CHAPTER 1
Introduction

We will adopt the overall goal of artificial intelligence (AI) to be ‘to build machines with minds, in the full and literal sense’ as prescribed by the Canadian philosopher John Haugeland (1985).

Not to create machines with a clever imitation of human-like intelligence. Or machines that exhibit behaviours that would be considered intelligent if done by humans – but to build machines that reason.

This book focuses on search methods for problem solving. We expect the user to define the goals to be achieved and the domain description, including the moves available with the machine. The machine then finds a solution employing first principles methods based on search. A process of trial and error. The ability to explore different options is fundamental to thinking.

As we describe subsequently, such methods are just amongst the many in the armoury of an intelligent agent. Understanding and representing the world, learning from past experiences, and communicating with natural language are other equally important abilities, but beyond the scope of *this* book. We also do not assume that the agent has meta-level abilities of being self-aware and having goals of its own. While these have a philosophical value, our goal is to make machines do something useful, with as general a problem solving approach as possible.

This and other definitions of what AI is do not prescribe *how to test* if a machine is intelligent. In fact, there is no clear-cut universally accepted definition of intelligence. To put an end to the endless debates on machine intelligence that ensued, the brilliant scientist Alan Turing proposed a behavioural test.

1.1 Can Machines Think?

Ever since the possibility of building intelligent machines arose, there have been raging debates on whether machine intelligence is possible or not. All kinds of arguments have been put forth both for and against the possibility. It was perhaps to put an end to these arguments that Alan Turing (1950) proposed his famous *imitation game*, which we now call the *Turing Test*. The test is simply this: if a machine interacts with a human using text messages and can fool human judges a sufficiently large fraction of times that they are chatting with another human, then we can say that the machine has passed the test and is intelligent.

https://doi.org/10.1017/9781009284325.002 Published online by Cambridge University Press

Since then, many programs have produced text based interactions that are convincingly human-like, for example, *ChatGPT*[1] being one of the latest. Advances in machine learning algorithms for building language models from large amounts of training data have enabled machines to churn out remarkably well structured impressive text. Humans are quite willing to believe that if it talks like a human, then it must think like a human. Even when the very first chat program *Eliza* threw back user sentences with an interrogative twist, its creator Edward Weizenbaum was shocked to discover that his secretary was confiding her personal woes to the program (Weizenbaum, 1966). Pamela McCorduck (2004) has observed in *Machines Who Think* that in medieval Europe people were willing to ascribe intelligence to mechanical toy statues that could nod or shake their head in response to a question.

Clearly relying on human impressions based on interaction in natural language is not the best way of determining whether a machine is intelligent or not. With more and more machines becoming good at generating text rivalling that produced by humans, a need is being felt for something that delves deeper and tests whether the machine is actually reasoning when answering questions.

Hector Levesque and colleagues have proposed a new test of intelligence which they call the *Winograd Schema Challenge*, after Terry Winograd who first suggested it (Levesque et al., 2012; Levesque, 2017). The idea is that the test cannot be answered by having a large language model or access to the internet but would need common sense knowledge about the world. The test subject is given a sentence that refers to two entities of the same kind and a pronoun that could refer to either one of them. The question is which one, and the task is called anaphora resolution. The ambiguity can easily be resolved by humans using common sense knowledge. The strategy is to have two variations of the sentence, each having a different word or a phrase that leads to different anaphora resolution. One of the versions is presented to the subject with a question about what the pronoun refers to. Guesswork on a series of such questions is only expected to produce about half the correct answers, whereas a knowledgeable (read intelligent) agent would do much better. The following is the example attributed to Winograd (1972).

- The town councillors refused to give the angry demonstrators a permit because they feared violence. Who feared violence?
  (a) The town councillors
  (b) The angry demonstrators
- The town councillors refused to give the angry demonstrators a permit because they advocated violence. Who advocated violence?
  (a) The town councillors
  (b) The angry demonstrators

In both cases, two options are given to the subject who has to choose one of the two. Here are two more examples of the Winograd Schema Challenge, with two sets of sentences, each one of which is presented and followed by a question.

