# MODULE 1

| | | | |
|---|---|---|---|
| Course Code | **1BAIA103/203** | CIE Marks | 50 |
| Teaching Hours/Week (L:T:P:) | 3:0:0:0 | SEE Marks | 50 |
| Total Hours of Pedagogy | 40 | Total Marks | 100 |
| Credits | 3 | Exam Hours | 3 |
| Examination type (SEE) | Theory | | |

**Course outcome (Course Skill Set)**
At the end of the course, the student will be able to:
CO1: Explain the concepts and types of artificial intelligence.
CO2: Illustrate basic machine learning methods for regression, classification and clustering.
CO3: Identify real-world applications across different disciplines.
CO4: Make use of prompt engineering techniques to interact with generative AI tools.
CO5: Outline recent trends in artificial intelligence and machine learning.
**Module-1**
**Introduction to Artificial Intelligence:** Artificial Intelligence, How Does AI Work? Advantages and Disadvantages of Artificial Intelligence, History of Artificial Intelligence, Types of Artificial Intelligence, Weak AI, Strong AI, Reactive Machines, Limited Memory, Theory of Mind, Self-Awareness, Is Artificial Intelligence Same as Augmented Intelligence and Cognitive Computing, Machine Learning and Deep Learning.
**Machine Intelligence:** Defining Intelligence, Components of Intelligence, Differences Between Human and Machine Intelligence, Agent and Environment, Search, Uninformed Search Algorithms, Informed Search Algorithms: Pure Heuristic Search, Best-First Search Algorithm (Greedy Search).
**Knowledge Representation:** Introduction, Knowledge Representation, Knowledge-Based Agent, Types of Knowledge.

## L1:

## 1.Introduction to Artificial Intelligence

### 1.1 What is Artificial Intelligence?

In 2004, John McCarthy defined artificial intelligence (AI) as the science and engineering of making intelligent machines, especially intelligent computer programs. However, much before this definition was coined, the birth of AI was marked by Alan Turing's seminal work, '*Computing Machinery and Intelligence*,' published in 1950. Alan Turing, also known as the 'father of computer science,' raised questions like ‘Can machines think?’ in this paper. Continuing with his work, he later proposed the ‘Turing Test’, where a human interrogator would try to distinguish between a computer and human text response.

Few languages that are popularly used to code AI applications are R, Python and Java.

**From a layman’s view,** artificial intelligence (AI), simply means the intelligence demonstrated by machines that help them to mimic the actions of humans. AI simulates natural intelligence

in machines that are programmed to learn from experiences, adjust to new inputs and perform human-like tasks.

**From a researcher’s view**, AI is a set of algorithms that generates results without having to be explicitly instructed to do so, thereby making machines capable of thinking and acting rationally and humanely. Most AI examples, from chess-playing computers to self-driving cars, heavily depend on deep learning and natural language processing techniques.

AI applications perform specialized tasks by processing large amounts of data and recognizing patterns in them. Besides learning from experience, AI applications can recognize objects, understand and respond to language and make decisions to solve real-world problems. AI perform tasks that normally require human intelligence. The main focus of this technology is to build machines and algorithms which are capable of performing computational tasks that would otherwise require human-like brain functions. While computers execute certain tasks like sorting, computing, memorizing, indexing, finding patterns, etc. far better than humans, they still cannot beat human skills for identifying emotions, recognizing faces, communication and conversation. This is where AI will play a crucial role to enable machines achieve human capabilities.

**NITI Aayog:** The National Strategy for Artificial Intelligence has defined AI as follows:

AI refers to the ability of machines to perform cognitive tasks like thinking, perceiving, learning, problem solving and decision-making. Initially conceived as a technology that could mimic human intelligence, AI has evolved in ways that far exceed its original conception. With incredible advances made in data collection, processing and computation power, intelligent systems can now be deployed to take over a variety of tasks, enable connectivity and enhance productivity.

### 1.1.1 How Does AI Work?

AI systems work effectively when fed with a large amount of labelled training data. This data is thoroughly analysed to discover correlations and patterns. These patterns are then used to make predictions about future states. For example, a chatbot fed with examples of text chats can learn to converse with humans in real-world applications. AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

#### Learning Processes

AI programs focus on acquiring data and creating rules for turning that data into actionable information. Rules, also known as algorithms, provide step-by-step instructions to complete a specific task.

#### Reasoning Processes

The success of an AI program depends on choosing the right algorithm to reach the desired outcome.

#### Self-Correction Processes

AI programs are designed to continually enhance their algorithms to provide the most accurate results.

Case Study: Why is artificial intelligence important?

AI allows business organizations to gain insights into their operations that they may not have been previously aware of. There are numerous tasks that AI-enabled machines can perform better than humans. This has acted as the driving force for exploring new business opportunities. It was because of AI that computer software could be used to connect riders to taxis. Uber, one of the largest companies in the world, is utilizing sophisticated machine learning algorithms to predict when people would need a greater number of rides in a particular area so that the company can get drivers on the road before they are needed.

### 1.1.2 Advantages and Disadvantages of Artificial Intelligence

Artificial intelligence, when used along with machine learning and deep learning techniques, is quickly evolving to processes massive amounts of data much faster and makes predictions more accurately than humanly possible. While huge volumes of data are being generated every day, only AI applications can utilize this data to quickly turn it into actionable information.

Google has become one of the largest players in AI. It provides a range of online services by using machine learning to understand how people use their services and then improving them.

#### Advantages

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

### Disadvantages

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

In 2017, Google's CEO, Sundar Pichai, pronounced that Google would operate as an 'AI first' company.

***Conclusion:*** Today, every big enterprise has used AI to improve its operations and gain advantage on their competitors.

## 1.2 History of Artificial Intelligence

Let us analyse the support for the modern field of AI, right from 1943 to the present day (Fig. 1.1).

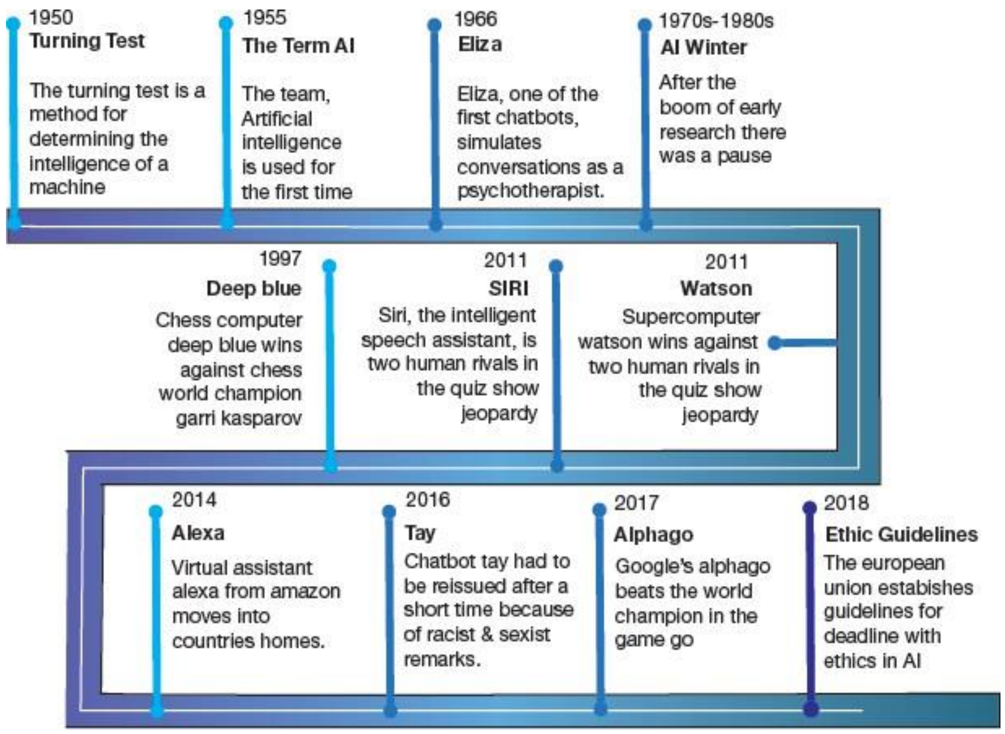

1950
Turning Test
The turning test is a method for determining the intelligence of a machine
1955
The Term AI
The team, Artificial intelligence is used for the first time
1966
Eliza
Eliza, one of the first chatbots, simulates conversations as a psychotherapist.
1970s-1980s
AI Winter
After the boom of early research there was a pause
1997
Deep blue
Chess computer deep blue wins against chess world champion garri kasparov
2011
SIRI
Siri, the intelligent speech assistant, is two human rivals in the quiz show jeopardy
2011
Watson
Supercomputer watson wins against two human rivals in the quiz show jeopardy
2014
Alexa
Virtual assistant alexa from amazon moves into countries homes.
2016
Tay
Chatbot tay had to be reissued after a short time because of racist & sexist remarks.
2017
Alphago
Google's alphago beats the world champion in the game go
2018
Ethic Guidelines
The european union estabishes guidelines for deadline with ethics in AI

**FIGURE 1.1** History of Aritificial Intelligence (AI)

In **1943**, Warren McCullough and Walter Pitts wrote a paper that proposed the first mathematical model for building a neural network.

**1950** was a significant year in the field of AI. A series of events took place this year including the following:

- Alan Turing demonstrated Turing Test, that could determine whether a machine is intelligent or not.
- Harvard undergraduates Marvin Minsky and Dean Edmonds built the first neural network computer.
- Claude Shannon publishes a paper on programming a computer for playing chess.
- Isaac Asimov published his work on the 'Three Laws of Robotics.'

In **1952**, Arthur Samuel developed a self-learning program to play checkers.

In **1954**, the IBM computer could translate 60 carefully selected Russian sentences into English.

In **1956**, John McCarthy coined the term 'artificial intelligence'. The scope and goals of AI were discussed in a conference later. It is from this point that we consider the birth of artificial intelligence as we know it today.

In the same year, Allen Newell and Herbert Simon demonstrated the first reasoning program.

In **1958**, John McCarthy developed the AI programming language, Lisp. He also published a paper discussing a complete AI system that could learn from experience as effectively as humans do.

In **1959**, Allen Newell, Herbert Simon and J.C. Shaw developed the General Problem Solver (GPS), a program designed to imitate human problem-solving.

In the same year, Arthur Samuel coined the term 'machine learning'.

In **1963**, John McCarthy started the AI Lab at Stanford.

MIT Professor Joseph Weizenbaum developed ELIZA, an early natural language processing program that laid the foundation for today's chatbots.

In **1969**, the first successful expert systems to diagnose blood infections was developed at Stanford.

In **1972**, logic programming language PROLOG was created.

**The 1974–1980** period is known as the 'First AI Winter,' because it was during this period that DARPA cutbacks in academic grants and funding for AI dried up, stalling further research.

In **1980**, DEC developed the first successful commercial expert system, R1.

In **1982** Japan launched the ambitious Fifth Generation Computer Systems (FGCS) project to develop supercomputers that could support performance and a platform for AI development. Later, in **1983**, even the US government funded research in advanced computing and artificial intelligence.

**1987–1993** marked the 'Second AI Winter.' In 1992, Japan terminated the FGCS project owing to failure in meeting its ambitious goals. The US also ended the project in 1993 after spending nearly $1 billion as the results were far from what were expected.

In **1997**, IBM's Deep Blue machine could beat world chess champion, Gary Kasparov.

In **2005**, STANLEY, a self-driving car, won the DARPA Grand Challenge. Even the US military started investing in autonomous robots like Boston Dynamics' 'Big Dog' and iRobot's 'PackBot.'

In **2008**, Google did a remarkable work in speech recognition and introduced the feature in its iPhone app.

In **2011**, Apple released Siri, an AI-powered virtual assistant that could be used through its iOS operating system.

In **2012**, Andrew Ng, founder of the Google Brain Deep Learning project, fed 10 million YouTube videos as a training set to a neural network and used deep learning algorithms to enable the neural network to recognize a cat without being told what a cat is.

In **2014**, Google makes the first self-driving car that could pass a state driving test. In the same year, Amazon released Alexa.

**In 2015,** Baidu's Minwa supercomputer used a convolutional neural network (special deep neural network) to identify and categorize images with a higher rate of accuracy than the average human.

In **2016** Google DeepMind's AlphaGo defeated world champion Go player Lee Sedol. Go is an ancient Chinese game and winning this game was a major breakthrough in the field of AI.

In the same year, the first 'robot citizen', a humanoid robot, Sophia was created that could perform facial recognition, verbal communication and facial expression.
In **2018**, Google released NLP engine BERT that reduced barriers in translation and understanding by machine learning applications. In the same year, Waymo One service was launched that allowed users to request a pick-up from one of the company's self-driving vehicles.
In **2020**, Baidu released its LinearFold AI algorithm to develop a vaccine during the early stages of the SARS-CoV-2 pandemic.
The algorithm could predict the RNA sequence of the virus in just 27 seconds, 120 times faster than other methods.

**Review Questions:**

**1.Define AI**

**2. How does AI work?**

**3.What are three cognitive skills?**

## L2:

### 1.3 Types of Artificial Intelligence

We can classify an artificial intelligent system into one of the following categories (Fig. 1.2).

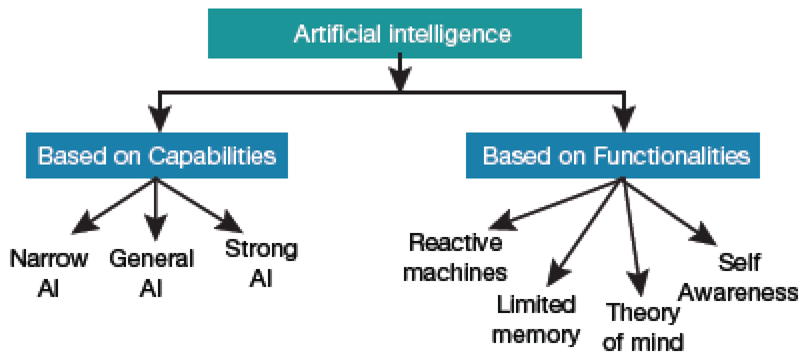


**FIGURE 1.2** Types of Artificial Intelligence

### 1.3.1 Weak AI

Weak AI, also known as narrow AI,

is specifically designed to perform a specific type of task. For example, Siri and Alexa are weak AI systems. These systems are already trained with appropriate responses to classify things accordingly. When you instruct Alexa to play a song, it responds by playing that song.

In fact, majority of AI applications that we use today (predicting weather, stock prices, optimizing business, etc.) come under this category. Weak AI systems operate within a limited context and are the most successful realization of artificial intelligence to date. Application of narrow AI has resulted in significant societal benefits. Google search, Image recognition software, self-driving cars and IBM's Watson are some examples of such systems.

### 1.3.2 Strong AI

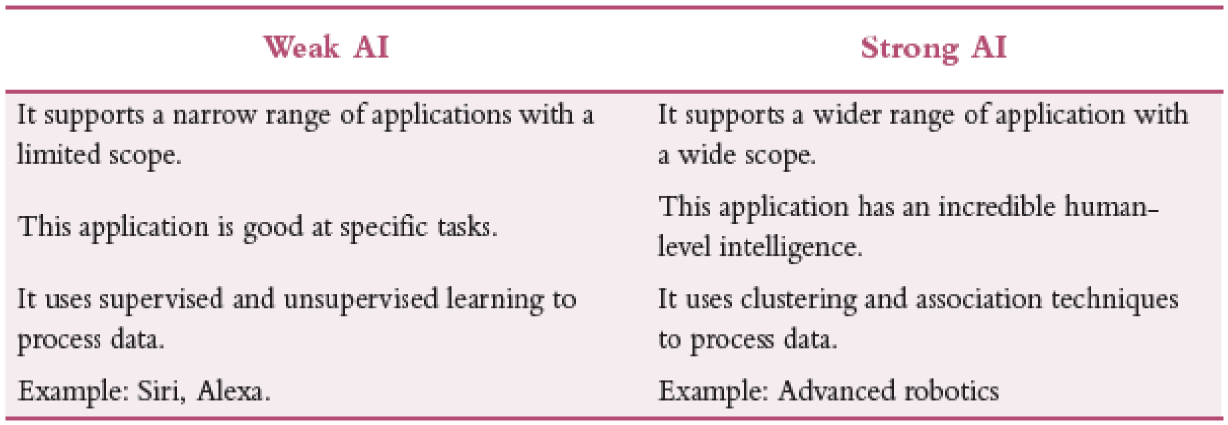
Strong AI, also known as artificial general intelligence or artificial super intelligence (ASI) or superintelligence, makes full attempt to resemble the human brain. It utilizes cognitive skills and fuzzy logic to perform tasks for which it had not been trained earlier. Such a system needs capability for visual perception, speech recognition, decision-making and translations between languages.

As of now, we see general AI only in sci-fi movies, where machines emulate human intelligence and think strategically to perform complex tasks. However, it is believed that soon, ASI will surpass the intelligence and ability of the human brain as demonstrated by HAL, the superhuman, rogue computer assistant in *2001: A Space Odyssey.*

Many people fear that intelligent robots would overrun humanity, but experts says that we need not worry about it as it is not likely to happen in the near future.

Moreover, artificial intelligence systems can also be categorized **into** four groups based on their functionality**.** This categorization can be given as discussed in Table 1.1.

**TABLE 1.1** Weak AI vs Strong AI



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

## 1.3.3 Reactive Machines

Reactive machines are very basic machines with no memory to store and thus use past experiences to determine future actions. They just perceive the world and react to it. IBM's Deep Blue, which defeated chess grandmaster Kasporov, is such an example. The AI system sees the pieces on a chess board and responds without referring to any of its prior experiences. Since it lacks capabilities to store past experiences, a reactive machine cannot improve with practice. A strong AI program must pass a Turing Test as well as the Chinese room test.

Reactive machines can perform only a limited number of specialized tasks. Such machines are more trustworthy and reliable as they react the same way to the same stimuli every time.

Deep Blue could only identify the pieces on a chess board and compute its next move based on the rules of chess and the present positions. The AI application could not pursue future potential moves by its opponent or plan its own pieces in a better position. Every move was treated as its own reality, without any connection with any other previous movement.

