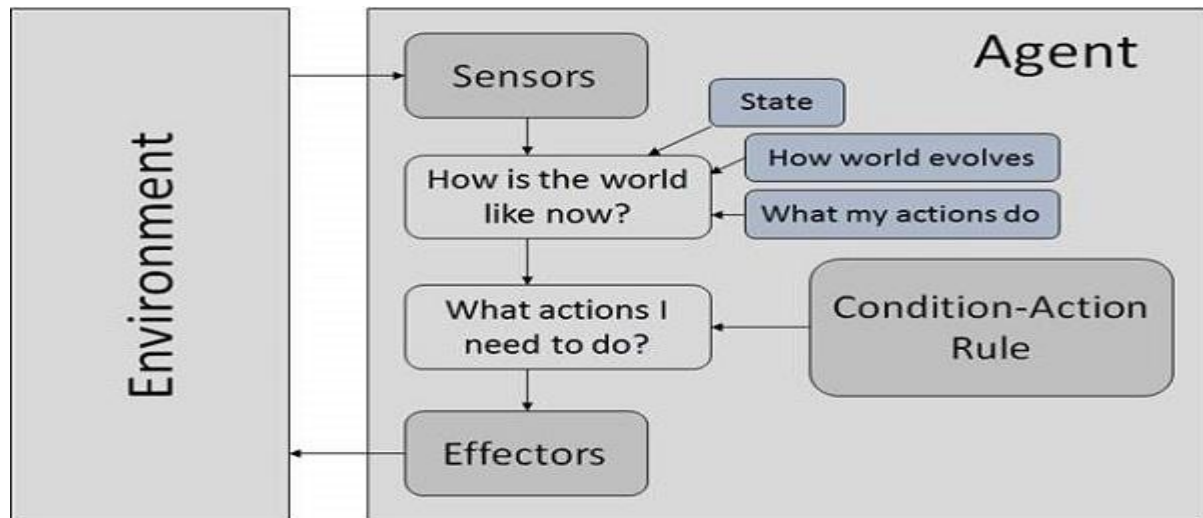
They use a model of the world to choose their actions. They maintain an internal state.

**Model** – knowledge about “how the things happen in the world”.

**Internal State** – It is a representation of unobserved aspects of current state depending on percept history.

**Updating the state requires the information about –**

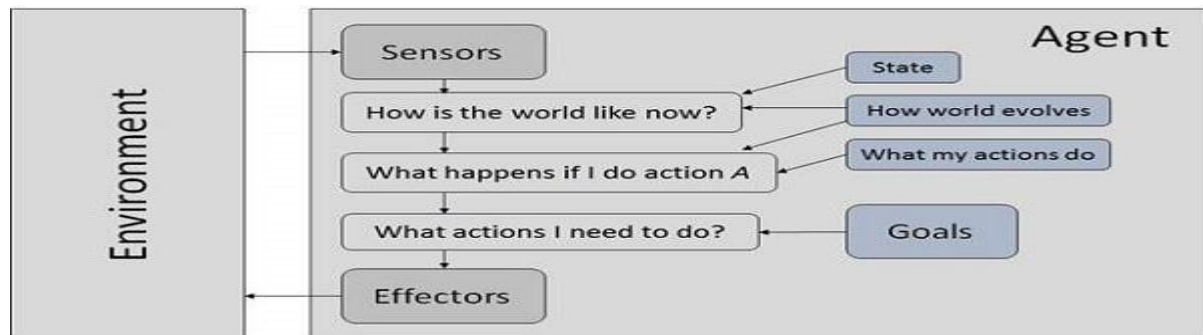
- How the world evolves.
- How the agent’s actions affect the world.



### Goal Based Agents

- They choose their actions in order to achieve goals. Goal-based approach is more flexible than reflex agent since the knowledge supporting a decision is explicitly modeled, thereby allowing for modifications.

**Goal** – It is the description of desirable situations.

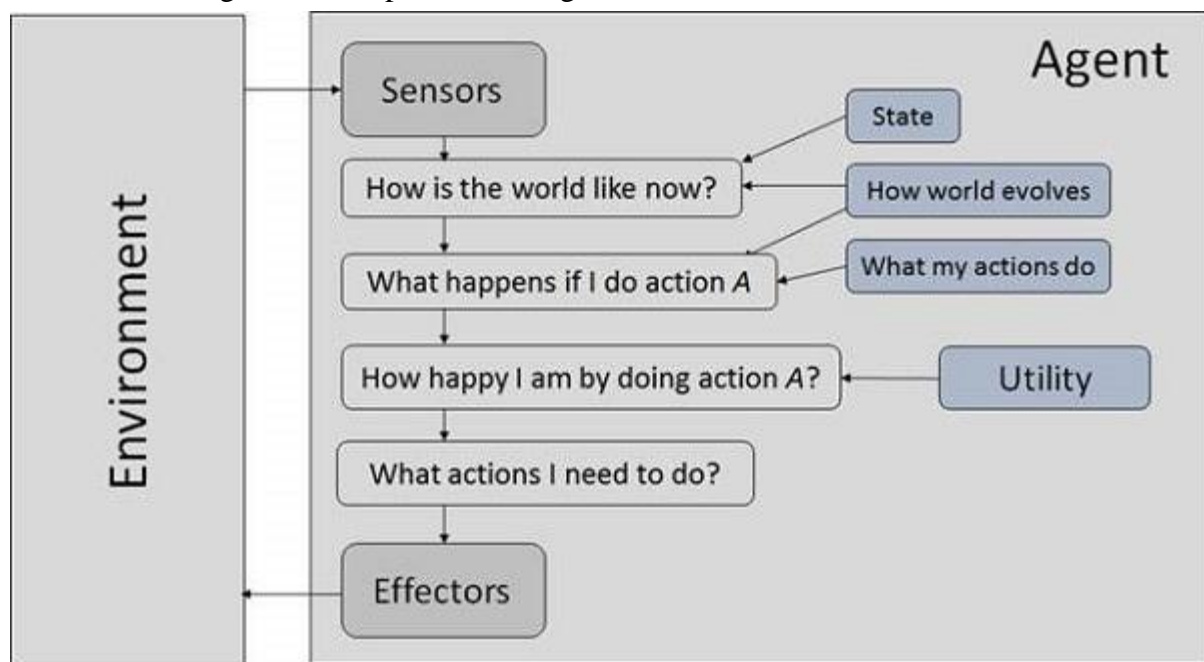


## Utility Based Agents

- They choose actions based on a preference (utility) for each state.

Goals are inadequate when –

- There are conflicting goals, out of which only few can be achieved.
- Goals have some uncertainty of being achieved and you need to weigh likelihood of success against the importance of a goal.



## The Nature of Environments

Some programs operate in the entirely **artificial environment** confined to keyboard input, database, computer file systems and character output on a screen.

In contrast, some software agents (software robots or softbots) exist in rich, unlimited softbots domains. The simulator has a **very detailed, complex environment**. The software agent needs to choose from a long array of actions in real time. A softbot designed to scan the online preferences of the customer and show interesting items to the customer works in the **real** as well as an **artificial** environment.

The most famous **artificial environment** is the **Turing Test environment**, in which one real and other artificial agents are tested on equal ground. This is a very challenging environment as it is highly difficult for a software agent to perform as well as a human.

## Turing Test

The success of an intelligent behavior of a system can be measured with Turing Test.

Two persons and a machine to be evaluated participate in the test. Out of the two persons, one plays the role of the tester. Each of them sits in different rooms. The tester is unaware of who is machine and who is a human. He interrogates the questions by typing and sending them to both intelligences, to which he receives typed responses.

This test aims at fooling the tester. If the tester fails to determine machine's response from the human response, then the machine is said to be intelligent.

## Properties of Environment /Types of Environments:

The environment has multifold properties –

- **Discrete / Continuous** – If there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete (For example, chess); otherwise it is continuous (For example, driving).
- **Observable / Partially Observable** – If it is possible to determine the complete state of the environment at each time point from the percepts it is observable; otherwise it is only partially observable.
- **Static / Dynamic** – If the environment does not change while an agent is acting, then it is static; otherwise it is dynamic.
- **Single agent / Multiple agents** – The environment may contain other agents which may be of the same or different kind as that of the agent.
- **Accessible / Inaccessible** – If the agent's sensory apparatus can have access to the complete state of the environment, then the environment is accessible to that agent.
- **Deterministic / Non-deterministic** – If the next state of the environment is completely determined by the current state and the actions of the agent, then the environment is deterministic; otherwise it is non-deterministic.
- **Episodic / Non-episodic** – In an episodic environment, each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself. Subsequent episodes do not depend on the actions in the previous episodes. Episodic environments are much simpler because the agent does not need to think ahead.

## **Another Material for Agents in AI**

There are multiple approaches that you might take to create Artificial Intelligence, based on what we hope to achieve with it and how will we measure its success. It ranges from extremely rare and complex systems, like self driving cars and robotics, to something that is a part of our daily lives, like face recognition, machine translation and email classification.

The article below gives an insight into what it takes to truly create Artificial Intelligence.

### **So you think you know what is Artificial Intelligence?**

*When you think of Artificial Intelligence, the first thing that comes to mind is either Robots or Machines with Brains*

*The path you take will depend upon what are the goals of your AI and how well you understand the complexity and feasibility of various approaches. In this article we will discuss the approach that is considered more feasible and general for scientific development, i.e. study of the design of rational/intelligent agents.*

### **What is an Agent?**

Anything that can be seen as

- **perceiving its environment through sensors**
- **acting upon it through actuators**

*It will run in cycles of perceiving, thinking and acting.* Take humans for example, we perceive our environment through our five senses(sensors), we think about it and then act using our body parts(actuators). Similarly, robotic agents perceive environment through sensors that we provide them(can be camera, microphones, infrared detectors), they do some computing(think) and then act using various motors/actuators attached for function. Now, it should be clear that the world around you is full of agents like your cell phone, vaccum cleaner, smart fridge, thermostat, camera and even yourself.

### **What is an Intelligent Agent?**

An agent which acts in a way that is expected to maximize to its performance measure, given the evidence provided by what it perceived and whatever built-in knowledge it has.

***The performance measure defines the criterion of success for an agent.***

Such agents are also known as **Rational Agents**. The rationality of the agent is measured by its performance measure, the prior knowledge it has, the environment it can perceive and actions it can perform.

***This concept is central in Artificial Intelligence.***

**The above properties of the intelligent agents are often grouped in the term PEAS, which stands for Performance, Environment, Actuators and Sensors.** So, for example a self driving car would be having following PEAS :-

- **Performance:** Safety, time, legal drive, comfort.
- **Environment:** Roads, other cars, pedestrians, road signs.
- **Actuators:** Steering, accelerator, brake, signal, horn.

- **Sensors:** Camera, sonar, GPS, speedometer, odometer, accelerometer, engine sensors, keyboard.

To satisfy real world use cases, the Artificial Intelligence itself needs to have a wide spectrum of intelligent agents. This introduces diversity in the types of agents and the environments we have.

### **Type of Environments**

A rational agent needs to be designed, keeping in mind the type of environment it will be used in. Below are the types:-

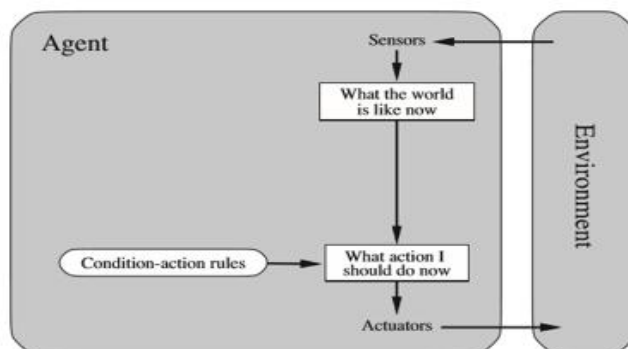
- **Fully observable and partially observable:** An agent's sensors give it *access to the complete state of the environment at each point in time*, if fully observable, otherwise not. For example, **chess is a fully observable environment, while poker is not**
- **Deterministic and Stochastic:** The *next state of the environment is completely determined by the current state and the action executed by the agent*. (If the environment is deterministic except for the actions of other agents, then the environment is strategic). *Stochastic environment is random in nature and cannot be completely determined*. For example, **8-puzzle has a deterministic environment, but driverless car does not**.
- **Static and Dynamic:** The *static environment is unchanged while an agent is deliberating*. (The environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does.). A *dynamic environment, on the other hand, does change*. **Backgammon has static environment and a roomba has dynamic**.
- **Discrete and Continuous :** A limited number of distinct, clearly defined perceptions and actions, constitute a discrete environment. E.g., **checkers is an example of a discrete environment, while self-driving car evolves in a continuous one**.
- **Single agent and Multi-agent :** *An agent operating just by itself has a single agent environment. However if there are other agents involved, then it's a multi agent environment*. **Self-driving cars have multi agent environment**.

There are other types of environments, episodic and sequential, known and unknown, that define scope of an agent.

### **Types of Agents**

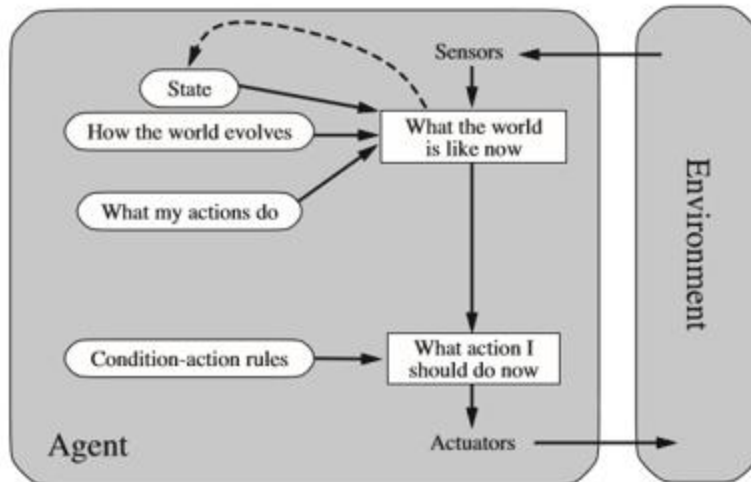
There are 4 types of agents in general, varying in the level of intelligence or the complexity of the tasks they are able to perform. All the types can improve their performance and generate better actions over time. These can be generalized as learning agents.

#### **Simple reflex agents**



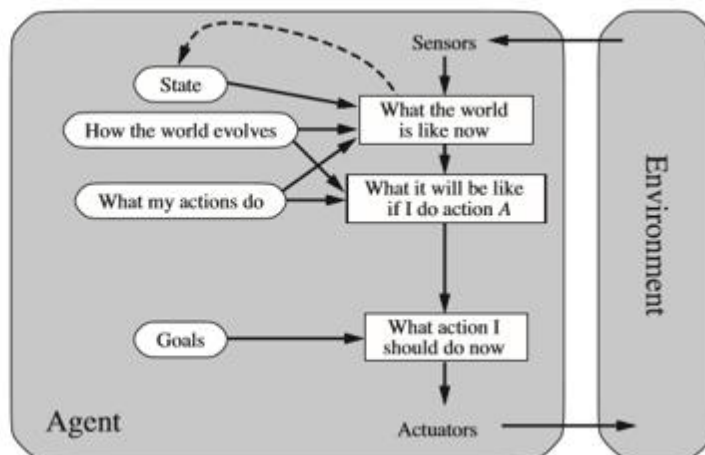
*These select an action based on the current state only, ignoring the history of perceptions. They can only work if the environment is fully observable, or the correct action is based on what is perceived currently.*

### ***Model-based reflex agents***



Agents keep track of partially observable environments. *These* have an internal state depending on perception history. *The environment/ world is modeled based on how it evolves independently from the agent, and how the agent actions affects the world.*

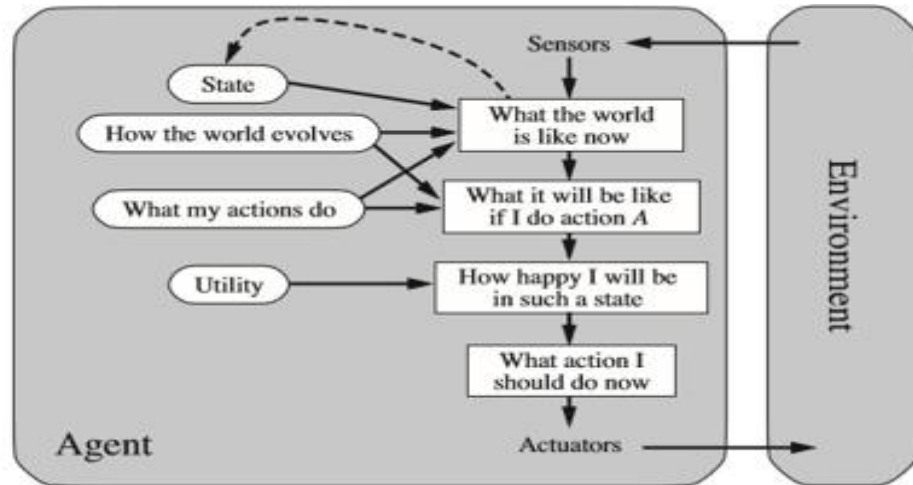
### ***Goal-based agents***



This is an improvement over model based agents, and used in cases where knowing the current state of the environment is not enough. Agents combine the provided goal information with the environment model, to chose the actions which achieve that goal.



## Utility-based agents

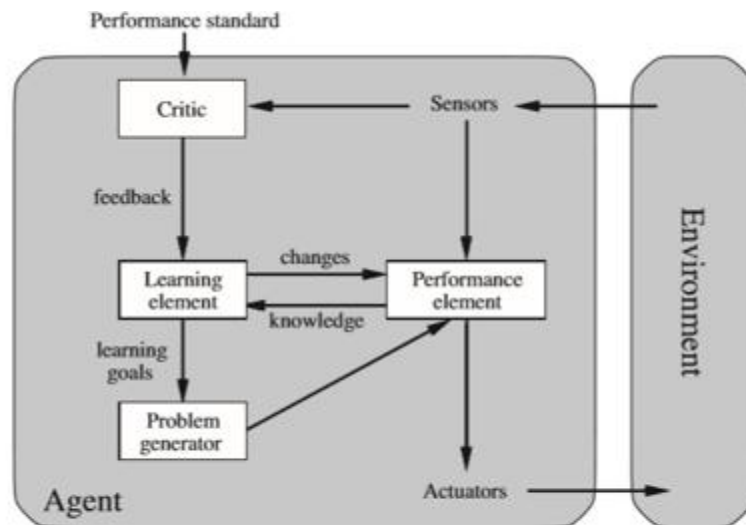


An improvement over goal based agents, helpful when achieving the desired goal is not enough. We might need to consider a cost. For example, we may look for quicker, safer, cheaper trip to reach a destination. This is denoted by a utility function. A utility agent will chose the action that maximizes the expected utility.

A general intelligent agent, also known as learning agent, was proposed by Alan Turing, and is now the preferred method for creating state-of-the-art systems in Artificial Intelligence.

All the agents described above can be generalized into these learning agents to generate better actions.

## Learning Agents

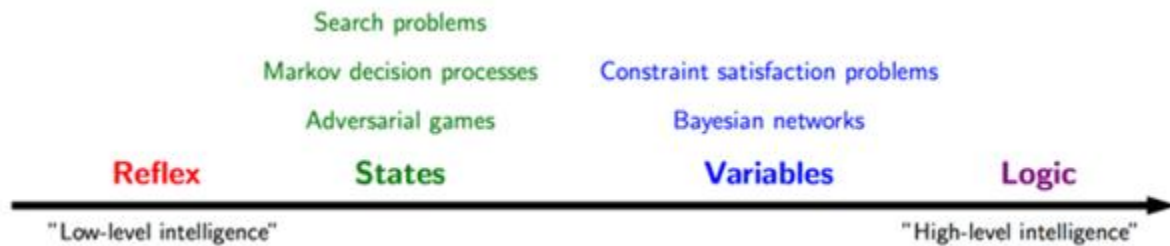


Learning element: responsible for making improvements—Performance element: responsible for selecting external actions. It is what we considered as agent so far.—Critic: How well is the

agent is doing w.r.t. a fixed performance standard.—Problem generator: allows the agent to explore.

## *Internal State Representation*

As the agents get complex, so does their internal structure. The way in which they store the internal state changes. By its nature, a simple reflex agent does not need to store a state, but other types do. The image below provides a high level representation of agent states, in order of increasing expressiveness power(left to right).



Credit : Percy Liang

- **Atomic representation:** In this case, *the state is stored as black box, i.e. without any internal structure.* For example, for Roomba(a robotic vaccum cleaner), the internal state is a patch already vaccumed, you don't have to know anything else. As depicted in the image, such representation works for model and goal based agents and *used in various AI algorithms such as search problems and adversarial games.*
- **Factored Representation:** The state, in this representation, is no longer a black box. ***It now has attribute-value pairs, also known as variables that can contain a value.*** For example, while finding a route, you have a GPS location and amount of gas in the tank. This adds a constraint to the problem. As depicted in the image, such representation works for goal based agents and ***used in various AI algorithms such as constraint satisfaction and bayesian networks.***
- **Structured Representation:** In this representation, ***we have relationships between the variables/ factored states. This induces logic in the AI algorithms.*** For example, in natural language processing, the states are whether the statement contains a reference to a person and whether the adjective in that statement represents that person. The relation in these states will decide, whether the statement was a sarcastic one. ***This is high level Artificial Intelligence, used in algorithms like first order logic, knowledge-based learning and natural language understanding.***

There is much more to these rational agents for Artificial Intelligence, and this was just an overview. As you can tell, the ***study of the design of rational agents is really important part of Artificial Intelligence***, as it has applications in a wide variety of fields. However, these agents don't work on their own, they need an AI algorithm to drive them. Most of these algorithms involve searching.



**Unit – II: Solving Problems by searching:** Problem Solving Agents, Example problems, Searching for Solutions, Uninformed Search Strategies, Informed search strategies, Heuristic Functions, Beyond Classical Search: Local Search Algorithms and Optimization Problems, Local Search in Continuous Spaces, Searching with Nondeterministic Actions, Searching with Partial observations, online search agents and unknown environments.

---

### Problem-solving agents:

In Artificial Intelligence, Search techniques are universal problem-solving methods. **Rational agents** or **Problem-solving agents** in AI mostly used these search strategies or algorithms to solve a specific problem and provide the best result. Problem-solving agents are the goal-based agents and use atomic representation. In this topic, we will learn various problem-solving search algorithms

### Search Algorithm Terminologies:

- **Search:** Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
  - a. **Search Space:** Search space represents a set of possible solutions, which a system may have.
  - b. **Start State:** It is a state from where agent begins **the search**.
  - c. **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
  - d. **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
  - e. **Actions:** It gives the description of all the available actions to the agent.
  - f. **Transition model:** A description of what each action do, can be represented as a transition model.
  - g. **Path Cost:** It is a function which assigns a numeric cost to each path.
  - h. **Solution:** It is an action sequence which leads from the start node to the goal node.
  - i. **Optimal Solution:** If a solution has the lowest cost among all solutions.

### Properties of Search Algorithms:

Following are the four essential properties of search algorithms to compare the efficiency of these algorithms:

**Completeness:** A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

**Optimality:** If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

**Time Complexity:** Time complexity is a measure of time for an algorithm to complete its task.

**Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

## Example problems

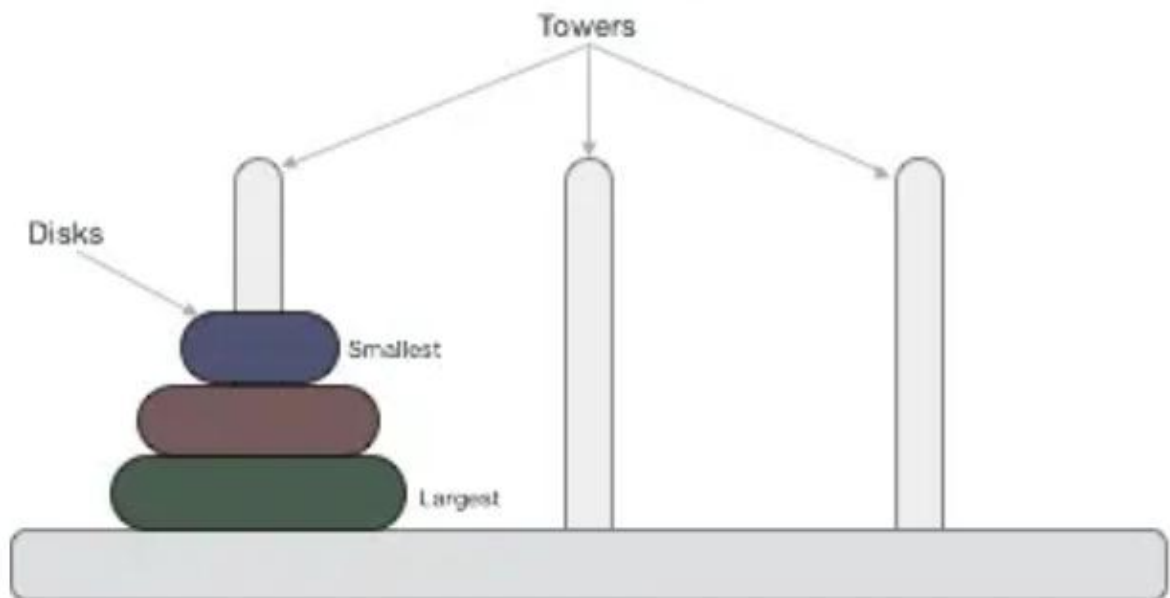
### 1. Problem: (Tower of Hanoi)

Tower of hanoi is mathematical game puzzle where we have three pile (pillars) and n numbers of disk.

#### This game has some rules (Rules of game)

1. Only one disk will move at a time.
2. The larger disk should always be on the bottom and the smaller disk on top of it.(Even during intermediate move)
3. Move only the uppermost disk.
4. All disk move to destination pile from source pile.

So, here we are trying to solve **that how many moves are required to solve a problem** (It depends on number of disk).



#### When we have two disk and 3 pillars (pile, A, B, C)

In the above diagram, following the rule of the game our target is move the disks from source pile (pillar) to the destination pillar. (let's take a look how many steps/ moves are required to make this happen).

**Step1:** Move small disk to the **auxiliary pillar (A)**.

**Step2:** Move large disk to the **Destination pillar (B)**.

### Step3: Move small disk to the **Destination pillar(B).4**

So, basically when we have **2 disks** we required **3** move to reach the destination .

### What if we have **3 , 4, 5...n disks ?**

Eventually, you figure out that there is some pattern to the puzzle and with each increment in disks, the pattern could be followed recursively.

Total move required to reach destination pillar is        means if we have 3 disks we required (4 moves to reach destination pillar) , if 4 disks 8 moves required and so on ...

**Note:** Tower of hanoi problem is an example of **recursion and backtracking**.

**Recursion and backtracking:** When a function calls itself, its called Recursion. (Breaking problem into smaller problems).

Let's suppose we have a problem **A** we subdivided it into **B, C, and D**. **In Backtracking** we attempt solving a sub-problem, and if we don't reach the desired solution, then undo whatever we did for solving that sub-problem, and try solving another sub-problem (**Until we get to the goal**).

**Note:** There must be a **termination condition** in the recursion problems.

### **PROGRAM:**

```
#include<stdio.h>
void TOH(int n,char x,char y,char z) {
    if(n>0) {
        TOH(n-1,x,z,y);
        printf("\n%c to %c",x,y);
        TOH(n-1,z,y,x);
    }
}
int main() {
    int n;
    printf("Enter the number of Rings on disc\n");
    scanf("%d",&n);
    TOH(n,'A','B','C');
}
```

### **Output**

Enter the number of Rings on disc

3

A to B

A to C

B to C  
A to B  
C to A  
C to B  
A to B

## 2. Problem: Water –Jug Problem

Problem Statement: There are 2 jugs with 4L and 3L capacity, you can fill the water in jugs with pump, No marking indicating the levels of jugs. We need to fill the 4L jug with exactly with 2L water capacity.

Assumptions:

1. There is a pump ,we can fill the jugs with it.
2. You can transfer water from one jug to another and vice-versa
3. You can pour water from any of the jug on to the ground
4. No marking mechanism on jugs indicating the levels.

STEPS TO SOLVE:

**S1:** State the description two integer(x,y)

$x \rightarrow 4L$  jug

$y \rightarrow 3L$  jug

$x = \{0,1,2,3,4\}$  ,  $y = \{0,1,2,3\}$  are the values it can take.

**S2:** Describe initial and goal state

Initial State: (0,0)

Goal State: (2,n), where n can be any size of y

**S3:** List all the actions in production rules

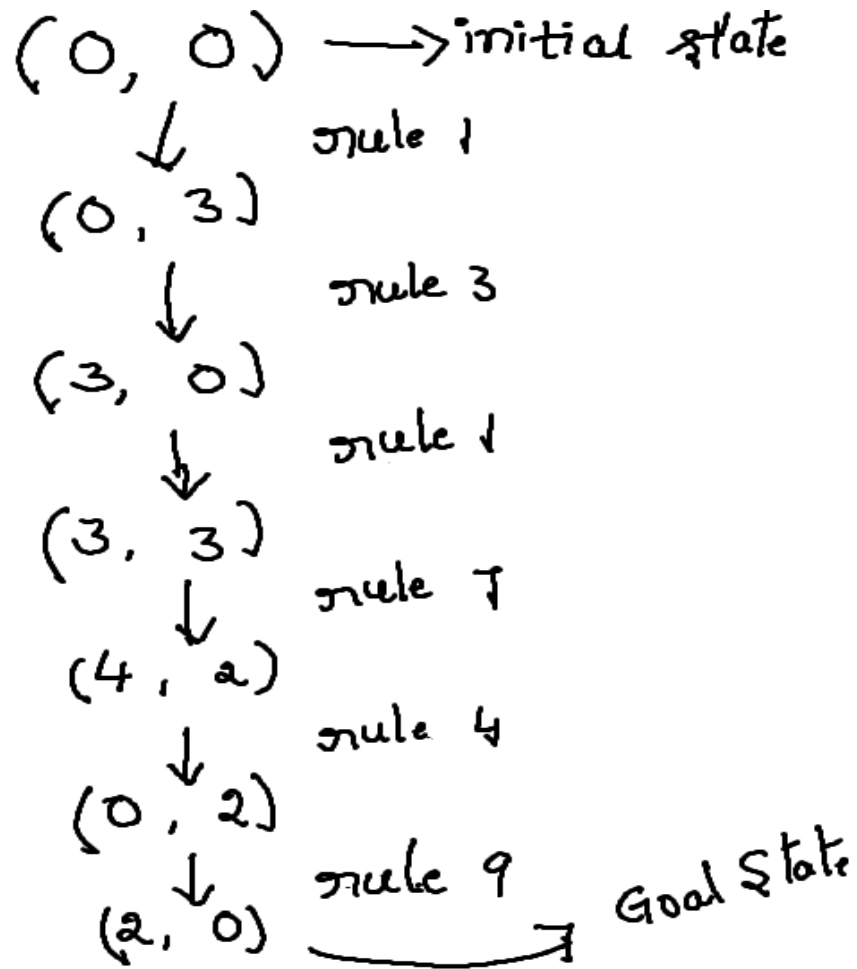
There are Nine Production Rules are there:

1. Fill 3L jug  $(x,y) \rightarrow (x,3)$  if  $y < 3$
2. Fill 4L jug  $(x,y) \rightarrow (4,y)$  if  $x < 4$

3. Empty 3L jug  $(x,y) \rightarrow (x,0)$  if  $y > 0$
4. Empty 4L jug  $(x,y) \rightarrow (0,y)$  if  $x > 0$
5. Move all from 3L to 4L  $(x,y) \rightarrow (x+y,0)$  if  $y > 0$  &  $x+y \leq 4$
6. Move all from 4L to 3L  $(x,y) \rightarrow (0,x+y)$  if  $x > 0$  &  $x+y \leq 3$
7. Transfer from 3L to 4L until 4L is full  $(x,y) \rightarrow (4,y-(4-x))$   $y > 0$  &  $x+y \leq 4$
8. Transfer from 4L to 3L until 3L is full  $(x,y) \rightarrow (x-(3-y),3)$  if  $x > 0$  &  $x+y \leq 4$
9. Transfer 2L water from 3L jug to 4L jug  $(x,y) \rightarrow (0,2) \rightarrow (2,0)$

From 1 to 8 are generalized rules, 9 is goal specific rule

S4:





### 3. 8 Puzzle Problem:

#### Puzzle Problem.

The 8 puzzle consists of eight numbered, movable tiles set in a 3x3 frame. One cell of the frame is always empty thus making it possible to move an adjacent numbered tile into the empty cell. Such a puzzle is illustrated in following diagram

We also know the **eight puzzle problem** by the name of **N puzzle problem** or **sliding puzzle problem**.