- The trophy doesn't fit in the brown suitcase because it's too big. What is too big?
  (a) the trophy
  (b) the suitcase

[1] ChatGPT: Optimizing Language Models for Dialogue. https://openai.com/blog/chatgpt/, accessed December 2022.

- The trophy doesn't fit in the brown suitcase because it's too small. What is too small?
  (a) the trophy
  (b) the suitcase

The following sentence is from the First Winograd Challenge at the International Joint Conference on AI in 2016 (Davis et al., 2017).

- John took the water bottle out of the backpack so that it would be lighter.
- John took the water bottle out of the backpack so that it would be handy.

What does 'it' refer to? Again, two options are given to the subject who is asked to choose one.

The authors report that the Winograd Schema Test was preceded by a pronoun disambiguation test in a single sentence, with examples chosen from naturally occurring text. Only those programs that did well in the first test were allowed to advance to the Winograd Schema Test. Here is an example from their paper which has been taken from the story 'Sylvester and the Magic Pebble'.

- The donkey wished a wart on its hind leg would disappear, and it did.

What vanished? The important thing is that such problems can be solved only if the subject is well versed with sufficient common sense knowledge about the world and also the structure of language.

A question one might ask is why should a test of intelligence be language based? After all, intelligence manifests itself in other ways as well. Could one of these also be an indicator of intelligence?

One area that has been proposed is in the arts, where creativity is the driving force. Computer generated art has time and again come to the limelight. Many artworks by *AARON*, the drawing artist created by Harold Cohen (1928–2016), have been demonstrated at AI conferences over the years (Cohen, 2016). A slew of text-to-image AI systems including DALL-E, Midjourney, and Stable Diffusion have all been released for public use recently.

Erik Belgum and colleagues have proposed a Turing Test for musical intelligence (Belgum et al., 1989). In the fall of 1997, Douglas Hofstadter organized a series of five public symposia centred on the burning question 'Are Computers Approaching Human-Level Creativity?' at Indiana University. This fourth symposium was about a particular computer program, David Cope's *EMI* (Experiments in Musical Intelligence) as a composer of music in the style of various classical composers (Cope, 2004). A two-hour concert took place in which compositions written by EMI and compositions written by famous human composers were performed without identification, and the audience was asked to vote for which pieces they thought were human-composed and which were computer-composed. Subsequently, David Coco-Pope published an article written by a computer program *EWI* (Experiments in Written Intelligence) in the style of Hofstadter, grudgingly conceded by Hofstadter himself at the end of the article (Hofstadter, 2009).

After the 2011 spectacular win by IBM's program *Watson* in the game of *Jeopardy* over two players who were widely considered to be the best that the game had seen, the company unveiled a program *Chef Watson* with the following claim – 'In our application, a computationally creative computer can automatically design and discover culinary recipes that are flavorful, healthy, and novel!'[2] The market is now abuzz with robots that can cook for you, for example, as reported in Cain (2022).

Recently, when DeepMind's *AlphaGo* program beat the reigning world champion Lee Sedol in the oriental game of *go*, the entire world sat up and took notice (Silver et al., 2016). This followed an equally impressive win almost twenty years earlier in 1997 when IBM's *Deep Blue* program beat the then world champion Garry Kasparov in the game of chess (Campbell et al., 2002). Both the games are two person board games in which programs can search game trees as described in Chapter 8. The challenge in these games is to search the huge trees that present themselves. While chess is played on an 8×8 board, *go* is played on a 19×19 board, which generates a much larger game tree. And yet a combination of machine learning and selective search proved invincible. Both these games are conceptually simple even though the search trees are large. In the author's opinion, only when a computer program can play the game of contract bridge at the level described in Ottlik and Kelsey (1983) can we legitimately stake a claim to have created an AI.

Meanwhile, one should perhaps take a cue from Alan Turing himself, move away from the bickering, and get on with the design and implementation of autonomous machines who[3] do useful things for us. In the summer of 1956, John McCarthy and Marvin Minsky had organized the Dartmouth Conference with the following stated goal – 'The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it' (McCorduck, 2004). *That* is the spirit of our quest for AI.