Google's AlphaGo is another example of a game-playing reactive machine. The application evaluates future moves using neural network. This gives it an edge over Deep Blue game.

### 1.3.4 Limited Memory

Limited memory machines retain data but only for a short period of time and can use it only for a limited time. They cannot keep adding data permanently to a library of their experiences. Such systems are usually used in autonomous vehicles since they need not store data (like *recent speed of nearby cars, distance between cars, the speed limit, etc.) that can help them navigate roads*.
AlphaGo also bested world-class competitors of the game, defeating champion Go player Lee Sedol in 2016.

No doubt, limited memory AI is more complex and is used for better possibilities than reactive machines. These machines continuously train a model to analyse and utilize new data. The model also improves using feedback received from humans or the environment and stored as data.

There are three major machine learning models that applies limited memory artificial intelligence:

1. **Reinforcement learning:** It enables continuous learning through repeated trial-and-error to make better predictions.
2. **Long Short-term Memory (LSTM):** Such models use past data to help predict

the next item in a sequence. More recent information is given a higher priority for prediction than past data. However, this does not mean that past data is not considered while making predictions.

3. **Evolutionary Generative Adversarial Networks (E-GAN):** The ML model evolves over time to explore new ways of utilizing previous experiences to jump formulate new decisions. The model is continuously striving to discover a better path. It makes use of simulations and statistics, to predict outcomes throughout its evolutionary mutation cycle.

### 1.3.5 Theory of Mind

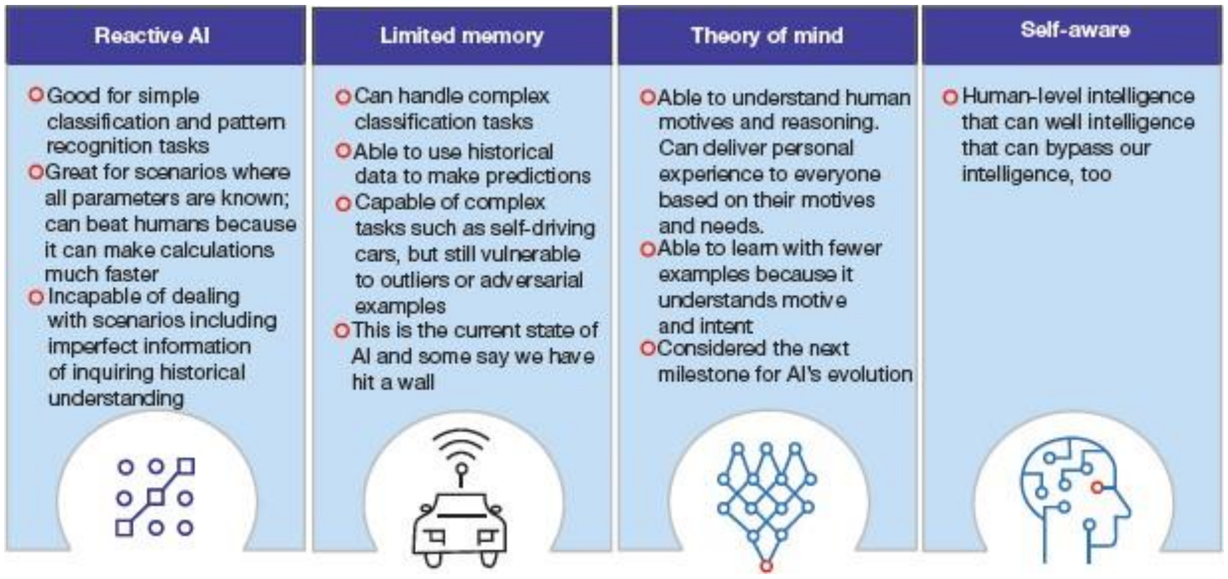
The theory of mind focuses on imitating the human brain. For this, it forms representations about the world, starting from thoughts, emotions and memories. However, such systems are only theoretical as of now but they may become reality very soon.

These machines must understand thoughts and emotions that affect the behaviour of one's self. Theory of Mind machines make decisions keeping in view the feelings experienced through self-reflection and determination.

### 1.3.6 Self-Awareness

Once Theory of Mind is established, then the next step would be incorporating self-awareness in AI systems. Self-awareness in machines enable them to possess human-level consciousness, understand the current state—its own existence in the world and use that information to deduce others feelings. Self-AI systems will be able to understand what others may need. They could interpret user's feelings by learning not only what they communicate to them but also how they communicate it.

Knowledge imbibed by researchers and its learning of the conscious context will help them to respond to an event. However, such systems do not yet exist but may become available in the near future.



| Reactive AI | Limited memory | Theory of mind | Self-aware |
|---|---|---|---|
| ○ Good for simple classification and pattern recognition tasks<br>○ Great for scenarios where all parameters are known; can beat humans because it can make calculations much faster<br>○ Incapable of dealing with scenarios including imperfect information of inquiring historical understanding | ○ Can handle complex classification tasks<br>○ Able to use historical data to make predictions<br>○ Capable of complex tasks such as self-driving cars, but still vulnerable to outliers or adversarial examples<br>○ This is the current state of AI and some say we have hit a wall | ○ Able to understand human motives and reasoning. Can deliver personal experience to everyone based on their motives and needs.<br>○ Able to learn with fewer examples because it understands motive and intent<br>○ Considered the next milestone for AI's evolution | ○ Human-level intelligence that can well intelligence that can bypass our intelligence, too |

**FIGURE 1.3** Types of AI Systems Based on Functionalities

**Review Questions:**

**1.What is Weak AI**

**2. What is Strong AI**

**3.What is Theory of Mind ?**

## L3:

### 1.4 Is Artificial Intelligence Same as Augmented Intelligence and Cognitive Computing?

Some experts think that *artificial intelligence* is the same as augmented intelligence. But this is not true. Let us understand the difference between the two terms.

**Augmented intelligence** is weak AI that simply improves products and services. For example, automatically highlighting vital information in business is a part of implementing augmented intelligence.

**Artificial intelligence** or true AI, strong AI or artificial general intelligence, is the future AI that would far surpasses the human brain's ability to understand and implement complex tasks. As of now, though lot of work is being done in this domain, AI remains within the realm of science fiction. Technologies like quantum computing can play an important role in making AGI a reality.

The term AI when used in reference to machines, makes machines obtain human intelligence by simulating how humans sense, learn, process and react to information in the environment.

**Cognitive computing** is a term that is especially coined for products and services that mimic that augments human thought processes.

## MACHINE LEARNING

**Let us try to understand the term 'machine learning' with the help of some real-world scenarios. Scenario 1:** Imagine that you are given the task of finding the missing number in the series-10,20,30,40, ?

Yes, you are correct. That's 50. But how did you get the answer?

You quickly did some computations in your mind and observed the pattern in the data values. The whole idea of machine learning is to imbibe the same thinking process in machines.

**Scenario 2:** Before going for playing a cricket match, you always use your past experiences to understand how a batsman/bowler plays. For example, if you know a bowler gives a straight ball after every 2 left balls and 1 right ball, then you can better use this information to prepare yourself for the next ball. In machine learning, we make machines also to learn from past experience/data.

**Scenario 3:** When you buy a new mobile phone, you can yourself compute the price of that phone vis-à-vis the configuration and brand. For example, you know a phone with 128 GB RAM, stereo speakers, iOS 16, hexacore would cost around 80K. So, given the details, we can predict the price.
Similarly, a machine learning program is developed to predict values based on

certain specifications. **Now let us go deeper in this terminology. Machine Learning** is a branch of computer science that analyses data and identifies patterns to teach a machine to deduce results and make decisions without any human intervention.

ML algorithms learn from experiences rather than instructions.

They automatically learn and improve by learning from their output. For this, humans do not have to write instructions for them to produce the desired output. They learn by analysing data sets and comparing the final output. In case of any error, they repeat the learning process until the accuracy of the outputs improve.

This automation not only saves human time and effort but also makes better decisions.

Thus, machine learning is an application of artificial intelligence (refer Fig. 1.4) that enables machines to automatically learn and improve from experiences without being explicitly programmed.

It enables computers (machines) to make data-driven decisions rather than being explicitly programmed for carrying out a certain task. These algorithms are designed to learn and improve over time when are exposed to new data. Thus, the process of teaching the machines is a structural process that can be broken down into three steps (Fig. 1.5).

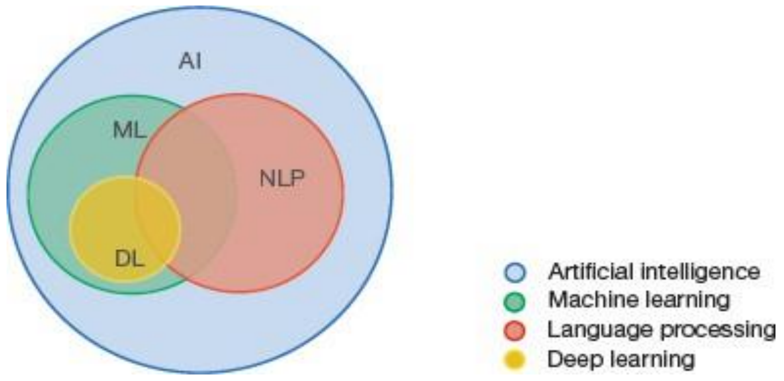


**FIGURE 1.4** Relationship between Artificial Intelligence, Machine Learning, Deep Learning and Natural Language Processing

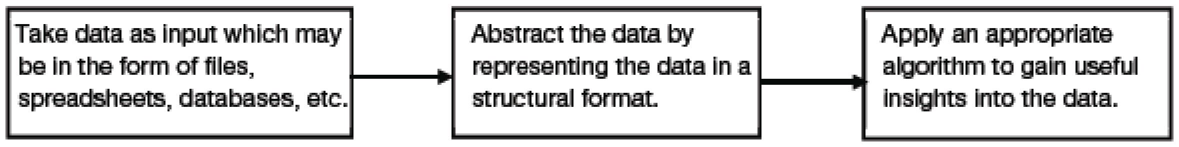

Take data as input which may be in the form of files, spreadsheets, databases, etc.
Abstract the data by representing the data in a structural format.
Apply an appropriate algorithm to gain useful insights into the data.

**FIGURE 1.5** Making the machines learn

## HOW IS AI RELATED TO MACHINE LEARNING?

Machine learning and artificial intelligence are two different techniques where machine learning is a tool for achieving artificial intelligence. The aim of AI is to create intelligent machines that can recognize human speech, have vision to see, assimilate knowledge, strategize and solve problems as humans do.

Machine learning provides machines the ability to learn, forecast and progress on their own without specifically being programmed. ML system improves by learning from more data and experiences.

Imagine you have been given the task of making a robot see, talk, walk, sense and learn. Which technique will you use—ML or AI? Of course, AI. This is because to achieve this task, multiple technologies come into picture (like NLP, computer vision, voice recognition, reinforcement learning, etc). Machine learning is just one of them that will be used only for learning purpose. Therefore, AI is the superset and ML is the subset of AI. That is, ML is only a way to achieve AI (Fig. 1.6).

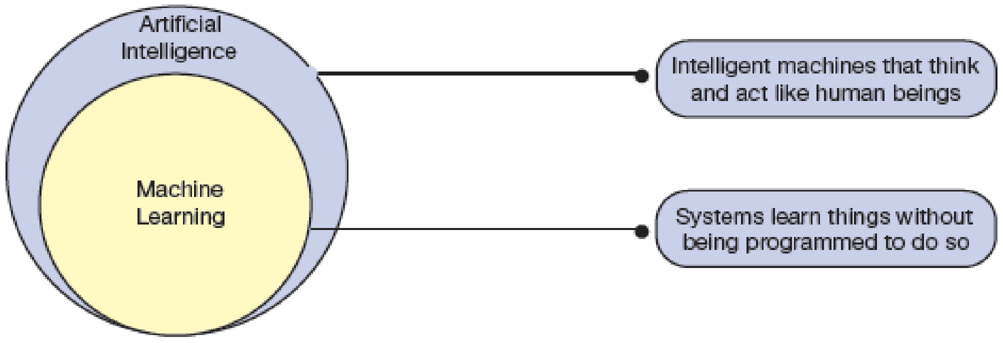
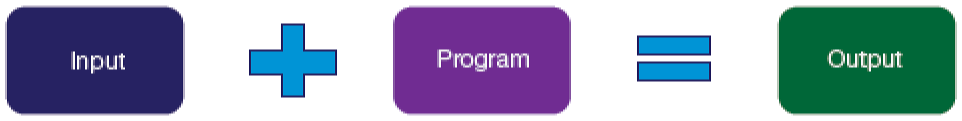

Artificial Intelligence
Machine Learning
Intelligent machines that think and act like human beings
Systems learn things without being programmed to do so

**FIGURE 1.6** Difference between AI and ML

## Traditional Programming vs Machine Learning

**Traditional programming** involves manually creating a program that accepts data and returns an output. It rests on a manual process as one has to manually formulate or code rules.

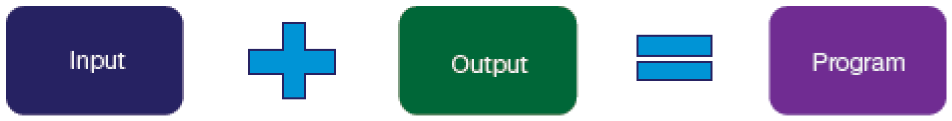


Traditional programming uses conventional procedural language like assembly language or a high-level language such as C, C++, Java, JavaScript, Python, etc. These languages code the rules and write lines of code manually. The program accepts input data, and based on the programming logic, generates the desired output.

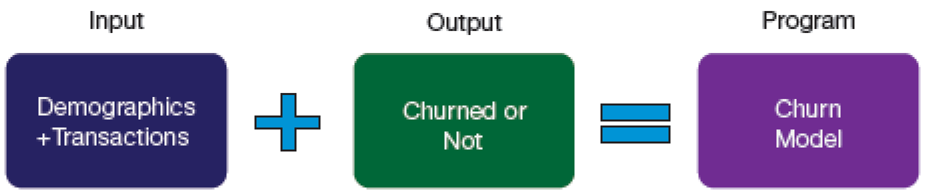
The conventional programming approach is algorithm dependent, and for a program, multiple algorithms can be used. The programmer can analyse the performance of all the algorithms to choose the best that meets the current requirements.

Recently, an advanced type of programming has revolutionized business by adding power to the applications working on intelligence and embedded analytics. **Machine learning** programming, also known as augmented analytics, accepts both data as well as output. It then generates the corresponding program for giving the specified output on that input. This gives insights that can be used to predict future outcomes.

The machine learning algorithm automatically formulates the rules from the data. Unlike traditional programming, machine learning is an automated process that adds the capability of embedded analytics in areas including natural language interfaces, automatic outlier detection, recommendations, etc. All of these features give better insights and reduce decision bias.



For example, if we give customer demographics and transactions as input and use historical customer churn rates as output, then the algorithm will formulate a program (also known as **predictive model)** that can predict if a customer will churn or not.



Therefore, the steps to create a model to predict outcomes can be given as follows:

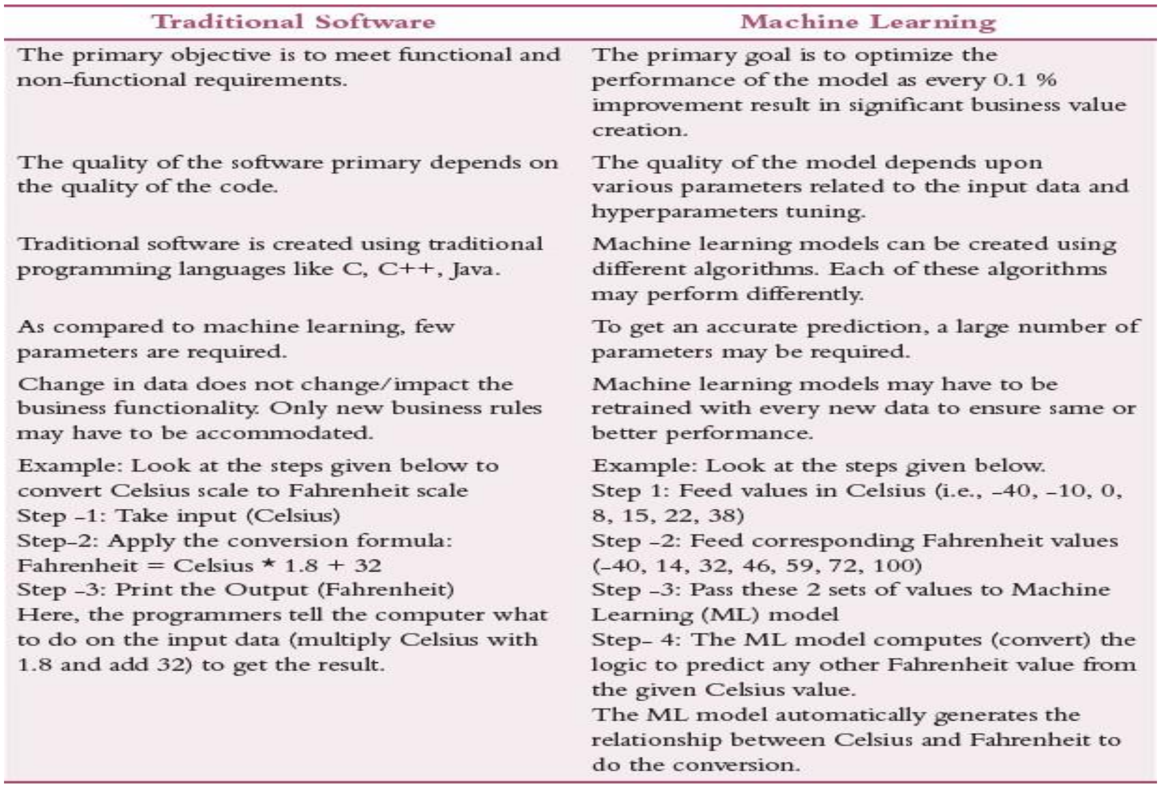
Step 1: Identify the business question you would like to ask. Step 2: Identify the historical input.

Step 3: Identify the historically observed output (when the condition is true and when it is false).