**N-puzzle** that consists of N tiles (N+1 tiles with an empty tile) where N can be 8, 15, 24 and so on.

In our example **N = 8**. (that is **square root of (8+1) = 3 rows and 3 columns**).

In the same way, if we have N = 15, 24 in this way, then they have Row and columns as follow (**square root of (N+1) rows and square root of (N+1) columns**).

That is if **N=15** than number of rows and columns= 4, and if **N= 24** number of rows and columns= 5.

So, basically in these types of problems we have given a **initial state or initial configuration (Start state) and a Goal state or Goal Configuration**.

Here We are solving a problem of **8 puzzle** that is a **3x3 matrix**.

**Note:** If we solve this problem with depth first search, then it will go to depth instead of exploring layer wise nodes.

**Time complexity:** In worst case time complexity in **BFS** is **O(b^d)** know as **order of b raise to power d**. In this **particular case it is (3^20)**.

**b**-branch factor

**d**-depth factor

Let's solve the problem with **Heuristic Search** that is **Informed Search (A\* , Best First Search (Greedy Search))**

To solve the problem with Heuristic search or informed search we have to calculate Heuristic values of each node to calculate **cost function**.

$$\text{Cost Function} \rightarrow c(x) = g(x) + f(x)$$

Where **c(x)** is cost function

**g(x)** → cost of reaching from current node to root node

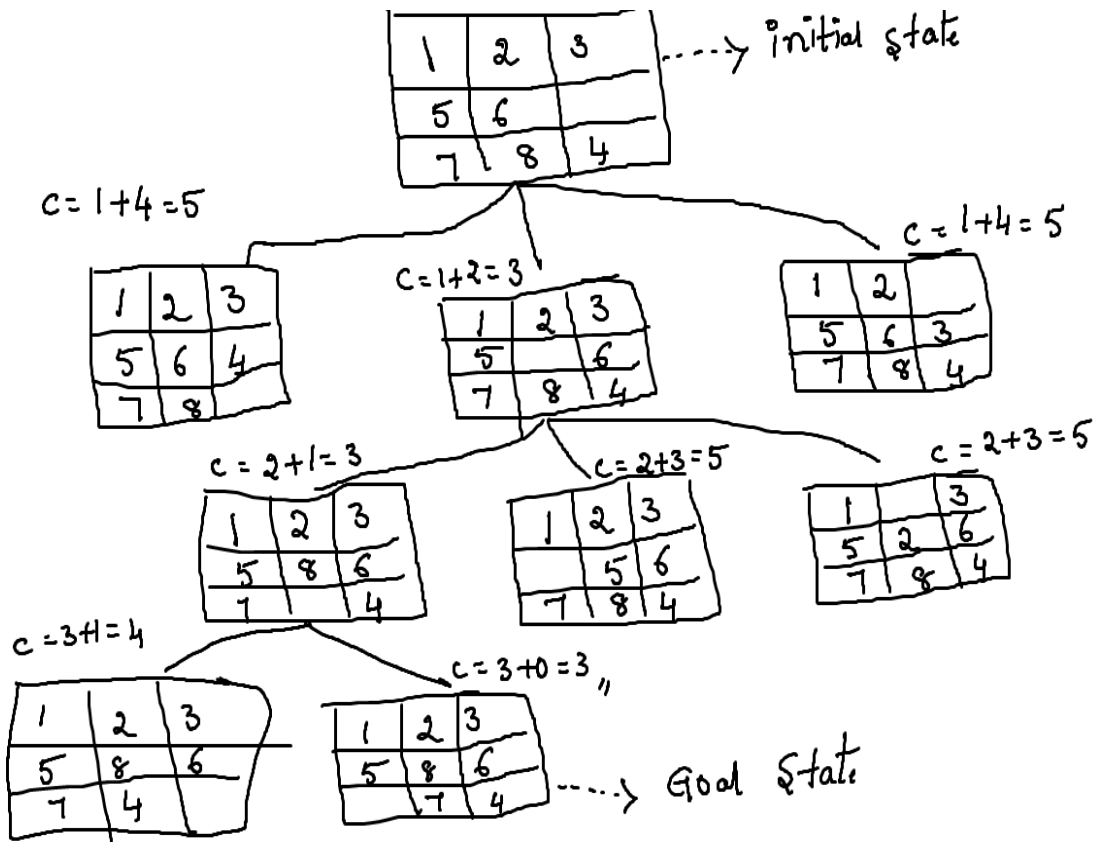
**f(x)** → approximate cost of reaching answer node from X (where X is current state)

1	2	3
5	6	
7	8	4

Initial State

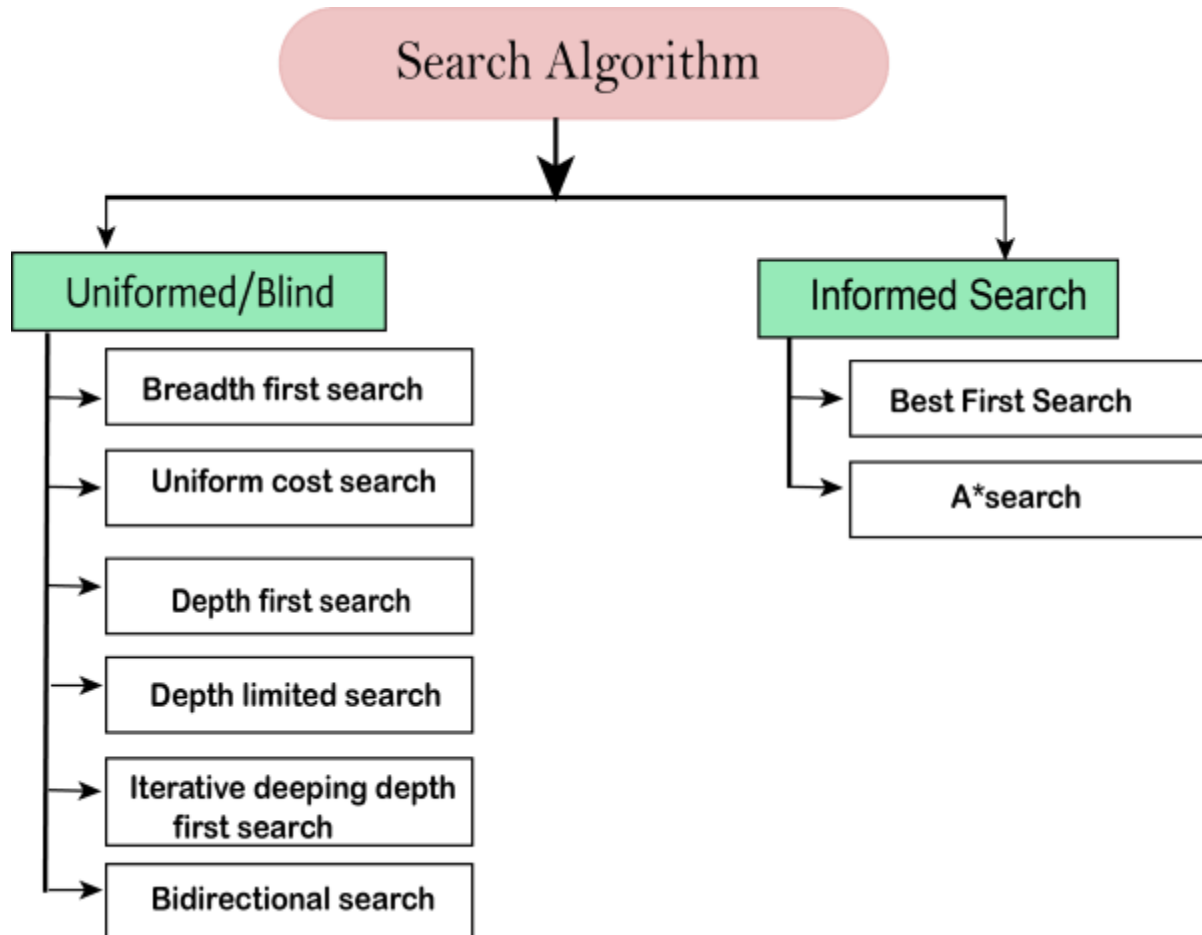
1	2	3
5	8	6
	7	4

Goal State



## Types of search algorithms

Based on the search problems we can classify the search algorithms into uninformed (Blind search) search and informed search (Heuristic search) algorithms.



### Uninformed/Blind Search:

The uninformed search does not contain any domain knowledge such as closeness, the location of the goal. It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes. Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, so it is also called blind search. It examines each node of the tree until it achieves the goal node.

**It can be divided into five main types:**

- Breadth-first search
- Uniform cost search
- Depth-first search

- Iterative deepening depth-first search
- Bidirectional Search

## Informed Search

- Informed search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy. Informed search is also called a Heuristic search.
- A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.

Informed search can solve much complex problem which could not be solved in another way.

An example of informed search algorithms is a traveling salesman problem.

1. Greedy Search
2. A\* Search

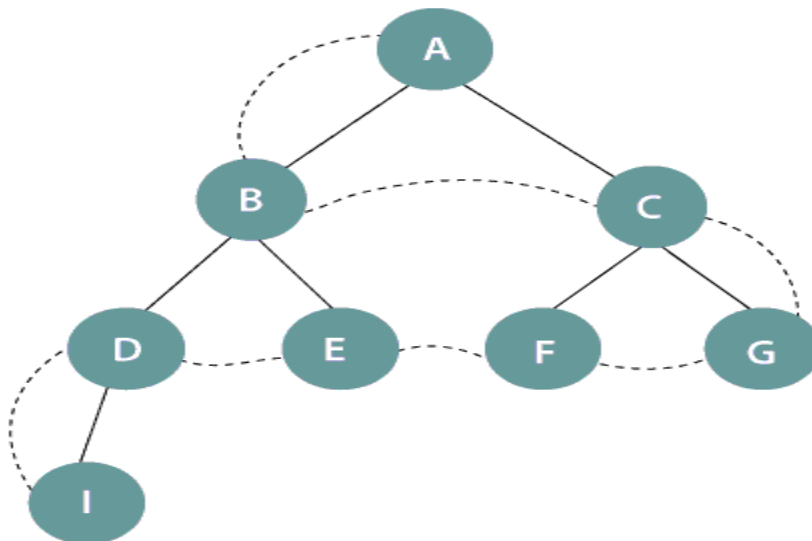
## Uninformed/Blind Search:

### Breadth-first search (BFS)

It is a simple search strategy where the root node is expanded first, then covering all other successors of the root node, further move to expand the next level nodes and the search continues until the goal node is not found.

**BFS** expands the shallowest (i.e., not deep) node first using FIFO (First in first out) order. Thus, new nodes (i.e., children of a parent node) remain in the queue and old unexpanded node which are shallower than the new nodes, get expanded first.

In BFS, goal test (**a test to check whether the current state is a goal state or not**) is applied to each node at the time of its generation rather when it is selected for expansion.



## Breadth-first search tree

In the above figure, it is seen that the nodes are expanded level by level starting from the root node **A** till the last node **I** in the tree. Therefore, the BFS sequence followed is: **A->B->C->D->E->F->G->I**.

## BFS Algorithm

- Set a variable **NODE** to the initial state, i.e., the *root node*.
- Set a variable **GOAL** which contains the value of the *goal state*.
- Loop each node by traversing level by level until the goal state is not found.
- While performing the looping, start removing the elements from the queue in *FIFO* order.
- If the goal state is found, **return goal state** otherwise continue the search.

**The performance measure of BFS is as follows:**

- **Completeness:** It is a complete strategy as it definitely finds the goal state.
- **Optimality:** It gives an optimal solution if the cost of each node is same.
- **Space Complexity:** The space complexity of BFS is **O(bd)**, i.e., it requires a huge amount of memory. Here, **b** is the **branching factor** and **d** denotes the **depth/level** of the tree
- **Time Complexity:** BFS consumes much time to reach the goal node for large instances.

## Disadvantages of BFS

- The biggest disadvantage of BFS is that it requires a lot of memory space, therefore it is a memory bounded strategy.
- BFS is time taking search strategy because it expands the nodes **breadthwise**.  
**Note:** BFS expands the nodes level by level, i.e., breadthwise, therefore it is also known as a **Level search technique**.

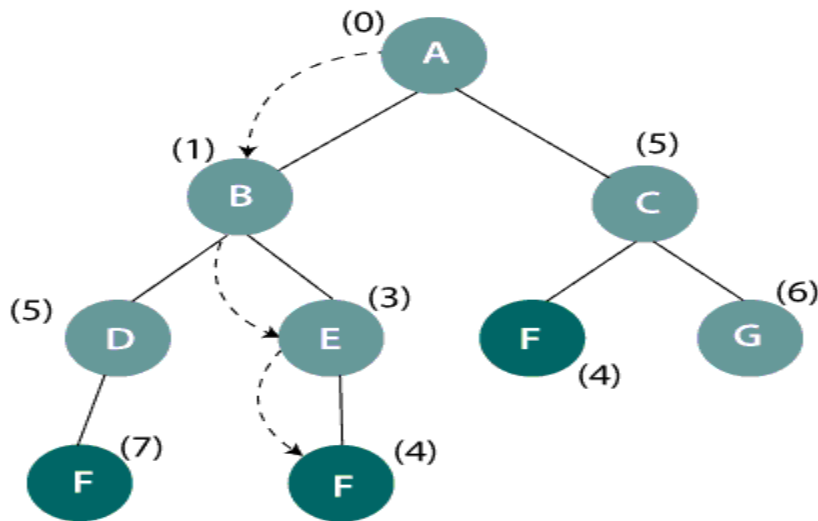
## Uniform-cost search

Unlike BFS, this uninformed search explores nodes based on their path cost from the root node. It expands a node **n** having the lowest path cost  $g(n)$ , where  $g(n)$  is the total cost from a root node to node **n**. Uniform-cost search is significantly different from the breadth-first search because of the following two reasons:

- First, the goal test is applied to a node only when it is selected for expansion **not when it is first generated** because the first goal node which is generated may be on a suboptimal path.
- Secondly, a goal test is added to a node, only when a better/ optimal path is found. Thus, uniform-cost search expands nodes in a sequence of their **optimal path cost** because before exploring any node, it searches the optimal path. Also, the step cost is positive so, paths never get shorter when a new node is added in the search.

## Uniform-cost search on a binary tree

In the above figure, it is seen that the goal-state is F and start/ initial state is A. There are three paths available to reach the goal node. We need to select an optimal path which may give the lowest total cost  $g(n)$ . Therefore, **A->B->E->F** gives the optimal path cost i.e., **0+1+3+4=8**.



## Uniform-cost search Algorithm

- Set a variable **NODE** to the initial state, i.e., *the root node and expand it*.
- After expanding the root node, select one node having the lowest path cost and expand it further. Remember, **the selection of the node should give an optimal path cost**.
- If the goal node is searched with optimal value, return **goal state**, else carry on the search.

### The performance measure of Uniform-cost search

- **Completeness:** It guarantees to reach the goal state.
- **Optimality:** It gives optimal path cost solution for the search.
- **Space and time complexity:** The worst space and time complexity of the uniform-cost search is  $O(b^{1+LC*/??})$ .

**Note:** When the path cost is same for all the nodes, it behaves similar to **BFS**.

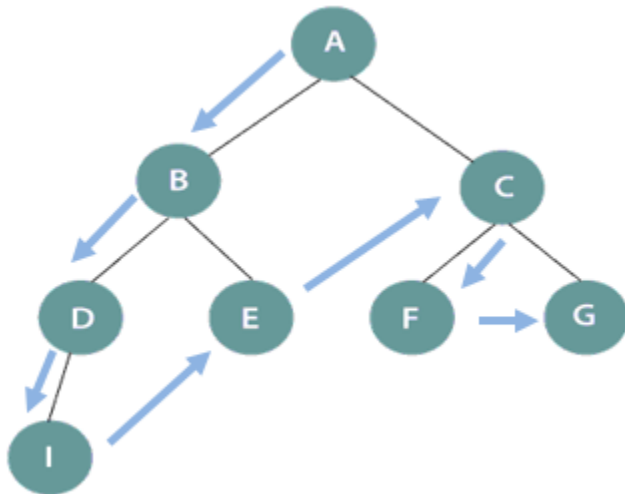
## Disadvantages of Uniform-cost search

- It does not care about the number of steps a path has taken to reach the goal state.
- It may stick to an infinite loop if there is a path with infinite zero cost sequence.
- It works hard as it examines each node in search of lowest cost path.

## Depth-first search

This search strategy explores the deepest node first, then backtracks to explore other nodes. It uses **LIFO (Last in First Out)** order, which is based on the stack, in order to expand the

unexpanded nodes in the search tree. The search proceeds to the deepest level of the tree where it has no successors. This search expands nodes till infinity, i.e., the depth of the tree.



## DFS search tree

In the above figure, DFS works starting from the initial node **A** (root node) and traversing in one direction deeply till node **I** and then backtrack to **B** and so on. Therefore, the sequence will be **A->B->D->I->E->C->F->G**.

## DFS Algorithm

- Set a variable **NODE** to the initial state, i.e., the *root node*.
- Set a variable **GOAL** which contains the value of the *goal state*.
- Loop each node by traversing deeply in one direction/path in search of the goal node.
- While performing the looping, start removing the elements from the stack in *LIFO* order.
- If the goal state is found, **return goal state** otherwise backtrack to expand nodes in other direction.

### The performance measure of DFS

- **Completeness:** DFS does not guarantee to reach the goal state.
- **Optimality:** It does not give an optimal solution as it expands nodes in one direction deeply.
- **Space complexity:** It needs to store only a single path from the root node to the leaf node. Therefore, DFS has **O(bm)** space complexity where **b** is the **branching factor**(i.e., **total no. of child nodes, a parent node have**) and **m** is the **maximum length of any path**.
- **Time complexity:** DFS has **O(bm)** time complexity.

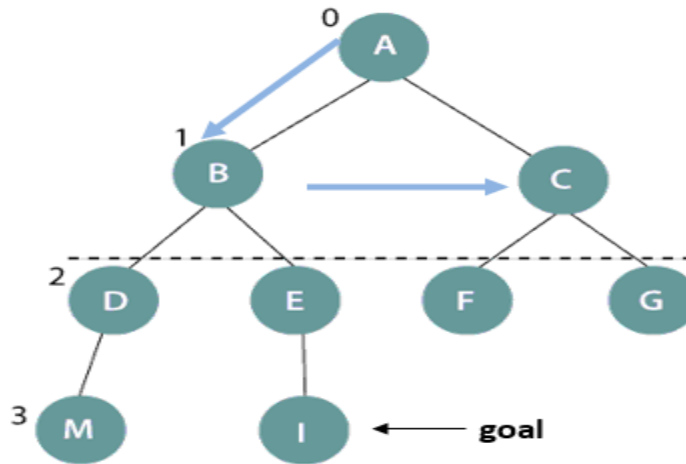
## Disadvantages of DFS

- It may get trapped in an infinite loop.
- It is also possible that it may not reach the goal state.
- DFS does not give an optimal solution.

**Note:** DFS uses the concept of **backtracking** to explore each node in a search tree.

## Depth-limited search

This search strategy is similar to DFS with a little difference. The difference is that in depth-limited search, we limit the search by imposing a **depth limit**  $l$  to the depth of the search tree. It does not need to explore till infinity. As a result, the **depth-first search is a special case of depth-limited search**, when the limit  $l$  is infinite.



### Depth-limited search on a binary tree

In the above figure, the **depth-limit is 1**. So, only level 0 and 1 get expanded in **A->B->C** DFS sequence, starting from the root node **A** till node **B**. It is not giving satisfactory result because we could not reach the goal node **I**.

## Depth-limited search Algorithm

- Set a variable **NODE** to the initial state, i.e., the *root node*.
- Set a variable **GOAL** which contains the value of the *goal state*.
- Set a variable **LIMIT** which carries a depth-limit value.
- Loop each node by traversing in **DFS** manner till the depth-limit value.
- While performing the looping, start removing the elements from the stack in *LIFO* order.
- If the goal state is found, **return goal state**. Else **terminate the search**.

### The performance measure of Depth-limited search

- **Completeness:** Depth-limited search does not guarantee to reach the goal node.
- **Optimality:** It does not give an optimal solution as it expands the nodes till the depth-limit.
- **Space Complexity:** The space complexity of the depth-limited search is **O(bl)**.
- **Time Complexity:** The time complexity of the depth-limited search is **O(bl)**.

## Disadvantages of Depth-limited search

- This search strategy is not complete.

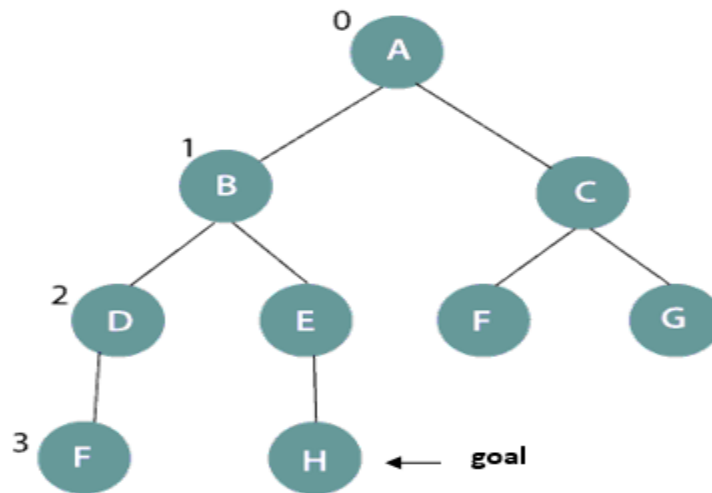


- It does not provide an optimal solution.

**Note:** Depth-limit search terminates with two kinds of failures: the **standard failure value** indicates “no solution,” and **cut-off value**, which indicates “no solution within the depth-limit.”

## Iterative deepening depth-first search/ Iterative deepening search

This search is a combination of BFS and DFS, as BFS guarantees to reach the goal node and DFS occupies less memory space. Therefore, iterative deepening search combines these two advantages of BFS and DFS to reach the goal node. It gradually increases the depth-limit from **0,1,2** and so on and reach the goal node.



In the above figure, the goal node is **H** and initial **depth-limit** =**[0-1]**. So, it will expand level 0 and 1 and will terminate with **A->B->C** sequence. Further, change the **depth-limit** =**[0-3]**, it will again expand the nodes from level 0 till level 3 and the search terminate with **A->B->D->F->E->H** sequence where **H** is the desired goal node.

Iterative deepening search Algorithm

- Explore the nodes in DFS order.
- Set a LIMIT variable with a limit value.
- Loop each node up to the limit value and further increase the limit value accordingly.
- Terminate the search when the goal state is found.

### The performance measure of Iterative deepening search

- **Completeness:** Iterative deepening search may or may not reach the goal state.
- **Optimality:** It does not give an optimal solution always.
- **Space Complexity:** It has the same space complexity as BFS, i.e., **O(bd)**.
- **Time Complexity:** It has **O(d)** time complexity.

### Disadvantages of Iterative deepening search

- The drawback of iterative deepening search is that it seems wasteful because it generates states multiple times.
- **Note:** Generally, iterative deepening search is required when the search space is large, and the depth of the solution is unknown.

## **Bidirectional search**

The strategy behind the bidirectional search is to run two searches simultaneously—**one forward search from the initial state and other from the backside of the goal**—hoping that both searches will meet in the middle. As soon as the two searches intersect one another, the bidirectional search terminates with the goal node. This search is implemented by replacing the goal test to check if the two searches intersect. Because if they do so, it means a solution is found.

### **The performance measure of Bidirectional search**

- **Complete:** Bidirectional search is complete.
- **Optimal:** It gives an optimal solution.
- **Time and space complexity:** Bidirectional search has  $O(bd/2)$

## **Disadvantage of Bidirectional Search**

- It requires a lot of memory space

## **Informed Search/ Heuristic Search in AI**

An informed search is more efficient than an uninformed search because in informed search, along with the current state information, some additional information is also present, which make it easy to reach the goal state.

Below we have discussed different types of informed search:

### **Best-first Search (Greedy search)**

A best-first search is a general approach of informed search. Here, a node is selected for expansion based on an evaluation function  $f(n)$ , where  $f(n)$  interprets the cost estimate value. The evaluation function expands that node first, which has the lowest cost. A component of  $f(n)$  is  $h(n)$  which carries the additional information required for the search algorithm, i.e.,  $h(n)$  = estimated cost of the cheapest path from the current node  $n$  to the goal node.

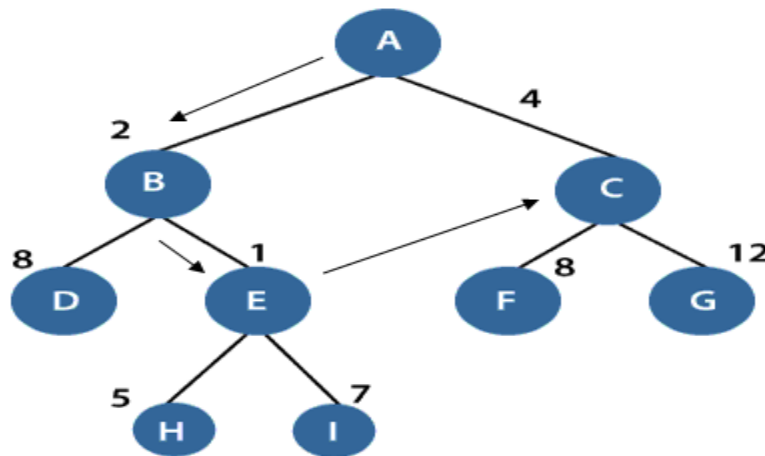
Note: If the current node  $n$  is a goal node, the value of  $h(n)$  will be 0.

Best-first search is known as a greedy search because it always tries to explore the node which is nearest to the goal node and selects that path, which gives a quick solution. Thus, it evaluates nodes with the help of the heuristic function, i.e.,  $f(n)=h(n)$ .

### **Best-first search Algorithm**

1. Set an OPEN list and a CLOSE list where the OPEN list contains visited but unexpanded nodes and the CLOSE list contains visited as well as expanded nodes.

- Initially, traverse the root node and visit its next successor nodes and place them in the OPEN list in ascending order of their heuristic value.
- Select the first successor node from the OPEN list with the lowest heuristic value and expand further.
- Now, rearrange all the remaining unexpanded nodes in the OPEN list and repeat above two steps.
- If the goal node is reached, terminate the search, else expand further.



In the above figure, the root node is A, and its next level successor nodes are B and C with  $h(B)=2$  and  $h(C)=4$ . Our task is to explore that node which has the lowest  $h(n)$  value. So, we will select node B and expand it further to node D and E. Again, search out that node which has the lowest  $h(n)$  value and explore it further.

### The performance measure of Best-first search Algorithm:

- Completeness: Best-first search is incomplete even in finite state space.
- Optimality: It does not provide an optimal solution.
- Time and Space complexity: It has  $O(bm)$  worst time and space complexity, where  $m$  is the maximum depth of the search tree. If the quality of the heuristic function is good, the complexities could be reduced substantially.

Note: Best first searches combine the advantage of BFS and DFS to find the best solution.

### Disadvantages of Best-first search

- BFS does not guarantee to reach the goal state.
- Since the best-first search is a greedy approach, it does not give an optimized solution.
- It may cover a long distance in some cases.

## A\* Search Algorithm

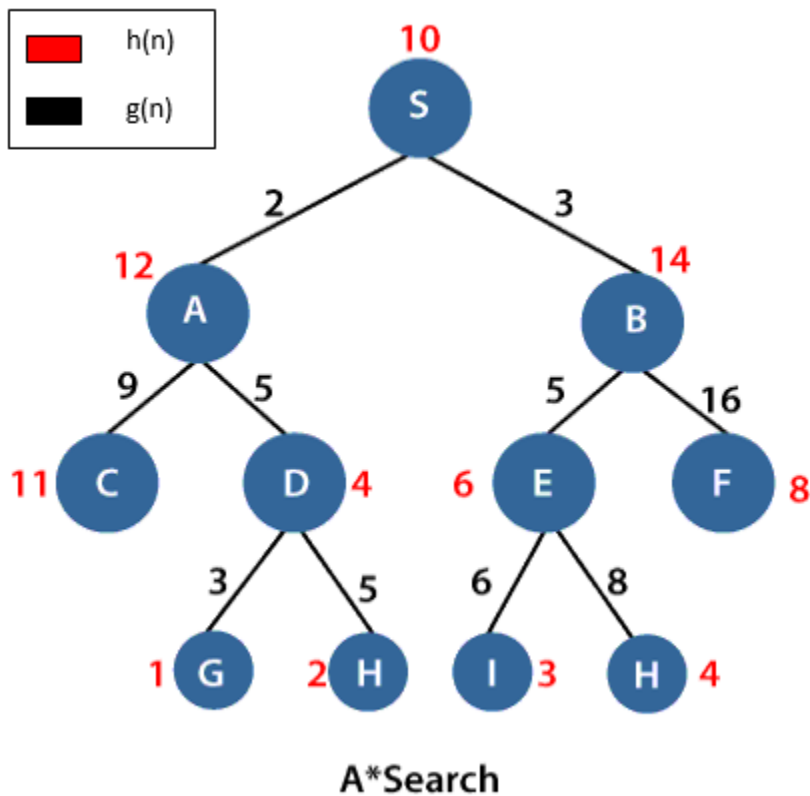
A\* search is the most widely used informed search algorithm where a node  $n$  is evaluated by combining values of the functions  $g(n)$  and  $h(n)$ . The function  $g(n)$  is the path cost from the start/initial node to a node  $n$  and  $h(n)$  is the estimated cost of the cheapest path from node  $n$  to the goal node. Therefore, we have

$$f(n) = g(n) + h(n)$$

where  $f(n)$  is the estimated cost of the cheapest solution through  $n$ .