1.2 Problem Solving

Our quest is for a machine that is autonomous and whose behaviour is goal directed. Whatever it does, it should do autonomously. We imagine a scenario in which the machine is an agent to serve the goals given it to it by a *user*. Current applications take a short horizon view of achieving specific goals, though we can imagine a persistent agent engaging with the human over long periods, perhaps even the user's lifetime. We ignore the doomsday scenarios in which machines overcome and subjugate humans, though this idea has been fashionable amongst certain sections of science writers. That so-called *singularity* is not even on the horizon (Larson, 2021).

We want our machines to solve problems for us. Given a set of goals, the machine must engage with the world to achieve those goals. The goals may be short term or long term, and the world in which the problem solving agent operates may be changing, even in the simplest case when the agent is the only one acting and effecting the change. The agent must sense its environment, deliberate over its goals, and act in the domain. The agent must not just be reactive, operating in a hard-coded *stimulus–response* cycle, but should be able to act flexibly in a *sense–deliberate–act* cycle autonomously. A schematic of an autonomous agent is shown in Figure 1.1.

In all life forms, deliberation happens in the brain. Incoming sensory data is processed in the context of what the creature already knows. Our understanding of the animal brain is that it is a collection of a large number of very simple processing components called *neurons*. Each

[2] https://researcher.watson.ibm.com/researcher/view_group.php?id = 5077, accessed December 2022.

[3] In the style of *Machines Who Think* by Pamela McCorduck.

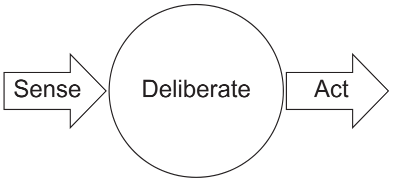

Figure 1.1 An autonomous agent operates in a three stage cycle. It receives input from its sensory mechanism, it deliberates over the inputs and its goals, and acts in the world.

neuron is connected to many other neurons and each connection has a weight that evolves with experience. This changing of weights is associated with the process of learning.

The neurons at the sensing end of the brain accept information coming in from various senses like sight, sound, smell, taste, and touch. The general model of processing is that once a neuron is activated, it sends a signal down its principal nerve called the axon, which distributes the signal to other connected neurons. The weights of the connections determine which connected neurons receive how much of the signal. Eventually the signals reach the neurons at the output end, sending signals down the motor neurons that activate muscles that produce sounds from the mouth and movement of the limbs.

Some simple creatures may be just reactive, recognizing food or prey and triggering appropriate actions, but as we move up the hierarchy, there may be more complex processing happening in the brain, involving memory (in Greek mythology, the dog Argos recognizes Odysseus at once when humans could not), planning (monkeys in Japan have been known to season their food with salt water), and reasoning (remember all those experiments with mice in mazes and Pavlov's dog). Whatever the cognitive capability of the creature, our view of their brains can be captured as shown in Figure 1.2.

Different life forms have differently sized brains relative to the sizes of their bodies. Earlier life forms had simple brains often referred to as the reptilian brain. In the 1960s, the American neuroscientist Paul MacLean (1990) formulated the *Triune Brain* model, which is based on

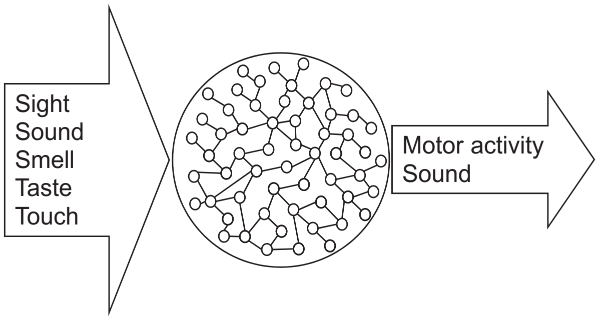

Figure 1.2 The neural animal brain. All life forms represent knowledge in the form of weights of connections between neurons in their brain and body. The numbers do not mean anything to us, and we say that the representation is sub-symbolic.

the division of the human brain into three distinct regions. MacLean's model suggests that the human brain is organized into a hierarchy, which itself is based on an evolutionary view of brain development. The three regions are as follows:

1. Reptilian or primal brain (basal ganglia) was the first to evolve and is the one in charge of our primal instincts.
2. Paleomammalian or emotional brain (limbic system) was the next to evolve and handles our emotions.
3. Neomammalian or rational brain (neocortex) which is responsible for what we call as thinking.