Machine learning language goes beyond traditional algorithmic solutions and trains the machine to solve different complex tasks by itself. Unlike conventional programming, machine learning uses a pre-written algorithm that learns how to solve the problem itself. It is a more sophisticated way of solving a problem.

***For Programmers:*** A data engineer cannot work as a substitute to a conventional programmer and vice versa. Every data engineer must know at least one coding language. However, a conventional programmer may not learn the machine learning algorithms.

The capabilities of both traditional programming and machine learning are altogether different and cannot replace each other (refer Table 1.2). However, machine learning supplements conventional programming. For example, while machine learning can build predictive algorithms, traditional programming, on the other hand, can be used to design user interface or perform data visualization. C***ase Study:*** It is not easy to filter Amna's photos from a set of 100 images using traditional programming. Even comparing every pixel value may not give accurate results. But the same task can be easily done with machine learning. We just need to input different photos of Amna, the machine learning model will



| Traditional Software | Machine Learning |
|---|---|
| The primary objective is to meet functional and non-functional requirements. | The primary goal is to optimize the performance of the model as every 0.1 % improvement result in significant business value creation. |
| The quality of the software primary depends on the quality of the code. | The quality of the model depends upon various parameters related to the input data and hyperparameters tuning. |
| Traditional software is created using traditional programming languages like C, C++, Java. | Machine learning models can be created using different algorithms. Each of these algorithms may perform differently. |
| As compared to machine learning, few parameters are required. | To get an accurate prediction, a large number of parameters may be required. |
| Change in data does not change/impact the business functionality. Only new business rules may have to be accommodated. | Machine learning models may have to be retrained with every new data to ensure same or better performance. |
| Example: Look at the steps given below to convert Celsius scale to Fahrenheit scale<br>Step -1: Take input (Celsius)<br>Step-2: Apply the conversion formula: Fahrenheit = Celsius * 1.8 + 32<br>Step -3: Print the Output (Fahrenheit)<br>Here, the programmers tell the computer what to do on the input data (multiply Celsius with 1.8 and add 32) to get the result. | Example: Look at the steps given below.<br>Step 1: Feed values in Celsius (i.e., -40, -10, 0, 8, 15, 22, 38)<br>Step -2: Feed corresponding Fahrenheit values (-40, 14, 32, 46, 59, 72, 100)<br>Step -3: Pass these 2 sets of values to Machine Learning (ML) model<br>Step- 4: The ML model computes (convert) the logic to predict any other Fahrenheit value from the given Celsius value.<br>The ML model automatically generates the relationship between Celsius and Fahrenheit to do the conversion. |

learn how she looks like.

## 1.5 Machine Learning and Deep Learning

Majority of AI systems are supported by breakthroughs in machine learning and deep learning techniques. These techniques together go hand-in-hand to deliver an efficient system that sometimes it is just difficult to understand the difference between artificial intelligence, machine learning and deep learning. However, venture capitalist Frank Chen highlighted this difference by stating '*Artificial intelligence is a set of algorithms and intelligence to try to mimic human intelligence. Machine learning is one of them, and deep learning is one of those machine learning techniques.'*

In simple words, machine learning techniques are used to feed data in to the computer so that statistical techniques can be applied to it to help machines 'learn' how to solve tasks or which it had not been specifically programmed. These computers free programmers from writing several lines of code. Machine learning techniques can be further classified into supervised learning (using labelled data sets) or unsupervised learning (using unlabelled data sets).

***Deep learning*** is an advanced machine learning technique that passes data inputs through a biologically-inspired multiple layers of neural network. Such ***a neural network architecture consists of a number of hidden layers through which the data is processed.*** Data processed in this architecture allows machines to go 'deep' in its learning, making connections and weighting input for the best results.

***How does AI work?***

AI algorithms solve specific tasks in the following manner:

1. Studying very large amounts of data.
2. Making generalizations or statistical estimations model (Refer Fig. 1.7).

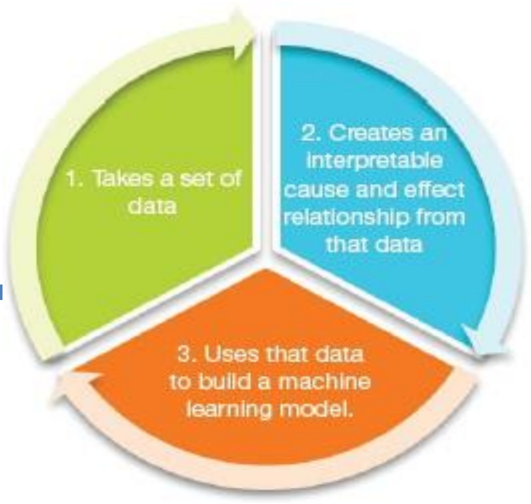

1. Takes a set of data
2. Creates an interpretable cause and effect relationship from that data
3. Uses that data to build a machine learning model.

**FIGURE 1.7** Building a ML model

**Features of AI**

1. Artificial intelligence is **autonomous** and can make independent decisions without any human intervention. These systems work silently in the background without the user's knowledge. The best part of an AI system is that they don't have to be programmed manually to respond to every situation. Rather, these systems themselves learn through input data and past experiences.

   Baidu, a search engine that works in the Chinese language, has a voice cloning tool, which can clone human voice with just 3-4 seconds of audio, as opposed to the previously taken 30 minutes
2. P**redict and adapt: AI systems can** understand data patterns to make decisions and predictions.
3. An AI system **continuously learns** from patterns in data.
4. An AI system is said to be **reactive as i**t perceives a problem and acts on perception.
5. **Data driven:** In the last several years, the cost of technology has gone down considerably. With the availability of cheap data storage (hard disk, etc.), fast processors (CPU, GPU or TPU) and sophisticated deep learning algorithms, huge volumes of data can now be easily extracted. This has led to the rise of data centric AI systems.
6. Data driven AI systems can make accurate predictions based on past experiences have outperformed humans. However, success of these systems depends largely on the availability of correctly labelled large datasets.
7. Finally, an AI system is **futuristic** since the scope of it being applied in different areas is continuously expanding.

**Review Questions:**

**1.What is ML?**

**2. What is Deep learning?**

**3.How does AI work?**

## L4:

## 3. Artificially Intelligent Machine

### 3.1 Defining Intelligence

Howard Gardner, the American developmental psychologist, has categorized intelligence into the following categories.

***Linguistic intelligence*** is the ability to speak, recognize and use mechanisms of phonology (speech sounds), syntax (grammar) and semantics (meaning). It is mainly exhibited by narrators and orators.

***Musical intelligence*** is the ability to create, communicate with and understand different sounds, pitch and rhythm. This intelligence is used by musicians, singers and composers.

***Logical-mathematical intelligence*** helps a person to use and understand complex and abstract ideas. Mathematicians and scientists extensively use this type of intelligence.

***Spatial intelligence*** is the ability to perceive visual or spatial information, change it and re-create visual images. Using this, 3D images can be created and manipulated (translated, transformed and rotated). This intelligence is generally applied by map readers, astronauts and physicists.

***Bodily-kinesthetic intelligence*** defines the ability to use the entire or part of the body to solve problems or manipulate objects. For example, this kind of intelligence is mostly used by players and dancers.

***Intrapersonal intelligence*** is the ability to distinguish between one's feelings, intentions and motivations. A few people like Gautam Buddha has this instinct.

***Interpersonal intelligence*** is the ability to recognize and differentiate between other people's feelings, beliefs and intentions. Mass communicators and interviewers use these skills to the maximum. A machine or a system is said to be **artificially intelligent** when it exhibits at least one and at most all intelligences.

## 3.2 Components of Intelligence

Intelligence is an intangible entity composed of the following (Fig. 3.1):

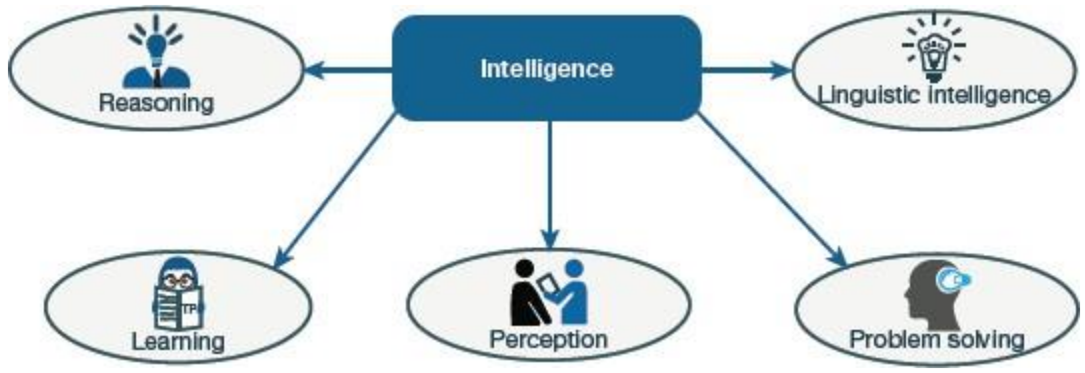
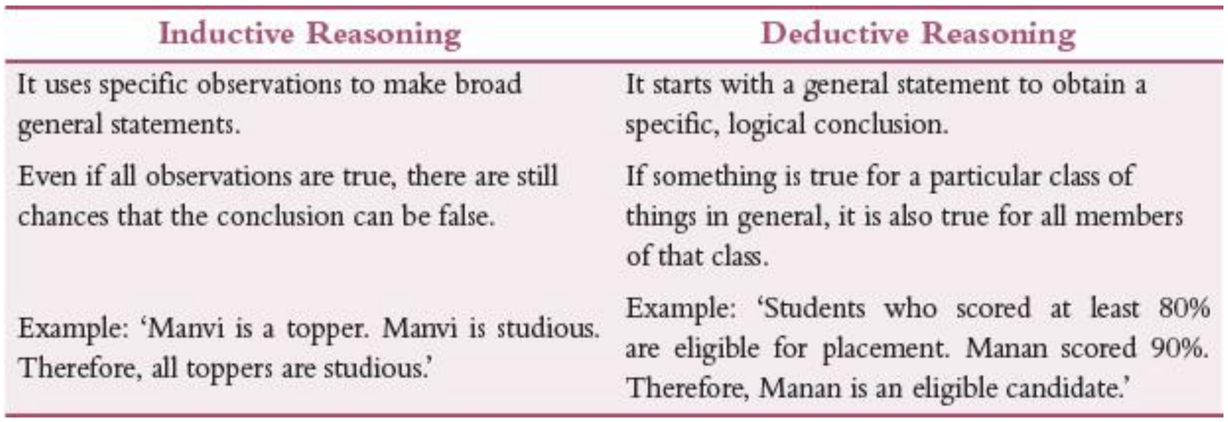


**FIGURE 3.1** Components of Intelligence

### Reasoning

It is the set of processes that is used in making decisions and prediction. There are two types of reasoning—inductive and deductive. The differences between these two techniques is given in Table 3.1.

**TABLE 3.1** Inductive vs Deductive Reasoning



| Inductive Reasoning | Deductive Reasoning |
|---|---|
| It uses specific observations to make broad general statements. | It starts with a general statement to obtain a specific, logical conclusion. |
| Even if all observations are true, there are still chances that the conclusion can be false. | If something is true for a particular class of things in general, it is also true for all members of that class. |
| Example: 'Manvi is a topper. Manvi is studious. Therefore, all toppers are studious.' | Example: 'Students who scored at least 80% are eligible for placement. Manan scored 90%. Therefore, Manan is an eligible candidate.' |

## Learning

It is a process of gaining knowledge or skill by studying, practicing, being taught or experiencing something. Learning is an ability that is possessed by humans, animals and AI-enabled systems to improve the awareness of the subjects of the study.

Learning can be categorized as follows.

1. **Auditory learning** promotes learning by listening and hearing. For example, students hear recorded lectures to understand a particular concept.
2. **Episodic learning** promotes learning by remembering the sequences of events that one has witnessed or experienced. This is linear and orderly. **Motor learning** helps to learn by precise movement of muscles. For example, picking objects.
3. **Observational learning** triggers learning by watching and imitating others. For example, children often learn by mimicking their parents.
4. **Perceptual learning** is done by recognizing stimuli that one has seen before. For example, identifying and classifying objects and situations.
5. **Relational learning** involves learning to differentiate among various stimuli on the basis of relational properties, rather than absolute properties. For example, if we know that the last time we cooked a dish, it had excess spices,

the next time, we would add spices in lesser quantity.

6. **Spatial learning** is done through visual stimuli such as images, colours, maps, etc. For example, we usually create a roadmap in our minds before actually moving to the road.
7. **Stimulus-response learning** facilitates subjects to perform a particular behaviour when a certain stimulus is received. For example, we usually shout when we touch a hot vessel.

### Problem Solving

It is the process in which one tries to find the desired solution in the present situation by following the path which is blocked by known or unknown hurdles. Problem solving uses **decision-making to** select the best suitable alternative to reach the desired goal.

### Perception

It is the process of acquiring, interpreting, selecting and organizing sensory information. While humans use sensory organs to perceive their environment, AI systems use data acquired by the sensors to do the same.

### Linguistic Intelligence

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

## 3.3 Differences Between Human and Machine Intelligence

1. Humans perceive by patterns, whereas machines perceive by analysing data with respect to a given set of rules.
2. Humans store and recall information by patterns but machines use searching algorithms to do the same.
3. Humans have intelligence to deduce any information that is missing or distorted but machines lack this ability to do it with a high level of accuracy.

**Review Questions:**

**1.What is Problem Solving?**

**2. What is learning?**

**3. Differencebetween Human and Machine ?**

## L5:

### 3.4 Agent and Environment

Artificial intelligence is all about studying rational agents, which may include anything that makes decisions. For example, a person, firm, machine or software that performs action(s) with the best outcome after considering past and current percepts at a given instance are examples of agents. These agents work in their environment and may interact with other agents in that environment. Thus, we can say that an AI system comprises an **agent and its environment**.

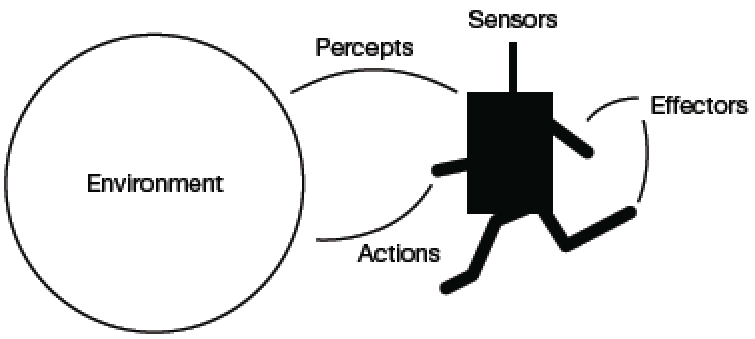


**FIGURE 3.2** AI Agent

In an artificially intelligent system, the agents act in their environment which may contain other agents. This **agent** (refer Fig. 3.2) can be anything that is capable of perceiving its environment. For this, it may have **sensors. Agents**

**also have effectors** that help them to act upon their environment**. *The different types of agents in an AI system include the following:***

1. H**uman agent:** A human agent has sensory organs (like eyes, ears, nose, tongue and skin) that acts as sensors, and other organs such as hands, legs, mouth as effectors for taking actions.
2. R**obotic agent:** A robotic agent uses cameras and infrared range finders as sensors, and various motors and actuators as effectors.
3. S**oftware agent:** Such an agent uses bit strings as its programs and actions.

### 3.4.1 KeyTerminology

#### Performance Measure of Agent

This criteria helps to determine how successful an agent is.

#### Behaviour of Agent

It is an action performed by an agent after any receiving a certain percept.

#### Percept

These are perceptual inputs given to an agent at a specific instance.

#### Percept Sequence

This is a list of all percepts received by an agent till date.

#### Agent Function

This is a map from the precept sequence to an action.

### 3.4.2 Rationality

Rationality is a feature that inculcates a sense of responsibility, sensibility and judgment. It empowers the agent to perform expected actions after perceiving.

The agent must perform all the actions to maximize its performance measure with respect to the percept sequence and the knowledge base that it has. Therefore, we can say that rationality of an agent depends on the following:

1. **Agent's performance** measure that gives the degree of success.
2. **Agent's percept sequence** received so far.
3. Agent's **prior knowledge** about the environment.
4. Agent's **actions** that can be performed.

A rational agent (refer Fig. 3.3) always performs the right action in the given percept sequence to maximize performance. Thus, a problem solved by an agent is characterized by performance measure, environment, actuators, and sensors (PEAS).

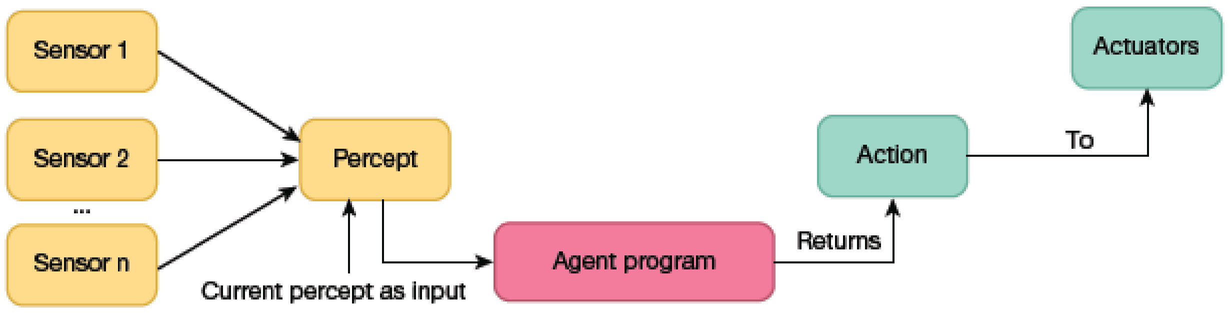


**FIGURE 3.3** Rational Agent

### 3.4.3 Structure of Intelligent Agents

The structure of an intelligent agent can be given as,

***Agent = Architecture + Agent Program***

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

### 3.4.4 Types of Agents

In this section, we will discuss different types of agents.

## Simple Reflex Agents

Simple reflex agents choose actions based only on the current percept. They are rational only if they make a correct decision based on the current precept. Their environment is completely observable.