So, in order to find the cheapest solution, try to find the lowest values of  $f(n)$ .

Let's see the below example to understand better.



In the above example,  $S$  is the root node, and  $G$  is the goal node. Starting from the root node  $S$  and moving towards its next successive nodes  $A$  and  $B$ . In order to reach the goal node  $G$ , calculate the  $f(n)$  value of node  $S$ ,  $A$  and  $B$  using the evaluation equation i.e.

$$f(n) = g(n) + h(n)$$

### Calculation of $f(n)$ for node S:

$$f(S) = (\text{distance from node } S \text{ to } S) + h(S)$$

- $0 + 10 = 10$ .

Calculation of  $f(n)$  for node A:

$$f(A) = (\text{distance from node } S \text{ to } A) + h(A)$$

- $2 + 12 = 14$

Calculation of  $f(n)$  for node B:  
 $f(B) = (\text{distance from node S to B}) + h(B)$

- $3 + 14 = 17$

Therefore, node A has the lowest  $f(n)$  value. Hence, node A will be explored to its next level nodes C and D and again calculate the lowest  $f(n)$  value. After calculating, the sequence we get is  $S \rightarrow A \rightarrow D \rightarrow G$  with  $f(n) = 13$  (lowest value).

## How to make A\* search admissible to get an optimized solution?

A\* search finds an optimal solution as it has the admissible heuristic function  $h(n)$  which believes that the cost of solving a problem is less than its actual cost. A heuristic function can either underestimate or overestimate the cost required to reach the goal node. But an admissible heuristic function never overestimates the cost value required to reach the goal state. Underestimating the cost value means the cost we assumed in our mind is less than the actual cost. Overestimating the cost value means the cost we assumed is greater than the actual cost, i.e.,

$$\left\{ \begin{array}{l} h'(n) \leq h(n) \text{ is underestimation of the cost value} \\ h'(n) \geq h(n) \text{ is overestimation of the cost value} \end{array} \right.$$

Here,  $h(n)$  is the actual heuristic cost value and  $h'(n)$  is the estimated heuristic cost value.

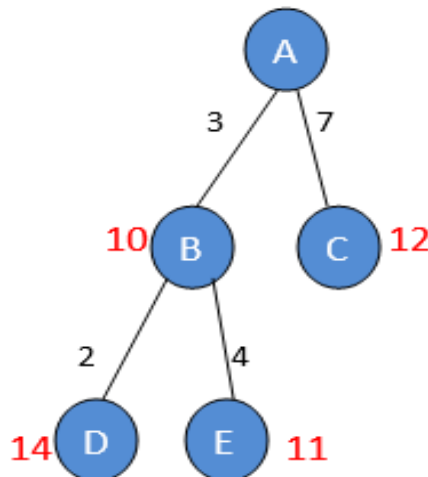
Note: An overestimated cost value may or may not lead to an optimized solution, but an underestimated cost value always lead to an optimized solution.

Let's understand with the help of an example:

Consider the below search tree where the starting/initial node is A and goal node is E. We have different paths to reach the goal node E with their different heuristic costs  $h(n)$  and path costs  $g(n)$ . The actual heuristic cost is  $h(n) = 18$ . Let's suppose two different estimation values:

$h_1'(n) = 12$  which is underestimated cost value

$h_2'(n) = 25$  which is overestimated cost value



So, when the cost value is overestimated, it will not take any load to search the best optimal path and acquire the first optimal path. But if the  $h(n)$  value is underestimated, it will try and reach the best optimal value of  $h(n)$  which will lead to a good optimal solution.

Note: Underestimation of  $h(n)$  leads to a better optimal solution instead of overestimating the value.

The performance measure of A\* search

- Completeness: The star(\*) in A\* search guarantees to reach the goal node.
- Optimality: An underestimated cost will always give an optimal solution.
- Space and time complexity: A\* search has  $O(bd)$  space and time complexities.

### **Disadvantage of A\* search**

- A\* mostly runs out of space for a long period.

## **HEURISTIC FUNCTIONS**

A Heuristic technique helps in solving problems, even though there is no guarantee that it will never lead in the wrong direction. There are heuristics of every general applicability as well as domain specific. The strategies are general purpose heuristics. In order to use them in a specific domain they are coupled with some domain specific heuristics. There are two major ways in which domain - specific, heuristic information can be incorporated into rule-based search procedure.

- In the rules themselves

- As a heuristic function that evaluates individual problem states and determines how desired they are.

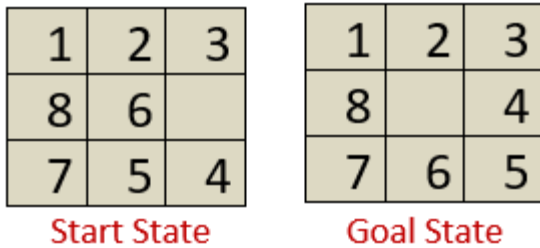
A heuristic function is a function that maps from problem state description to measures desirability, usually represented as number weights. The value of a heuristic function at a given node in the search process gives a good estimate of that node being on the desired path to solution. Well designed heuristic functions can provide a fairly good estimate of whether a path is good or not. ( " The sum of the distances traveled so far" is a simple heuristic function in the traveling salesman problem) . the purpose of a heuristic function is to guide the search process in the most profitable directions, by suggesting which path to follow first when more than one path is available. However in many problems, the cost of computing the value of a heuristic function would be more than the effort saved in the search process. Hence generally there is a trade-off between the cost of evaluating a heuristic function and the savings in search that the function provides.

Heuristic Functions in AI: As we have already seen that an informed search make use of heuristic functions in order to reach the goal node in a more prominent way. Therefore, there are several pathways in a search tree to reach the goal node from the current node. The selection of a

good heuristic function matters certainly. A good heuristic function is determined by its efficiency. More is the information about the problem, more is the processing time.

Some toy problems, such as 8-puzzle, 8-queen, tic-tac-toe, etc., can be solved more efficiently with the help of a heuristic function. Let's see how:

Consider the following 8-puzzle problem where we have a start state and a goal state. Our task is to slide the tiles of the current/start state and place it in an order followed in the goal state. There can be four moves either left, right, up, or down. There can be several ways to convert the current/start state to the goal state, but, we can use a heuristic function  $h(n)$  to solve the problem more efficiently.

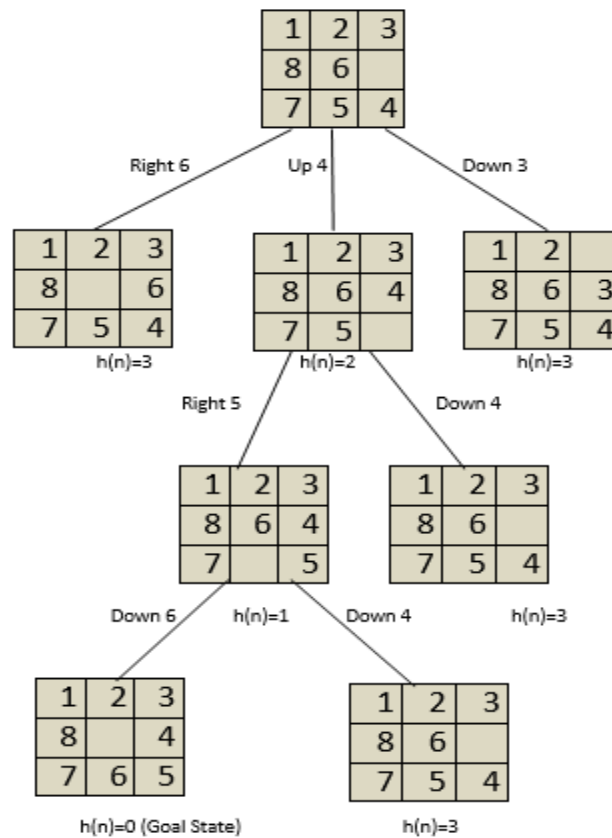


A heuristic function for the 8-puzzle problem is defined below:

$$h(n) = \text{Number of tiles out of position.}$$

So, there is total of three tiles out of position i.e., 6, 5 and 4. Do not count the empty tile present in the goal state). i.e.  $h(n)=3$ . Now, we require to minimize the value of  $h(n) = 0$ .

We can construct a state-space tree to minimize the  $h(n)$  value to 0, as shown below:



It is seen from the above state space tree that the goal state is minimized from  $h(n)=3$  to  $h(n)=0$ . However, we can create and use several heuristic functions as per the requirement. It is also clear from the above example that a heuristic function  $h(n)$  can be defined as the information required to solve a given problem more efficiently. The information can be related to the nature of the state, cost of transforming from one state to another, goal node characteristics, etc., which is expressed as a heuristic function.

### **Properties of a Heuristic search Algorithm**

Use of heuristic function in a heuristic search algorithm leads to following properties of a heuristic search algorithm:

- **Admissible Condition:** An algorithm is said to be admissible, if it returns an optimal solution.
- **Completeness:** An algorithm is said to be complete, if it terminates with a solution (if the solution exists).
- **Dominance Property:** If there are two admissible heuristic algorithms A1 and A2 having  $h_1$  and  $h_2$  heuristic functions, then A1 is said to dominate A2 if  $h_1$  is better than  $h_2$  for all the values of node  $n$ .
- **Optimality Property:** If an algorithm is complete, admissible, and dominating other algorithms, it will be the best one and will definitely give an optimal solution.

## **Local Search Algorithms and Optimization Problem**

The informed and uninformed search expands the nodes systematically in two ways:

- keeping different paths in the memory and selecting the best suitable path,  
Which leads to a solution state required to reach the goal node? But beyond these “classical search algorithms,” we have some “local search algorithms” where the path cost does not matter, and only focus on solution-state needed to reach the goal node.  
A local search algorithm completes its task by traversing on a single current node rather than multiple paths and following the neighbors of that node generally.  
Although local search algorithms are not systematic, still they have the following two advantages:
- Local search algorithms use a very little or constant amount of memory as they operate only on a single path.
- Most often, they find a reasonable solution in large or infinite state spaces where the classical or systematic algorithms do not work.

Does the local search algorithm work for a pure optimized problem?

Yes, the local search algorithm works for pure optimized problems. A pure optimization problem is one where all the nodes can give a solution. But the target is to find the best state out of all according to the objective function. Unfortunately, the pure optimization problem fails to find high-quality solutions to reach the goal state from the current state.

Note: An objective function is a function whose value is either minimized or maximized in different contexts of the optimization problems. In the case of search algorithms, an objective function can be the path cost for reaching the goal node, etc.

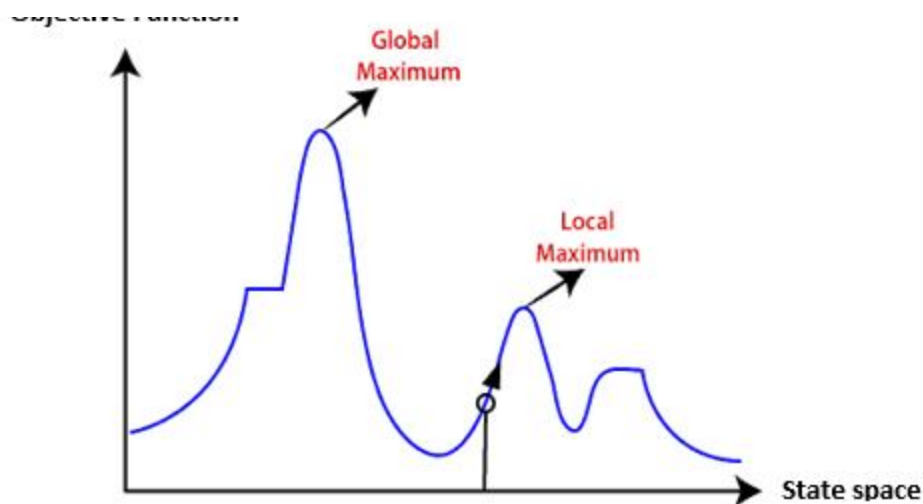
### **Working of a Local search algorithm**

Let's understand the working of a local search algorithm with the help of an example:



Consider the below state-space landscape having both:

- Location: It is defined by the state.
- Elevation: It is defined by the value of the objective function or heuristic cost function.



A one-dimensional state-space landscape in which elevation corresponds to the objective function

The local search algorithm explores the above landscape by finding the following two points:

- Global Minimum: If the elevation corresponds to the cost, then the task is to find the lowest valley, which is known as Global Minimum.
- Global Maxima: If the elevation corresponds to an objective function, then it finds the highest peak which is called as Global Maxima. It is the highest point in the valley.

We will understand the working of these points better in Hill-climbing search.

Below are some different types of local searches:

- Hill-climbing Search
- Simulated Annealing
- Local Beam Search

## Hill Climbing Algorithm in AI

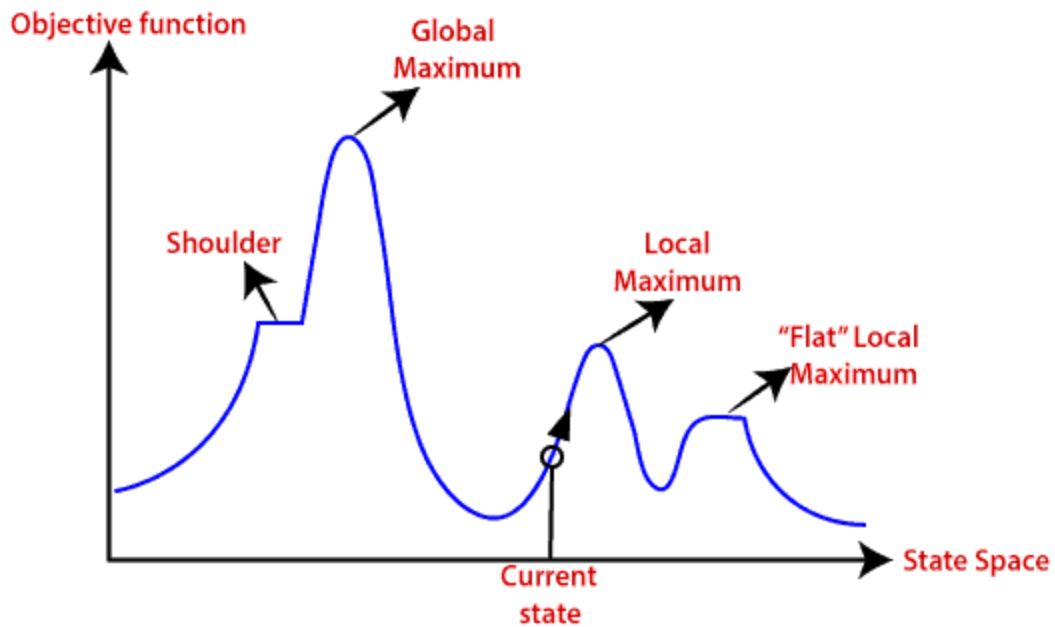
**Hill Climbing Algorithm:** Hill climbing search is a local search problem. *The purpose of the hill climbing search is to climb a hill and reach the topmost peak/ point of that hill.* It is based on the heuristic search technique where the person who is climbing up on the hill estimates the direction which will lead him to the highest peak.

### State-space Landscape of Hill climbing algorithm

To understand the concept of hill climbing algorithm, consider the below landscape representing the **goal state/peak** and the **current state** of the climber. The topographical regions shown in the figure can be defined as:

- **Global Maximum:** It is the highest point on the hill, which is the goal state.
- **Local Maximum:** It is the peak higher than all other peaks but lower than the global maximum.

- **Flat local maximum:** It is the flat area over the hill where it has no uphill or downhill. It is a saturated point of the hill.
- **Shoulder:** It is also a flat area where the summit is possible.
- **Current state:** It is the current position of the person.

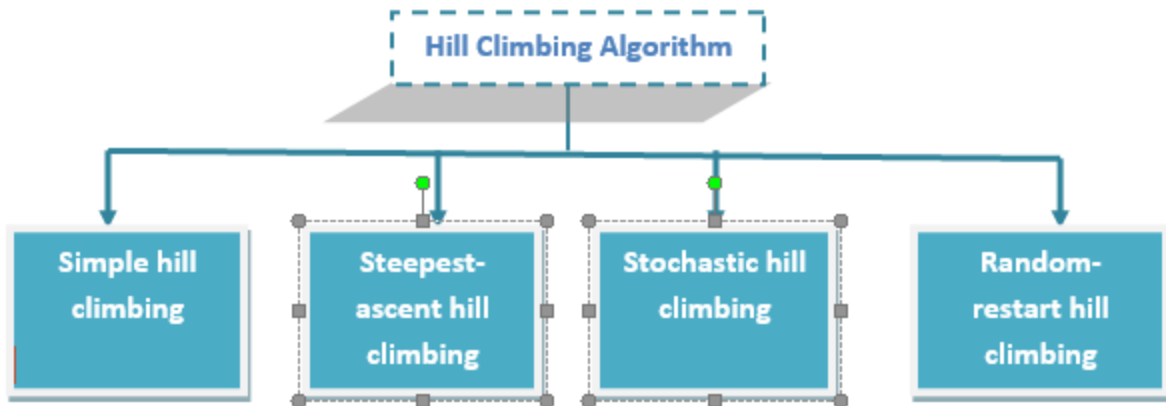


**A one-dimensional state-space landscape in which elevation corresponds to the objective function**

### Types of Hill climbing search algorithm

There are following types of hill-climbing search:

- Simple hill climbing
- Steepest-ascent hill climbing
- Stochastic hill climbing
- Random-restart hill climbing



## Simple hill climbing search

Simple hill climbing is the simplest technique to climb a hill. The task is to reach the highest peak of the mountain. Here, the movement of the climber depends on his move/steps. If he finds his next step better than the previous one, he continues to move else remain in the same state. This search focuses only on his previous and next step.

### Simple hill climbing Algorithm

1. Create a **CURRENT** node, **NEIGHBOUR** node, and a **GOAL** node.
2. If the **CURRENT node=GOAL node**, return **GOAL** and terminate the search.
3. Else **CURRENT node<= NEIGHBOUR node**, move ahead.
4. Loop until the goal is not reached or a point is not found.

### Steepest-ascent hill climbing

Steepest-ascent hill climbing is different from simple hill climbing search. Unlike simple hill climbing search, It considers all the successive nodes, compares them, and choose the node which is closest to the solution. Steepest hill climbing search is similar to **best-first search** because it focuses on each node instead of one.

Note: Both simple, as well as steepest-ascent hill climbing search, fails when there is no closer node.

## Steepest-ascent hill climbing algorithm

1. Create a **CURRENT** node and a **GOAL** node.
2. If the **CURRENT node=GOAL node**, return **GOAL** and terminate the search.
3. Loop until a better node is not found to reach the solution.
4. If there is any better successor node present, expand it.
5. When the **GOAL** is attained, return **GOAL** and terminate.

### Stochastic hill climbing

Stochastic hill climbing does not focus on all the nodes. It selects one node at random and decides whether it should be expanded or search for a better one.

### Random-restart hill climbing

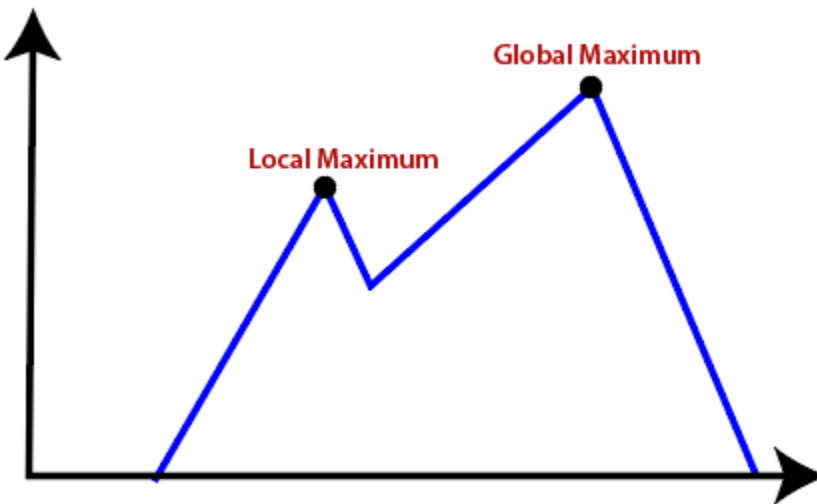
Random-restart algorithm is based on **try and try strategy**. It iteratively searches the node and selects the best one at each step until the goal is not found. The success depends most commonly

on the shape of the hill. If there are few plateaus, local maxima, and ridges, it becomes easy to reach the destination.

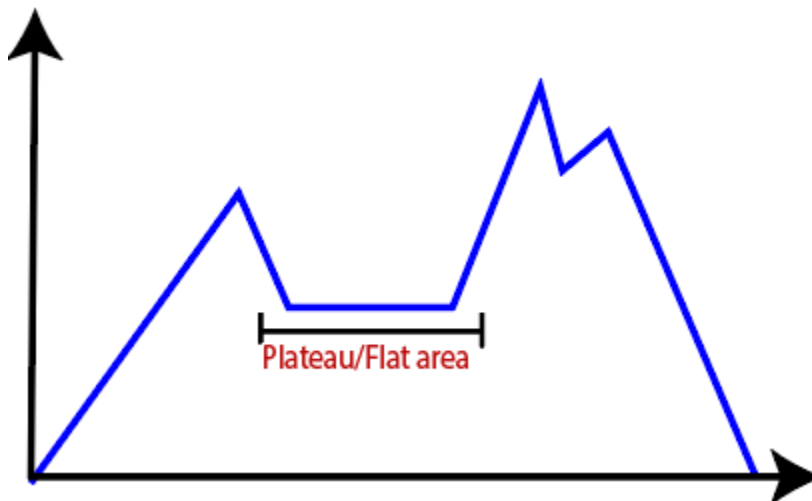
## Limitations of Hill climbing algorithm

Hill climbing algorithm is a fast and furious approach. It finds the solution state rapidly because it is quite easy to improve a bad state. But, there are following limitations of this search:

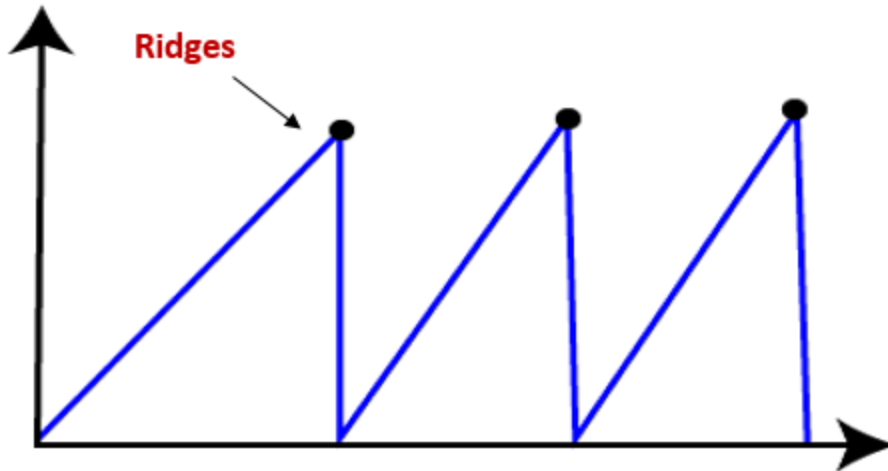
- **Local Maxima:** It is that peak of the mountain which is highest than all its neighboring states but lower than the global maxima. It is not the goal peak because there is another peak higher than it.



- **Plateau:** It is a flat surface area where no uphill exists. It becomes difficult for the climber to decide that in which direction he should move to reach the goal point. Sometimes, the person gets lost in the flat area.



- **Ridges:** It is a challenging problem where the person finds two or more local maxima of the same height commonly. It becomes difficult for the person to navigate the right point and stuck to that point itself.



## Simulated Annealing

Simulated annealing is similar to the hill climbing algorithm. It works on the current situation. It picks a **random move** instead of picking **the best move**. If the move leads to the improvement of the current situation, it is always accepted as a step towards the solution state, else it accepts the move having a **probability less than 1**. This search technique was first used in **1980** to solve **VLSI layout** problems. It is also applied for factory scheduling and other large optimization tasks.

## Local Beam Search

Local beam search is quite different from random-restart search. It keeps track of **k** states instead of just one. It selects **k** randomly generated states, and expand them at each step. If any state is a goal state, the search stops with success. Else it selects the best **k** successors from the complete list and repeats the same process. In random-restart search where each search process runs independently, but in local beam search, the necessary information is shared between the parallel search processes.

## Disadvantages of Local Beam search

- This search can suffer from a lack of diversity among the **k** states.
  - It is an expensive version of hill climbing search.
- Note:** A variant of Local Beam Search is **Stochastic Beam Search** which selects **k** successors at random rather than choosing the best **k** successors.

## SEARCHING WITH NONDETERMINISTIC ACTIONS

we assumed that the environment is fully observable and deterministic and that the agent knows what the effects of each action are. Therefore, the agent can calculate exactly which state results from any sequence of actions and always knows which state it is in. Its percepts provide no new information after each action, although of course they tell the agent the initial state. When the environment is either partially observable or nondeterministic (or both), percepts become useful. In a partially observable environment, every percept helps narrow down the set of possible states the agent might be in, thus making it easier for the agent to achieve its goals. When the environment is nondeterministic, percepts tell the agent which of the possible outcomes of its actions has actually occurred. In both cases, the future percepts cannot be determined in advance and the agent's future actions will depend on those future percepts. CONTINGENCY PLAN So the solution to a problem is not a sequence but a contingency plan (also known as a strategy) that specifies what to do depending on what percepts are received..

### The erratic vacuum world

As an example, we use the vacuum world,

Recall that the state space has eight states, as shown in Figure 4.9. There are three actions—Left, Right, and Suck—and the goal is to clean up all the dirt (states 7 and 8). If the environment is observable, deterministic, and completely known, then the problem is trivially solvable by any of the algorithms in Chapter 3 and the solution is an action sequence. For example, if the initial state is 1, then the action sequence [Suck,Right,Suck] will reach a goal state, 8.

Now suppose that we introduce nondeterminism in the form of a powerful but erratic vacuum cleaner. In the erratic vacuum world, the Suck action works as follows:

- When applied to a dirty square the action cleans the square and sometimes cleans up dirt in an adjacent square, too.

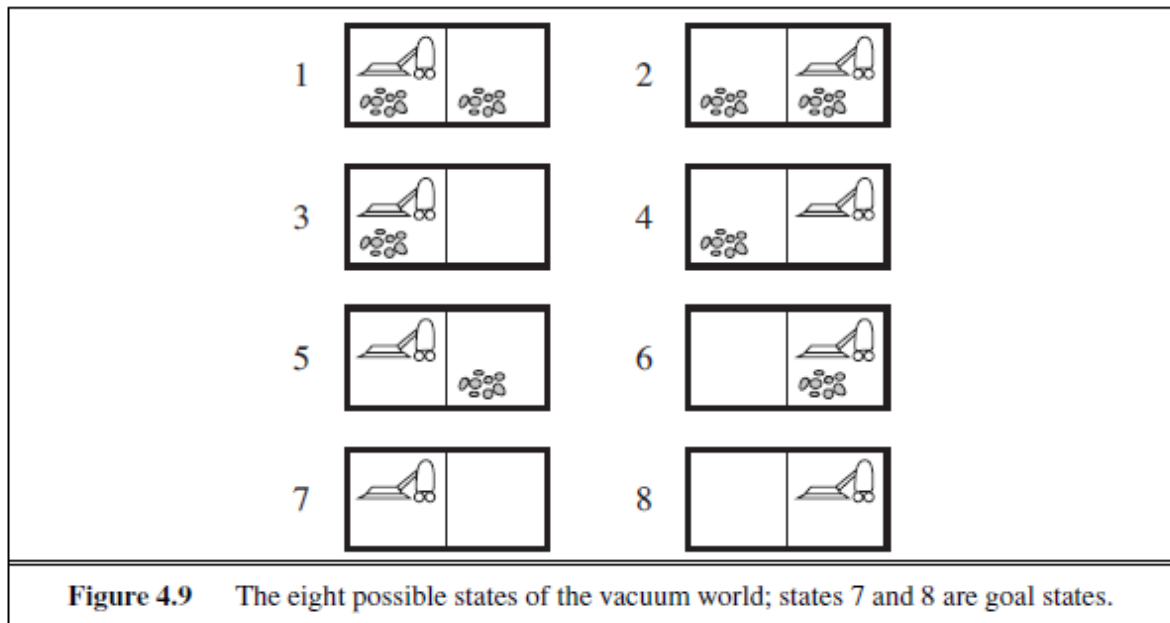
When applied to a clean square the action sometimes deposits dirt on the carpet.<sup>9</sup>

To provide a precise formulation of this problem, we need to generalize the notion of a transition model from Chapter 3. Instead of defining the transition model by a RESULT function that returns a single state, we use a RESULTS function that returns a set of possible outcome states.

For example, in the erratic vacuum world, the Suck action in state 1 leads to a state in the set {5, 7}—the dirt in the right-hand square may or may not be vacuumed up. We also need to generalize the notion of a solution to the problem. For example, if we start in state 1, there is no single sequence of actions that solves the problem. Instead, we need a contingency plan such as the following:

**[Suck, if State =5 then [Right, Suck] else [ ]]** .

Thus, solutions for nondeterministic problems can contain nested if–then–else statements; this



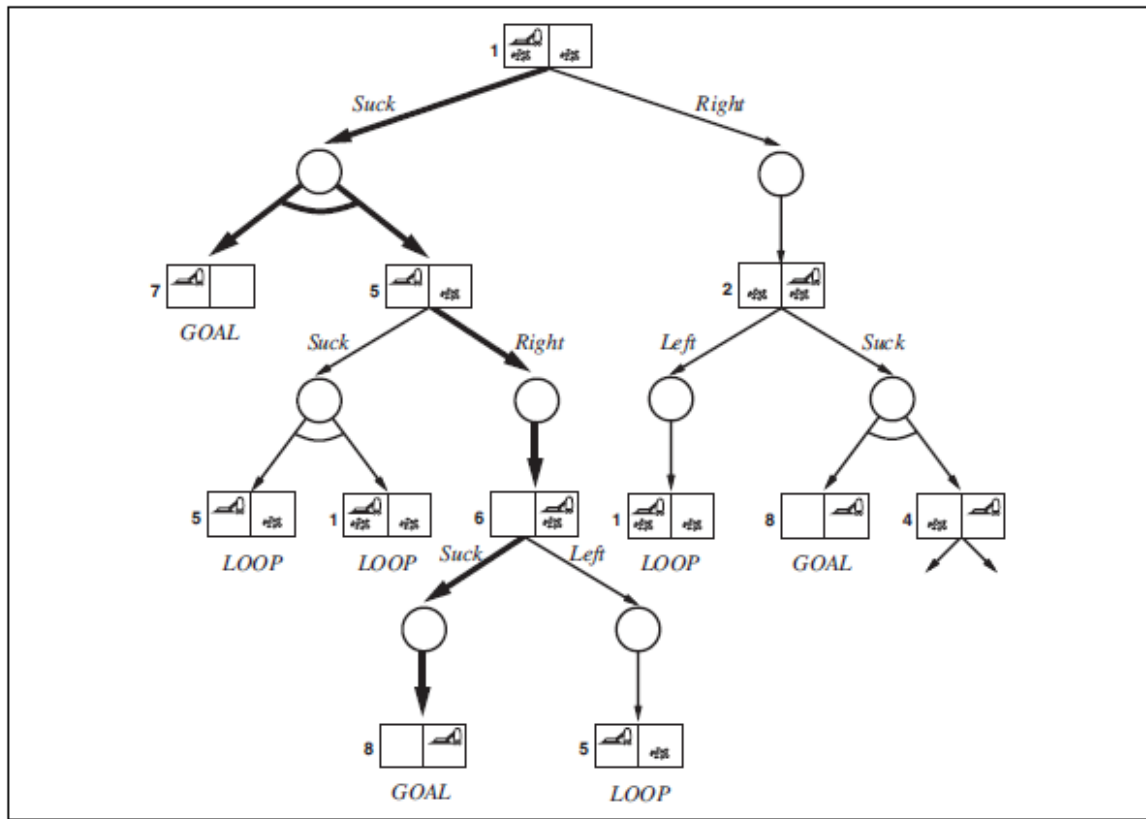
means that they are trees rather than sequences. This allows the selection of actions based on contingencies arising during execution. Many problems in the real, physical world are contingency problems because exact prediction is impossible. For this reason, many people keep their eyes open while walking around or driving.