According to MacLean, the hierarchical organization of the human brain represents the gradual acquisition of the brain structures through evolution. The human brain, considered by many to be the most complex piece of matter in the universe, is made up of a cerebrum, the brain stem, and the cerebellum. The cerebrum is considered to be the seat of thought and, in humans, comprises two halves, each having an inner white core and an outer cerebral cortex made up of grey matter.

It is generally believed that the larger the brain, the greater the cognitive abilities of the owner.

1.3 Neural Networks

A neuron is a simple device that computes a simple function of the inputs it receives. Collections of interconnected neurons can do complex computations. Insights into animal brains have prompted many researchers to pursue the path of creating *artificial neural networks* (ANNs).

An ANN is a computational model that can be trained to perform certain tasks by repeatedly showing a stimulus and the expected response. It is best suited for the classification task. The earliest neural network was the *perceptron* (McCulloch and Pitts, 1943; Rosenblatt, 1958) which had one layer of neurons and could serve as a binary linear classifier. That is, whenever two classes in some space could be separated by a line or a plane in appropriate dimensions, the perceptron could be trained to learn the position and the orientation of the separator. Research in this area suffered a setback when Minsky and Papert (1969) showed its limitations – it could only classify linearly separable classes. For example, one cannot draw a line to separate the shaded circles, representing data from class A, from the unshaded ones, representing data from class B, as shown in Figure 1.3.

The work in neural networks was revived with the publication of the *Backpropagation* algorithm. In the mid-1980s, Rumelhart, Hinton, and Williams (1986) showed that a *multi-layer perceptron* could be trained to learn any non-linear classifier. They also popularized the Backpropagation algorithm that fed the error at the output layer back via the hidden layer, adjusting the weights of the connections (McClelland and Rumelhart, 1986a, 1986b).

Figure 1.4 shows the schematic diagram of a typical feedforward neural network. On the left is the input layer where the discretized input is fed in, activating nodes in the layer. Then, activation spreads from the left to the output layer on the right via the nodes in the hidden layer. In Figure 1.4, there are five output nodes, which could stand for five class labels. In the simplest case, when the network has learned to classify the input, which could be an image, one output

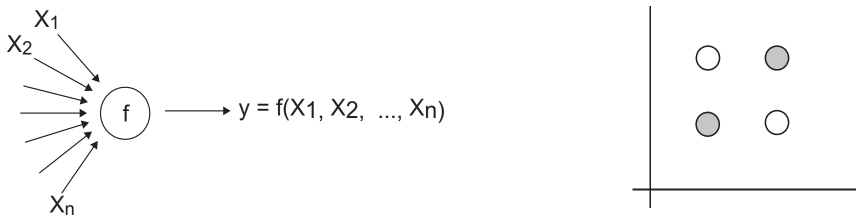

Figure 1.3 A neuron is a simple processing device that receives signals and generates an impulse as shown on the left. On the right is an example of a classification problem in which no line can be drawn to separate the shaded circle from the unshaded ones.

node is activated, indicating the class label. What the neural network has learnt is the *association* between the pattern of activation in the input layer and the class label at the output layer.

The fact that the input may be an image of a scene is only in the mind of the user, as is the name given to the class label. In the figure, the names are five animals, but the neural network has no idea that one is talking of animals, or a particular animal like a horse or a bear. It just *knows* which label to activate with a given image.