They work using **condition-action rule** that maps a state (condition) to an action. If the condition is true, then the action is taken, else not. However, the problem with a simple reflex agent is that they succeed only when the environment is fully observable. If the environment is partially observable, then the agent may get stuck in infinite loops. In such a situation, the agent can escape the loop only if it can randomize its actions.

Other issues with such agents are given below.

1. Possess very limited intelligence
2. Has knowledge of any state other than the current state.
3. Any change in the environment will eventually end up updating the collection of rules.

## Model-Based Reflex Agents

Model reflex agents (refer Fig. 3.4) use a model of the world to choose their actions. For this, they need to maintain an internal state. Here, **model** means knowledge about 'how things happen in the world'.

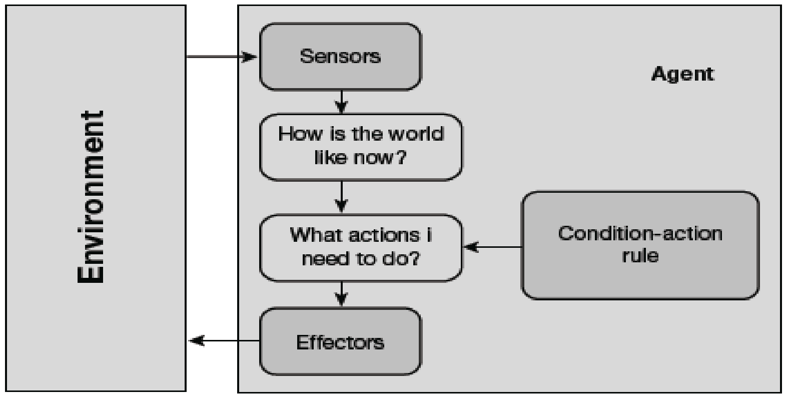


**FIGURE 3.4** Model reflex agent

**Internal state** represents unobserved aspects of the current state depending on percept history.

***To update the state, information is required to understand,***

- How the world evolves and
- How the agent's actions affect the world.

## Goal-Based Agents

Goal means a description of desirable situations. Goal-based agents (refer Fig. 3.5) choose their actions to achieve goals. This provides more flexibility than reflex agent as the knowledge supporting a decision is explicitly modeled and permits any sort of modifications.

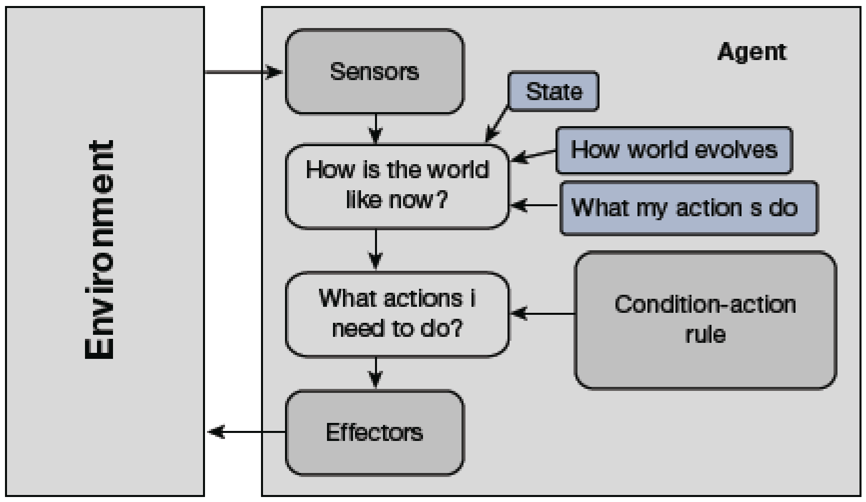


**FIGURE 3.5** Goal-based agent

## Utility-Based Agents

Many a time, goals are either conflicting or are difficult to achieve. In such situations, utility-based agents (refer Fig. 3.6) can be used to achieve goals that are more important than others. These agents choose actions based on a preference (utility) for each state.

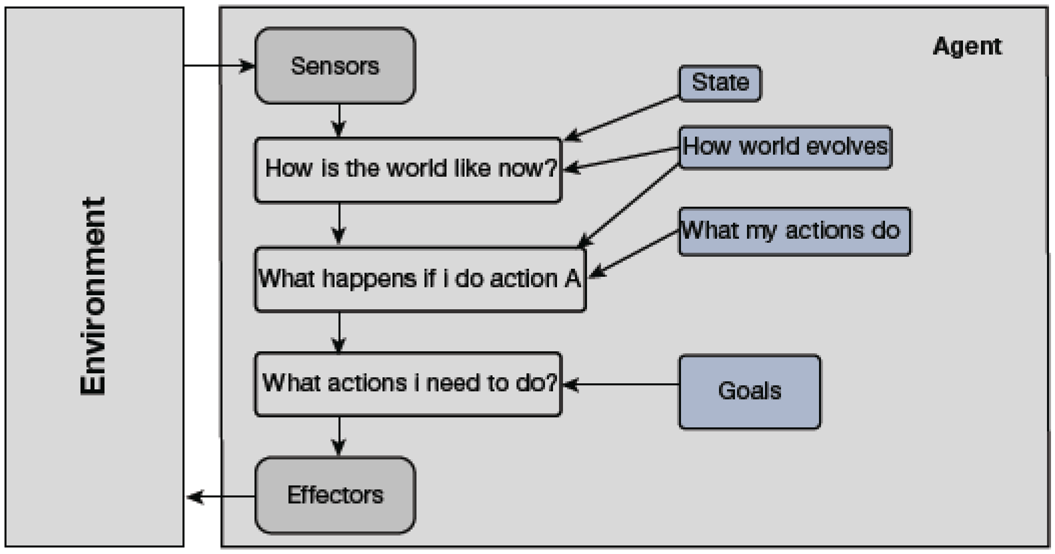

Environment
Agent
Sensors
State
How is the world like now?
How world evolves
What my actions do
What happens if i do action A
What actions i need to do?
Goals
Effectors

**FIGURE 3.6** Utility-based agents

However, at times, achieving the desired goal is not enough and happiness of the agent is also important. In such a scenario, utility describes how **'happy'** the agent is. A utility agent chooses the action that maximizes the expected utility, that is, the associated degree of happiness.

## Learning Agent

A learning agent (refer Fig. 3.7) learns from its past experiences. It is designed to possess learning capabilities. Initially, it starts working with basic knowledge and then acts and adapts automatically through learning.

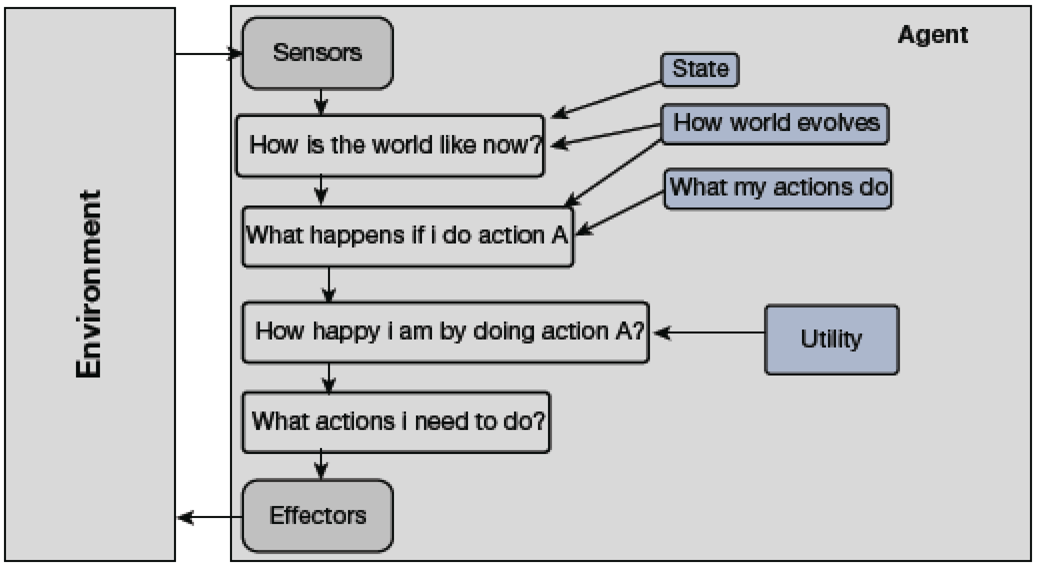

Environment
Agent
Sensors
State
How is the world like now?
How world evolves
What my actions do
What happens if i do action A
How happy i am by doing action A?
Utility
What actions i need to do?
Effectors

**FIGURE 3.7** Learning agent

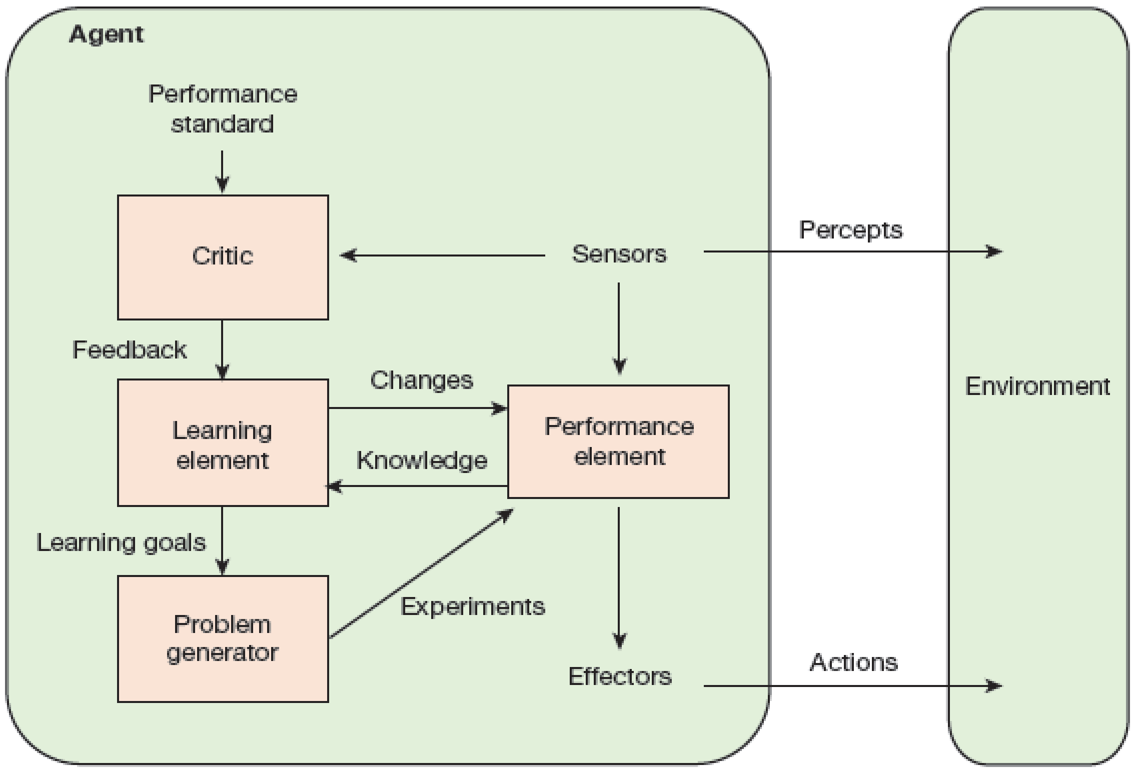
Such an agent has four main conceptual components, as follows:

**Learning element,** which is responsible for making improvements by learning from the environment

**Critic** (refer Fig. 3.8) is the element that provides feedback to the learning element, describing how well the agent is doing with respect to a fixed performance standard.

**Performance element** chooses a particular external action to be taken.

**FIGURE 3.8** Critic



**Problem generator** suggests actions that result in new and informative experiences.

## 3.4.5 The Nature of Environments

When building an artificially intelligent system, some programs would be confined to keyboard input, database, computer file systems and character output on a screen while other software agents (like software robots or softbots) would operate in unlimited domains. The simulator for a softbot has a **very detailed, complex environment**. The software agent must take

a specific action by choosing from a long array of actions in real time.

For example, a softbot whose job is to scan customer preferences and display them relevant items works in the **real** as well as an **artificial** environment.

The most famous **artificial environment** is the **Turing Test environment**, in which artificial agents are tested with a real agent (on equal ground). The software agent is trained to perform as well as a human.

### Turing Test

Turing test is widely used to determine the success of an intelligent behaviour of a system. Two persons and a machine or intelligent system or software agent to be evaluated participate in the test. One of the two persons becomes the tester. The other person and the software agent are then made to sit in different rooms. The tester does not know who is the machine and who is a human. He asks them questions by typing and sending them to both, and receives typed responses.

The aim of this test is to fool the tester. If the tester fails to determine whether the reply is coming from a machine or a human, then the machine is said to be intelligent.

## 3.4.6 Types of Environments

As stated earlier, environment is the place where the agent would work. It gives rewards and specifies states and actions of the agents. Although environment in an AI system is quite vast, we can identify a small number of dimensions to categorize them as given below.

### Discrete/Continuous

If an environment has a limited number of distinct, clearly defined, states, then it is said to be discrete. For example, a software agent designed to play chess game works in a discrete environment as there are limited moves that a player can make. However, a self-driving car operates in a continuous environment.

While a discrete environment has a finite number of percepts and actions that can be performed within it, a continuous environment has no such constraints.

## Known Vs Unknown

Known and unknown are based on an agent’s state of knowledge to perform an action and they are not actually a feature of an environment. In a known environment, the agent knows the results for all actions but in an unknown environment, the agent has to learn how it should work to perform an action. Usually, a known environment is partially observable and an unknown environment is fully observable.

While known environments are good examples of **exploitation, unknown environments are best for exploration. Reinforcement learning** is extensively applied in an unknown environment.

## Observable/Partially Observable

If an agent can determine the complete state of the environment at each time point from the percepts, then it is said to be observable; otherwise, it is only partially observable. Correspondingly, in an unobservable environment, the agent has no sensors. In a fully observable environment, the complete state of the environment *relevant to the choice of action* of the agent can be captured using its sensors.

While a fully observable environment (refer Fig. 3.9) does not require to maintain the internal state to keep track history of the world, a not fully observable environment, on the other hand, must maintain an internal state to keep track of the world. An environment with sensors can be partially observable because of noise, inaccuracy of the sensors, due to the framework of the task itself or because parts of the state are missing from the sensor data.

For example, the classic chess game is fully observable as one agent can perceive the positions and the moves of the other agent. But in the **Kriegspiel** version of chess, the game environment is partially observable as only invalid moves can be observed.

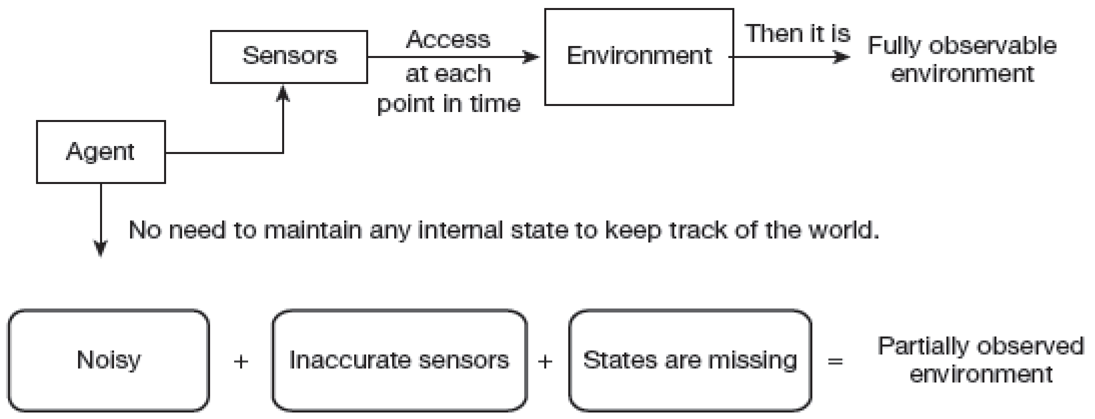

Sensors
Access at each point in time
Environment
Then it is
Fully observable environment
Agent
No need to maintain any internal state to keep track of the world.
Noisy
+
Inaccurate sensors
+
States are missing
=
Partially observed environment

**FIGURE 3.9** Observable/Partially Observable Environment

### Static/Dynamic

A static environment does not change when an agent is acting, but a dynamic environment does. Agents in static environments, therefore, do not take into account the state of the environment during the action and are thus easy to deal with. However, in a dynamic environment, agents must consider the world while performing each action. For example, self-driving cars work in a dynamic environment, but games like crossword puzzles and chess operate in a static one.

Between these two extremes, we can also have **semi-dynamic environments** which do not change with the passage of time but the agent's performance score does.

### Single Agent/Multiple Agents

The environment may contain other agents which may or may not be of the same kind of the agent. For example, vacuum cleaning environment is a single agent but a chess game is a two-agent environment.

In a multi-agent environment, the performance measure of one agent depends on the action performed by another agent. A multi-agent environment (refer Fig. 3.10) may be competitive or cooperative. For example, chess is an example of a

**competitive** environment and two taxi-driving agents must avoid collisions and work to maximize the performance of both the agents.

Thus, it operates in a **cooperative** environment.

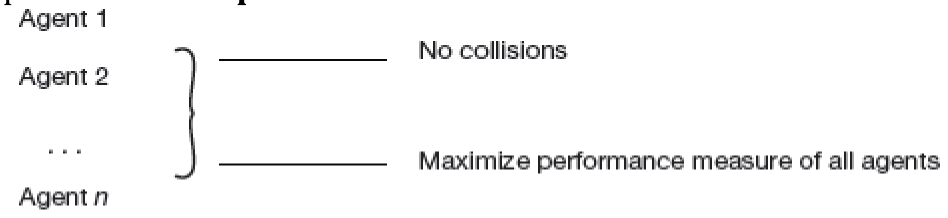


**FIGURE 3.10** Multi-agent environment

## Accessible/Inaccessible

In an accessible environment, the sensory apparatus of the agent has access to the complete state of the environment. This means that it can obtain complete and accurate information about the state's environment, but this is not so in case of an inaccessible environment.