#### AND–OR search trees

The next question is how to find contingent solutions to nondeterministic problems. As in Chapter 3, we begin by constructing search trees, but here the trees have a different character. In a deterministic environment, the only branching is introduced by the agent’s own choices in each state. We call these nodes OR nodes. OR NODE In the vacuum world, for example, at an OR node the agent chooses Left or Right or Suck. In a nondeterministic environment, branching is also introduced by the environment’s choice of outcome for each action. We call these AND NODE nodes AND nodes.

For example, the Suck action in state 1 leads to a state in the set {5, 7},

so the agent would need to find a plan for state 5 and for state 7. These two kinds of nodes AND–OR TREE alternate, leading to an AND–OR tree as illustrated in Figure A solution for an AND–OR search problem is a subtree that (1) has a goal node at every leaf, (2) specifies one action at each of its OR nodes, and (3) includes every outcome branch at each of its AND nodes. The solution is shown in bold lines in the figure; it corresponds to the plan given in Equation (4.3). (The plan uses if–then–else notation to handle the AND branches, but when there are more than two branches at a node, it might be better to use a case



**Figure 4.10** The first two levels of the search tree for the erratic vacuum world. State nodes are OR nodes where some action must be chosen. At the AND nodes, shown as circles, every outcome must be handled, as indicated by the arc linking the outgoing branches. The solution found is shown in bold lines.

### Searching with partial observations

We now turn to the problem of partial observability, where the agent's percepts do not suffice to pin down the exact state. As noted at the beginning of the previous section, if the agent is in one of several possible states, then an action may lead to one of several possible outcomes—even if the environment is deterministic.

The key concept required for solving partially observable problems is the **belief BELIEF STATE**, representing the agent's current belief about the possible physical states it might be in, given the sequence of actions and percepts up to that point. We begin with the simplest scenario



for studying belief states, which is when the agent has no sensors at all; then we add in partial sensing as well as nondeterministic actions.

#### 4.4.1 Searching with no observation

When the agent's percepts provide no information at all, we have what is called a sensorless problem or sometimes a conformant problem. At first, one might think the sensorless agent has no hope of solving a problem if it has no idea what state it's in; in fact, sensorless problems are quite often solvable. Moreover, sensorless agents can be surprisingly useful, primarily because they don't rely on sensors working properly. In manufacturing systems, for example, many ingenious methods have been developed for orienting parts correctly from an unknown initial position by using a sequence of actions with no sensing at all. The high cost of sensing is another reason to avoid it: for example, doctors often prescribe a broadspectrum antibiotic rather than using the contingent plan of doing an expensive blood test, then waiting for the results to come back, and then prescribing a more specific antibiotic and perhaps hospitalization because the infection has progressed too far.

We can make a sensorless version of the vacuum world. Assume that the agent knows the geography of its world, but doesn't know its location or the distribution of dirt. In that case, its initial state could be any element of the set  $\{1, 2, 3, 4, 5, 6, 7, 8\}$ . Now, consider what happens if it tries the action Right. This will cause it to be in one of the states  $\{2, 4, 6, 8\}$ —the agent now has more information! Furthermore, the action sequence [Right,Suck] will always end up in one of the states  $\{4, 8\}$ . Finally, the sequence [Right,Suck,Left,Suck] is guaranteed COERCION to reach the goal state 7 no matter what the start state. We say that the agent can coerce the world into state 7.

To solve sensorless problems, we search in the space of belief states rather than physical states.<sup>10</sup> Notice that in belief-state space, the problem is fully observable because the agent always knows its own belief state. Furthermore, the solution (if any) is always a sequence of actions. This is because, as in the ordinary problems of Chapter 3, the percepts received after each action are completely predictable—they're always empty! So there are no contingencies to plan for. This is true even if the environment is nondeterministic. It is instructive to see how the belief-state search problem is constructed. Suppose the underlying physical problem  $P$  is defined by ACTIONSP, RESULTP, GOAL-TESTP, and STEP-COSTP. Then we can define the corresponding sensorless problem as follows:

- **Belief states:** The entire belief-state space contains every possible set of physical states. If  $P$  has  $N$  states, then the sensorless problem has up to  $2^N$  states, although many may be unreachable from the initial state.
- **Initial state:** Typically the set of all states in  $P$ , although in some cases the agent will have more knowledge than this.
- **Actions:** This is slightly tricky. Suppose the agent is in belief state  $b = \{s_1, s_2\}$ , but ACTIONSP( $s_1$ )  $\neq$  ACTIONSP( $s_2$ ); then the agent is unsure of which actions are legal. If we assume that illegal actions have no effect on the environment, then it is safe to take the union of all the actions in any of the physical states in the current belief state  $b$ :

$$\text{ACTIONS}(b) = \bigcup_{s \in b} \text{ACTIONS}_P(s)$$

On the other hand, if an illegal action might be the end of the world, it is safer to allow only the *intersection*, that is, the set of actions legal in *all* the states. For the vacuum world, every state has the same legal actions, so both methods give the same result.

- **Transition model:** The agent doesn't know which state in the belief state is the right one; so as far as it knows, it might get to any of the states resulting from applying the action to one of the physical states in the belief state. For deterministic actions, the set of states that might be reached is

$$b' = \text{RESULT}(b, a) = \{s' : s' = \text{RESULT}_P(s, a) \text{ and } s \in b\}. \quad (4.4)$$

With deterministic actions,  $b'$  is never larger than  $b$ . With nondeterminism, we have

$$\begin{aligned} b' = \text{RESULT}(b, a) &= \{s' : s' \in \text{RESULTS}_P(s, a) \text{ and } s \in b\} \\ &= \bigcup_{s \in b} \text{RESULTS}_P(s, a), \end{aligned}$$

which may be larger than  $b$ , as shown in Figure 4.13. The process of generating the new belief state after the action is called the **prediction step**; the notation  $b' = \text{PREDICT}_P(b, a)$  will come in handy.

- **Goal test:** The agent wants a plan that is sure to work, which means that a belief state satisfies the goal only if *all* the physical states in it satisfy  $\text{GOAL-TEST}_P$ . The agent may *accidentally* achieve the goal earlier, but it won't *know* that it has done so.
- **Path cost:** This is also tricky. If the same action can have different costs in different states, then the cost of taking an action in a given belief state could be one of several values. (This gives rise to a new class of problems, which we explore in Exercise 4.9.) For now we assume that the cost of an action is the same in all states and so can be transferred directly from the underlying physical problem.

Even with this improvement, however, sensorless problem-solving as we have described

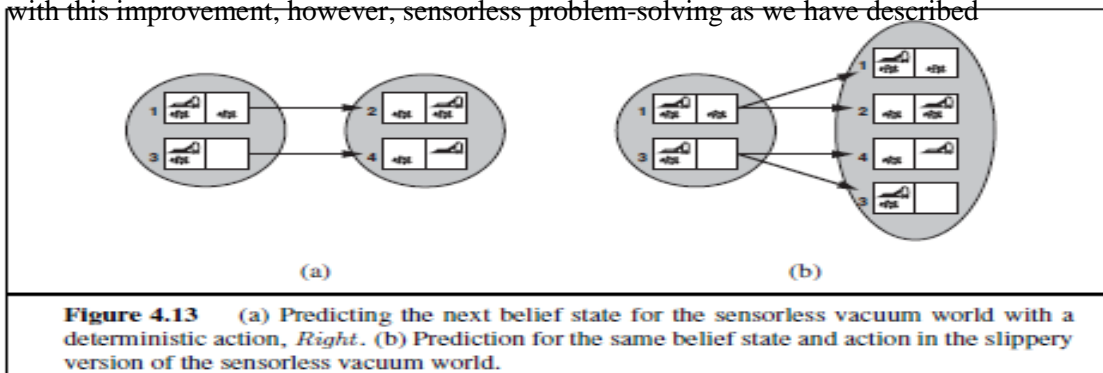
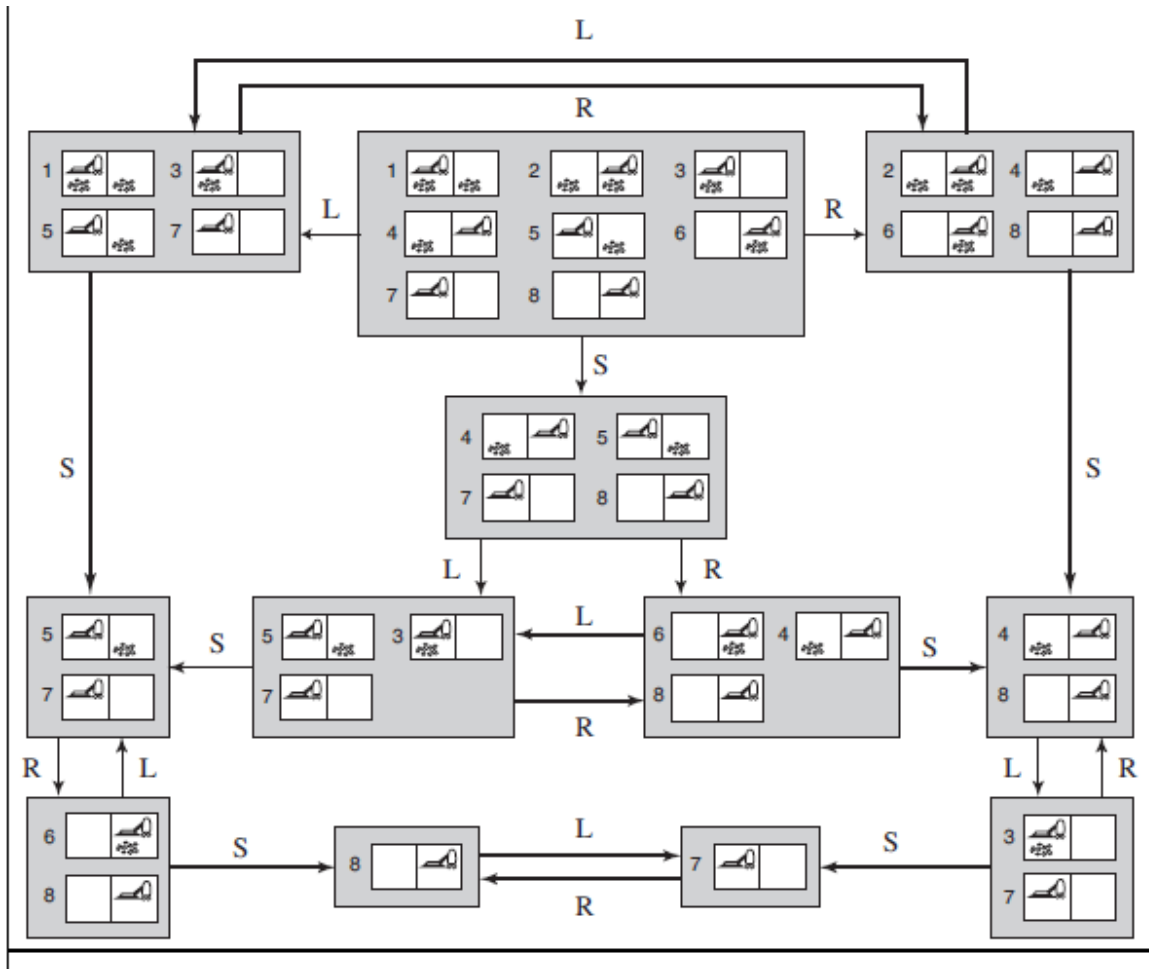


Figure 4.14 shows the reachable belief-state space for the deterministic, sensorless vacuum world. There are only 12 reachable belief states out of  $2^8 = 256$  possible belief states.

The preceding definitions enable the automatic construction of the belief-state problem formulation from the definition of the underlying physical problem. Once this is done, we can apply any of the search algorithms of Chapter 3. In fact, we can do a little bit more than that. In “ordinary” graph search, newly generated states are tested to see if they are identical to existing states. This works for belief states, too; for example, in Figure 4.14, the action sequence [*Suck, Left, Suck*] starting at the initial state reaches the same belief state as [*Right, Left, Suck*], namely,  $\{5, 7\}$ . Now, consider the belief state reached by [*Left*], namely,  $\{1, 3, 5, 7\}$ . Obviously, this is not identical to  $\{5, 7\}$ , but it is a *superset*. It is easy to prove (Exercise 4.8) that if an action sequence is a solution for a belief state  $b$ , it is also a solution for any subset of  $b$ . Hence, we can discard a path reaching  $\{1, 3, 5, 7\}$  if  $\{5, 7\}$  has already been generated. Conversely, if  $\{1, 3, 5, 7\}$  has already been generated and found to be solvable, then any *subset*, such as  $\{5, 7\}$ , is guaranteed to be solvable. This extra level of pruning may dramatically improve the efficiency of sensorless problem solving.

it is seldom feasible in practice. The difficulty is not so much the vastness of the belief-state space—even though it is exponentially larger than the underlying physical state space; in most cases the branching factor and solution length in the belief-state space and physical state space are not so different. The real difficulty lies with the size of each belief state. For example, the initial belief state for the  $10 \times 10$  vacuum world contains  $100 \times 2^{100}$  or around  $10^{32}$  physical states—far too many if we use the atomic representation, which is an explicit list of states.

One solution is to represent the belief state by some more compact description. In English, we could say



the agent knows “Nothing” in the initial state; after moving *Left*, we could say, “Not in the rightmost column,” and so on. Chapter 7 explains how to do this in a formal representation scheme. Another approach is to avoid the standard search algorithms, which treat belief states as black boxes just like any other problem state. Instead, we can look *inside* the

Figure 4.14 The reachable portion of the belief-state space for the deterministic, sensorless vacuum world. Each shaded box corresponds to a single belief state. At any given point, the agent is in a particular belief state but does not know which physical state it is in. The initial belief state (complete ignorance) is the top center box. Actions are represented by labeled links. Self-loops are omitted for clarity.

belief states and develop **incremental belief-state search** algorithms that build up the solution one physical state at a time. For example, in the sensorless vacuum world, the initial belief state is  $\{1, 2, 3, 4, 5, 6, 7, 8\}$ , and we have to find an action sequence that works in all 8 states. We can do this by first finding a solution that works for state 1; then we check if it works for state 2; if not, go back and find a different solution for state 1, and so on. Just as an AND–OR search has to find a solution for every branch at an AND node, this algorithm has to find a solution for every state in the belief state; the difference is that AND–OR search can find a different solution for each branch, whereas an incremental belief-state search has to find *one* solution that works for *all* the states. The main advantage of the incremental approach is that it is typically able to detect failure quickly—when a belief state is unsolvable, it is usually the case that a small subset of the belief state, consisting of the first few states examined, is also unsolvable

### Searching with observations

For a general partially observable problem, we have to specify how the environment generates percepts for the agent. For example, we might define the local-sensing vacuum world to be one in which the agent has a position sensor and a local dirt sensor but has no sensor capable of detecting dirt in other squares. The formal problem specification includes a PERCEPT( $s$ ) function that returns the percept received in a given state. (If sensing is nondeterministic, then we use a PERCEPTS function that returns a set of possible percepts.) For example, in the local-sensing vacuum world, the PERCEPT in state 1 is [A, Dirty]. Fully observable problems are a special case in which PERCEPT( $s$ )= $s$  for every state  $s$ , while sensorless problems are a special case in which PERCEPT( $s$ )= $\text{null}$ .

When observations are partial, it will usually be the case that several states could have produced any given percept. For example, the percept [A, Dirty] is produced by state 3 as well as by state 1. Hence, given this as the initial percept, the initial belief state for the local-sensing vacuum world will be  $\{1, 3\}$ . The ACTIONS, STEP-COST, and GOAL-TEST are constructed from the underlying physical problem just as for sensorless problems, but the transition model is a bit more complicated. We can think of transitions from one belief state to the next for a particular action as occurring in three stages, as shown in Figure 4.15:

- The **prediction stage** is the same as for sensorless problems: given the action  $a$  in belief state  $b$ , the predicted belief state is  $\hat{b} = \text{PREDICT}(b, a)$ .<sup>11</sup>
- The **observation prediction stage** determines the set of percepts  $o$  that could be observed in the predicted belief state:

$$\text{POSSIBLE-PERCEPTS}(\hat{b}) = \{o : o = \text{PERCEPT}(s) \text{ and } s \in \hat{b}\} .$$

- The **update stage** determines, for each possible percept, the belief state that would result from the percept. The new belief state  $b_o$  is just the set of states in  $\hat{b}$  that could have produced the percept:

$$b_o = \text{UPDATE}(\hat{b}, o) = \{s : o = \text{PERCEPT}(s) \text{ and } s \in \hat{b}\} .$$

Notice that each updated belief state  $b_o$  can be no larger than the predicted belief state  $\hat{b}$ ; observations can only help reduce uncertainty compared to the sensorless case. Moreover, for deterministic sensing, the belief states for the different possible percepts will be disjoint, forming a *partition* of the original predicted belief state.

Putting these three stages together, we obtain the possible belief states resulting from a given action and the subsequent possible percepts:

**RESULTS(b, a) = {bo : bo = UPDATE(PREDICT(b, a), o) and  
POSSIBLE-PERCEPTS(PREDICT(b, a))} .**

Again, the nondeterminism in the partially observable problem comes from the inability to predict exactly which percept will be received after acting; underlying nondeterminism in the physical environment may contribute to this inability by enlarging the belief state at the prediction stage, leading to more percepts at the observation stage.

### 4.4.3 Solving partially observable problems

The preceding section showed how to derive the RESULTS function for a nondeterministic belief-state problem from an underlying physical problem and the PERCEPT function. Given such a formulation, the AND–OR search algorithm of Figure 4.11 can be applied directly to derive a solution. Figure 4.16 shows part of the search tree for the local-sensing vacuum world, assuming an initial percept [A, Dirty]. The solution is the conditional plan

**[Suck, Right, if Bstate = {6} then Suck else [ ]]** .

Notice that, because we supplied a belief-state problem to the AND–OR search algorithm, it returned a conditional plan that tests the belief state rather than the actual state. This is as it should be: in a partially observable environment the agent won't be able to execute a solution that requires testing the actual state.

As in the case of standard search algorithms applied to sensorless problems, the AND–OR search algorithm treats belief states as black boxes, just like any other states. One can improve on this by checking for previously generated belief states that are subsets or supersets of the current state, just as for sensorless problems. One can also derive incremental search algorithms, analogous to those described for sensorless problems, that provide substantial speedups over the black-box approach.

**Unit – III: Reinforcement Learning:** Introduction, Passive Reinforcement Learning, Active Reinforcement Learning, Generalization in Reinforcement Learning, Policy Search, applications of RL

**Natural Language Processing:** Language Models, Text Classification, Information Retrieval, Information Extraction.

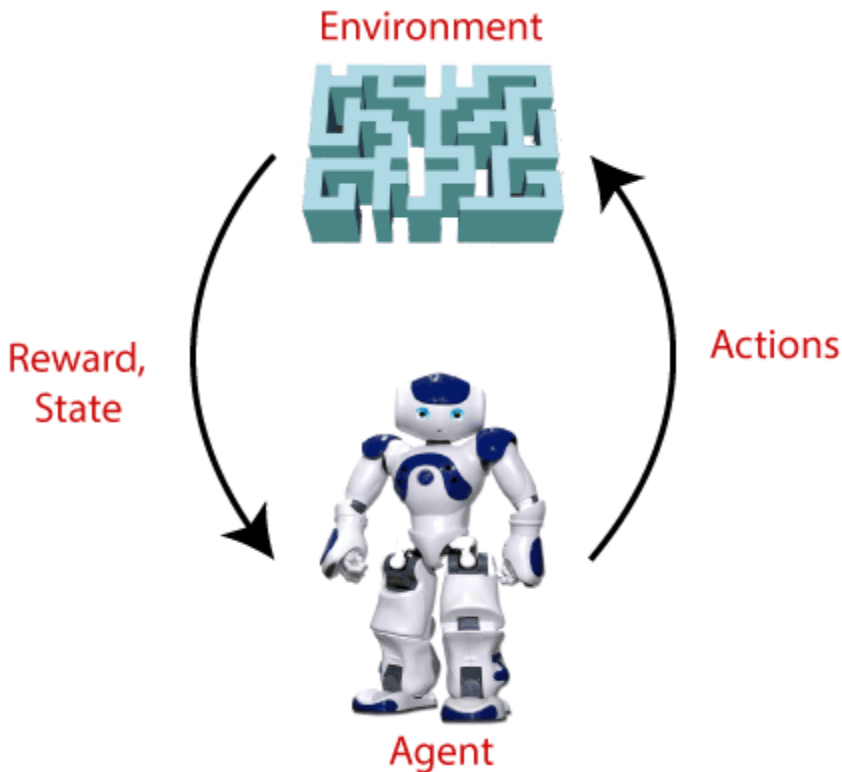
---

**Reinforcement Learning:** Introduction

## What is Reinforcement Learning?

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike **supervised learning**.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that "**Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that.**" How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- It is a core part of **Artificial intelligence**, and all **AI agent** works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.
- **Example:** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.

- The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.



---

## Terms used in Reinforcement Learning

- **Agent ()**: An entity that can perceive/explore the environment and act upon it.
- **Environment ()**: A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action ()**: Actions are the moves taken by an agent within the environment.
- **State ()**: State is a situation returned by the environment after each action taken by the agent.
- **Reward ()**: A feedback returned to the agent from the environment to evaluate the action of the agent.

- **Policy ( $\pi$ ):** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value ( $V$ ):** It is expected long-term return with the discount factor and opposite to the short-term reward.
- **Q-value ( $Q$ ):** It is mostly similar to the value, but it takes one additional parameter as a current action ( $a$ ).

## Key Features of Reinforcement Learning

- In RL, the agent is not instructed about the environment and what actions need to be taken.
- It is based on the hit and trial process.
- The agent takes the next action and changes states according to the feedback of the previous action.
- The agent may get a delayed reward.
- The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

## Approaches to implement Reinforcement Learning

There are mainly three ways to implement reinforcement-learning in ML, which are:

### 1. Value-based:

The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy  $\pi$ .

### 2. Policy-based:

Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward. The policy-based approach has mainly two types of policy:

- **Deterministic:** The same action is produced by the policy ( $\pi$ ) at any state.
  - **Stochastic:** In this policy, probability determines the produced action.
3. **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no



particular solution or algorithm for this approach because the model representation is different for each environment.

---

## Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below:

1. Policy
2. Reward Signal
3. Value Function
4. Model of the environment

**1) Policy:** A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states. A policy is the core element of the RL as it alone can define the behavior of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:

**For deterministic policy:**  $a = \pi(s)$

**For stochastic policy:**  $\pi(a | s) = P[At = a | St = s]$

**2) Reward Signal:** The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a **reward signal**. These rewards are given according to the good and bad actions taken by the agent. The agent's main objective is to maximize the total number of rewards for good actions. The reward signal can change the policy, such as if an action selected by the agent leads to low reward, then the policy may change to select other actions in the future.

**3) Value Function:** The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the **immediate signal for each good and bad action**, whereas a value function specifies **the good state and action for the future**. The value function depends on the reward as, without reward, there could be no value. The goal of estimating values is to achieve more rewards.

**4) Model:** The last element of reinforcement learning is the model, which mimics the behavior of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward.

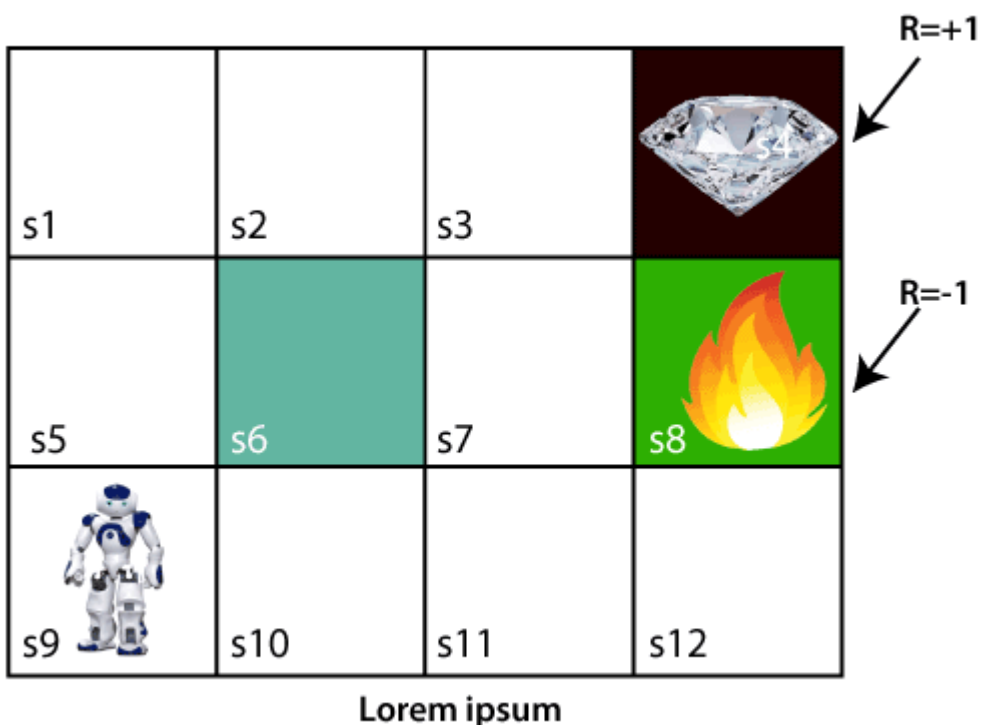
The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems **with the help of the model** are termed as the **model-based approach**. Comparatively, an approach **without using a model** is called a **model-free approach**.

## How does Reinforcement Learning Work?

To understand the working process of the RL, we need to consider two main things:

- **Environment:** It can be anything such as a room, maze, football ground, etc.
- **Agent:** An intelligent agent such as AI robot.

Let's take an example of a maze environment that the agent needs to explore. Consider the below image:







In the above image, the agent is at the very first block of the maze. The maze is consisting of an  $S_6$  block, which is a **wall**,  $S_8$  a **fire pit**, and  $S_4$  a **diamond block**.

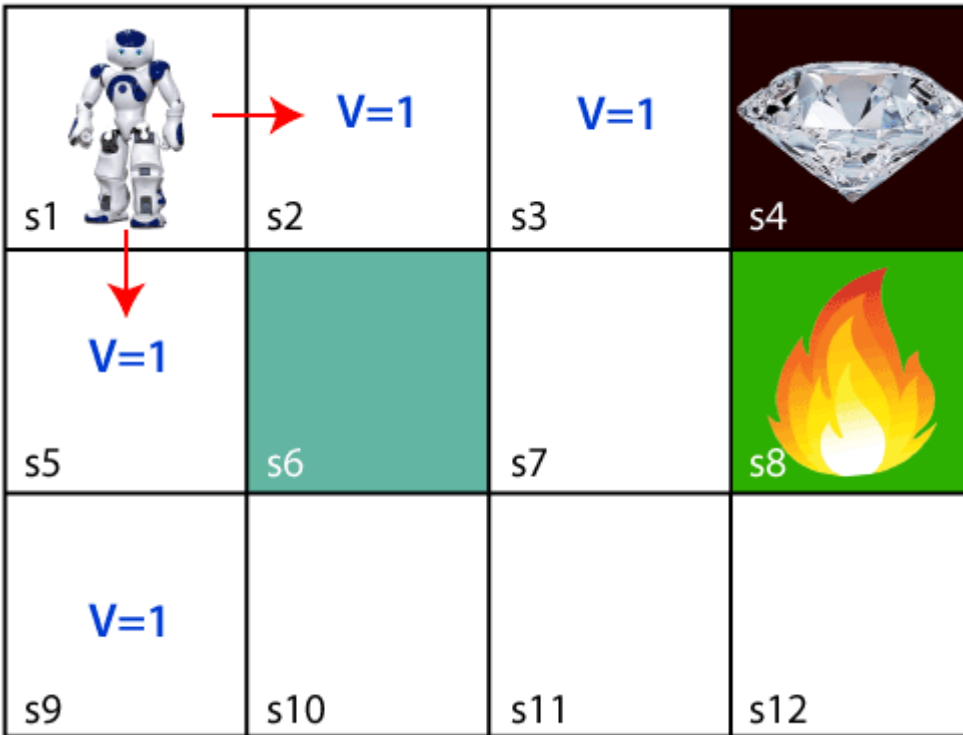
The agent cannot cross the  $S_6$  block, as it is a solid wall. If the agent reaches the  $S_4$  block, then get the **+1 reward**; if it reaches the fire pit, then gets **-1 reward point**. It can take four actions: **move up, move down, move left, and move right**.

The agent can take any path to reach to the final point, but he needs to make it in possible fewer steps. Suppose the agent considers the path **S9-S5-S1-S2-S3**, so he will get the +1-reward point.

The agent will try to remember the preceding steps that it has taken to reach the final step. To memorize the steps, it assigns 1 value to each previous step. Consider the below step:

<b>V=1</b> s1	<b>V=1</b> s2	<b>V=1</b> s3	 s4
<b>V=1</b> s5	 s6	s7	 s8
 <b>V=1</b> s9	s10	s11	s12

Now, the agent has successfully stored the previous steps assigning the 1 value to each previous block. But what will the agent do if he starts moving from the block, which has 1 value block on both sides? Consider the below diagram:



It will be a difficult condition for the agent whether he should go up or down as each block has the same value. So, the above approach is not suitable for the agent to reach the destination. Hence to solve the problem, we will use the **Bellman equation**, which is the main concept behind reinforcement learning.

It will be a difficult condition for the agent whether he should go up or down as each block has the same value. So, the above approach is not suitable for the agent to reach the destination. Hence to solve the problem, we will use the **Bellman equation**, which is the main concept behind reinforcement learning.