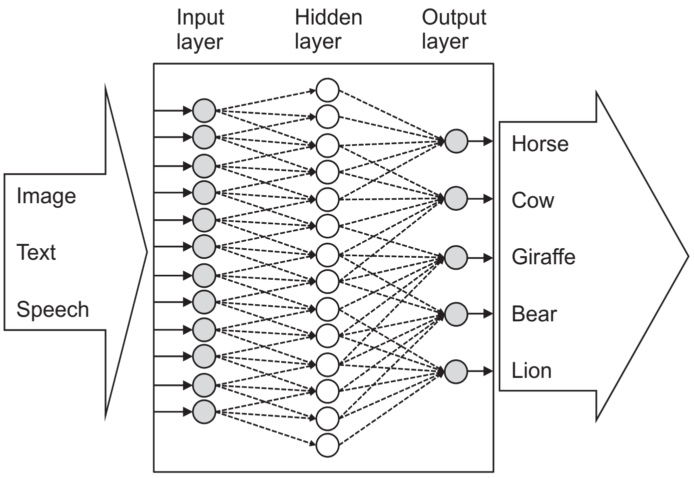

Figure 1.4 A feedforward artificial neural network learns a function from the input space to the output classes. Learning happens via the adjustment of edge weights. The labels of the output classes are meaningful only to the user.

This knowledge is not explicit or symbolic in the network. It is buried in the weights of the edges from nodes in one layer to the next one. These weights are instrumental in directing the activation from the input layer to the relevant output layer node. Nowhere in the network is there any indication that one is looking at a giraffe or a lion. Such representations of knowledge are often called *sub-symbolic* in contrast with the explicit symbolic representations we humans commonly use.

Neural networks learn what they learn by a process of *training*. The most common form of training is called *supervised learning*, in which a user presents input patterns to the network, and for each input shows what the output label should be. Every time an input pattern is presented, the network makes its own decision of what the activation value of the class label is. For example, if a bear is shown to the network, it might compute the output values as [0.2, 0.1, 0.0, 0.4, 0.3] when the expected output is [0, 0, 0, 1, 0], indicating that it is the fourth node (the bear). The error in the actual output defines a loss function that BACKPROP (as it is also known) aims to minimize. Most variations of the algorithm compute the gradient of the loss function with respect to the weights and do a small change in the edge weights in each cycle to reduce the loss. This can be viewed as *gradient descent*, an algorithm we look at later in the book.

The other forms of learning that are popular are *unsupervised* learning in which algorithms can learn to identify clusters in data, and *reinforcement* learning in which feedback from the world is used to adjust relevant weights. Reinforcement learning has achieved great success in game playing programs that learn how to play by playing hundreds of thousands of games against themselves, learning how to evaluate board positions from the outcomes of the games.

1.3.1 Deep neural networks

In principle the three layered network could learn any function, but in practice it was hard to do so, requiring a large number of neurons in the hidden layer. Geoffrey Hinton persevered with neural networks and showed in 2012 that deep networks with many hidden layers can achieve phenomenal success in computer vision – recognizing thousands of *types* of objects. Alex Krizhevsky in collaboration with Ilya Sutskever and his PhD advisor Geoffrey Hinton implemented the convolutional neural network (CNN) named *AlexNet* (Krizhevsky et al., 2017). This program did phenomenally well on the task of image recognition in the ImageNet Large Scale Visual Recognition Challenge in 2012. The network achieved a top-5 error of 15.3%, much better than the runner up. Their paper claimed that the *depth of the model* was essential for its high performance. Since then, neural networks with many layers have been doing very well in pattern recognition tasks. In 2015, a deep CNN with over 100 layers from Microsoft Research Asia outperformed AlexNet. Even though many layers make them computationally expensive, the use of graphics processing units (GPUs) during training has made them feasible. Figure 1.5 shows a schematic of a deep neural network.