For example, an empty room whose state can be defined by its temperature is an example of an accessible environment, whereas information about an event on the Earth is an inaccessible environment.

## Deterministic/Non-deterministic

In a deterministic environment (refer Fig. 3.11), the next state of the environment can be easily determined by the current state as well as the actions of the agent. This is not true for a non- deterministic environment. A n**on-deterministic** environment description maximizes an agent's performance for all possible outcome of its actions. Note that in a deterministic, fully observable environment, the agent does worry about uncertainty.

In a **Stochastic** environment, uncertainty about outcomes is quantified in terms of **probabilities**.

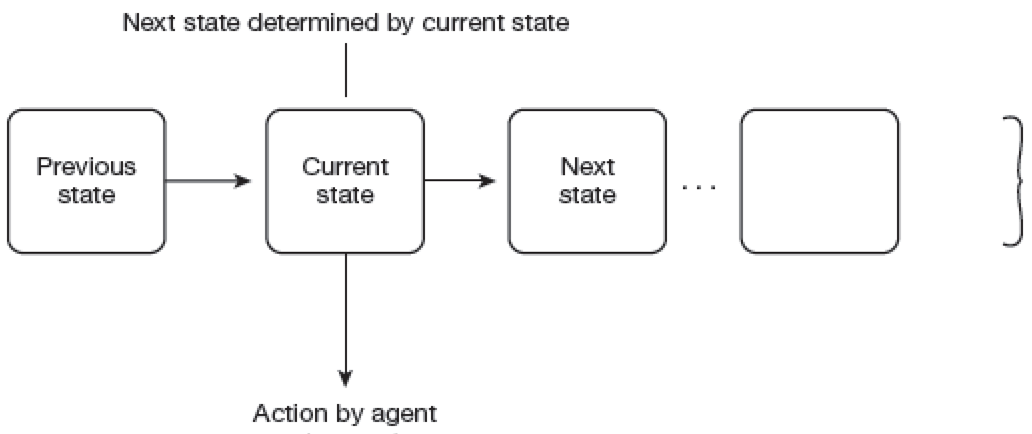


**FIGURE 3.11** Deterministic/Non-deterministic Environment

For example, the game of ludo is non-deterministic as the dice will generate a number randomly, creating uncertainty in the environment. On the contrary, chess can be considered a deterministic environment to a certain extent as the state of the game can be determined by estimating the other agent's moves although there can be uncertainty due to the other agent in the game. Most real-life situations are so complex that it is not possible to track of all the unobserved aspects. In such cases, we can consider them as deterministic.

## Episodic/Non-episodic

In an episodic environment (refer Fig. 3.12), each episode has an agent perceiving and then acting. The quality of its action depends just on the episode itself. Subsequent episodes do not depend on the actions performed in the previous episodes.

Episodic environments are, therefore, simpler as the agent does not need to think ahead. There are however, a series of one- shot actions, and only the current percept is required for the action. On the other hand, in a sequential or non-episodic environment, an agent requires memory to store past actions so that it can determine the next best actions. Therefore, the current decision can affect all future decisions.

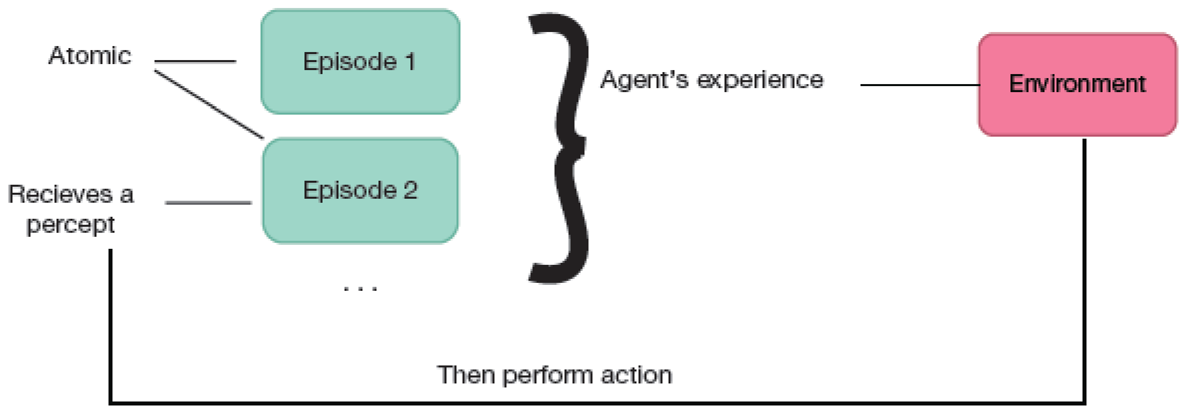


**FIGURE 3.12** Episodic/Non-episodic Environment

**Review Questions:**

**1.What is Agent?**

**2. What are the types of Agent?**

**3. What are the type of Environment ?**

**L6:**

## 3.5 Search Algorithms

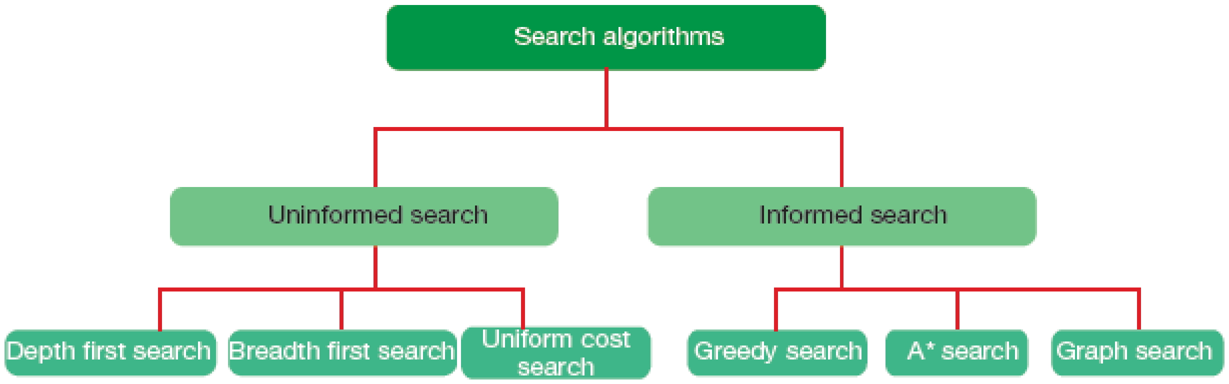
We have read that artificial Intelligence is the study of building rational agents. These agents make use of some or the other search algorithms in the background to achieve their tasks. For example, some single-player games like tile games, Sudoku, crossword, etc., use search algorithms to deduce a particular position in the game.

***A search problem consists of the following:***

1. **State space** is the set of all possible states that can be attained by an agent.
2. **Start state** is the state from where the searching is done.
3. **Goal test** is a function that checks if the current state is the goal state or not.

4. **Solution** to a search problem is a sequence of actions (also known as **plan)** that transforms the start state to the goal state. This plan is realized using search algorithms.

### 3.5.1 Types of Search Algorithms



There are many search algorithms but, in this section, we will discuss six of the most popularly used ones. For better understanding, we will categorize them under two headings— informed and uninformed as shown in Fig. 3.13.

**FIGURE 3.13** Different Types of Search Algorithms

### 3.5.2 Properties of Search Algorithms

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

## 3.6 Uninformed Search Algorithms

A uninformed search, also known as a blind search algorithm, has no additional information about the goal state other than the one provided in the problem definition. The only information they have is on how to traverse or visit the nodes in the tree.

In such algorithms, the machine blindly follows the technique irrespective of whether it is right or wrong, efficient or inefficient. There can be multiple plans to reach the goal state from the start state. All these paths differ by the order and/or length of actions.

In this section, we will read about depth first search, breadth first search and uniform cost search algorithms. Each of these algorithms has the following information.

**Problem graph,** the start node S to the goal node G.

**Strategy,** stating the path followed by the graph reach G.

**Fringe,** which is a data structure to store all the possible states (nodes) that can be reached from the current state.

**Tree,** that depicts path while traversing to the goal node.

**Solution plan, that** specifies the sequence of nodes from S to G.

**Path/Step cost** are integers that represent the cost to move from one node to another node.

### 3.6.1 Depth First Search (DFS)

In this search algorithm, the tree is traversed from the root node. The key will be searched till the leaf node of a particular branch. If the key is not found, then the search process backtracks to the point from where the other branch was left unexplored. This process is repeated for that other branch either until the key is found or the entire tree is completely traversed.

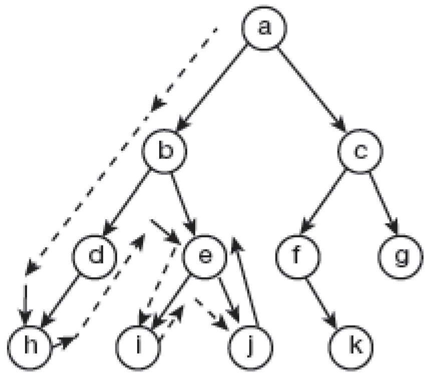


**FIGURE 3.14** Depth-first search

In Fig. 3.14, we can clearly visualize the working of a DFS algorithm. Here, searching starts from the root node A and then traverses nodes B, D and H. Now, H is the leaf node, so we retrace the path to reach node B to traverse its unexplored branch. Nodes E and I are thus traversed. Now, I is the leaf node, so again, the path is retraced to follow the unexplored branch of node E. As a result, node J is visited and then it is found that all nodes in branches of node B have been traced. So, the algorithm will move further by exploring the other untraced branch of the root node. Here, node C, followed by nodes F, K and G are explored.

From these observations, we can conclude that DFS algorithm occupies a lot of memory space, and takes a lot of time to execute when the solution is at the bottom or end of the tree. DFS is implemented using LIFO, that is, stack data structure.

The time complexity of Depth-First Search (DFS) is O(V + E), where V is the number of vertices and E is the number of edges, because it visits each vertex and edge at most once. The space complexity of DFS is O(V) because it uses a stack (either the call stack for recursive DFS or an explicit stack for iterative DFS) to keep track of the path, and in the worst case, this stack can hold all the vertices in the graph

**Though the** DFS algorithm is complete if the search tree is finite and a solution exists, **it** is not optimal as the number of steps in reaching the solution, or the cost spent in reaching it is high.

## Advantages

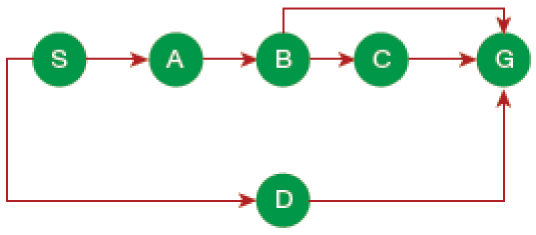
1. It uses less memory as it stores only the nodes on the path from the root node to the current node.
2. It takes less time to reach the goal node/state as compared to the BFS algorithm.

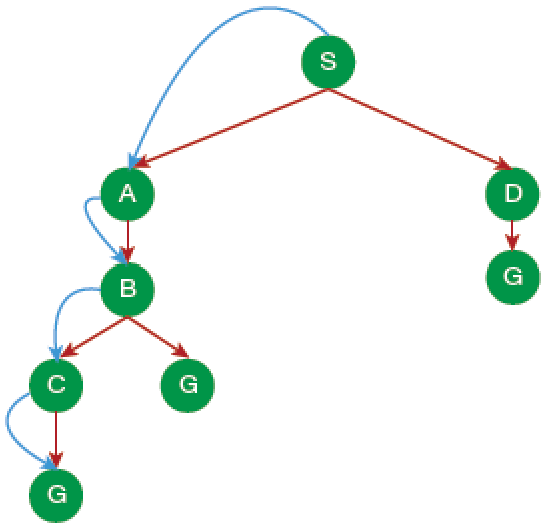
## Disadvantages

1. Sometimes, many states keep recurring. In such a scenario, there is no guarantee of finding the solution.
2. In certain cases, the algorithm may get stuck in an infinite loop while going deep searching for nodes in the tree. A possible solution to this problem is to choose an appropriate value of cut-off depth. If the ideal cut-off is *d*, and the chosen cut-off is less than *d*, then algorithm fails. But, if the chosen cut-off is more than *d*, the execution time increases.
3. Complexity of the algorithm depends on the number of paths.
4. DFS cannot check duplicate nodes.

**Example:** Explore the path that will be explored using the DFS algorithm to reach node G from S.

DFS creates the same set of nodes as breadth-first method, different order.

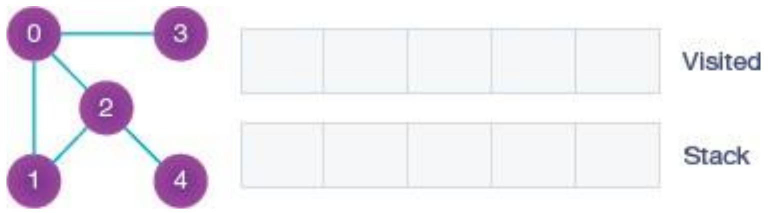




**Solution:** To explore the path, a search tree for the graph will be created. Since DFS traverses the tree using the 'deepest node first' technique, it would always pick the deeper branch until it reaches the solution or until all nodes of the current branch have been traversed. The traversal is shown in blue arrows.

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

**Example:** Consider the graph given below and state its DFS traversal.



**Solution:** We start with node 0. Exploring its branch to node 1, we move to node 2 and node 4. From node 4, we backtrack to node 3. Hence, the DFS path can be given as, 0 -> 1 -> 2 -> 4 -> 3.

### 3.6.2 Depth-Limited Search Algorithm (DLS)

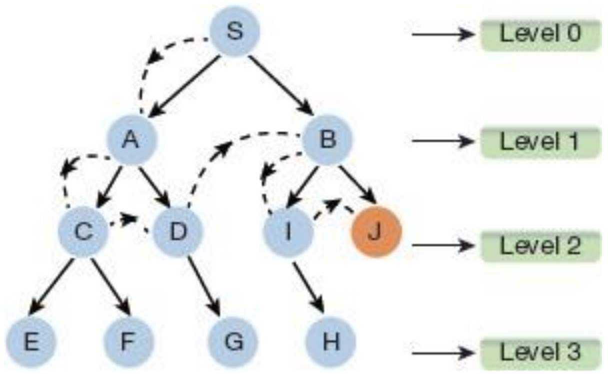
DLS is similar to the DFS algorithm with an addition of a predetermined limit. This limit helps to overcome the limitations of the infinite path in the DFS. In DLS, the node at the depth limit will be treated as it is a leaf node (or a node with no successor nodes).

The DLS algorithm is terminated in any of the three cases:

1. First, if the problem does not have any solution. This is known as standard error failure.
2. Second, if the problem does not have any solution within a given depth limit. This is known as cut-off failure.
3. Third, when the solution is found.

For example, in the tree given in Fig. 3.15, if level is limited to 2, then the search technique will not go to level 3. Therefore,

nodes E, F, G and H will not be traversed.



Thus, the DLS search algorithm is complete if the solution is present within the specified depth limit. While time complexity of DLS algorithm is **$O(b^{\ell})$**, space complexity is O**$(b \times \ell)$**, where l is the depth limit and d is the depth of the tree.

**FIGURE 3.15** Depth limited search

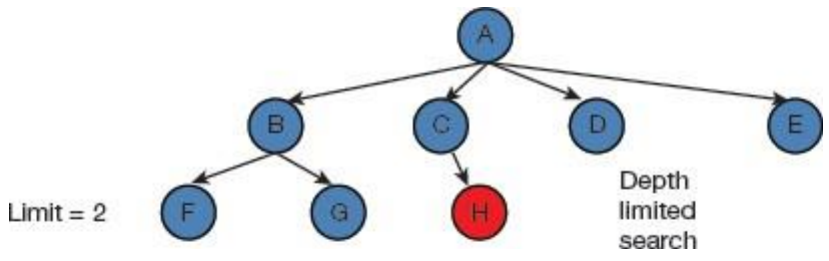
## Advantages

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

## Disadvantages

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

**Example:** Traverse the given tree to search node H using DLS with predefined limit as 2.



**Solution:** We start with Node A at level 0.

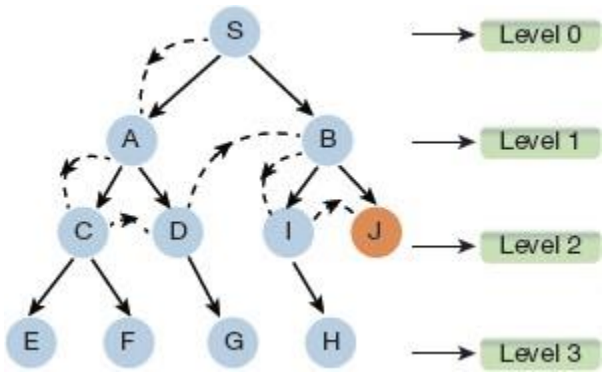
The search process then continues to explore nodes B, C, D, and E bat level 1.

Now, we go to level 2 and explore the child nodes of node B. When H is not found there, a backtrack is done to level 1.

Child nodes of node C are searched to find node H. Finally, node H is found at level 2 as a child node of C and the algorithm terminates.

**Example:** Traverse the given tree to search node H using DLS with predefined limit as 2.

**Solution:** In the tree, node H is not present till level 2, so the algorithm terminates returning a cutoff failure. The path traversed for searching is marked with a dashed line.



This image explains the DLS implementation and could be referred to for better understanding.

### 3.6.3 Breadth First Search (BFS)

In this algorithm, nodes in the tree are traversed breadthwise to reach the goal node. Searching begins from the root node and expands the successor node breadthwise before traversing depth-wise.

As evident from Fig. 3.16, a BFS algorithm starts from the root node A and then traverses node B followed by node C. This means that children of the root node are traversed before exploring other nodes on any branch. After traversing all nodes at a particular level, control then passes to the next level.

Therefore, after node C, all nodes from D to G are traversed. This is followed by traversal of nodes H to K. Thus, the BFS algorithm explores all neighbour nodes at the present depth before moving the nodes at the next depth level.