## The Bellman Equation

The Bellman equation was introduced by the Mathematician **Richard Ernest Bellman in the year 1953**, and hence it is called as a Bellman equation. It is associated with dynamic programming and used to calculate the values of a decision problem at a certain point by including the values of previous states.

It is a way of calculating the value functions in dynamic programming or environment that leads to modern reinforcement learning.

The key-elements used in Bellman equations are:

- Action performed by the agent is referred to as "a"
- State occurred by performing the action is "s."
- The reward/feedback obtained for each good and bad action is "R."
- A discount factor is Gamma " $\gamma$ ."

The Bellman equation can be written as:

$$1. V(s) = \max [R(s,a) + \gamma V(s')] ]$$

Where,

**$V(s)$  = value calculated at a particular point.**

**$R(s,a)$  = Reward at a particular state  $s$  by performing an action.**

**$\gamma$  = Discount factor**

**$V(s')$  = The value at the previous state.**

In the above equation, we are taking the max of the complete values because the agent tries to find the optimal solution always.

So now, using the Bellman equation, we will find value at each state of the given environment. We will start from the block, which is next to the target block.

**For 1st block:**

$V(s_3) = \max [R(s,a) + \gamma V(s')]$ , here  $V(s') = 0$  because there is no further state to move.

$V(s_3) = \max[R(s,a)] \Rightarrow V(s_3) = \max[1] \Rightarrow \mathbf{V(s_3) = 1}$ .

**For 2nd block:**

$V(s_2) = \max [R(s,a) + \gamma V(s')]$ , here  $\gamma = 0.9$  (lets),  $V(s') = 1$ , and  $R(s, a) = 0$ , because there is no reward at this state.

$V(s_2) = \max[0.9(1)] \Rightarrow V(s) = \max[0.9] \Rightarrow \mathbf{V(s_2) = 0.9}$

**For 3rd block:**

$V(s_1) = \max [R(s,a) + \gamma V(s')]$ , here  $\gamma = 0.9$  (lets),  $V(s') = 0.9$ , and  $R(s, a) = 0$ , because there is no reward at this state also.

$$V(s_1) = \max[0.9(0.9)] \Rightarrow V(s_3) = \max[0.81] \Rightarrow \mathbf{V(s_1) = 0.81}$$

**For 4th block:**

$V(s_5) = \max [R(s,a) + \gamma V(s')]$ , here  $\gamma = 0.9$  (lets),  $V(s') = 0.81$ , and  $R(s, a) = 0$ , because there is no reward at this state also.





$$V(s_5) = \max[0.9(0.81)] \Rightarrow V(s_5) = \max[0.73] \Rightarrow \mathbf{V(s_5) = 0.73}$$

**For 5th block:**





$V(s_9) = \max [R(s,a) + \gamma V(s')]$ , here  $\gamma = 0.9$  (lets),  $V(s') = 0.73$ , and  $R(s, a) = 0$ , because there is no reward at this state also.

$$V(s_9) = \max[0.9(0.73)] \Rightarrow V(s_4) = \max[0.66] \Rightarrow \mathbf{V(s_4) = 0.66}$$



**Consider the below image:**

V=0.81 s1	V=0.9 s2	V=1 s3	 s4
V=0.73 s5		s7	 s8
 V=0.66 s9	s10	s11	s12

Now, we will move further to the 6<sup>th</sup> block, and here agent may change the route because it always tries to find the optimal path. So now, let's consider from the block next to the fire pit.

V=0.81 s1	V=0.9 s2	V=1 s3	 s4
V=0.73 s5		 s7	 s8
V=0.66 s9	s10	s11	s12

Now, the agent has three options to move; if he moves to the blue box, then he will feel a bump if he moves to the fire pit, then he will get the -1 reward. But here we are taking only positive rewards, so for this, he will move to upwards only. The complete block values will be calculated using this formula. Consider the below image:

V=0.81 s1	V=0.9 s2	V=1 s3	 s4
V=0.73 s5	s6	V=0.9 s7	 s8
V=0.66 s9	V=0.73 s10	V=0.81 s11	V=0.73 s12

## Types of Reinforcement learning

There are mainly two types of reinforcement learning, which are:

- **Positive Reinforcement**
- **Negative Reinforcement**

### **Positive Reinforcement:**

The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior.

This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

### **Negative Reinforcement:**

The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition.



It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior.

## How to represent the agent state?

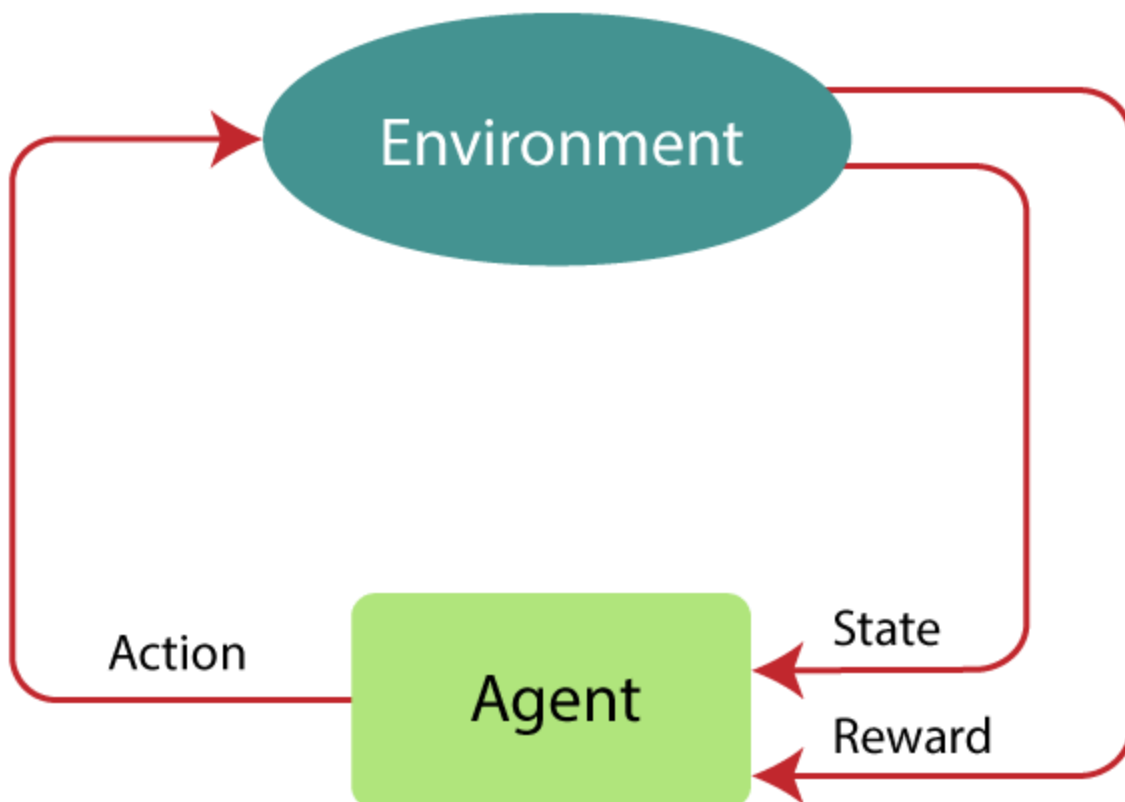
We can represent the agent state using the **Markov State** that contains all the required information from the history. The State  $S_t$  is Markov state if it follows the given condition:

$$P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \dots, S_t]$$

The Markov state follows the **Markov property**, which says that the future is independent of the past and can only be defined with the present. The RL works on fully observable environments, where the agent can observe the environment and act for the new state. The complete process is known as Markov Decision process, which is explained below:

## Markov Decision Process

Markov Decision Process or MDP, is used to **formalize the reinforcement learning problems**. If the environment is completely observable, then its dynamic can be modeled as a **Markov Process**. In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.



MDP is used to describe the environment for the RL, and almost all the RL problem can be formalized using MDP.

MDP contains a tuple of four elements  $(S, A, P_a, R_a)$ :

- A set of finite States  $S$
- A set of finite Actions  $A$
- Rewards received after transitioning from state  $S$  to state  $S'$ , due to action  $a$ .
- Probability  $P_a$ .

MDP uses **Markov property**, and to better understand the MDP, we need to learn about it.

### Markov Property:

It says that ***"If the agent is present in the current state  $S_1$ , performs an action  $a_1$  and move to the state  $s_2$ , then the state transition from  $s_1$  to  $s_2$  only depends on the current state and future action and states do not depend on past actions, rewards, or states."***

Or, in other words, as per Markov Property, the current state transition does not depend on any past action or state. Hence, MDP is an RL problem that satisfies the Markov property. Such as in a **Chess game, the players only focus on the current state and do not need to remember past actions or states.**

### Finite MDP:

A finite MDP is when there are finite states, finite rewards, and finite actions. In RL, we consider only the finite MDP.

### Markov Process:

Markov Process is a memoryless process with a sequence of random states  $S_1, S_2, \dots, S_t$  that uses the Markov Property. Markov process is also known as Markov chain, which is a tuple  $(S, P)$  on state  $S$  and transition function  $P$ . These two components ( $S$  and  $P$ ) can define the dynamics of the system.

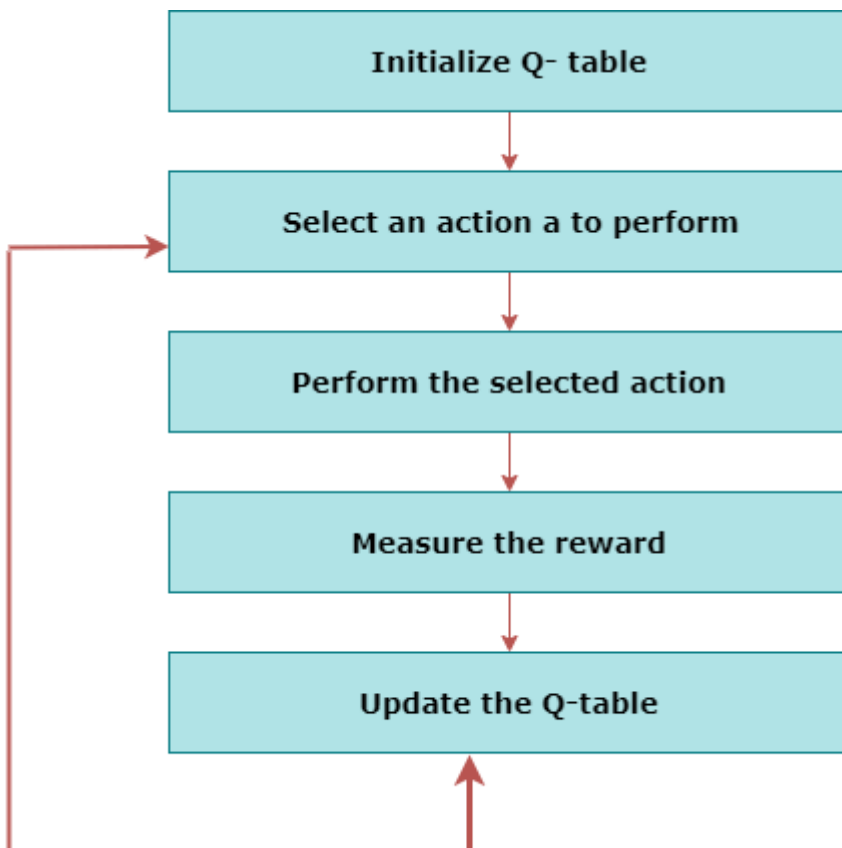
---

## Reinforcement Learning Algorithms

Reinforcement learning algorithms are mainly used in AI applications and gaming applications. The main used algorithms are:

- **Q-Learning:**

- Q-learning is an **Off policy RL algorithm**, which is used for the temporal difference Learning. The temporal difference learning methods are the way of comparing temporally successive predictions.
- It learns the value function  $Q(S, a)$ , which means how good to take action "a" at a particular state "s."
- The below flowchart explains the working of Q- learning:



- **State Action Reward State action (SARSA):**

- SARSA stands for **State Action Reward State action**, which is an **on-policy** temporal difference learning method. The on-policy control method selects the action for each state while learning using a specific policy.
- The goal of SARSA is to calculate the  $Q \pi (s, a)$  for the selected current policy  $\pi$  and all pairs of (s-a).
- The main difference between Q-learning and SARSA algorithms is that **unlike Q-learning, the maximum reward for the next state is not required for updating the Q-value in the table.**

- In SARSA, new action and reward are selected using the same policy, which has determined the original action.
- The SARSA is named because it uses the quintuple **Q(s, a, r, s', a')**. Where,
  - s: original state**
  - a: Original action**
  - r: reward observed while following the states**
  - s' and a': New state, action pair.**
- **Deep Q Neural Network (DQN):**
  - As the name suggests, DQN is a **Q-learning using Neural networks**.
  - For a big state space environment, it will be a challenging and complex task to define and update a Q-table.
  - To solve such an issue, we can use a DQN algorithm. Where, instead of defining a Q-table, neural network approximates the Q-values for each action and state.


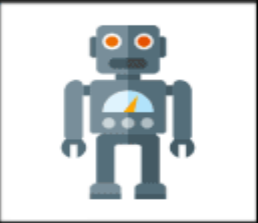
Now, we will expand the Q-learning.

### Q-Learning Explanation:

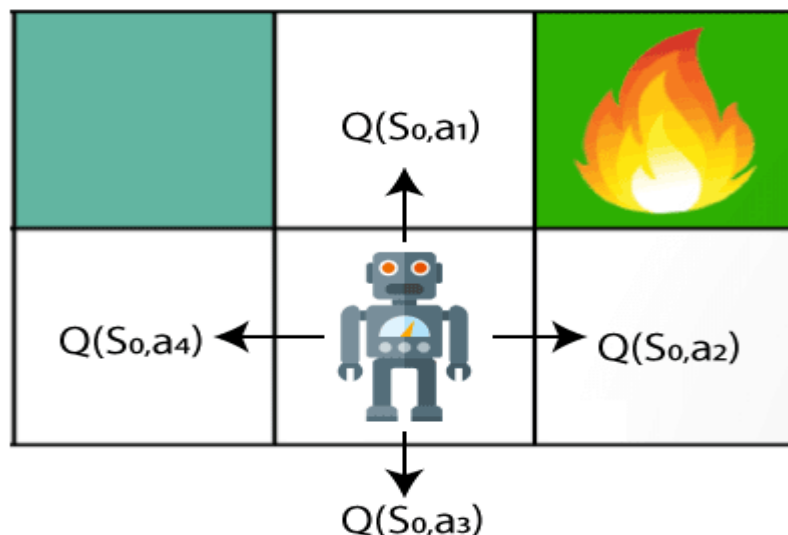
- Q-learning is a popular model-free reinforcement learning algorithm based on the Bellman equation.
- **The main objective of Q-learning is to learn the policy which can inform the agent that what actions should be taken for maximizing the reward under what circumstances.**
- It is an **off-policy RL** that attempts to find the best action to take at a current state.
- The goal of the agent in Q-learning is to maximize the value of Q.
- The value of Q-learning can be derived from the Bellman equation. Consider the Bellman equation given below:

$$V(s) = \max [R(s,a) + \gamma \sum_{s'} P(s, a, s')V(s')] ]$$

In the equation, we have various components, including reward, discount factor ( $\gamma$ ), probability, and end states  $s'$ . But there is no any Q-value is given so first consider the below image:

	$V(s_2)$	
$V(s_1)$		$V(s_3)$

In the above image, we can see there is an agent who has three values options,  $V(s_1)$ ,  $V(s_2)$ ,  $V(s_3)$ . As this is MDP, so agent only cares for the current state and the future state. The agent can go to any direction (Up, Left, or Right), so he needs to decide where to go for the optimal path. Here agent will take a move as per probability bases and changes the state. But if we want some exact moves, so for this, we need to make some changes in terms of Q-value. Consider the below image:



Q- represents the quality of the actions at each state. So instead of using a value at each state, we will use a pair of state and action, i.e.,  $Q(s, a)$ . Q-value specifies that which action is more lubricative than others, and according to the best Q-value, the agent takes his next move. The Bellman equation can be used for deriving the Q-value.

To perform any action, the agent will get a reward  $R(s, a)$ , and also he will end up on a certain state, so the Q -value equation will be:

$$Q(S, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') V(s')$$

Hence, we can say that,  $V(s) = \max [Q(s, a)]$

$$Q(S, a) = R(s, a) + \gamma \sum_{s'} (P(s, a, s') \max_{a'} Q(s', a'))$$

**The above formula is used to estimate the Q-values in Q-Learning.**

**What is 'Q' in Q-learning?**

The Q stands for **quality** in **Q-learning**, which means it specifies the quality of an action taken by the agent.

### Q-table:

A Q-table or matrix is created while performing the Q-learning. The table follows the state and action pair, i.e., [s, a], and initializes the values to zero. After each action, the table is updated, and the q-values are stored within the table.

The RL agent uses this Q-table as a reference table to select the best action based on the q-values.

## Difference between Reinforcement Learning and Supervised Learning

The Reinforcement Learning and Supervised Learning both are the part of machine learning, but both types of learnings are far opposite to each other. The RL agents interact with the environment, explore it, take action, and get rewarded. Whereas supervised learning algorithms learn from the labeled dataset and, on the basis of the training, predict the output.

The difference table between RL and Supervised learning is given below:

<b>Reinforcement Learning</b>	<b>Supervised Learning</b>
RL works by interacting with the environment.	Supervised learning works on the existing dataset.
The RL algorithm works like the human brain works when making some decisions.	Supervised Learning works as when a human learns things in the supervision of a guide.
There is no labeled dataset is present	The labeled dataset is present.
No previous training is provided to the learning agent.	Training is provided to the algorithm so that it can predict the output.
RL helps to take decisions sequentially.	In Supervised learning, decisions are made when input is given.

## Reinforcement Learning Applications



**Robotics:** RL is used in **Robot navigation, Robo-soccer, walking, juggling**, etc.

**Control:** RL can be used for **adaptive control** such as Factory processes, admission control in telecommunication, and Helicopter pilot is an example of reinforcement learning.

**Game Playing:** RL can be used in **Game playing** such as tic-tac-toe, chess, etc.

**Chemistry:** RL can be used for optimizing the chemical reactions.

**Business:** RL is now used for business strategy planning.

**Manufacturing:** In various automobile manufacturing companies, the robots use deep reinforcement learning to pick goods and put them in some containers.

**Finance Sector:** The RL is currently used in the finance sector for evaluating trading strategies.

### Conclusion:

From the above discussion, we can say that Reinforcement Learning is one of the most interesting and useful parts of Machine learning. In RL, the agent explores the environment by exploring it without any human intervention. It is the main learning algorithm that is used in Artificial Intelligence. But there are some cases where it should not be used, such as if you



have enough data to solve the problem, then other ML algorithms can be used more efficiently. The main issue with the RL algorithm is that some of the parameters may affect the speed of the learning, such as delayed feedback.

## 1. **Policy-Based Learning Approach:**

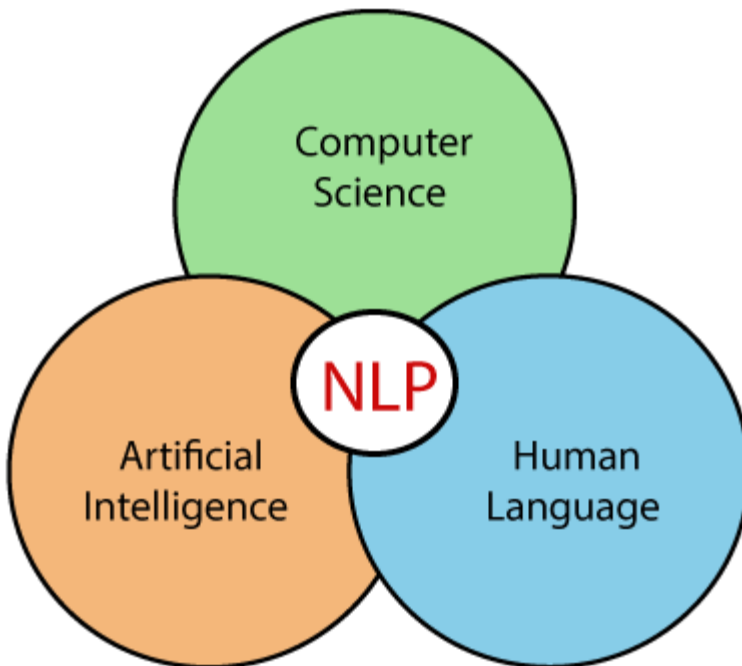
Policy-based Learning searches directly for the optimal policy which achieves the maximum future reward. In policy-based approach, we want to directly optimize the policy function  $\pi(s)$  without using a value function. The policy is what defines the agent behavior at a given time. We learn a policy function. This lets us map each state to the best corresponding action.

This approach has two types of policy:

1. **Deterministic:** a policy at a given state will always return the same action.
2. **Stochastic:** output a distribution probability over actions.

## What is NLP?

NLP stands for **Natural Language Processing**, which is a part of **Computer Science**, **Human language**, and **Artificial Intelligence**. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as **translation**, **automatic summarization**, **Named Entity Recognition (NER)**, **speech recognition**, **relationship extraction**, and **topic segmentation**.



## History of NLP

### (1940-1960) - Focused on Machine Translation (MT)

The Natural Languages Processing started in the year 1940s.

**1948** - In the Year 1948, the first recognisable NLP application was introduced in Birkbeck College, London.

**1950s** - In the Year 1950s, there was a conflicting view between linguistics and computer science. Now, Chomsky developed his first book syntactic structures and claimed that language is generative in nature.

In 1957, Chomsky also introduced the idea of Generative Grammar, which is rule based descriptions of syntactic structures.

### (1960-1980) - Flavored with Artificial Intelligence (AI)

In the year 1960 to 1980, the key developments were:

### **Augmented Transition Networks (ATN)**

Augmented Transition Networks is a finite state machine that is capable of recognizing regular languages.

### **Case Grammar**

Case Grammar was developed by **Linguist Charles J. Fillmore** in the year 1968. Case Grammar uses languages such as English to express the relationship between nouns and verbs by using the preposition.

In Case Grammar, case roles can be defined to link certain kinds of verbs and objects.

**For example:** "Neha broke the mirror with the hammer". In this example case grammar identify Neha as an agent, mirror as a theme, and hammer as an instrument.

In the year 1960 to 1980, key systems were:

### **SHRDLU**

SHRDLU is a program written by **Terry Winograd** in 1968-70. It helps users to communicate with the computer and moving objects. It can handle instructions such as "pick up the green boll" and also answer the questions like "What is inside the black box." The main importance of SHRDLU is that it shows those syntax, semantics, and reasoning about the world that can be combined to produce a system that understands a natural language.

### **LUNAR**

LUNAR is the classic example of a Natural Language database interface system that is used ATNs and Woods' Procedural Semantics. It was capable of translating elaborate natural language expressions into database queries and handle 78% of requests without errors.

### **1980 - Current**

Till the year 1980, natural language processing systems were based on complex sets of hand-written rules. After 1980, NLP introduced machine learning algorithms for language processing.

In the beginning of the year 1990s, NLP started growing faster and achieved good process accuracy, especially in English Grammar. In 1990 also, an electronic text introduced, which provided a good resource for training and examining natural language programs. Other factors may include the availability of computers with fast CPUs and more memory. The major factor behind the advancement of natural language processing was the Internet.

Now, modern NLP consists of various applications, like **speech recognition**, **machine translation**, and **machine text reading**. When we combine all these applications then it allows the

artificial intelligence to gain knowledge of the world. Let's consider the example of AMAZON ALEXA, using this robot you can ask the question to Alexa, and it will reply to you.

## Advantages of NLP

- NLP helps users to ask questions about any subject and get a direct response within seconds.
- NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
- NLP helps computers to communicate with humans in their languages.
- It is very time efficient.
- Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

## Disadvantages of NLP

A list of disadvantages of NLP is given below:

- NLP may not show context.
- NLP is unpredictable
- NLP may require more keystrokes.
- NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

## Components of NLP

There are the following two components of NLP -

### 1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks -

- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

### 2. Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

*Note: The NLU is difficult than NLG.*

### Difference between NLU and NLG

NLU	NLG
NLU is the process of reading and interpreting language.	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs.	It produces constructing natural language outputs from non-linguistic inputs.

## Applications of NLP

There are the following applications of NLP -

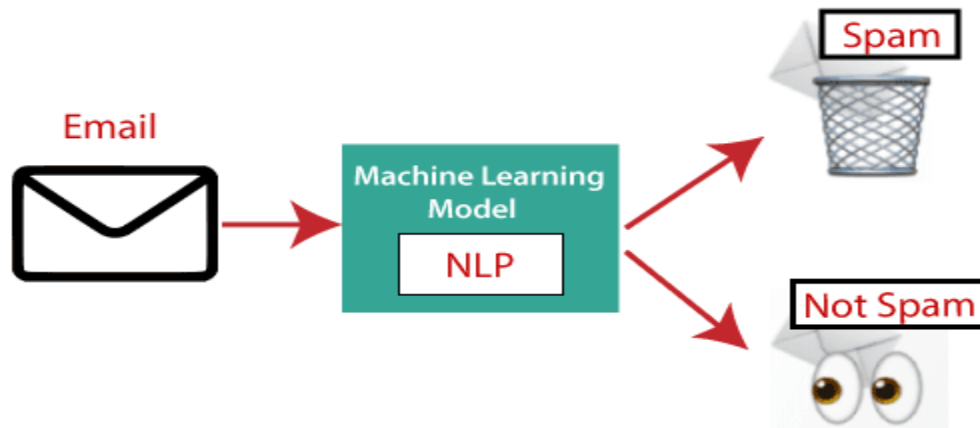
### 1. Question Answering

Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.



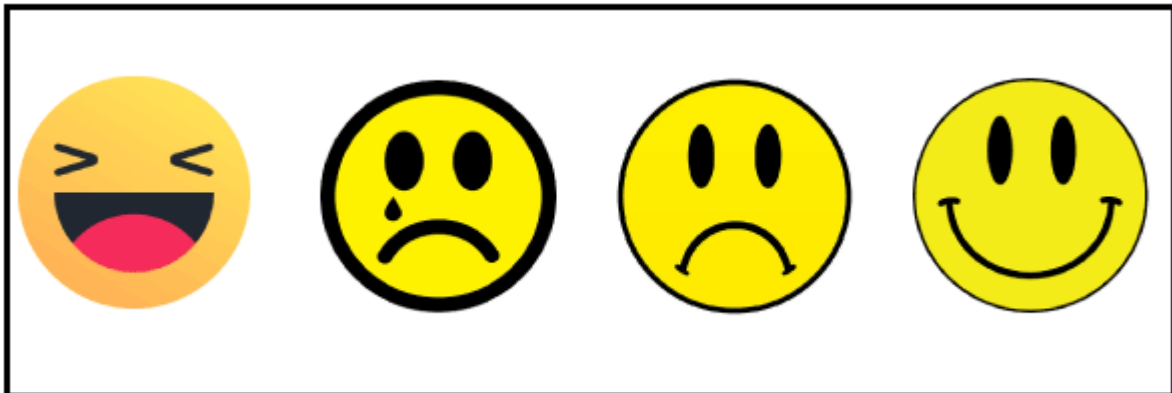
## 2. Spam Detection

Spam detection is used to detect unwanted e-mails getting to a user's inbox.



## 3. Sentiment Analysis

Sentiment Analysis is also known as **opinion mining**. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (Natural Language Processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)



## 4. Machine Translation

Machine translation is used to translate text or speech from one natural language to another natural language.



**Example:** Google Translator

## 5. Spelling correction

Microsoft Corporation provides word processor software like MS-word, PowerPoint for the spelling correction.



## 6. Speech Recognition

Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

## 7. Chatbot

Implementing the Chatbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.

## 8. Information extraction

Information extraction is one of the most important applications of NLP. It is used for extracting structured information from unstructured or semi-structured machine-readable documents.

## 9. Natural Language Understanding (NLU)

It converts a large set of text into more formal representations such as first-order logic structures that are easier for the computer programs to manipulate notations of the natural language processing.

---

# How to build an NLP pipeline

There are the following steps to build an NLP pipeline -

## Step1: Sentence Segmentation

Sentence Segment is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences.

**Example:** Consider the following paragraph -

**Independence Day is one of the important festivals for every Indian citizen. It is celebrated on the 15th of August each year ever since India got independence from the British rule. The day celebrates independence in the true sense.**

**Sentence Segment produces the following result:**

1. "Independence Day is one of the important festivals for every Indian citizen."
2. "It is celebrated on the 15th of August each year ever since India got independence from the British rule."
3. "This day celebrates independence in the true sense."

## Step2: Word Tokenization

Word Tokenizer is used to break the sentence into separate words or tokens.

**Example:**

JavaTpoint offers Corporate Training, Summer Training, Online Training, and Winter Training.

Word Tokenizer generates the following result:

"JavaTpoint", "offers", "Corporate", "Training", "Summer", "Training", "Online", "Training", "and", "Winter", "Training", "."

## Step3: Stemming

Stemming is used to normalize words into its base form or root form. For example, celebrates, celebrated and celebrating, all these words are originated with a single root word "celebrate." The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

**For Example**, intelligence, intelligent, and intelligently, all these words are originated with a single root word "intelligen." In English, the word "intelligen" do not have any meaning.

## Step 4: Lemmatization



Lemmatization is quite similar to the Stemming. It is used to group different inflected forms of the word, called Lemma. The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

**For example:** In lemmatization, the words intelligence, intelligent, and intelligently has a root word intelligent, which has a meaning.

### **Step 5: Identifying Stop Words**

In English, there are a lot of words that appear very frequently like "is", "and", "the", and "a". NLP pipelines will flag these words as stop words. **Stop words** might be filtered out before doing any statistical analysis.

**Example:** He **is** a good boy.

*Note: When you are building a rock band search engine, then you do not ignore the word "The."*

### **Step 6: Dependency Parsing**

Dependency Parsing is used to find that how all the words in the sentence are related to each other.

### **Step 7: POS tags**

POS stands for parts of speech, which includes Noun, verb, adverb, and Adjective. It indicates that how a word functions with its meaning as well as grammatically within the sentences. A word has one or more parts of speech based on the context in which it is used.

**Example: "Google"** something on the Internet.

In the above example, Google is used as a verb, although it is a proper noun.

### **Step 8: Named Entity Recognition (NER)**

Named Entity Recognition (NER) is the process of detecting the named entity such as person name, movie name, organization name, or location.