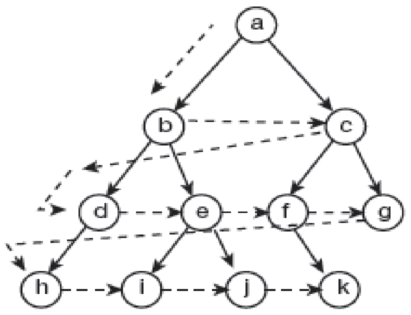


**FIGURE 3.16** Breadth-first search

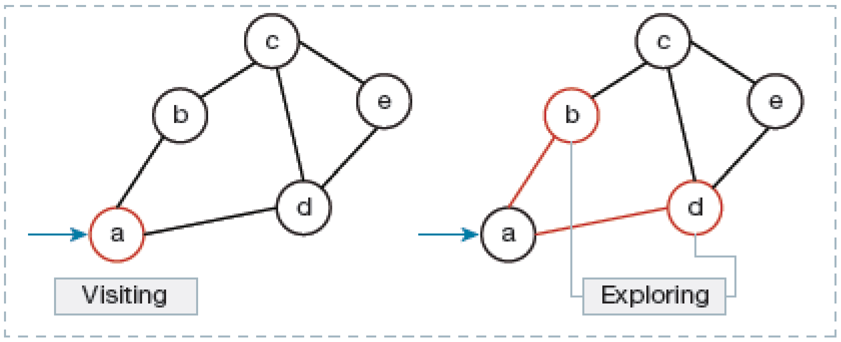


**FIGURE 3.17 BFS Tree starting from a random node**

In Fig. 3.17, BFS algorithm selects a random initial node (also known as source node or root node) and traverses the graph layer-wise in such a way that all the nodes and their respective children nodes are visited and explored. Here,

**Visiting a node** means to visit or select a node. And,

**Exploring a node** means exploring the adjacent nodes (child nodes) of the selected node.

## Advantages

1. BFS will find a solution if it exists.
2. If a given problem has multiple solutions, then BFS will give the minimal solution requiring least number of steps.
3. The architecture of the BFS algorithm is simple and robust.
4. It constructs the shortest path of traversing through the nodes of the graph so that the graph can be traversed in the smallest number of iterations.
5. The algorithm does not get stuck in an infinite loop.

## Disadvantages

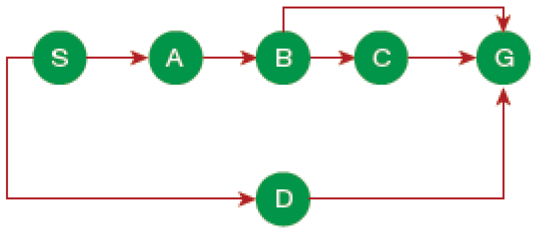
1. Since each level of node is saved for creating the next one, it consumes a lot of memory space. Space requirement to store nodes is exponential.
2. It takes less time if the solution is far away from the root node—at the

bottom or at the end.Its complexity depends on the number of nodes.

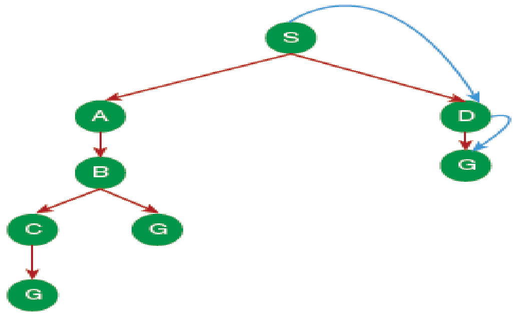
The BFS algorithm is implemented using a FIFO queue. It is a complete algorithm, that is, it finds a solution if it exits. The algorithm is optimal, if step cost = 1 (that is, if either there is no cost or if all step costs are same). Moreover, it has both time and space complexity given by, $O(b^d)$. Here, b is the branching factor and d is the depth of the tree. Note that total number of nodes created in the worst case is $b + b^2 + b^3 + \ldots + b^d$.

*While time complexity is equivalent to the number of nodes traversed in BFS s***pace complexity, on the other hand, it is e***quivalent to how large can the fringe get.* Due to high precision and robust implementation, BFS is used in multiple real-life solutions like P2P networks, Web Crawlers and Network Broadcasting.

**Example: Find the** BFS traversal from node S to node G.



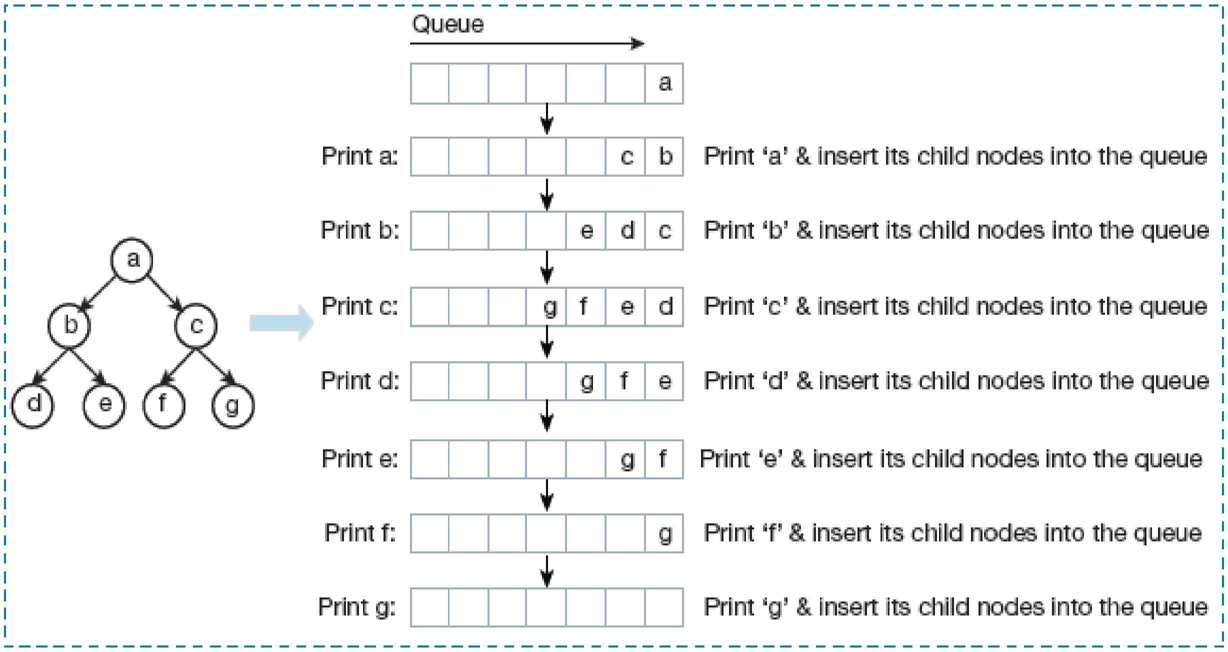
**Solution:** The BFS creates a tree and traverses it using the principle



'shallowest node first'. So, at note S, node D will be traversed followed by node G. Thus, **Path is** S -> D -> G

*In a BFS algorithm, nodes in the path = the depth of the shallowest solution = number of nodes in level .*

**Example:** Consider the graph. Using node A as the source node, traverse the graph and trace the working of the algorithm using a queue.


Queue
a
Print a:
c b
Print 'a' & insert its child nodes into the queue
Print b:
e d c
Print 'b' & insert its child nodes into the queue
Print c:
g f e d
Print 'c' & insert its child nodes into the queue
Print d:
g f e
Print 'd' & insert its child nodes into the queue
Print e:
g f
Print 'e' & insert its child nodes into the queue
Print f:
g
Print 'f' & insert its child nodes into the queue
Print g:
Print 'g' & insert its child nodes into the queue
a
b
c
d
e
f
g

***Step 1:*** Insert root node 'a' into the Queue.

***Step 2:*** Remove node 'a' from the queue, print it and insert the child nodes of 'a' in the queue. Thus, nodes 'b' and 'c' are inserted.

***Step 3:*** The queue is not empty and has node 'b' and 'c'. Since 'b' is the first node in the queue, remove it, print it and insert the child nodes of 'b' into the queue. That is, insert node 'd' and 'e'.

Repeat these steps until the queue gets empty. Do not insert those nodes in the queue that are already visited.

**Breadth-First Search Algorithm Pseudocode**

```
Input: s as the source node
BFS (G, s)
let Q be queue.
Q.enqueue( s )
 mark s as visited
while ( Q is not empty)
    v = Q.dequeue( )
    for all neighbours w of v in Graph G
        if w is not visited
            Q.enqueue( w )
            mark w as visited
```

According to the pseudocode, s is the root node of the graph, G. initially, s is inserted in the queue. Then, all child nodes of ‘s’ are marked. These child nodes are visited after removing ‘s’ from the queue. At each step of the algorithm, child nodes, w are inserted into the queue to further visit its child nodes. The process is repeated under the queue is empty.

## Applications of Breadth-First Search Algorithm

BFS is a simple graph traversal method that is widely used in the following areas.

### Crawlers in Search Engines

BFS algorithm is used for indexing web pages. In this application, each web page is considered as a node in a graph.

The algorithm starts traversing from the source page and follows all the links associated with that page.

### GPS Navigation Systems

GPS systems use BFS algorithm to find neighbouring locations.

### Find the Shortest Path for an Unweighted Graph

For an unweighted graph, the shortest path can be easily calculated by choosing a path with the least number of edges. BFS algorithm can do this by traversing a minimum number of nodes starting from the source node. Similarly, either breadth- first search or depth-first traversal methods can be used to find a spanning tree.

### Broadcasting

In computer networks, data is broken down into small packets before transmission across the communication media. These packets use an algorithm to compute a path to the destination. For messages that are broadcasted across all the nodes in a network, BFS is a preferred choice.

### Peer-to-Peer Networking

BFS is used in peer-to-peer network as a traversal method to find all the neighbouring nodes. For example, BitTorrent uses BFS for peer-to- peer communication.

**Review Questions:**

**1.What is BFS?**

**2. What are the types of algorithms?**

**3. What is DFS ?**

**4. What is DLS ?**

## L7:

### 3.6.4 Uniform Cost Search (UCS)

The UCS algorithm is used to find an optimal solution to the goal state when the step costs are not the same. In such a scenario, the algorithm computes the cumulative cost to expand each node from the root node to the goal node. It neither traverses depth nor breadth, but searches for the next node with the lowest cost. Sorting is done in increasing cost of the path to a node so that the algorithm can explore paths in the increasing order of cost. UCS always expands the node with the least cost.

It is identical to BFS if each transition has the same cost. Remember that,

**Cost of a node** is defined as:

*cost(node) = cumulative cost of all nodes from root*

cost(root) = 0

In Fig. 3.18, if S is the start node and G is the goal state, then node A is explored as it has the lowest cost (the other node G has a higher cost). Now, node A has two child nodes—B and C. C having the lowest step cost is traversed and from there, node D, followed by G is reached. Though UFS performs well for this algorithm, it may not work with all cases.

Thus, we see that uniform-cost search (UCS) expands nodes based on their path costs form the root node. It can be used to traverse any graph/tree where the optimal cost is the traversal criteria. The uniform-cost search algorithm is implemented by the priority queue, giving maximum priority to the lowest cumulative cost. If you observe carefully, you will find that UCS is equivalent to BFS algorithm if the path cost of all edges is the same.

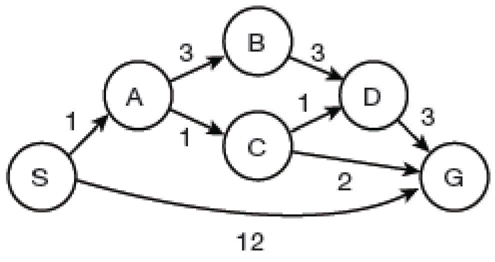


**FIGURE 3.18** Uniform cost search

The UCS algorithm is a complete and optimal algorithm with time and space complexity and space complexity O(b$^{(c/ε)}$), where ε is the lowest cost and c is the optimal cost.

## Advantages

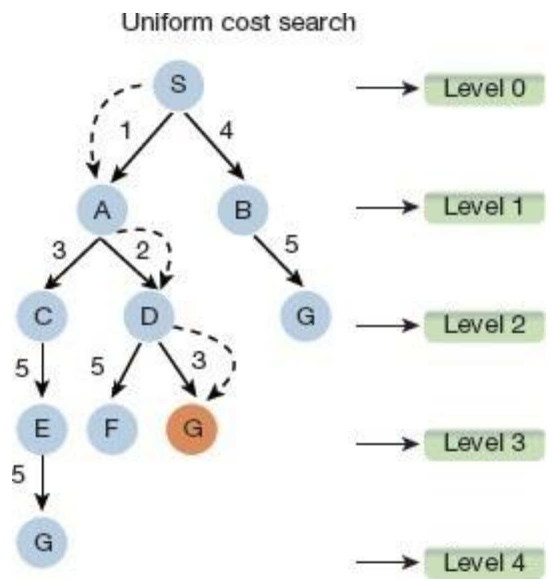
1. It finds an optimal solution by considering the least cost at every state.

2. UCS is complete only if states are finite and if there is no loop with zero weight.
3. UCS is optimal only if there is no negative cost.

## Disadvantages

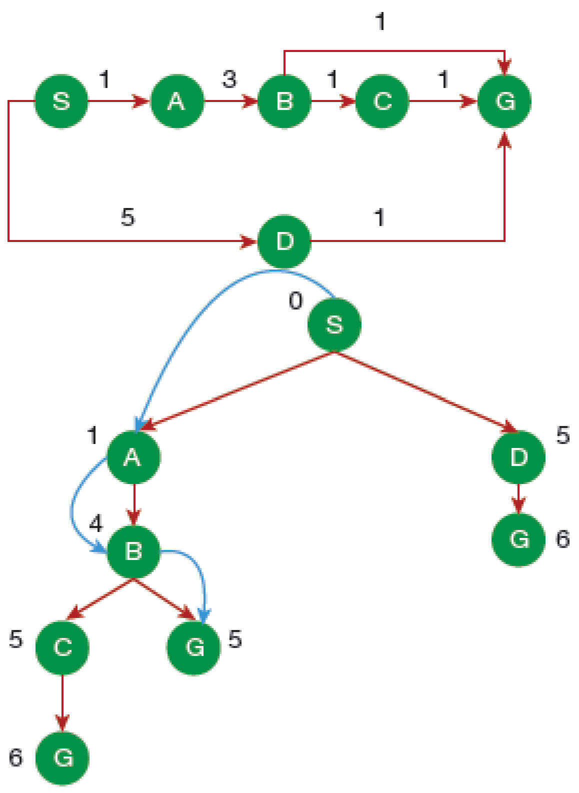
1. The algorithm may get stuck in an infinite loop as it considers only cost and not the number of steps taken to reach the goal state.
2. It explores nodes in every 'direction'.
3. It does not have any information about the location of the goal state.
4. It requires more space for storing information about nodes.
5. The UCS must explore all paths including the long ones.

**Example:** Consider the tree given below and its UCS traversal to reach node G from S.



**Example: Find the path and cost** to move from node S to node G in the graph given below.

**Solution:** For a better clarity, we can draw an equivalent search tree for the graph. Using UCS, the path with the least cumulative cost is chosen.



**Path:** S -> A -> B -> G

**Cost:** 5

### 3.6.5 Iterative Deepening Depth-First Search (IDDFS)

The iterative deepening algorithm combines the features of DFS and BFS algorithms to find out the best depth limit that can be used to implement the DLS algorithm. IDDFS does so by gradually increasing the limit until the goal is found.

The algorithm performs depth-first search to level 1, starts over, executes a complete depth-first search to level 2 and continues until the solution is found. It never creates a node until all lower nodes are generated. It only saves a stack of nodes.

IDDFS gets benefitted by fast search technique of BFS algorithm and memory efficiency of the depth-first search algorithm. This algorithm is widely used when search space is large, and depth of the goal node is not known.

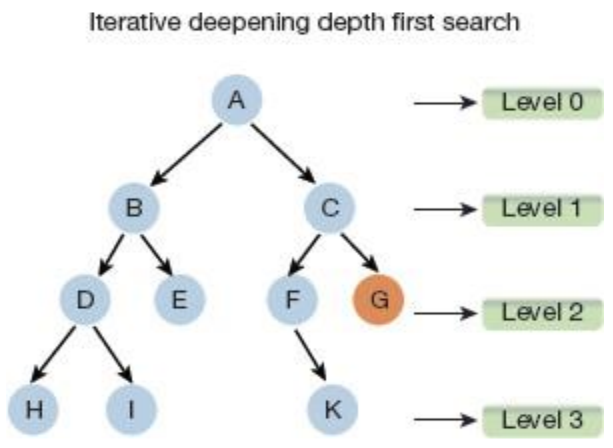
## Advantages

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

## Disadvantages

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

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



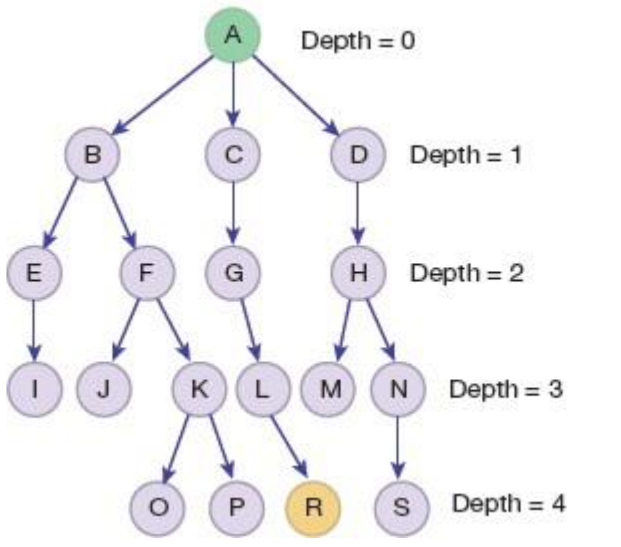
**Solution:** In the first iteration, node A at level 0 is explored. In the second iteration, nodes B and C are traversed at level 1. In the third iteration, nodes D, E, F and G are traversed. In the fourth iteration, node H is reached.

The algorithm ends when it finds a solution at depth *d*. The number of nodes created at depth *d* is $b^{d}$ and at depth *d-1* is $b^{d-1}$.

**Time complexity** of IDDFS algorithm is $O(b^{d})$ and its space complexity is O(bd), where b is the branching factor and depth of the tree is d.

In IDDFS, we perform DFS up to a certain 'limited depth,' and keep increasing this 'limited depth' after every iteration.

**Example:** Consider the tree given below and demonstrate the application of IDDFS.



The tree can be visited as: A B E F C G D H

Depth = {0, 1, 2, 3, 4}

| Depth limits | IDDFS |
|---|---|
| 0 | A |
| 1 | A B C D |
| 2 | A B E F C G D H |
| 3 | A B E I F J K C G L D H M N |
| 4 | A B E I F J K O P C G L R D H M N S |

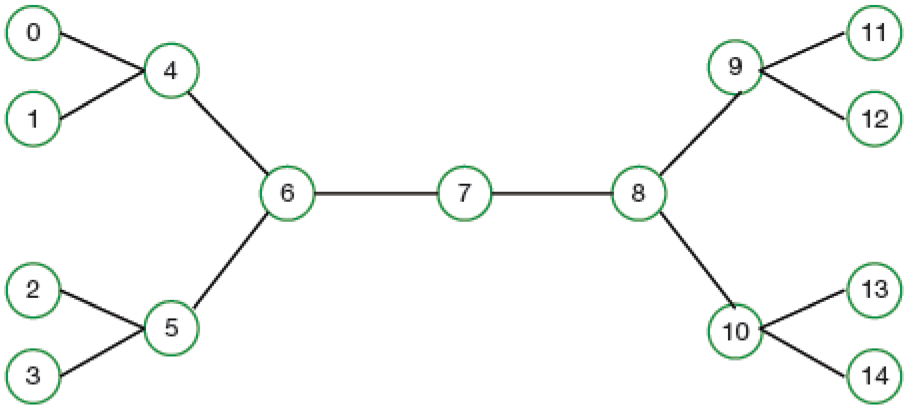
### 3.6.6 Bidirectional Search

In a bidirectional search, searching is done **from both the directions simultaneously. It searches forward from initial state and backward from goal state till both meet to identify a common state.** The path from initial state is concatenated with the inverse path from the goal state. Each search is done only up to half of the total path.

**The a**lgorithm finds the smallest path from the source node to the goal node. Thus, bidirectional search replaces single search graph with two smaller sub graphs—one starting from initial or start node to the goal node (termed as forward search) and the other starting from goal node towards the source node (known as backward search).

**The search process terminates when the two graphs intersect.** Bidirectional search can be guided by a heuristic estimate of remaining distance from source to goal and vice versa. Heuristic means finding the shortest path from the current node in the graph to the goal node.

**Example:** Consider the graph given below and apply bidirectional search on it to reach goal node 14 from source node 0.



**Solution:** Two searches are executed simultaneously-, one from node 0 and the other from node 14. Both these searches intersect at node 7. At this point, we have found a path from node 0 to 7 and from node 14 to 7. The search process terminates successfully, thereby avoiding unnecessary exploration.

## Advantages

1. It is faster than other algorithms
2. **It avoids unnecessary** exploration of nodes
3. Time and space complexity of searching is $O(b^{d/2} + b^{d/2})$, which is far less than $O(b^{d})$. Here, **b** is the branching factor of the tree and d is the distance of goal node from the source node.
4. The algorithm drastically reduces searching time by supporting simultaneous searches.
5. It takes less memory capacity to store all the searches.
6. Bidirectional search is complete if BFS is used in both searches.
7. The algorithm is optimal if BFS is used for search and paths have uniform cost.

## Disadvantages

1. The algorithm can be used only when the goal state is clearly known.
2. It is difficult to implement.

3. The algorithm must be robust enough to correctly deduce the intersection point where it can terminate. Otherwise, the algorithm may get stuck in an infinite loop.
4. It is very difficult to search backwards through all states.

## When to Use Bidirectional Approach?

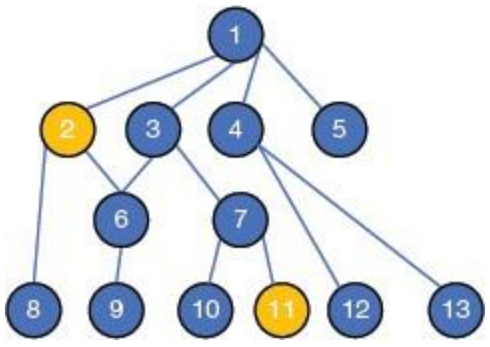
Bi-directional search can be best used in the following situations:

- When both the starting node and the goal node are unique and completely defined.
- When the branching factor is exactly the same in both directions.

**Example:** Consider the graph given below and demonstrate the application of bidirectional search from Start forward search from node 2 and backward search from node 11.

Do BFS from both directions.

In the forward search process, nodes 1, 6 and 8 are explored. In the backward search process, mode 7 is explored. But no intersection node is encountered.



Next explore node 3 while doing a forward search and node 3 while doing a backward search. Here, we find intersecting node as node 3, so the algorithm terminates and the path is- 2 -> 1 -> 3-> 7 -> 11.

## 3.7 Informed Search Algorithms

The uninformed search algorithms that we have learnt so far had no knowledge about search space. But informed search algorithm, as the name suggests, contains some information about how far we are from the goal, cost of the path, how to reach the goal, etc. This knowledge helps agents to explore less to the search space and reach the goal node more efficiently.

The informed search algorithm which is extensively used in a large space is also known as heuristic search as it uses a heuristic function. The heuristic function takes the current state of the agent as its input and estimates how close the agent is from the goal. Though the heuristic method may not always give the best solution, it is, however, guaranteed to find a good solution in reasonable time.

A heuristic function h(n), calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive. We can write,

$h(n) <= h^\star(n)$

**where h(n) is heuristic cost, and $h^\star(n)$ is the estimated cost.**

**This means that the heuristic cost should be less than or equal to the estimated cost.**

### 3.7.1 Pure Heuristic Search

To solve large problems with a large number of possible states, problem-specific knowledge needs to be added to increase the efficiency of search algorithms. In such a situation, heuristic search algorithm performs well.

Heuristic search is the simplest form of heuristic search algorithms that expands nodes based on their heuristic value
h(n). The algorithm maintains two lists—OPEN and CLOSED. All nodes that have already been expanded are placed in the CLOSED list, while others that have not

been expanded are in the OPEN list.

On each iteration, nodes with the lowest heuristic value are expanded. Then, the heuristic function is applied to the child nodes and they are placed in the open list according to their heuristic value. The shorter paths are saved and the longer ones are disposed. The process continues unit a goal state is found.

In this section, we will discuss two important informed search algorithms—b**est first search algorithm (greedy search) and A* search algorithm.**

### 3.7.2 Best-First Search Algorithm (Greedy Search)

Greedy best-first search algorithm combines depth-first search and breadth-first search algorithms and always selects the path which appears best at that moment. It uses the heuristic function and best-first search at each step to choose the most promising node. The most promising node is the one that is closest to the goal node. The greedy best first algorithm is implemented by the priority queue. Heuristic evaluation functions calculate the cost of optimal path between two states.

***Best first search algorithm steps:***

**Step 1:** Insert the starting node in the OPEN list.

**Step 2:** Stop and return failure if the OPEN list is empty,

**Step 3:** From the OPEN list, remove the node having the lowest value of h(n), and insert it in the CLOSED list.

**Step 4:** Expand the node (removed in Step 3) and generate its successors.

**Step 5:** Check each successor to find if it is the goal node or not. If any successor node is found to be the goal node, then return success and terminate the search, else go to Step 6.

**Step 6:** Check if each successor node is in either OPEN or CLOSED list. If the node is not present in any of the lists, then

add it to the OPEN list.

**Step 7:** Go to Step 2.

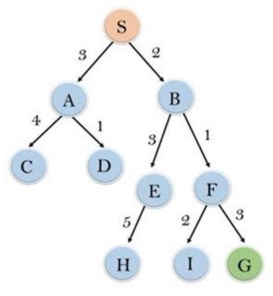
## Advantages

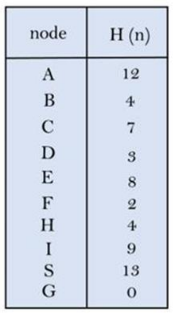
1. It takes advantage of both BFS and DFS algorithms.
2. Best first search algorithm is more efficient than BFS and DFS algorithms.

## Disadvantages

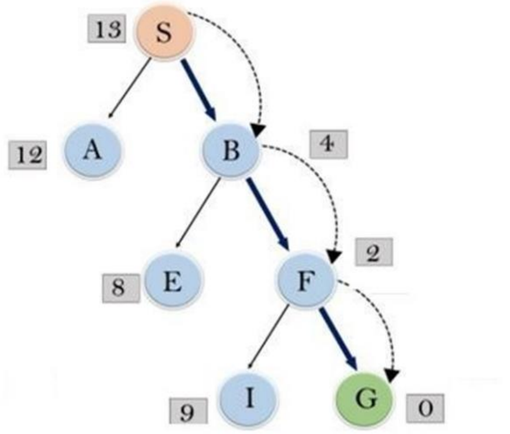
1. In some worst-case scenarios, best first search can behave as an unguided depth-first search.
2. Like DFS, best first search algorithm can get stuck in a loop.
3. Best first search algorithm is not an optimal algorithm.

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





| node | H (n) |
|---|---|
| A | 12 |
| B | 4 |
| C | 7 |
| D | 3 |
| E | 8 |
| F | 2 |
| H | 4 |
| I | 9 |
| S | 13 |
| G | 0 |



Solution:

**Expand the nodes of S and put in the CLOSED list**

**Initialization:** Open [A, B], Closed [S]

**Iteration 1:** Open [A], Closed [S, B]

**Iteration 2:** Open[E,F,A], Closed [S,B]
: Open [E, A], Closed [S, B, F]

**Iteration 3:** Open [I,G,E,A], Closed [S,B,F]
: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F----> G**

**Time Complexity:** The worst case time complexity of Greedy best first search is O($b^m$).

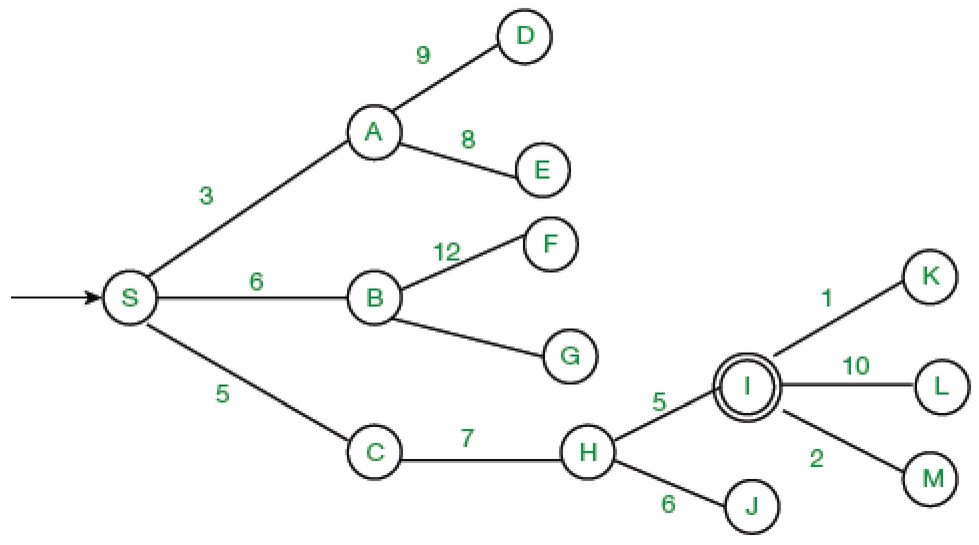
**Space Complexity:** The worst case space complexity of Greedy best first search is O($b^m$). Where, m is the maximum depth of the search space.

**Complete:** Greedy best-first search is also incomplete, even if the given state space is finite.

**Optimal:** Greedy best first search algorithm is not optimal.

The worst-case time complexity and space complexity of greedy best first search is O($b^m$) where, m is the maximum depth of the search space. Although the algorithm is complete, it can, at times, behave like an incomplete algorithm even if the given state space is finite. This makes the Greedy best first search algorithm not an optimal one.

**Example:** Apply greedy best first search algorithm on the graph given below to reach node I from node S.



**Solution:** Add node S in the CLOSED list and place its successors in the OPEN list. Open [A, B, C], Closed [S]

Remove A from the OPEN list as it has minimum h(n), place it in CLOSED list and put its successors in the OPEN list.

Open [B, C, E, D], Closed [S, A]

Remove C from the OPEN list as it has minimum h(n), place it in CLOSED list and put its successors in the OPEN list.

Open [B, E, D, H], Closed [S, A, C]

Remove B from the OPEN list as it has minimum h(n), place it in CLOSED list and put its successors in the OPEN list.

Open [E, D, H, F, G], Closed [S, A, C, B]

Remove H from the OPEN list as it has minimum h(n), place it in CLOSED list and put its successors in the OPEN list. Since I is the successor of node H, the algorithm returns success.

Open [E, D, F, G, I, J], Closed [S, A, C, B, H]

### 3.7.3 A⋆ Search Algorithm

The A⋆ search algorithm, best-known form of best first search,

uses heuristic function h(n), and g(n) which specifies the cost to reach the node n from the start state (refer Fig. 3.19). It solves the searching problem efficiently by expanding less search tree to find shortest path and give an optimal result faster. That is, it avoids expanding paths that are already expensive, but expands most promising paths first.

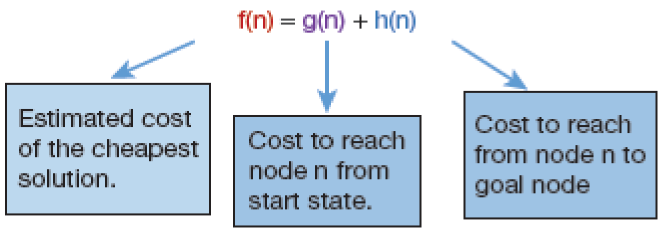

f(n) = g(n) + h(n)
Estimated cost of the cheapest solution.
Cost to reach node n from start state.
Cost to reach from node n to goal node

**FIGURE 3.19** Calculating Cost of A⋆ Algorithm

The A⋆ search algorithm, uses search heuristic as well as the cost to reach the node. Sum of both the costs gives a number.

**Review Questions:**

**1.What is Greedy search?**

**2. What are the types of informed search algorithms?**

**3. What is Bidirectional search?**

# L8:

## 4.Knowledge Representation

### 4.1 Introduction

Artificial intelligence as technology has always fascinated human beings. Multiple science fiction novels and movies have demonstrated the use of AI-powered systems (like robots) that can think, act, understand complex information and make smart decisions based on it.

Although this all looks like a fantastic development, we agree that it is not easy to make machines that behave exactly like humans. This is because humans have conscience and this conscience develops gradually in us with our knowledge, experiences and memories.

To build AI systems that have conscience, we need to inculcate knowledge in them. This chapter is, therefore, dedicated towards representing knowledge in machines.

### Knowledge and Intelligence

Knowledge of real world plays a crucial role in intelligence and creating AI agents that demonstrate intelligent behaviour. Such an agent acts by sensing its environment and using knowledge to act intelligently. However, for this, the knowledge or experience about the input is mandatory. The relationship between knowledge and intelligence can be clearly understood from Fig. 4.1. You can see that if we remove the knowledge component, then the decision maker will not be able to sense the environment accurately and take appropriate decisions.

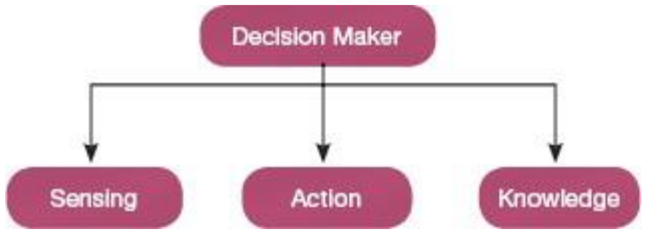


**FIGURE 4.1** Relationship between Knowledge and Intelligence

## 4.2 Knowledge Representation

Humans are best at understanding, reasoning and interpreting things they know (knowledge). Humans use their knowledge to perform various actions in the real world. ***But how machines perform actions, comes under the domain of knowledge representation and reasoning***.

Knowledge representation and reasoning (KR, KRR) is a part of artificial intelligence which focuses on how AI agents think and how their thinking contributes to intelligent behaviour. A knowledge-based system represents information about the real world in such a way that a computer can easily understand and utilize this knowledge to solve complex real-world problems like diagnosing a medical condition or communicating with humans in natural language.

Knowledge representation represents knowledge available in the form of beliefs, intentions and judgments that an intelligent agent must possess for automated reasoning. One of the primary purposes of knowledge representation includes modelling intelligent behaviour for an agent.

Knowledge representation is not just storing data in a database. It goes beyond this aspect to facilitate an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

### 4.2.1 What Knowledge Needs to be Represented?

Humans have intuition, intentions, prejudices, beliefs, judgments, common sense, etc. apart from knowledge about certain facts. We need to incorporate all this information in a machine-understandable format and make the AI system truly intelligent. For this, we need to represent the following knowledge in AI systems:

1. **Object:** Information and facts about all objects (relevant in context). For example, in a self-driving car, vehicles and roads are objects.
2. **Events:** Information and facts about actions which occur in the real world. For example, in a self-driving car, an event can be applying breaks when an object comes in front of it.
3. **Performance:** The manner in which actions are performed. It describes the behaviour related to how to do things.
4. **Meta-knowledge:** It is the knowledge about knowledge (what we know).
5. **Facts:** Facts are the truths about the real world that needs to be represented for an intelligent agent.
6. **Knowledge base:** It is the most important component of a knowledge-based agent that stores a group of sentences (technical sentences, not simple English language ones).

### 4.2.2 What is Knowledge?

We know that knowledge is the basic element of our brain as it helps us to know and understand the things logically and is gained by experiences of facts, data and situations. A knowledgeable person performs all the actions in a better way. Everyone has five types of knowledge illustrated in Fig. 4.2 and enumerated as follows.