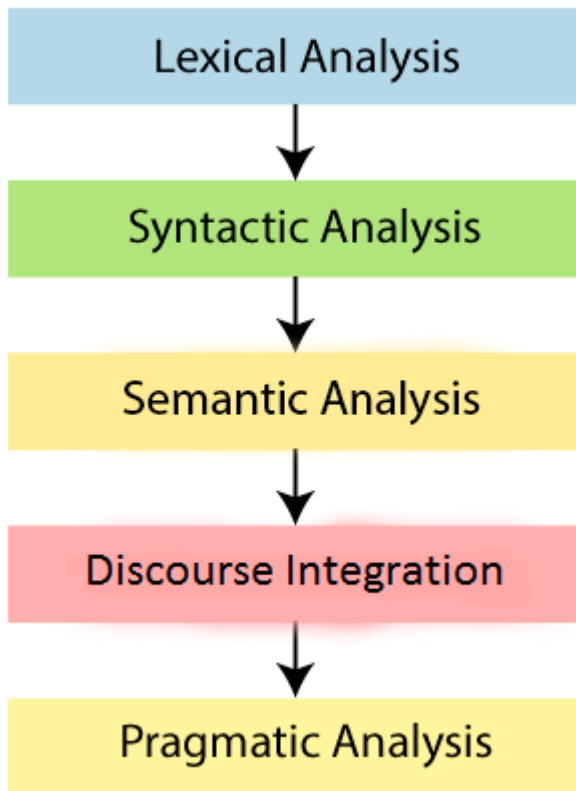
**Example: Steve Jobs** introduced iPhone at the Macworld Conference in San Francisco, California.

### **Step 9: Chunking**

Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

# Phases of NLP

There are the following five phases of NLP:



## 1. Lexical Analysis and Morphological

The first phase of NLP is the Lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences, and words.

## 2. Syntactic Analysis (Parsing)

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

**Example:** Agra goes to the Poonam

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

## 3. Semantic Analysis

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences.

## 4. Discourse Integration

Discourse Integration depends upon the sentences that precedes it and also invokes the meaning of the sentences that follow it.

## 5. Pragmatic Analysis

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

**For Example:** "Open the door" is interpreted as a request instead of an order.

## Why NLP is difficult?

NLP is difficult because Ambiguity and Uncertainty exist in the language.

### Ambiguity

There are the following three ambiguity -

- **Lexical Ambiguity**

Lexical Ambiguity exists in the presence of two or more possible meanings of the sentence within a single word.

#### Example:

Manya is looking for a **match**.

In the above example, the word match refers to that either Manya is looking for a partner or Manya is looking for a match. (Cricket or other match)

- **Syntactic Ambiguity**

Syntactic Ambiguity exists in the presence of two or more possible meanings within the sentence.

#### Example:

I saw the girl with the binocular.

In the above example, did I have the binoculars? Or did the girl have the binoculars?

- **Referential Ambiguity**

Referential Ambiguity exists when you are referring to something using the pronoun.

**Example:** Kiran went to Sunita. She said, "I am hungry."

In the above sentence, you do not know that who is hungry, either Kiran or Sunita.

## NLP APIs

Natural Language Processing APIs allow developers to integrate human-to-machine communications and complete several useful tasks such as speech recognition, chatbots, spelling correction, sentiment analysis, etc.

A list of NLP APIs is given below:

- **IBM Watson API**  
IBM Watson API combines different sophisticated machine learning techniques to enable developers to classify text into various custom categories. It supports multiple languages, such as English, French, Spanish, German, Chinese, etc. With the help of IBM Watson API, you can extract insights from texts, add automation in workflows, enhance search, and understand the sentiment. The main advantage of this API is that it is very easy to use.  
**Pricing:** Firstly, it offers a free 30 days trial IBM cloud account. You can also opt for its paid plans.
- **Chatbot API**  
Chatbot API allows you to create intelligent chatbots for any service. It supports Unicode characters, classifies text, multiple languages, etc. It is very easy to use. It helps you to create a chatbot for your web applications.  
**Pricing:** Chatbot API is free for 150 requests per month. You can also opt for its paid version, which starts from \$100 to \$5,000 per month.
- **Speech to text API**  
Speech to text API is used to convert speech to text  
**Pricing:** Speech to text API is free for converting 60 minutes per month. Its paid version starts from \$500 to \$1,500 per month.
- **Sentiment Analysis API**  
Sentiment Analysis API is also called as '**opinion mining**' which is used to identify the tone of a user (positive, negative, or neutral)  
**Pricing:** Sentiment Analysis API is free for less than 500 requests per month. Its paid version starts from \$19 to \$99 per month.
- **Translation API by SYSTRAN**  
The Translation API by SYSTRAN is used to translate the text from the source language to the target language. You can use its NLP APIs for language detection, text segmentation, named entity recognition, tokenization, and many other tasks.  
**Pricing:** This API is available for free. But for commercial users, you need to use its paid version.
- **Text Analysis API by AYLIEN**  
Text Analysis API by AYLIEN is used to derive meaning and insights from the textual content. It is available for both free as well as paid from \$119 per month. It is easy to use.  
**Pricing:** This API is available free for 1,000 hits per day. You can also use its paid version, which starts from \$199 to \$1,399 per month.
- **Cloud NLP API**  
The Cloud NLP API is used to improve the capabilities of the application using natural language processing technology. It allows you to carry various natural language processing functions like sentiment analysis and language detection. It is easy to use.  
**Pricing:** Cloud NLP API is available for free.

- **Google Cloud Natural Language API**

Google Cloud Natural Language API allows you to extract beneficial insights from unstructured text. This API allows you to perform entity recognition, sentiment analysis, content classification, and syntax analysis in more the 700 predefined categories. It also allows you to perform text analysis in multiple languages such as English, French, Chinese, and German.

**Pricing:** After performing entity analysis for 5,000 to 10,000,000 units, you need to pay \$1.00 per 1000 units per month.

## NLP Libraries

**Scikit-learn:** It provides a wide range of algorithms for building machine learning models in Python.

**Natural language Toolkit (NLTK):** NLTK is a complete toolkit for all NLP techniques.

**Pattern:** It is a web mining module for NLP and machine learning.

**TextBlob:** It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase extraction, or pos-tagging.

**Quepy:** Quepy is used to transform natural language questions into queries in a database query language.

**SpaCy:** SpaCy is an open-source NLP library which is used for Data Extraction, Data Analysis, Sentiment Analysis, and Text Summarization.

**Gensim:** Gensim works with large datasets and processes data streams.

## Difference between Natural language and Computer Language

Natural Language	Computer Language
Natural language has a very large vocabulary.	Computer language has a very limited vocabulary.
Natural language is easily understood by humans.	Computer language is easily understood by the machines.
Natural language is ambiguous in nature.	Computer language is unambiguous.

## 1. Text classification

Text clarification is the process of categorizing the text into a group of words. By using NLP, text classification can automatically analyze text and then assign a set of predefined tags or categories based on its context. NLP is used for sentiment analysis, topic detection, and language detection. There is mainly three text classification approach-

- Rule-based System,
- Machine System
- Hybrid System.

In the rule-based approach, texts are separated into an organized group using a set of handicraft linguistic rules. Those handicraft linguistic rules contain users to define a list of words that are characterized by groups. For example, words like Donald Trump and Boris Johnson would be categorized into politics. People like LeBron James and Ronaldo would be categorized into sports.

Machine-based classifier learns to make a classification based on past observation from the data sets. User data is pre-labeled as train and test data. It collects the classification strategy from the previous inputs and learns continuously. Machine-based classifier usage a bag of a word for feature extension.

### The Bag of Words Representation

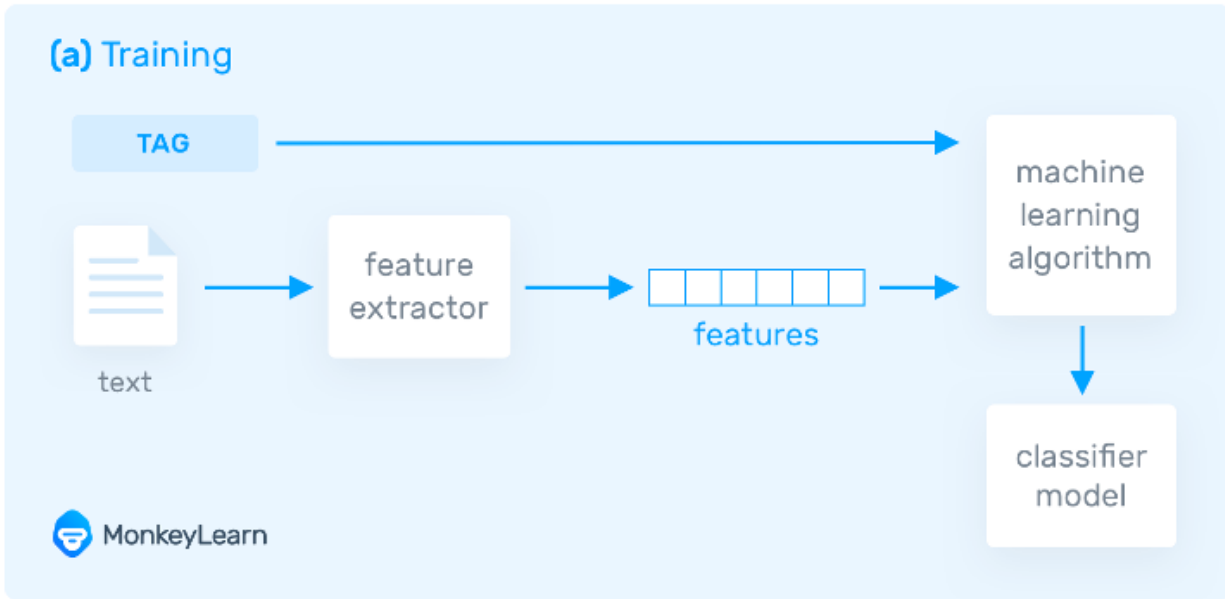
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

15

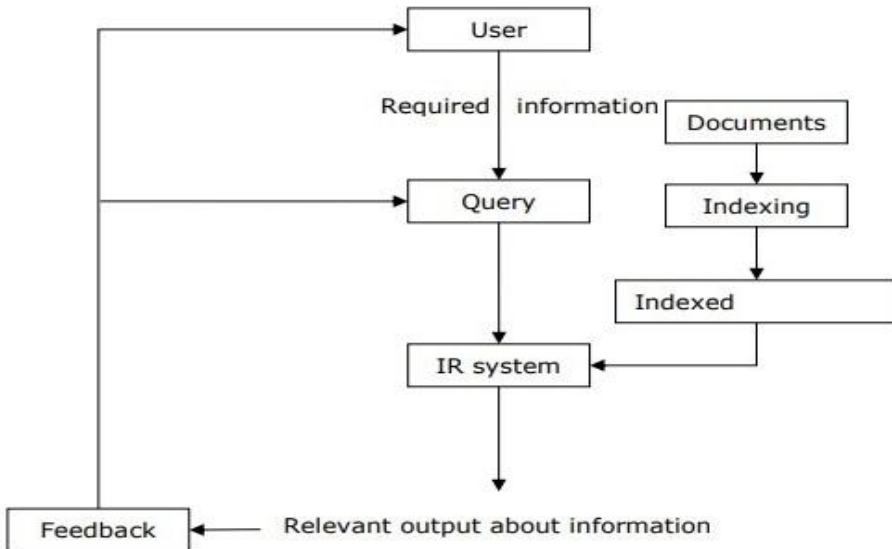
In a bag of words, a vector represents the frequency of words in a predefined dictionary of a word list. We can perform NLP using the following machine learning algorithms: Naïve Bayes, SVM, and Deep Learning.



The third approach to text classification is the Hybrid Approach. Hybrid approach usage combines a rule-based and machine Based approach. Hybrid based approach usage of the rule-based system to create a tag and use machine learning to train the system and create a rule. Then the machine-based rule list is compared with the rule-based rule list. If something does not match on the tags, humans improve the list manually. It is the best method to implement text classification

## NLP - Information Retrieval

Information retrieval (IR) may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information.



The system assists users in finding the information they require but it does not explicitly return the answers of the questions. It informs the existence and location of documents that might consist of the required information. The documents that satisfy user's requirement are called relevant documents. A perfect IR system will retrieve only relevant documents.

With the help of the above diagram, we can understand the process of information retrieval (IR)

It is clear from the above diagram that a user who needs information will have to formulate a request in the form of query in natural language. Then the IR system will respond by retrieving the relevant output, in the form of documents, about the required information.

## Classical Problem in Information Retrieval (IR) System

The main goal of IR research is to develop a model for retrieving information from the repositories of documents. Here, we are going to discuss a classical problem, named ad-hoc retrieval problem, related to the IR system.

In ad-hoc retrieval, the user must enter a query in natural language that describes the required information. Then the IR system will return the required documents related to the desired information. For example, suppose we are searching something on the Internet and it gives some exact pages that are relevant as per our requirement but there can be some non-relevant pages too. This is due to the ad-hoc retrieval problem.

## Aspects of Ad-hoc Retrieval

Followings are some aspects of ad-hoc retrieval that are addressed in IR research –

- How users with the help of relevance feedback can improve original formulation of a query?
- How to implement database merging, i.e., how results from different text databases can be merged into one result set?
- How to handle partly corrupted data? Which models are appropriate for the same?

## Information Retrieval (IR) Model

Mathematically, models are used in many scientific areas having objective to understand some phenomenon in the real world. A model of information retrieval predicts and explains what a user will find in relevance to the given query. IR model is basically a pattern that defines the above-mentioned aspects of retrieval procedure and consists of the following –

- A model for documents.
- A model for queries.
- A matching function that compares queries to documents.

Mathematically, a retrieval model consists of –

D – Representation for documents.

R – Representation for queries.

F – The modeling framework for D, Q along with relationship between them.



$R(q, d_i)$  – A similarity function which orders the documents with respect to the query. It is also called ranking.

## Types of Information Retrieval (IR) Model

An information model (IR) model can be classified into the following three models –

### Classical IR Model

It is the simplest and easy to implement IR model. This model is based on mathematical knowledge that was easily recognized and understood as well. Boolean, Vector and Probabilistic are the three classical IR models.

### Non-Classical IR Model

It is completely opposite to classical IR model. Such kind of IR models are based on principles other than similarity, probability, Boolean operations. Information logic model, situation theory model and interaction models are the examples of non-classical IR model.

### Alternative IR Model

It is the enhancement of classical IR model making use of some specific techniques from some other fields. Cluster model, fuzzy model and latent semantic indexing (LSI) models are the example of alternative IR model.

## Design features of Information retrieval (IR) systems

Let us now learn about the design features of IR systems –

### Inverted Index

The primary data structure of most of the IR systems is in the form of inverted index. We can define an inverted index as a data structure that list, for every word, all documents that contain it and frequency of the occurrences in document. It makes it easy to search for ‘hits’ of a query word.

### Stop Word Elimination

Stop words are those high frequency words that are deemed unlikely to be useful for searching. They have less semantic weights. All such kind of words are in a list called stop list. For example, articles “a”, “an”, “the” and prepositions like “in”, “of”, “for”, “at” etc. are the examples of stop words. The size of the inverted index can be significantly reduced by stop list. As per Zipf’s law, a stop list covering a few dozen words reduces the size of inverted index by almost half. On the other hand, sometimes the elimination of stop word may cause elimination of the term that is useful for searching. For example, if we eliminate the alphabet “A” from “Vitamin A” then it would have no significance.

## Stemming

Stemming, the simplified form of morphological analysis, is the heuristic process of extracting the base form of words by chopping off the ends of words. For example, the words laughing, laughs, laughed would be stemmed to the root word laugh.

In our subsequent sections, we will discuss about some important and useful IR models.

## IR MODELS

### 1.The Boolean Model

It is the oldest information retrieval (IR) model. The model is based on set theory and the Boolean algebra, where documents are sets of terms and queries are Boolean expressions on terms. The Boolean model can be defined as –

- **D** – A set of words, i.e., the indexing terms present in a document. Here, each term is either present (1) or absent (0).
- **Q** – A Boolean expression, where terms are the index terms and operators are logical products – AND, logical sum – OR and logical difference – NOT
- **F** – Boolean algebra over sets of terms as well as over sets of documents

If we talk about the relevance feedback, then in Boolean IR model the Relevance prediction can be defined as follows –

- **R** – A document is predicted as relevant to the query expression if and only if it satisfies the query expression as –

$$((text \vee information) \wedge retrieval \wedge \sim theory)$$

We can explain this model by a query term as an unambiguous definition of a set of documents.

For example, the query term “economic” defines the set of documents that are indexed with the term “economic”.

Now, what would be the result after combining terms with Boolean AND Operator? It will define a document set that is smaller than or equal to the document sets of any of the single terms. For example, the query with terms “social” and “economic” will produce the documents set of documents that are indexed with both the terms. In other words, document set with the intersection of both the sets.

Now, what would be the result after combining terms with Boolean OR operator? It will define a document set that is bigger than or equal to the document sets of any of the single terms. For example, the query with terms “social” or “economic” will produce the documents set of documents that are indexed with either the term “social” or “economic”. In other words, document set with the union of both the sets.

### Advantages of the Boolean Mode

The advantages of the Boolean model are as follows –

- The simplest model, which is based on sets.

- Easy to understand and implement.
- It only retrieves exact matches
- It gives the user, a sense of control over the system.

## Disadvantages of the Boolean Model

The disadvantages of the Boolean model are as follows –

- The model's similarity function is Boolean. Hence, there would be no partial matches. This can be annoying for the users.
- In this model, the Boolean operator usage has much more influence than a critical word.
- The query language is expressive, but it is complicated too.
- No ranking for retrieved documents.

## 2. Vector Space Model

Due to the above disadvantages of the Boolean model, Gerard Salton and his colleagues suggested a model, which is based on Luhn's similarity criterion. The similarity criterion formulated by Luhn states, "the more two representations agreed in given elements and their distribution, the higher would be the probability of their representing similar information."

Consider the following important points to understand more about the Vector Space Model –

- The index representations (documents) and the queries are considered as vectors embedded in a high dimensional Euclidean space.
- The similarity measure of a document vector to a query vector is usually the cosine of the angle between them.

## Cosine Similarity Measure Formula

Cosine is a normalized dot product, which can be calculated with the help of the following formula –

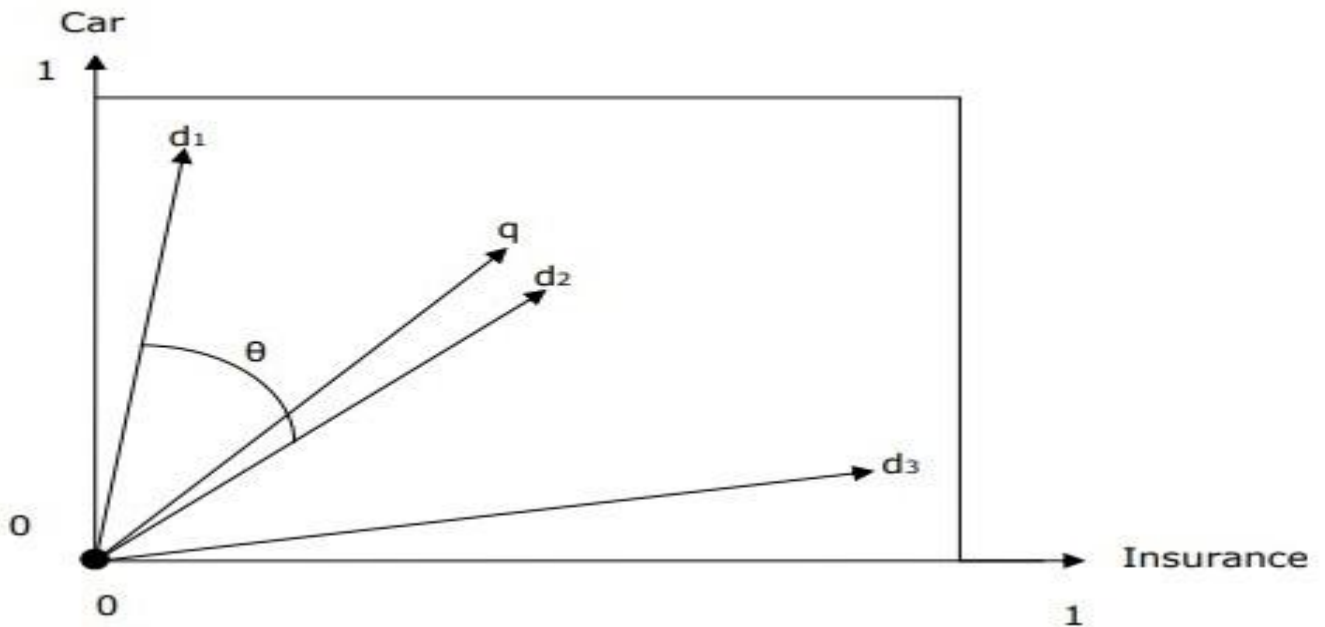
$$\text{Score}(d \rightarrow q) = \frac{\sum_{k=1}^m d_k \cdot q_k}{\sqrt{\sum_{k=1}^m (d_k)^2} \cdot \sqrt{\sum_{k=1}^m (q_k)^2}}$$

$$\text{Score}(d \rightarrow q) = 1 \text{ when } d = q$$

$$\text{Score}(d \rightarrow q) = 0 \text{ when } d \text{ and } q \text{ share no items}$$

## Vector Space Representation with Query and Document

The query and documents are represented by a two-dimensional vector space. The terms are car and insurance. There is one query and three documents in the vector space.



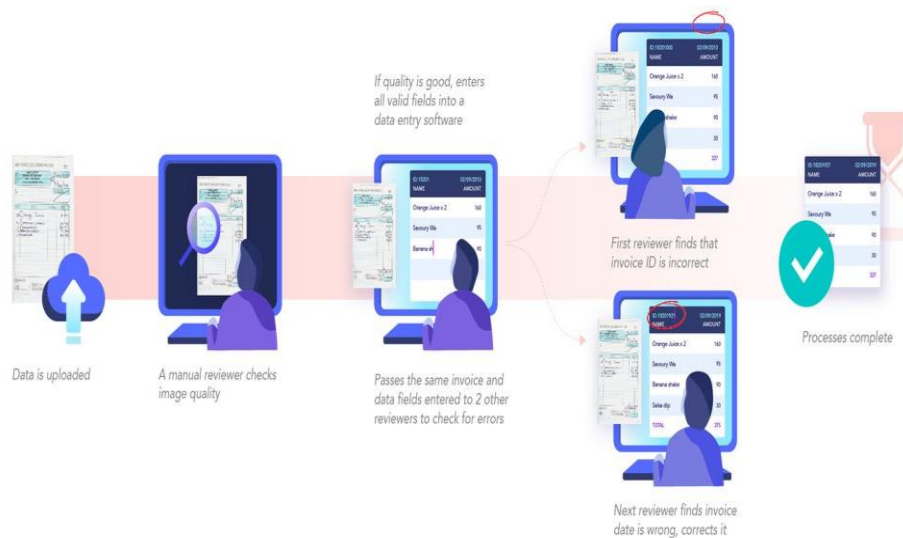
The top ranked document in response to the terms car and insurance will be the document d2 because the angle between q and d2 is the smallest. The reason behind this is that both the concepts car and insurance are salient in d2 and hence have the high weights. On the other side, d1 and d3 also mention both the terms but in each case, one of them is not a centrally important term in the document.

## Information Extraction

Information Extraction is the process of parsing through unstructured data and extracting essential information into more editable and structured data formats.

For example, consider we're going through a company's financial information from a few documents. Usually, we search for some required information when the data is digital or manually check the same. But with information extraction NLP algorithms, we can automate the data extraction of all required information such as tables, company growth metrics, and other financial details from various kinds of documents (PDFs, Docs, Images etc.).

Below is a screenshot explaining how we can extract information from an Invoice.



## Information Extraction Workflow

Information Extraction from text data can be achieved by leveraging Deep Learning and NLP techniques like Named Entity Recognition. However, if we build one from scratch, we should decide the algorithm considering the type of data we're working on, such as invoices, medical reports, etc. This is to make sure the model is specific to a particular use case. We'll be learning more about this in the following sections.

## How Does Information Extraction Work?

To understand the mechanics of Information Extraction NLP algorithms, we should understand the kind of data we are working on. This will help us to sort out the information we want to extract from the unstructured data. For example, for invoice related information, the algorithm should understand the invoice items, company name, billing address etc. While working on medical reports, it should identify and extract patient names, drug information, and other general reports.

After curating the data, we'll then start applying the information extraction NLP techniques, to process and build models around the data. Below are some of the most common techniques that are frequently used.

## Tokenization

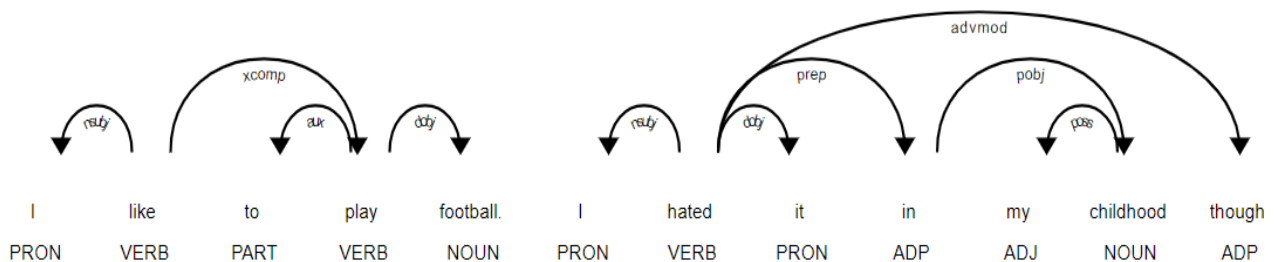
Computers usually won't understand the language we speak or communicate with. Hence, we break the language, basically the words and sentences, into tokens and then load it into a program. The process of breaking down language into tokens is called tokenization.

For example, consider a simple sentence: "NLP information extraction is fun". This could be tokenized into:

1. One-word (sometimes called unigram token): NLP, information, extraction, is, fun
2. Two-word phrase (bigram tokens): NLP information, information extraction, extraction is, is fun, fun NLP
3. Three-word sentence (trigram tokens): NLP information extraction, information extraction is, extraction is fun

## Parts of Speech Tagging

Tagging parts of speech is very crucial for information extraction from text. It'll help us understand the context of the text data. We usually refer to text from documents as "unstructured data" – data with no defined structure or pattern. Hence, with POS tagging we can use techniques that will provide the context of words or tokens used to categorise them in specific ways.

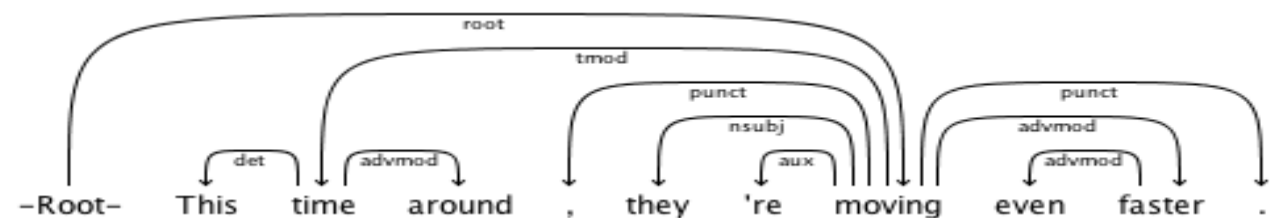


### Parts of Speech Tagging

In parts of speech tagging, all the tokens in the text data get categorised into different word categories, such as nouns, verbs, adjectives, prepositions, determiners, etc. This additional information connected to words enables further processing and analysis, such as sentiment analytics, lemmatization, or any reports where we can look closer at a specific class of words.

## Dependency Graphs

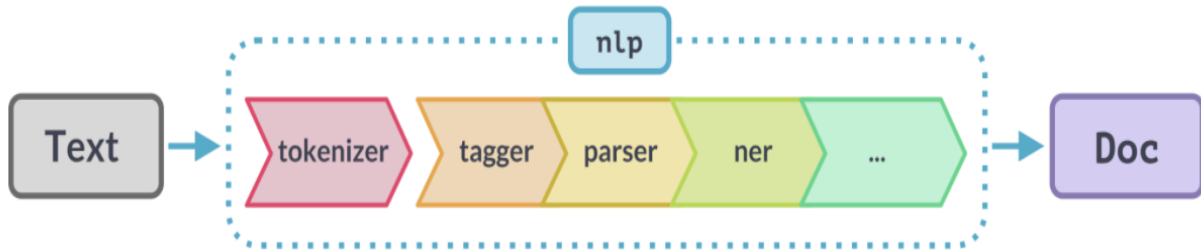
Dependency graphs help us find relationships between neighbouring words using directed graphs. This relation will provide details about the dependency type (e.g. Subject, Object etc.). Following is a figure representing a dependency graph of a short sentence. The arrow directed from the word *faster* indicates that *faster* modifies *moving*, and the label `advmod` assigned to the arrow describes the exact nature of the dependency.



Dependency Graph Example

## NER with Spacy

Spacy is an open-source NLP library for advanced Natural Language Processing in Python and Cython. It's well maintained and has over 20K stars on Github. To extract information with spacy NER models are widely leveraged.



NLP Pipelines for building models with Spacy

**Unit-IV: Natural Language for Communication:** Phrase structure grammars, Syntactic Analysis, Augmented Grammars and semantic Interpretation, Machine Translation, Speech Recognition

**Perception:** Image Formation, Early Image Processing Operations, Object Recognition by appearance, Reconstructing the 3D World, Object Recognition from Structural information, Using Vision.

---

### **Phrase structure grammars:**

Despite the exceptions, the notion of a **lexical category** (also known as a **part of speech**) such as *noun* or *adjective* is a useful generalization—useful in its own right, but more so when we string together lexical categories to form **syntactic categories** such as *noun phrase* or *verb phrase*, and combine these syntactic categories into trees representing the **phrase structure** of sentences: nested phrases, each marked with a category.

Grammatical formalisms can be classified by their **generative capacity**: the set of languages they can represent. Chomsky (1957) describes four classes of grammatical formalisms that differ only in the form of the rewrite rules. The classes can be arranged in a hierarchy, where each class can be used to describe all the languages that can be described by a less powerful class, as well as some additional languages. Here we list the hierarchy, most powerful class first:

**Recursively enumerable** grammars use unrestricted rules: both sides of the rewrite rules can have any number of terminal and nonterminal symbols, as in the rule  $A B C \rightarrow D E$ . These grammars are equivalent to Turing machines in their expressive power.

**Context-sensitive grammars** are restricted only in that the right-hand side must contain at least as many symbols as the left-hand side. The name “context-sensitive” comes from the fact that a rule such as  $A X B \rightarrow A Y B$  says that an  $X$  can be rewritten as a  $Y$  in the context of a preceding  $A$  and a following  $B$ . Context-sensitive grammars can represent languages such as  $a^n b^n c^n$  (a sequence of  $n$  copies of  $a$  followed by the same number of  $b$ s and then  $c$ s).

In **context-free grammars** (or **CFGs**), the left-hand side consists of a single nonterminal symbol. Thus, each rule licenses rewriting the nonterminal as the right-hand side in *any* context. CFGs are popular for natural-language and programming-language grammars, although it is now widely accepted that at least some natural languages have constructions that are not context-free (Pullum, 1991). Context-free grammars can represent  $a^n b^n$ , but not  $a^n b^n c^n$ .

**Regular** grammars are the most restricted class. Every rule has a single nonterminal on the left-hand side and a terminal symbol optionally followed by a nonterminal on the right-hand side. Regular grammars are equivalent in power to finite state machines. They are poorly suited for programming languages, because they cannot represent constructs such as balanced opening and closing parentheses (a variation of the  $a^n b^n$  language). The closest they can come is representing  $a^* b^*$ , a sequence of any number of  $a$ s followed by any number of  $b$ s.

### **Syntactic Analysis**

Syntactic analysis – or parsing – analyzes text using basic grammar rules to identify sentence structure, how words are organized, and how words relate to each other.



Some of its main sub-tasks include:

- **Tokenization** consists of breaking up a text into smaller parts called tokens (which can be sentences or words) to make text easier to handle.
- **Part of speech tagging (PoS tagging)** labels tokens as verb, adverb, adjective, noun, etc. This helps infer the meaning of a word (for example, the word “book” means different things if used as a verb or a noun).
- **Lemmatization & stemming** consist of reducing inflected words to their base form to make them easier to analyze.
- **Stop-word removal** removes frequently occurring words that don’t add any semantic value, such as I, they, have, like, yours, etc.

## **Machine Translation:**

What is machine translation?

Machine translation is the process of using artificial intelligence (AI) to automatically translate content from one language (the source) to another (the target) without any human input.

Translation was one of the first applications of computing power, starting in the 1950s. Unfortunately, the complexity of the task was far higher than early computer scientists’ estimates, requiring enormous data processing power and storage far beyond the capabilities of early machines.

It was only in the early 2000s that the software, data, and required hardware became capable of doing basic machine translation. Early developers used statistical databases of languages to “teach” computers to translate text. Training these machines involved a lot of manual labor, and each added language required starting over with the development for that language.

In 2016, Google had an experimental team testing the use of neural learning models and artificial intelligence (AI) to train translation engines. When the small team’s methodology was tested against Google’s main statistical machine translation engine, it proved far faster and more effective across many languages. In addition, it ‘learned’ as it was used generating constant improvement in quality. Neural machine translation proved so effective that Google changed course and adopted it as their primary development model. Other major providers including Microsoft and Amazon soon followed suit, and modern machine translation became a viable addition to translation technology. Many translation management systems (TMSs) now incorporate MT into their solutions for their user’s workflows

### **What types of machine translation are there?**

The three most common types of machine translation include:

#### **Rule-based machine translation (RBMT)**

The earliest form of MT, rule-based MT, has several serious disadvantages including requiring significant amounts of human post-editing, the requirement to manually add languages, and low quality in general. It has some uses in very basic situations where a quick understanding of meaning is required.

## Statistical machine translation (SMT)

Statistical MT builds a statistical model of the relationships between words, phrases, and sentences in a text. It applies the model to a second language to convert those elements to the new language. Thereby, it improves on rule-based MT but shares many of the same problems.

## Neural machine translation (NMT)

As mentioned above, the neural MT model uses artificial intelligence to learn languages and constantly improve that knowledge, much like the neural networks in the human brain. It is more accurate, easier to add languages, and much faster once trained. Neural MT is rapidly becoming the standard in MT engine development.

## Which machine translation type should I use?

In general, the decision on which machine translation type you should use depends on:

- **The budget available:** Neural MT is more expensive to train than statistical MT, but the quality improvement is well worth any cost difference. Many systems are deprecating their older statistical models in favor of neural learning.
- **The industry involved:** Some industries demand complex and technical language that may require more sophisticated processing. Neural MT provides this.
- **The language pairs you need:** Statistical MT is often sufficient for certain language pairs such as Latin-based languages with similar grammatical rules and syntax.
- **The amount of data you have:** Neural MT requires the processing of large quantities of text for it to learn and for you to reap the benefits.
- **Internal vs. customer-facing content:** Customer-facing content, like marketing or sales materials that reflect brand quality, requires the most sophisticated combination of machine translation and experienced human translators doing post-editing. Basic employee communication or internal documentation may be able to be achieved by using basic machine translation when time and cost are factors

## Which machine translation engine should I use?

The major developers of machine translation technology—Google, Microsoft, and Amazon—all currently use a type of neural MT as their preferred methodology since it allows for both more nuanced translation and constantly adding language pairs. This growth capability is made possible by the engines' ability to learn and improve as they are used more.

## Generic machine translation engines

Machine translation works on training data. Depending on your needs, the data can be generic or custom. Generic data is simply the total of all the data learned from all the translations performed over time by the machine translation engine (MTE). It enables a generalized translation tool for all kinds of applications, including text, voice, and full documents, including formatting.

Specialized training data is data fed to an MT to build a specialization in a subject matter area like engineering, programming, design, or any discipline with its own glossaries.

#### ***Google Translate***

Generally considered one of the leading machine translation engines, based on usage, number of languages, and integration with search. Google Translate was the first MTE based on neural language processing that learns from repeated usage.

#### ***Amazon Translate***

Amazon Translate is also neural-based and is closely integrated with Amazon Web Services (AWS). Some evidence suggests Amazon Translate is more accurate with certain languages, notably Chinese, however it is important when making comparisons to understand that all of these engines are constantly learning and improving.

#### ***Microsoft Translator***

Another cloud-based neural engine, Microsoft Translator is closely integrated with MS Office and other Microsoft products, providing instant access to translation abilities within a document or other software.

#### ***DeepL***

DeepL is the product of a smaller company based in Germany and is exclusively devoted to the development of a machine translation engine claiming more nuanced and natural output based on their proprietary neural AI.

#### **Custom machine translation engines**

These major, general-purpose MTEs are the big players. However, there are many specialized engines developed for specific translation management systems, scientific disciplines, and other specialized uses. They are created by taking a basic platform and training it in a discipline based on providing data specific to that discipline

#### **What are the advantages of machine translation?**

Before the introduction of neural learning, MT was still very much a beta product generating translations whose quality varied wildly, veering sometimes into being humorously poor or unreadable. Modern machine translation engines have largely changed all of that and now serve as an indispensable tool in the translation process. It can be used “as is” for less critical applications or combined with human post-editing to speed up traditional translation workflows.

#### **Speed and volume**

MT is fast, translating millions of words almost instantaneously, while continually improving as more content is translated. For very high-volume projects, MT can not only handle volume at speed, but it can also work with content management systems to organize and tag that content. This makes it possible to retain organization and context as the content is translated into multiple languages.

#### **Large language selection**

With the major providers offering 50-100 languages or more, translations can be done simultaneously across multiple languages for global product rollouts and updates to documentation.

### Reduced costs and faster turnaround

The combination of high-speed throughput, as well as the ability to select from existing language pairs covering dozens of combinations, means the use of MT can cut costs and time to deliver translations, even when human translators are still post-editing the work. Basically, MT does the initial heavy lifting by providing basic but useful translations. The human translator then refines these basic versions to more closely reflect the original intent of the content and ensure proper localization per region.

### Automated integration into translation workflows

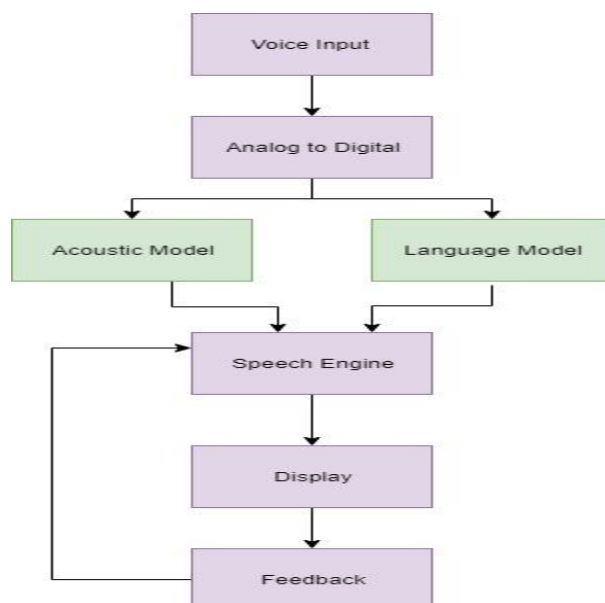
Many translation management systems integrate one or more kinds of MT into their workflow. They include settings to automatically run a translation and send that off as part of the human translator content package. Given the low cost and lack of any latency in the MT step, there is really no reason to not include the machine-translated content in the automation of workflows, especially for internal documentation and communication (rather than customer-facing and brand-oriented)

## Speech Recognition

### What is Speech Recognition?

Speech recognition is also known as automatic speech recognition (ASR), computer speech recognition, or speech to text (STT), which means understanding voice by the computer and performing any required task. It develops methods and technologies that implement the recognition and translation of spoken language into text by computers.

### Working Mechanism of Speech Recognition



### Where Is It Used?

- System control/navigation, e.g., GPS-connected digital maps

- Commercial/industrial applications in the car steering system
- Also, voice dialing hands-free use of mobile in the car

## Speech Recognition Techniques

The main objective of speech recognition is for a machine to be able to “listen,” “understand,” and “act upon” the information provided through the voice input. Automatic speaker recognition aims to analyze, extract, characterize, and recognize information about the speaker’s identity.

**Hence, the speaker recognition system works in three stages, as follows:**

1. Analysis
2. Feature extraction
3. Modeling

### 2.1 Speech Analysis Technique

Speaker identity can be shown by a different type of information that is present in speech data. This incorporates speaker-specific information due to the vocal tract, excitation source, and behavior feature. This stage deals with a suitable frame size for segmenting speech signals for further analysis and extracting.

### 2.2. Feature Extraction Technique

The speech feature extraction technique is the process of placing words in groups or classes is about decreasing the dimensionality of the input vector while maintaining the discriminating power of the signal. From the basic formation of speaker identification and verification system, we know that the number of training and test vector needed for the classification problem grows with the dimension of the given input; therefore, we need feature extraction of the speech signal.

### 2.3. Modeling

The modeling technique aims to create speaker models using a speaker-specific feature vector. Further, Speaker recognition and Speaker identification are the parts of Modeling. The speaker identification technique identifies by itself, who is speaking based on individual information integrated into a speech signal.

## *What Is an Acoustic Model?*

In brief, an acoustic model is a file that consists of statistical representations of each of the distinguishable sounds that makes up a word. Statistical representations assigned to the label called a phoneme. The English language has approximately 40 different sounds used in speech recognition. Therefore, we have 40 different phonemes.

## *What Is a Language Model?*

In brief, language models are used to limit the search in a decoder by limiting the number of possible worlds that are required to consider at any one point in the search. Finally, the result is faster execution and higher accuracy of the model.

## Types of Speech Recognition Software's

1. **Speaker Dependent**

Those individuals who will be using the system train the Speaker dependent systems. These systems are capable of achieving a high better command count than 95% accuracy for word recognition. The drawback of this approach is that the system only responds accurately only to the individual who trained the system. This is the most common approach implemented in software for personal computers.

2. **Speaker Independent**

Speaker independent is a system trained to respond to a word regardless of who is speaking. Therefore the system must respond to a large variety of speech patterns, inflections, and enunciation is of the target word. The command word count is generally lower than the speaker-dependent, whereas high accuracy can still be maintained within processing limits. Industrial requirements need speaker-independent voice systems more often, such as the AT&T system are used in the telephone systems.

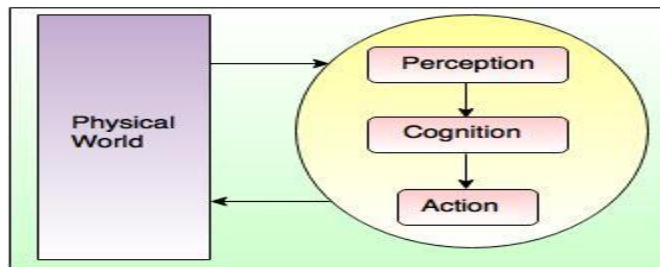
## Example of Speech Recognition

1. Speech Recognition is implemented in the front end or the back end medical document processes. Front end speech recognition is where the provider dictates the speech recognition engine. Spoken words displayed as recognized. The dictator is answerable for editing and signing off the Document. Back end recognition, also known as deferred speech recognition, is where the provider dictates into a digital dictation system. The voice is routed through a speech-recognition machine, and the draft document is recognized. It is routed along with the original voice file to the editor (where the draft file is edited and report finalized). Back end or deferred recognition is widely used in the industry currently.
2. Substantial efforts dedicated in the last decade to the test and evaluation of speech recognition in fighter aircraft. Speech recognizers operated successfully in fighter aircraft. Applications like setting radio frequencies, commanding an autopilot system, setting steer-point coordinates and weapons release parameters, and also controlling flight display.
3. In-car systems, simple voice commands used to initiate phone calls, select radio stations, or play music from a compatible smartphone, MP3 player, or music-loaded flash drive. In addition, voice recognition capabilities vary between car make and model.

## Perception:

### What is perception in AI?

- Perception is a process to interpret, acquire, select and then organize the sensory information that is captured from the real world.  
For example: Human beings have sensory receptors such as touch, taste, smell, sight and hearing. So, the information received from these receptors is transmitted to human brain to organize the received information.
- According to the received information, action is taken by interacting with the environment to manipulate and navigate the objects.
- Perception and action are very important concepts in the field of Robotics. The following figures show the complete autonomous robot.



**Fig: Autonomous Robot**

- There is one important difference between the artificial intelligence program and robot. The AI program performs in a computer stimulated environment, while the robot performs in the physical world.

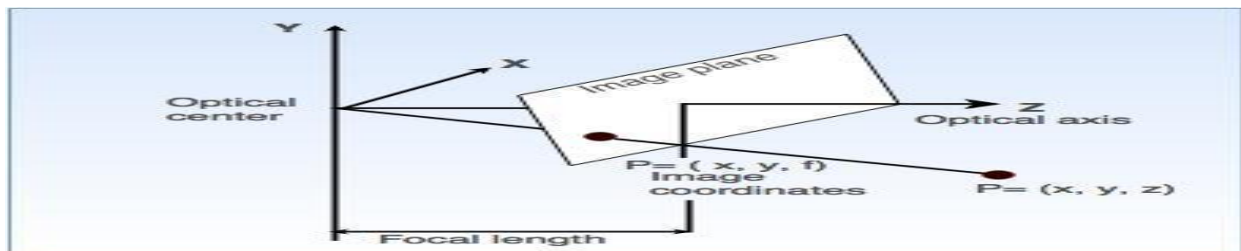
#### **For example:**

In chess, an AI program can be able to make a move by searching different nodes and has no facility to touch or sense the physical world.

However, the chess playing robot can make a move and grasp the pieces by interacting with the physical world.

#### **Image formation in digital camera**

Image formation is a physical process that captures object in the scene through lens and creates a 2-D image.



**Fig: Geometry of Image Formation in the pinhole camera**

Let's understand the geometry of a pinhole camera shown in the following diagram.

In the above figure, an optical axis is perpendicular to the image plane and image plane is generally placed in front of the optical center.

So, let P be the point in the scene with coordinates (X,Y,Z) and P' be its image plane with coordinates (x, y, z).

If the focal length from the optical center is f, then by using properties of similar triangles, equation is derived as,

$$-x/f = X/Z \text{ so } x = -fX/Z \text{ .....equation (i)}$$

$$-y/f = -Y/Z \text{ so } y = -fY/Z \text{ .....equation (ii)}$$

These equations define an image formation process called as **perspective projection**.

### ***What is the purpose of edge detection?***

- Edge detection operation is used in an image processing.
- The main goal of edge detection is to construct the ideal outline of an image.
- Discontinuity in brightness of image is affected due to:
  - i) Depth discontinuities
  - ii) Surface orientation discontinuities
  - iii) Reflectance discontinuities
  - iv) Illumination.

### ***3D-Information extraction using vision***

#### **Why extraction of 3-D information is necessary?**

The 3-D information extraction process plays an important role to perform the tasks like manipulation, navigation and recognition. It deals with the following aspects:

#### **1. Segmentation of the scene**

- The segmentation is used to arrange the array of image pixels into regions. This helps to match semantically meaningful entities in the scene.
- The goal of segmentation is to divide an image into regions which are homogeneous.
- The union of the neighboring regions should not be homogeneous.
- Thresholding is the simplest technique of segmentation. It is simply performed on the object, which has an homogeneous intensity and a background with a different intensity level and the pixels are partitioned depending on their intensity values.

#### **2. To determine the position and orientation of each object**

- Determination of the position and orientation of each object relative to the observer is important for manipulation and navigation tasks.

For example: Suppose a person goes to a store to buy something. While moving around he must know the locations and obstacles, so that he can make the plan and path to avoid them.

- The whole orientation of image should be specified in terms of a three dimensional rotation.



### **3. To determine the shape of each and every object**

- When the camera moves around an object, the distance and orientation of that object will change but it is important to preserve the shape of that object.  
For example: If an object is cube, that fact does not change, but it is difficult to represent the global shape to deal with wide variety of objects present in the real world.
- If the shape of an object is same for some manipulating tasks, it becomes easy to decide how to grasp that object from a particular place.
- The object recognition plays most significant role to identify and classify the objects as an example only when the geometric shapes are provided with color and texture.

However, a question arises that, how should we recover 3-D image from the pinhole camera? There are number of techniques available in the visual stimulus for 3D-image extraction such as motion, binocular stereopsis, texture, shading, and contour. Each of these techniques operates on the background assumptions about physical scene to provide interpretation

**Unit-IV: Natural Language for Communication:** Phrase structure grammars, Syntactic Analysis, Augmented Grammars and semantic Interpretation, Machine Translation, Speech Recognition

**Perception:** Image Formation, Early Image Processing Operations, Object Recognition by appearance, Reconstructing the 3D World, Object Recognition from Structural information, Using Vision.

### **Fundamentals of Image Formation**

Image formation is an analog to digital conversion of an image with the help of 2D Sampling and Quantization techniques that is done by the capturing devices like cameras. In general, we see a 2D view of the 3D world.

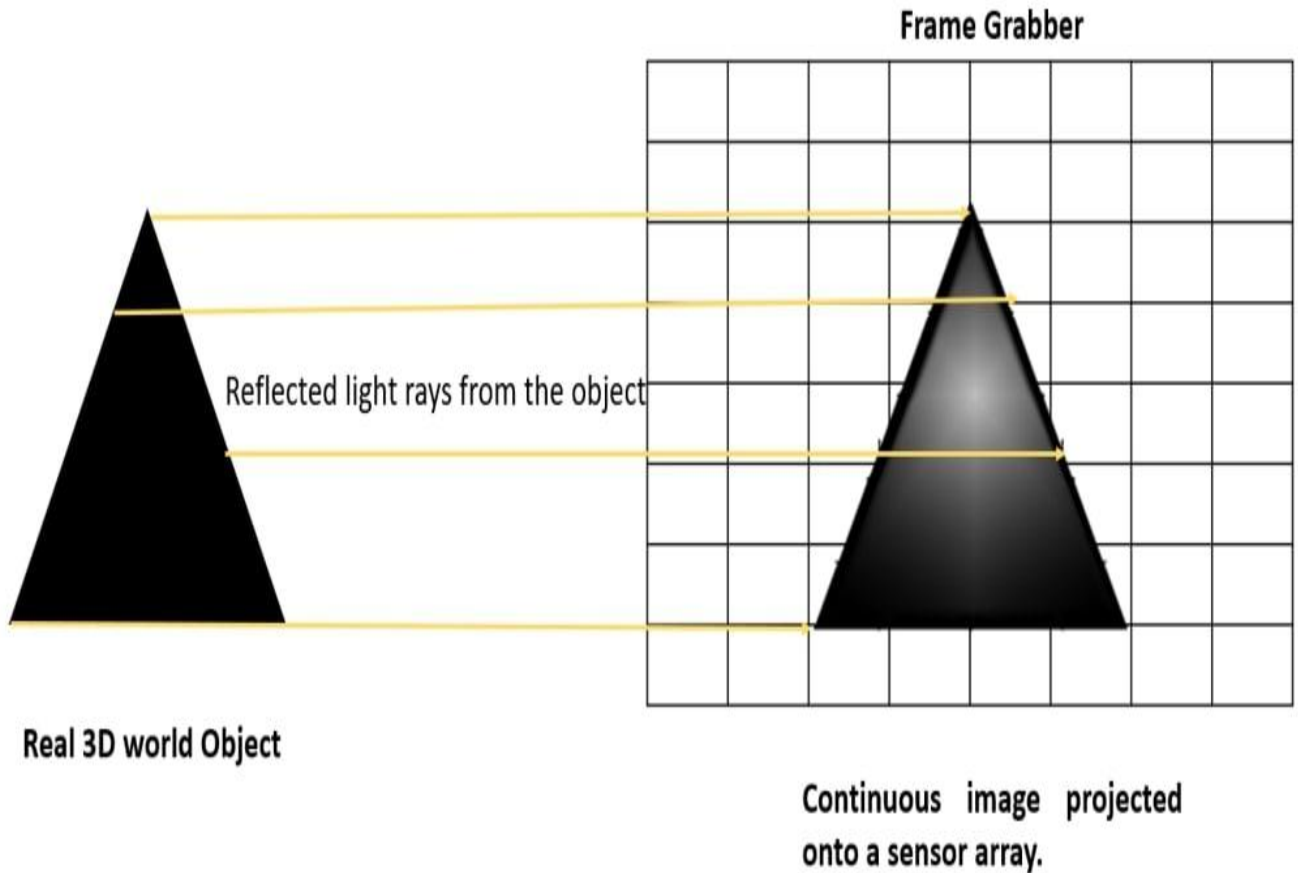
In the same way, the formation of the analog image took place. It is basically a conversion of the 3D world that is our analog image to a 2D world that is our Digital image.

Generally, a frame grabber or a digitizer is used for sampling and quantizing the analog signals.

### **Imaging:**

The mapping of a 3D world object into a 2D digital image plane is called *imaging*. In order to do so, each point on the 3D object must correspond to the image plane. We all know that light reflects from every object that we see thus enabling us to capture all those light-reflecting points in our image plane.

Various factors determine the quality of the image like spatial factors or the lens of the capturing device.



### **Color and Pixelation:**

In digital imaging, a frame grabber is placed at the image plane which is like a sensor. It aims to focus the light on it and the continuous image is pixelated via the reflected light by the 3D object. The light that is focused on the sensor generates an electronic signal.

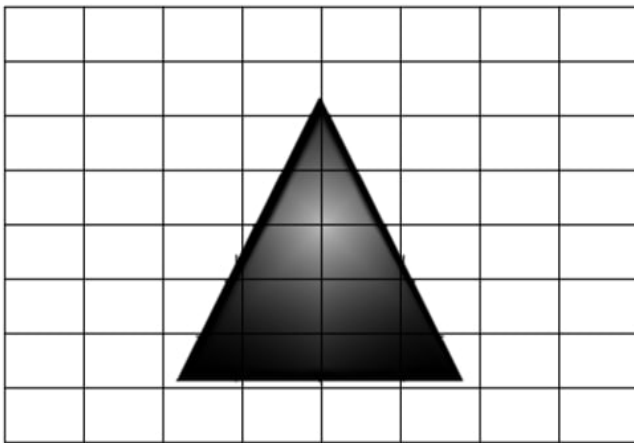
Each pixel that is formed may be colored or grey depending on the intensity of the sampling and quantization of the light that is reflected and the electronic signal that is generated via them.

All these pixels form a digital image. The density of these pixels determines the image quality. The more the density the more the clear and high-resolution image we will get.

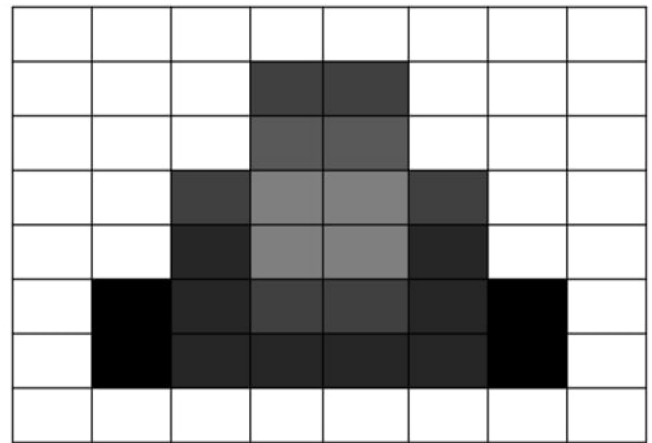
### **Forming a Digital Image:**

In order to form or create an image that is digital in nature, we need to have a continuous conversion of data into a digital form. Thus, we require two main steps to do so:

- **Sampling (2D):** Sampling is a spatial resolution of the digital image. And the rate of sampling determines the quality of the digitized image. The magnitude of the sampled image is determined as a value in image processing. It is related to the coordinate's values of the image.
- **Quantization:** Quantization is the number of grey levels in the digital image. The transition of the continuous values from the image function to its digital equivalent is called quantization. It is related to the intensity values of the image.
- The normal human being acquires a high level of quantization levels to get the fine shading details of the image. The more quantization levels will result in the more clear image.



**Continuous image projected onto a sensor array.**



**Result of image sampling and quantization.**

### **Early Image Processing Operations**

The purpose of early image processing was to improve the quality of the image. It was aimed for human beings to improve the visual effect of people. In image processing, the input is a low-quality image, and the output is an image with improved quality. Common image processing include **image enhancement, restoration, encoding, and compression.**

## Fundamental Image Processing Steps

### Image Acquisition

Image acquisition is the first step in image processing. This step is also known as preprocessing in image processing. It involves retrieving the image from a source, usually a hardware-based source.

### Image Enhancement

Image enhancement is the process of bringing out and highlighting certain features of interest in an image that has been obscured. This can involve changing the brightness, contrast, etc.

### Image Restoration

Image restoration is the process of improving the appearance of an image. However, unlike image enhancement, image restoration is done using certain mathematical or probabilistic models.

### Color Image Processing

Color image processing includes a number of color modeling techniques in a digital domain. This step has gained prominence due to the significant use of digital images over the internet.

### Wavelets and Multiresolution Processing

Wavelets are used to represent images in various degrees of resolution. The images are subdivided into wavelets or smaller regions for data compression and for pyramidal representation.

### Compression

Compression is a process used to reduce the storage required to save an image or the bandwidth required to transmit it. This is done particularly when the image is for use on the Internet.

## Morphological Processing

Morphological processing is a set of processing operations for morphing images based on their shapes.

## Segmentation

Segmentation is one of the most difficult steps of image processing. It involves partitioning an image into its constituent parts or objects.

## Representation and Description

After an image is segmented into regions in the segmentation process, each region is represented and described in a form suitable for further computer processing. Representation deals with the image's characteristics and regional properties. Description deals with extracting quantitative information that helps differentiate one class of objects from the other.

## Recognition

Recognition assigns a label to an object based on its description.

## Applications of Image Processing

### Medical Image Retrieval

Image processing has been extensively used in medical research and has enabled more efficient and accurate treatment plans. For example, it can be used for the early detection of breast cancer using a sophisticated nodule detection algorithm in breast scans. Since medical usage calls for highly trained image processors, these applications require significant implementation and evaluation before they can be accepted for use.

### Traffic Sensing Technologies

In the case of traffic sensors, we use a video image processing system or VIPS. This consists of a) an image capturing system b) a telecommunication system and c) an image processing system. When capturing video, a VIPS has several detection zones

which output an “on” signal whenever a vehicle enters the zone, and then output an “off” signal whenever the vehicle exits the detection zone. These detection zones can be set up for multiple lanes and can be used to sense the traffic in a particular station.



Left - normal traffic image | Right - a VIPS image with detection zones ([source](#))

Besides this, it can auto record the license plate of the vehicle, distinguish the type of vehicle, monitor the speed of the driver on the highway and lots more.

### Image Reconstruction

Image processing can be used to recover and fill in the missing or corrupt parts of an image. This involves using image processing systems that have been trained extensively with existing photo datasets to create newer versions of old and damaged photos.

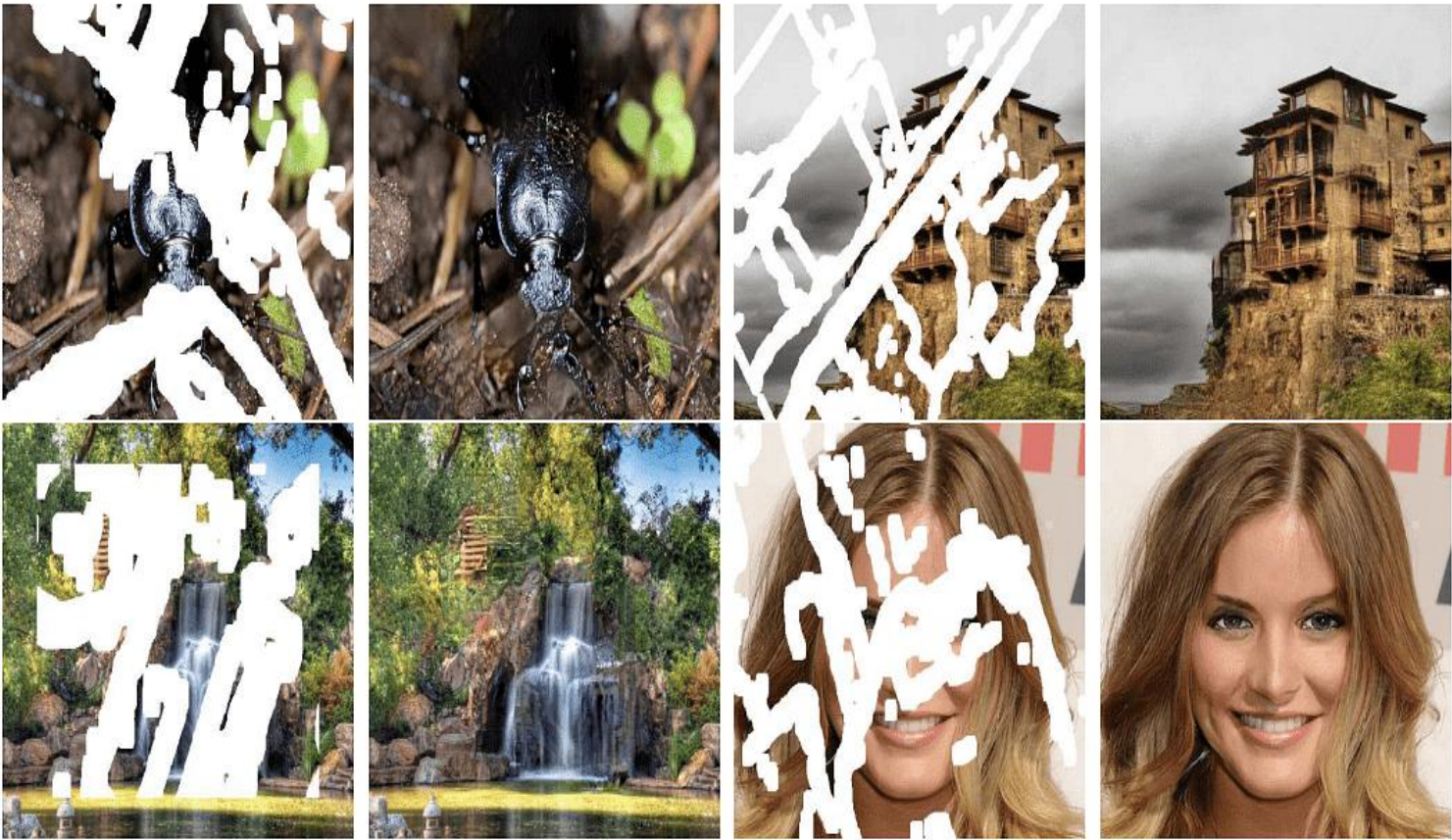


Fig: Reconstructing damaged images using image processing ([source](#))

### Face Detection

One of the most common applications of image processing that we use today is face detection. It follows [deep learning algorithms](#) where the machine is first trained with the specific features of human faces, such as the shape of the face, the distance between the eyes, etc. After teaching the machine these human face features, it will start to accept all objects in an image that resemble a human face. Face detection is a vital tool used in security, biometrics and even filters available on most social media apps these days.