1. **Meta knowledge** is the knowledge about knowledge.
2. **Heuristic knowledge** is the knowledge about a specific topic. For example, it can be knowledge of some experts in a filed or subject. Heuristic knowledge is

treated as the rule of thumb as it is based on previous experiences and awareness of approaches, which are good to work but not guaranteed.

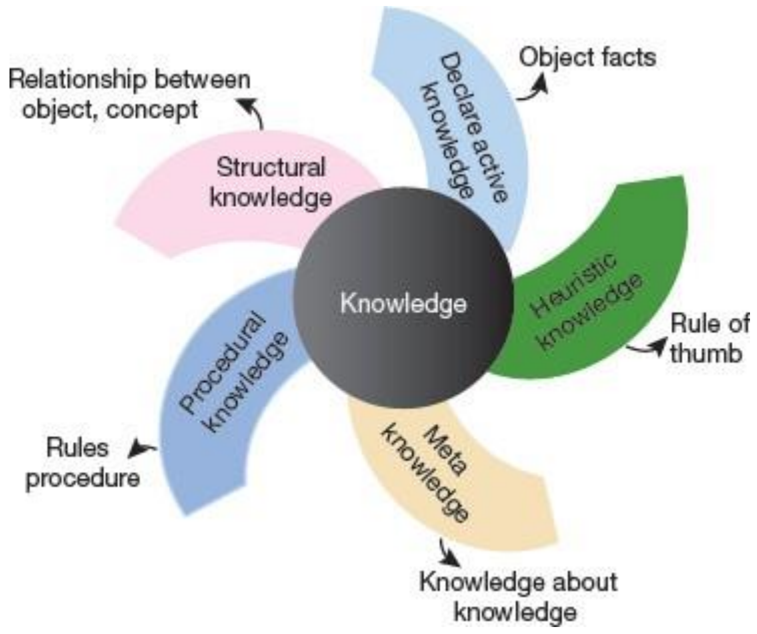


**FIGURE 4.2** Types of knowledge

3. **Procedural knowledge,** also known as imperative knowledge, gives information about how to achieve or do something. This knowledge includes rules, strategies, procedures, agendas, etc. that can be directly applied to perform any task.

4. **Declarative knowledge** is the information that we have about an object. This knowledge helps us to describe a particular concept, fact, object and its attributes. Declarative knowledge is simpler than procedural language and is also called descriptive knowledge as it is usually represented using declarative sentences.

5. **Structural knowledge** is the basic knowledge to solve complex problems. It describes relationships between various concepts or objects such as kind of, part of, and grouping of something.

### 4.2.3 What is Logic?

Logic is the main component of any knowledge as it facilitates a person to draw conclusion by filtering required information from large volumes of sentences.

In AI, knowledge is represented using logic. Every logic has three main elements.

### Syntax

Every language has its own syntax to specify the sequence of constructs that makes a complete sentence in that language. It is therefore not wrong to say that syntax is the representation of a language.

### Semantics

Semantics is used to check if a syntactically correct sentence is logically correct and meaningful or not. Therefore, semantics defines the sense of the sentence which relates to the real world. For example, consider the statement:

*Ram is riding a bike.*

This sentence is syntactically as well as semantically correct. However, the sentence,

*Bike is riding Ram.*

is syntactically correct but semantically incorrect.

### Logical Inference

Inference means to deduce conclusions in the context of some fact or problem. Correspondingly, logical inference uses inference algorithms to think all the possible reasons that can give a proper result.

## 4.2.4 Cycle of Knowledge Representation in AI

An AI system has several components (Fig. 4.3) that helps it to exhibit an intelligent behaviour. These components include the following:

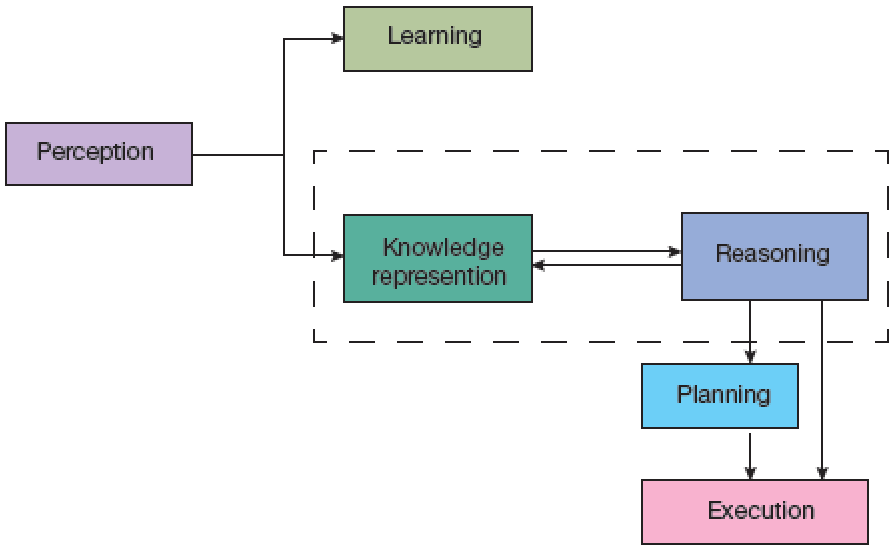


**FIGURE 4.3** Components of an AI system

## Perception

The perception component retrieves data from the environment, discovers the source(s) of noise, checks if the AI was damaged by anything and defines response to be given when any sense has been detected. Data from the environment is gathered using different types of sensors in the form of video, audio, text, time, temperature or any other sensor-based input.

## Learning

The learning component learns from the data that is captured by the perception component. The aim here is to develop a system that can be taught instead of being programmed. With learning, the system focuses on self-improvement through

knowledge acquisition, inference, acquisition of heuristics, faster searches, etc.

## Knowledge Representation and Reasoning

This is the main component which shows human-like intelligence in the machines. Knowledge representation techniques (refer Fig. 4.4) are designed to understand using a top-down approach focussing in detail, what an agent needs to know to behave intelligently. The KRR component also defines how automated reasoning procedures can make this knowledge available as needed.

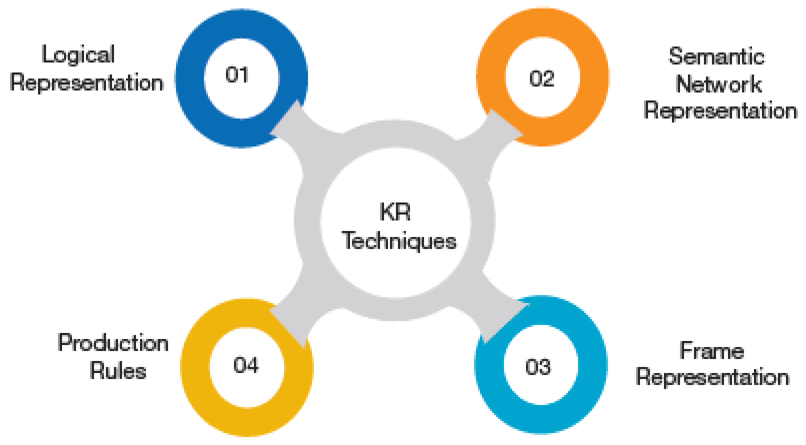


**FIGURE 4.4** Forms of Knowledge Representation

**Planning and Execution**

This component analyses knowledge representation and reasoning. The planning component selects an initial state, enumerates preconditions and effects, and a sequence of actions to achieve the goal state. Once planning is done, actions are executed to get the desired results.

### 4.2.5 Knowledge Representation Requirements

A good knowledge representation system must have the following properties.

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

## 4.3 Knowledge-Based Agent

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

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

Knowledge-based agents represent knowledge that can be reasoned to act intelligently according to the requirements. Usually, these agents store knowledge about its surroundings in the form of sentences.

### 4.3.1 The Architecture of Knowledge-Based Agent

Knowledge-based agents represent the world with some formal representation to act intelligently. For this, they contain a knowledge-base and an inference system.

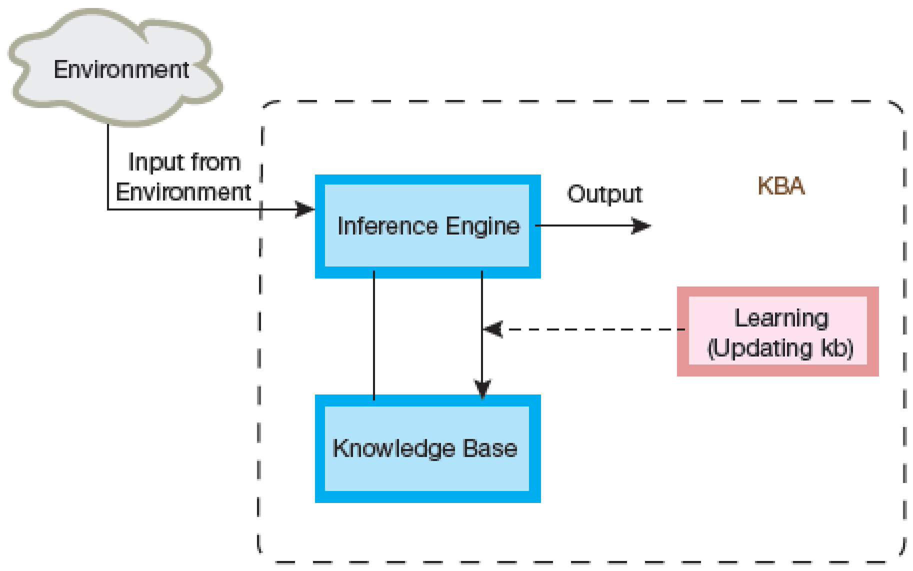


**FIGURE 4.5** Architecture of Knowledge-Based Agent

Knowledge base (KB) is the most important component of a knowledge-based agent. It stores real facts of world using sentences which are explained in knowledge representation language. Inference engine (IE) is the knowledge-based system engine that infers new knowledge in the system.

Figure 4.5 shows a generalized architecture for a knowledge- based agent (KBA). According to Fig. 4.5,

***Step 1:*** The KBA perceives the environment to take input from it.

***Step 2:*** This input is taken by the inference engine of KBA.

***Step 3:*** The inference engine interacts with the knowledge base to make decisions based on the knowledge stored in it.

Note that the learning element of the KBA regularly updates the knowledge base by learning new knowledge. This knowledge base is nothing but a collection of sentences (technical information not same as a sentence in English) that are expressed in a language known as knowledge representation language and stores fact about the world.

In AI, a sentence is not an English language sentence, but a fact represented in a language known as knowledge.

The inference system derives new sentences from old ones and also facilitates addition of new sentence(s) to the knowledge base. As stated earlier, a sentence is a proposition about the world. New information is deduced from the knowledge base by applying logical rules (either forward chaining or backward chaining).

### 4.3.2 Operations Performed By KBA

Whenever the KBA is called to add/update knowledge in the KB or to answer queries made to the KB, it acts intelligently by performing a set of important functions that are given below.

**TELL** operation tells the knowledge base what knowledge it already has about its environment and what additional knowledge it needs to know. TELL is also used to inform the knowledge base which action was selected and performed by the agent.

**ASK** operation asks the knowledge base what action it should perform. The agent gets its answer from the knowledge base.

**PERFORM** operation performs the selected action.

### 4.3.3 A Generic Knowledge-Based Agent

Given below is the structure outline of a generic knowledge- based agent program (Fig. 4.6).

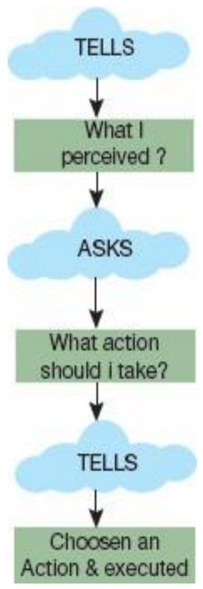


**FIGURE 4.6** Mechanism of an agent program

```
function KB-AGENT(percept):
persistent: KB, a knowledge base
            t : time ( counter initially 0)
TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
Action = ASK(KB, MAKE-ACTION-QUERY(t))
TELL(KB, MAKE-ACTION-SENTENCE(action, t))
t = t + 1
return action
```

The knowledge-based agent accepts the information it perceives from the environment as input and returns an action as the output. The agent maintains the knowledge base (KB). It has some background knowledge of the real world and maintains a counter, t with initial value zero to indicate the time for the whole process. Each time the function is called, the KBA performs operations as given below.

***TELLS*** the KB what it perceives. For this, the MAKE-PERCEPT- SENTENCE function is used to generate a sentence that informs about what information the agent had perceived at the given time.

***ASKS*** KB what action it should take. The MAKE-ACTION-QUERY function is executed to generate a sentence asking which action should be performed at the current time.

***TELLS*** the KB about the action that was chosen. The MAKE- ACTION-SENTENCE function generates a sentence informing about the chosen action that was executed.

### 4.3.4 Various Levels of Knowledge-Based Agent

A knowledge-based agent can be viewed at different levels as follows.

#### Knowledge Level

This is the first level of a KBA. In this level, we need to specify what the agent knows, and what its goals would be. This information is then used to fix the behaviour of the agent. For example, if an agent has to go from point A to point B and it knows the optimum way to reach the destination, then this information is added at the knowledge level (using sentences).

#### Logical Level

At this level, the knowledge representation of the knowledge stored in the knowledge base is understood. For this, sentences are encoded into different logics. Therefore, knowledge is encoded into logical sentences. Continuing with the above example, agent deduces (understands) logic to reach destination, that is, point B.

#### Implementation Level

This is the physical representation of logic and knowledge. At this level, the agent performs actions according to the knowledge obtained from the logical and knowledge level. For example, the agent that wants to move from point A to B, implement the knowledge and logic to reach the destination.

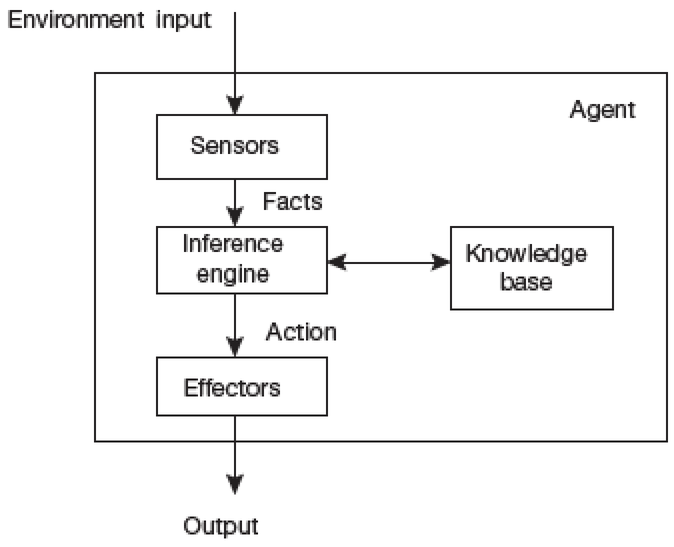


**FIGURE 4.7** Adjusting AC temperature

**Example:** In an automated air conditioner, the inbuilt knowledge stored in the system states that 'It would adjust its temperature according to the weather.' Now, when to adjust and how to adjust is logic. The actual working of adjusting the temperature is at the implementation level of the knowledge- based agent.

### 4.3.5 Approaches to Designing a Knowledge-Based Agent

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

#### Declarative Approach

In this approach, the knowledge-based agent is created by initializing it with an empty knowledge base and then telling the agent all the sentences with which we want to start with.

These sentences are told to the empty system one by one until the system becomes knowledgeable enough to deal with the environment.

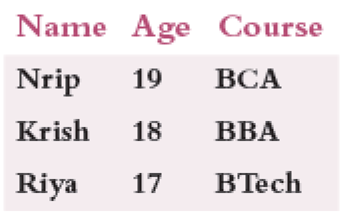
#### Procedural Approach

In the procedural approach, the desired behaviour is directly encoded in the agent as a program code. Therefore, in this approach, we are actually specifying the desired behaviour of the system through coding in programming languages like LISP, Prolog.

Practically, a **hybrid approach** is used that combines features of both declarative as well as procedural approaches. In the hybrid approach, declarative knowledge is compiled into more efficient procedural code.

## 4.4 Types of Knowledge

We can express different types of knowledge in a knowledge representation system.



| Name | Age | Course |
|---|---|---|
| Nrip | 19 | BCA |
| Krish | 18 | BBA |
| Riya | 17 | BTech |

**FIGURE 4.8** A simple relation/ table of students

### Simple Relational Knowledge

In this technique, facts about a set of objects are systematically stored using relations (or tables) in database systems (refer Fig. 4.8). These relations depict relationship between different entities. However, the main shortcoming of this approach is that there is little opportunity for inference.

Inheritable knowledge Inheritable knowledge is one that stores data using a hierarchy of classes. We start representing using generalized classes and move down to specialized classes in the hierarchy. For example, consider Fig. 4.9 which shows such a relationship. Note that Zinya and Vidhi both are instances of under-graduate and post-graduate students, respectively.
Inheritable knowledge shows the relation between instance and class, and is therefore known as instance relation of IS-A relation. In this diagram, objects and values are represented in boxed nodes.

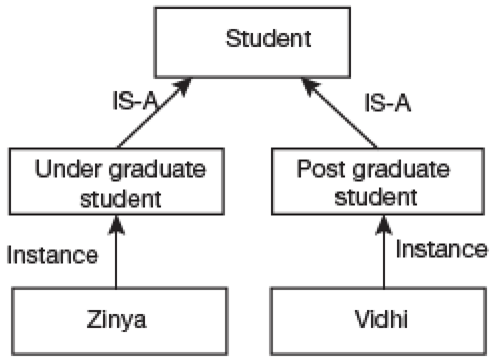


**FIGURE 4.9** Inferential Knowledge Representation

## Inferential Knowledge

Inferential Knowledge Represents Knowledge in the Form of Formal Logic and Can be Used to Derive More Facts Accurately. For example,

**Statement 1**: Diya is a student. **Statement 2**: All

students are bright. Then, it can be represented as;

Student(Diya)

∀x = Student (x) ———-> Bright (x)

**Review Questions:**

**1.What is knowledge representation?**

**2. What are the types of knowledge?**

**3. How to design a knowledge based Agent ?**