**Unit-V: Robotics:** Introduction, Robot Hardware, Robotic Perception, Planning to move, Planning uncertain movements, Moving, Robotic software architectures, application domains

**Philosophical foundations:** Weak AI, Strong AI, Ethics and Risks of AI, Agent Components, Agent Architectures are we going in the right direction, what if AI does succeed.

---

### **Robotics: Introduction**

→Robotics is the term used in artificial intelligence that deals with a study of creating intelligent and efficient robots.

→Robots are multifunctional, re-programmable, automatic industrial machine designed for replacing human in hazardous work.

→Robotics is a domain of application that applies both the works of hardware and software similar to embedded systems

Robots can be work as:-

- An automatic machine sweeper
- In space
- A machine removing mines in a war field
- An automatic car for a child to play with
- In military, etc.

### **What is Robotics?**

Robotics is a branch of Artificial Intelligence (AI), it is mainly composed of electrical engineering, mechanical engineering and computer science engineering for construction, designing and application of robots.

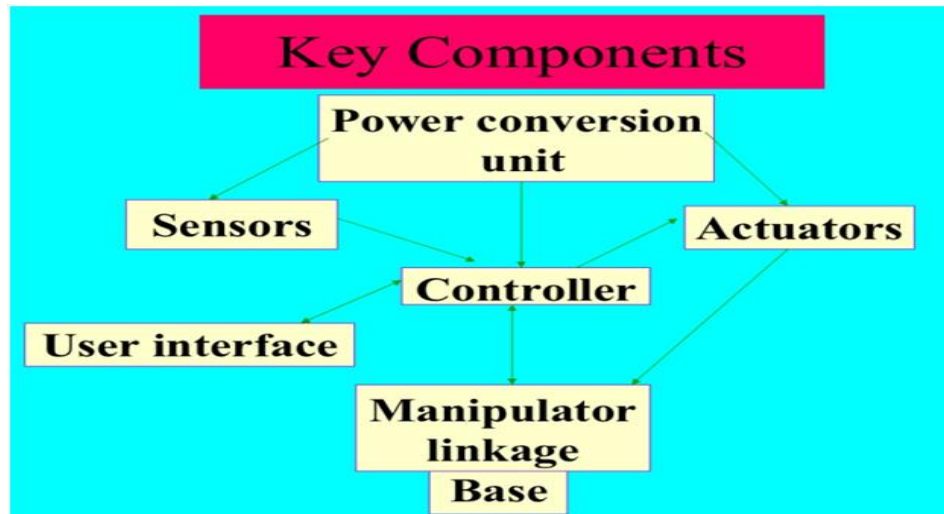
Robotics is science of building or designing an application of robots. The aim of robotics is to design an efficient robot.

### **Aspects of Robotics**

- The robots have **electrical components** for providing power and control the machinery.
- They have **mechanical construction**, shape, or form designed to accomplish a particular task.
- It contains some type of **computer program** that determines what, when and how a robot does something.

## Components of Robot/Robot Hardware:

Consider the key components of robotics are:-



- **Power Supply** - The working power to the robot is provided by batteries, hydraulic, solar power, or pneumatic power sources.
- **Actuators** - Actuators are the energy conversion device used inside a robot. The major function of actuators is to convert energy into movement.
- **Electric motors (DC/AC)**- Motors are electromechanical component used for converting electrical energy into its equivalent mechanical energy. In robots motors are used for providing rotational movement.
- **Sensors** - Sensors provide real time information on the task environment. Robots are equipped with tactile sensor it imitates the mechanical properties of touch receptors of human fingerprints and a vision sensor is used for computing the depth in the environment.
- **Controller** - Controller is a part of robot that coordinates all motion of the mechanical system. It also receives an input from immediate environment through various sensors. The heart of robot's controller is a microprocessor linked with the input/output and monitoring device. The command issued by the controller activates the motion control mechanism, consisting of various controller, actuators and amplifier.

## Robotic software architectures:

A methodology for structuring algorithms is called software **architecture**. Architecture includes languages and tools for writing programs, as well as an overall philosophy for how Programs can be brought together.

Modern-day software architectures for robotics must decide how to combine reactive control and model-based deliberative planning. In many ways, reactive and deliberate techniques have orthogonal strengths and weaknesses. Reactive control is sensor-driven and appropriate for making low-level decisions in real time. However, it rarely yields a plausible solution at the global level, because global control decisions depend on information that cannot be sensed at the time of decision making. For such problems, deliberate planning is a more appropriate choice.

Consequently, most robot architectures use reactive techniques at the lower levels of control and deliberative techniques at the higher levels. We encountered such a combination in our discussion of PD controllers, where we combined a (reactive) PD controller with a (deliberate) path planner. Architectures that combine reactive and deliberate techniques are called **hybrid architectures**.

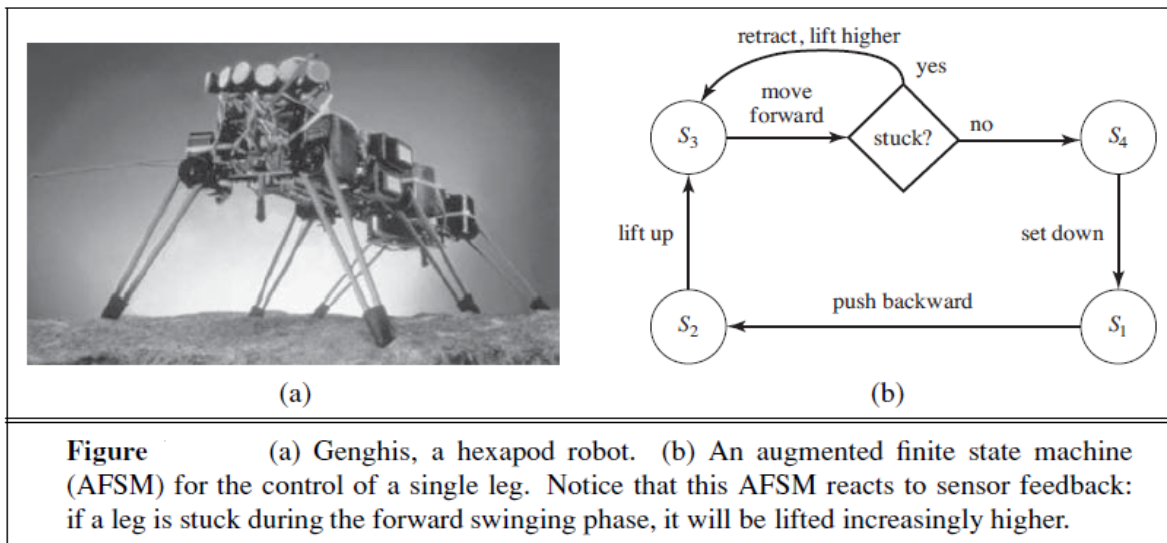
Three Basic Architecture of Robotics Software are:

1. Subsumption architecture
2. Three-layer architecture
3. Pipeline architecture

### **1.Subsumption architecture:**

The subsumption architecture (Brooks, 1986) is a framework for assembling reactive controllers out of finite state machines. Nodes in these machines may contain tests for certain sensor variables, in which case the execution trace of a finite state machine is conditioned on the outcome of such a test. Arcs can be tagged with messages that will be generated when traversing them, and that are sent to the robot's motors or to other finite state machines. Additionally, finite state machines possess internal timers (clocks) that control the time it takes to traverse an arc. The resulting machines are referred to as augmented finite state machines, or AFSMs, where the augmentation refers to the use of clocks.

An example of a simple AFSM is the four-state machine shown in below Figure which generates cyclic leg motion for a hexapod walker. This AFSM implements a cyclic controller, whose execution mostly does not rely on environmental feedback. The forward swing phase, however, does rely on sensor feedback. If the leg is stuck, meaning that it has failed to execute the forward swing, the robot retracts the leg, lifts it up a little higher, and attempts to execute the forward swing once again. Thus, the controller is able to react to contingencies arising from the interplay of the robot and its environment



Unfortunately, the subsumption architecture has its own problems.

**First**, the AFSMs are driven by raw sensor input, an arrangement that works if the sensor data is reliable and contains all necessary information for decision making, but fails if sensor data has to be integrated in nontrivial ways over time. Subsumption-style controllers have therefore mostly been applied to simple tasks, such as following a wall or moving towards visible light sources.

**Second**, the lack of deliberation makes it difficult to change the task of the robot. A subsumption style robot usually does just one task, and it has no notion of how to modify its controls to accommodate different goals (just like the dung beetle on page 39). Finally, subsumptionstyle controllers tend to be difficult to understand. In practice, the intricate interplay between dozens of interacting AFSMs (and the environment) is beyond what most human programmers can comprehend. For all these reasons, the subsumption architecture is rarely used in robotics, despite its great historical importance. However, it has had an influence on other architectures, and on individual components of some architecture.

## 2. Three-layer architecture

Hybrid architectures combine reaction with deliberation. The most popular hybrid architecture is the **three-layer architecture**, which consists of a

- *Reactive layer*
- *Executive layer*
- *Deliberative layer.*

The **reactive layer** provides low-level control to the robot. It is characterized by a tight sensor–action loop. Its decision cycle is often on the order of milliseconds.

The **executive layer** (or sequencing layer) serves as the glue between the reactive layer and the deliberative layer. It accepts directives by the deliberative layer, and sequences them for the

reactive layer. For example, the executive layer might handle a set of via-points generated by a deliberative path planner, and make decisions as to which reactive behavior to invoke. Decision cycles at the executive layer are usually in the order of a second. The executive layer is also responsible for integrating sensor information into an internal state representation. For example, it may host the robot's localization and online mapping routines

The **deliberative layer** generates global solutions to complex tasks using planning. Because of the computational complexity involved in generating such solutions, its decision cycle is often in the order of minutes. The deliberative layer (or planning layer) uses models for decision making. Those models might be either learned from data or supplied and may utilize state information gathered at the executive layer.

Variants of the three-layer architecture can be found in most modern-day robot software systems. The decomposition into three layers is not very strict. Some robot software systems possess additional layers, such as user interface layers that control the interaction with people, or a multi agent level for coordinating a robot's actions with that of other robots operating in the same environment.

### 3. Pipeline architecture

Another architecture for robots is known as the **pipeline architecture**. Just like the subsumption architecture, the pipeline architecture executes multiple processes in parallel. However, the specific modules in this architecture resemble those in the three-layer architecture. Figure shows an example pipeline architecture, which is used to control an autonomous car.

- Data enters this pipeline at the **sensor interface layer**.
- The **perception layer** then updates the robot's internal models of the environment based on this data.
- Next, these models are handed to the **planning and control layer**, which adjusts the robot's internal plans, turns them into actual controls for the robot.
- Those are then communicated back to the vehicle through the **vehicle interface layer**

The key to the pipeline architecture is that this all happens in parallel. While the perception layer processes the most recent sensor data, the control layer bases its choices on slightly older data. In this way, the pipeline architecture is similar to the human brain. We don't switch off our motion controllers when we digest new sensor data. Instead, we perceive, plan, and act all at the same time. Processes in the pipeline architecture run asynchronously, and all computation is data-driven. The resulting system is robust, and it is fast

## APPLICATION DOMAINS

Here are some of the prime application domains for robotic technology

**1. Industry and Agriculture.** Traditionally, robots have been fielded in areas that require difficult human labor, yet are structured enough to be amenable to robotic automation. The best example is the assembly line, where manipulators routinely perform tasks such as assembly, part placement, material handling, welding, and painting

**2. Transportation.** Robotic transportation has many facets: from autonomous helicopters that deliver payloads to hard-to-reach locations, to automatic wheelchairs that transport people who are unable to control wheelchairs by themselves, to autonomous straddle carriers that outperform skilled human drivers when transporting containers from ships to trucks on loading docks

**3. Robotic cars.** Most of us use cars every day. Many of us make cell phone calls while driving. Some of us even text. The sad result: more than a million people die every year in traffic accidents. Robotic cars like BOSS and STANLEY offer hope: Not only will they make driving much safer, but they will also free us from the need to pay attention to the road during our daily commute.

**4. Health care.** Robots are increasingly used to assist surgeons with instrument placement when operating on organs as intricate as brains, eyes, and hearts. Figure 25.28(b) shows such a system. Robots have become indispensable tools in a range of surgical procedures, such as hip replacements, thanks to their high precision. In pilot studies, robotic devices have been found to reduce the danger of lesions when performing colonoscopy

**5. Hazardous environments.** Robots have assisted people in cleaning up nuclear waste, most notably in Chernobyl and Three Mile Island. Robots were present after the collapse of the World Trade Center, where they entered structures deemed too dangerous for human search and rescue crews.

**6. Exploration.** Robots have gone where no one has gone before, including the surface of Mars. Robotic arms assist astronauts in deploying and retrieving satellites and in building the International Space Station. Robots also help explore under the sea. They are routinely used to acquire maps of sunken ships

**7. Personal Services.** Service is an up-and-coming application domain of robotics. Service robots assist individuals in performing daily tasks. Commercially available domestic service robots include autonomous vacuum cleaners, lawn mowers, and golf caddies. The world's most popular mobile robot is a personal service robot

**8. Entertainment.** Robots have begun to conquer the entertainment and toy industry. We see **robotic soccer**, a competitive game very much like human soccer, but played with autonomous mobile robots. Robot soccer provides great opportunities for research in AI, since it raises a range of problems relevant to many other, more serious robot applications. Annual robotic soccer competitions have attracted large numbers of AI researchers and added a lot of excitement to the field of robotics.

**9. Human augmentation.** A final application domain of robotic technology is that of human augmentation. Researchers have developed legged walking machines that can carry people around, very much like a wheelchair. Several research efforts presently focus on the development of devices that make it easier for people to walk or move their arms by providing additional forces through extra skeletal attachments

Types of AI

For all the labels, there are only three main types of AI:

1. Weak AI

2. Strong AI,

### 3. Super AI

#### *1. Weak AI*

Weak AI is both the most limited and the most common of the three types of AI. It's also known as narrow AI or artificial narrow intelligence (ANI).

Weak AI refers to any AI tool that focuses on doing one task really well. That is, it has a narrow scope in terms of what it can do. The idea behind weak AI isn't to mimic or replicate human intelligence. Rather, it's to simulate human behaviour.

***“Weak AI is nowhere near matching human intelligence, and it isn't trying to.”***

A common misconception about weak AI is that it's barely intelligent at all — more like artificial stupidity than AI. But even the smartest seeming AI of today are only weak AI.

In reality, then, narrow or weak AI is more like an intelligent specialist. It's highly intelligent at completing the specific tasks it's programmed to do.

#### *2. Strong AI*

The next of the types of AI is strong AI, which is also known as general AI or artificial general intelligence (AGI). Strong AI refers to AI that exhibits human-level intelligence. So, it can understand, think, and act the same way a human might in any given situation.

In theory, then, anything a human can do, a strong AI can do too.

***“We don't yet have strong AI in the world; it exists only in theory.”***

For a start, Moravec's paradox has us struggling to replicate the basic human functions like sight or movement. (Though image and facial recognition mean that AI is now learning to 'see' and categorise.)

Add to this that currently, AI is only capable of the few things we program into it, and it's clear that strong AI is a long way off. It's thought that to achieve true strong AI, we would need to make our machines conscious.

#### *3. Super AI*

But if strong AI already mimics human intelligence and ability, what's left for the last of the types of AI?

Super AI is AI that surpasses human intelligence and ability. It's also known as artificial superintelligence (ASI) or superintelligence. It's the best at everything — maths, science, medicine, hobbies, you name it. Even the brightest human minds cannot come close to the abilities of super AI.



*“Of the types of AI, super AI is the one most people mean when they talk about robots taking over the world.”*

Or about AI overthrowing or enslaving humans. (Or most other science fiction AI tropes.)

But rest assured, super AI is purely speculative at this point. That is, it's not likely to exist for an exceedingly long time (if at all).

Note:

**So far, we've only achieved the first of the three types of AI — weak AI. As research continues, it's reasonable to strive for strong AI.**

## **Ethics and Risks of AI**

Top 9 ethical issues in artificial intelligence

### **1. Unemployment. What happens after the end of jobs?**

The hierarchy of labour is concerned primarily with automation. As we've invented ways to automate jobs, we could create room for people to assume more complex roles, moving from the physical work that dominated the pre-industrial globe to the cognitive labour that characterizes strategic and administrative work in our globalized society.

Look at trucking: it currently employs millions of individuals in the United States alone. What will happen to them if the self-driving trucks promised by Tesla's Elon Musk become widely available in the next decade? But on the other hand, if we consider the lower risk of accidents, self-driving trucks seem like an ethical choice. The same scenario could happen to office workers, as well as to the majority of the workforce in developed countries.

### **2. Inequality. How do we distribute the wealth created by machines?**

Our economic system is based on compensation for contribution to the economy, often assessed using an hourly wage. The majority of companies are still dependent on hourly work when it comes to products and services. But by using artificial intelligence, a company can drastically cut down on relying on the human workforce, and this means that revenues will go to fewer people. Consequently, individuals who have ownership in AI-driven companies will make all the money.

### **3. Humanity. How do machines affect our behaviour and interaction?**

Artificially intelligent bots are becoming better and better at modelling human conversation and relationships. In 2015, a bot named Eugene Goostman won the Turing Challenge for the first

time. In this challenge, human raters used text input to chat with an unknown entity, then guessed whether they had been chatting with a human or a machine. Eugene Goostman fooled more than half of the human raters into thinking they had been talking to a human being.

This milestone is only the start of an age where we will frequently interact with machines as if they are humans; whether in customer service or sales. While humans are limited in the attention and kindness that they can expend on another person, artificial bots can channel virtually unlimited resources into building relationships.

#### **4. Artificial stupidity. How can we guard against mistakes?**

Intelligence comes from learning, whether you're human or machine. Systems usually have a training phase in which they "learn" to detect the right patterns and act according to their input. Once a system is fully trained, it can then go into test phase, where it is hit with more examples and we see how it performs.

Obviously, the training phase cannot cover all possible examples that a system may deal with in the real world. These systems can be fooled in ways that humans wouldn't be. For example, random dot patterns can lead a machine to "see" things that aren't there. If we rely on AI to bring us into a new world of labour, security and efficiency, we need to ensure that the machine performs as planned, and that people can't overpower it to use it for their own ends

#### **5. Racist robots. How do we eliminate AI bias?**

Though artificial intelligence is capable of a speed and capacity of processing that's far beyond that of humans, it cannot always be trusted to be fair and neutral. Google and its parent company Alphabet are one of the leaders when it comes to artificial intelligence, as seen in Google's Photos service, where AI is used to identify people, objects and scenes. But it can go wrong, such as when a camera missed the mark on racial sensitivity, or when a software used to predict future criminals showed bias against black people.

We shouldn't forget that AI systems are created by humans, who can be biased and judgemental. Once again, if used right, or if used by those who strive for social progress, artificial intelligence can become a catalyst for positive change.

#### **6. Security. How do we keep AI safe from adversaries?**

The more powerful a technology becomes, the more can it be used for nefarious reasons as well as good. This applies not only to robots produced to replace human soldiers, or autonomous weapons, but to AI systems that can cause damage if used maliciously. Because these fights won't be fought on the battleground only, cybersecurity will become even more important. After all, we're dealing with a system that is faster and more capable than us by orders of magnitude.

## **7. Evil genies. How do we protect against unintended consequences?**

It's not just adversaries we have to worry about. What if artificial intelligence itself turned against us? This doesn't mean by turning "evil" in the way a human might, or the way AI disasters are depicted in Hollywood movies. Rather, we can imagine an advanced AI system as a "genie in a bottle" that can fulfill wishes, but with terrible unforeseen consequences.

In the case of a machine, there is unlikely to be malice at play, only a lack of understanding of the full context in which the wish was made. Imagine an AI system that is asked to eradicate cancer in the world. After a lot of computing, it spits out a formula that does, in fact, bring about the end of cancer – by killing everyone on the planet. The computer would have achieved its goal of "no more cancer" very efficiently, but not in the way humans intended it.

## **8. Singularity. How do we stay in control of a complex intelligent system?**

The reason humans are on top of the food chain is not down to sharp teeth or strong muscles. Human dominance is almost entirely due to our ingenuity and intelligence. We can get the better of bigger, faster, stronger animals because we can create and use tools to control them: both physical tools such as cages and weapons, and cognitive tools like training and conditioning.

This poses a serious question about artificial intelligence: will it, one day, have the same advantage over us? We can't rely on just "pulling the plug" either, because a sufficiently advanced machine may anticipate this move and defend itself. This is what some call the "singularity": the point in time when human beings are no longer the most intelligent beings on earth.

## **9. Robot rights. How do we define the humane treatment of AI?**

While neuroscientists are still working on unlocking the secrets of conscious experience, we understand more about the basic mechanisms of reward and aversion. We share these mechanisms with even simple animals. In a way, we are building similar mechanisms of reward and aversion in systems of artificial intelligence. For example, reinforcement learning is similar to training a dog: improved performance is reinforced with a virtual reward.

## RISKS in AI

### 7 Dangerous Risks of Artificial Intelligence

- Automation-spurred job loss
- Privacy violations
- 'Deepfakes'
- Algorithmic bias caused by bad data
- Socioeconomic inequality
- Market volatility
- Weapons automatization

## AI - Agents & Environments

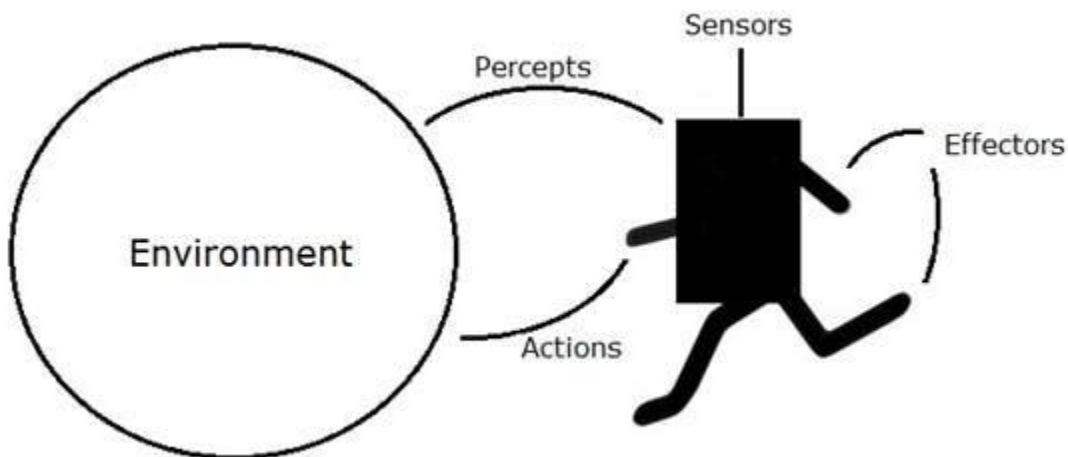
AI Agent can have mental properties like knowledge, belief, intention etc

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

### What are Agent and Environment?

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

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



### Agent Terminology

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

## Rationality

Rationality is nothing but status of being reasonable, sensible, and having good sense of judgment.

Rationality is concerned with expected actions and results depending upon what the agent has perceived. Performing actions with the aim of obtaining useful information is an important part of rationality.

## What is Ideal Rational Agent?

An ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure, on the basis of –

- Its percept sequence
- Its built-in knowledge base

Rationality of an agent depends on the following –

- The **performance measures**, which determine the degree of success.
- Agent's **Percept Sequence** till now.
- The agent's **prior knowledge about the environment**.
- The **actions** that the agent can carry out.

A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept sequence. The problem the agent solves is characterized by Performance Measure, Environment, Actuators, and Sensors (PEAS).

## The Structure of Intelligent Agents

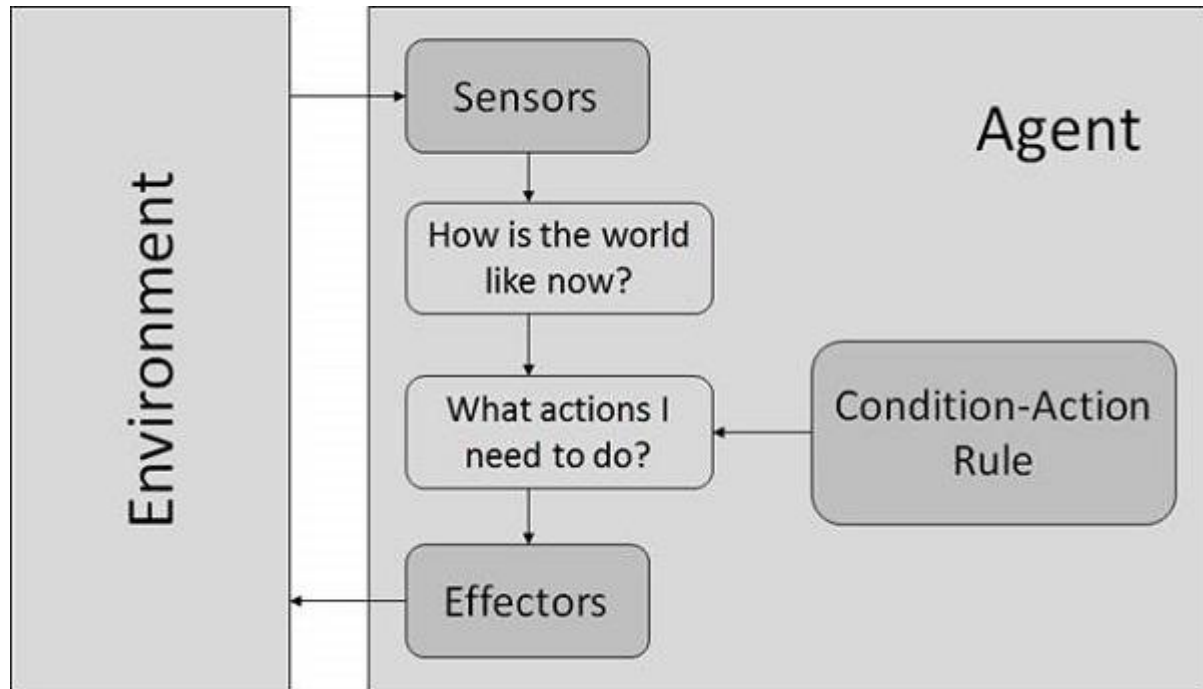
Agent's structure can be viewed as –

- Agent = Architecture + Agent Program
- Architecture = the machinery that an agent executes on.
- Agent Program = an implementation of an agent function.

## Simple Reflex Agents

- They choose actions only based on the current percept.
- They are rational only if a correct decision is made only on the basis of current percept.
- Their environment is completely observable.

**Condition-Action Rule** – It is a rule that maps a state (condition) to an action.



## Model Based Reflex Agents

They use a model of the world to choose their actions. They maintain an internal state.

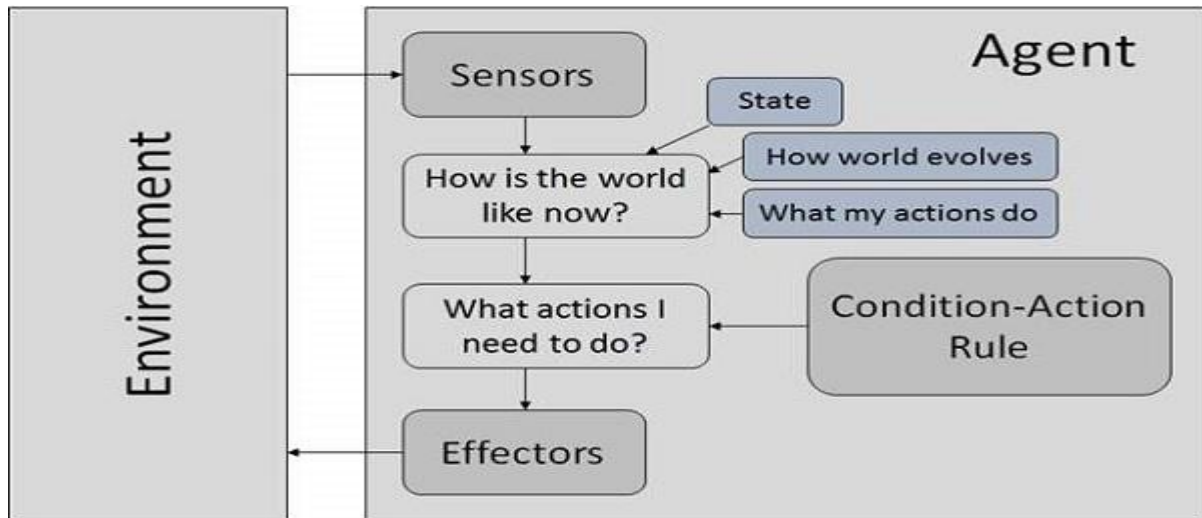
**Model** – knowledge about “how the things happen in the world”.

**Internal State** – It is a representation of unobserved aspects of current state depending on percept history.

**Updating the state requires the information about –**

- How the world evolves.

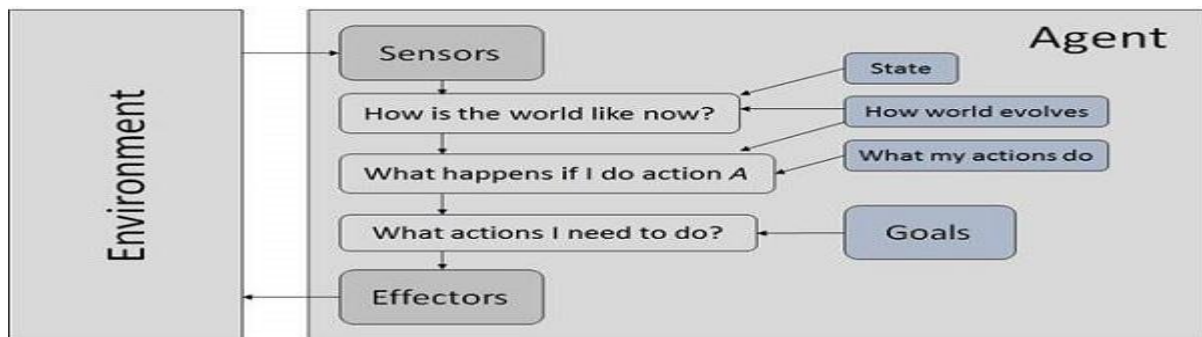
- How the agent's actions affect the world.



### Goal Based Agents

- They choose their actions in order to achieve goals. Goal-based approach is more flexible than reflex agent since the knowledge supporting a decision is explicitly modeled, thereby allowing for modifications.

**Goal** – It is the description of desirable situations.



### Utility Based Agents

- They choose actions based on a preference (utility) for each state.

Goals are inadequate when –

- There are conflicting goals, out of which only few can be achieved.

- Goals have some uncertainty of being achieved and you need to weigh likelihood of success against the importance of a goal.