The development of newer architectures and newer algorithms was instrumental in the spurt of interest in deep neural networks. Equally responsible perhaps was the explosion in the amount of data available on the internet, for example, the millions of images with captions uploaded by users, along with rapid advances in the computing power available. In 2018, three scientists, Geoffrey Hinton, Yann LeCun, and Yoshua Bengio, were jointly awarded the Turing Award for their work in this area. Deep networks got further impetus with the availability

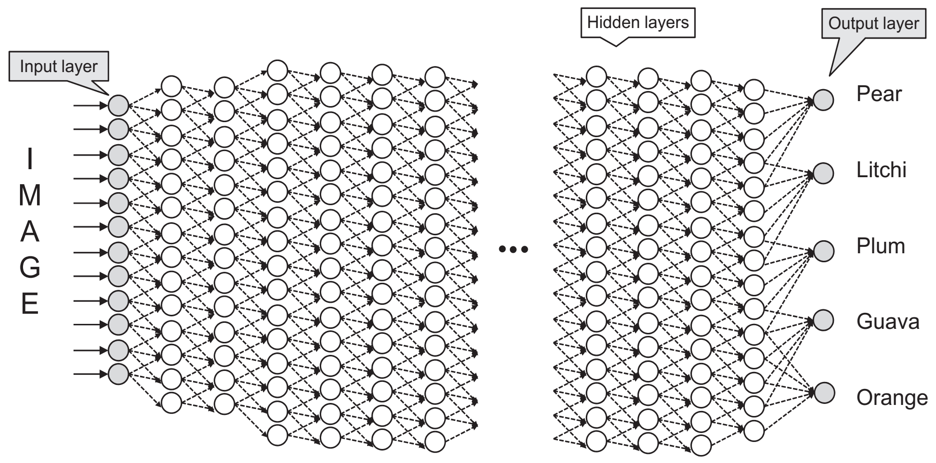

Figure 1.5 The schematic diagram of a deep neural network.

of open source software like *Tensorflow*[4] from Google that makes the task of implementing machine learning models easier for researchers.

More recently, *generative neural networks* have been successfully deployed for language generation and even creating paintings, for example, from OpenAI.[5] Generative models embody a form of unsupervised learning from large amounts of data, and are then trained to *generate data* like the one the algorithms were trained on. After having been fed with millions of images and text and their associated captions, they have now learnt to generate similar pictures or stories from similar text commands. Programs like *ChatGPT, Imagen,* and *DALL-E* have created quite a flurry amongst many users on the internet.

Deep neural networks are very good at the task of pattern recognition. Qualitatively, they are no different from the earlier networks, but in terms of performance they are far superior. The main task they are very good at is classification, a task that some researchers have commented is accomplished 'in the blink of an eye' by all life forms (Darwiche, 2018). The question one might ask is what after that?

Both in the case of generative models and deep neural network based classification, one must remember that the programs are throwing back at us whatever data has been fed to them. They do not *understand* what they are writing or drawing even though there is some correlation between the input query or command and the output generated.

For understanding and acting upon such perceived data, one needs to create models of the world to reason with. This is best done by explicit symbolic representations, which have the added benefit that they can contribute to explanations.

[4] https://developers.google.com/machine-learning/crash-course/first-steps-with-tensorflow/toolkit, accessed December 2022.

[5] https://openai.com/blog/generative-models/, accessed December 2022.

1.4 Symbolic AI

Imagine that you are the coach of a football team watching the game when your team is down two–nil. You have been watching the game for the best part of seventy minutes. The need of the hour is to make a couple of changes.[6] You pull out two players who are playing under their par and send in two substitutes.

Can a neural network take such a decision? Clearly not. The knowledge of the neural network is a kind of long term memory arrived at by training on many past examples. The neural network does not have the apparatus to represent the world around it in a dynamic scenario. What an agent also needs is short term memory that represents the current problem, and facilitates reasoning about the situation and planning of actions. We will talk about short term and long term memory in a little more detail in Chapter 7. But now we introduce the main idea at the core of this book – symbolic reasoning.

1.4.1 Symbols, language, and knowledge

Arguably, humankind broke away from the rest of the animal world with the development of language. The ability to give *names* to concepts combined with a shared understanding of what words mean not only has been a boon for communication (remember the boy who shouted wolf?) but has also provided a basis for representing complex concepts.

The core of *language*, whether spoken or written, is the *symbol*. A symbol is a perceptible something that stands for something else. The study of how signs and symbols are created and how they are interpreted is called *semiotics*. We are all familiar with road signs indicating the presence of schools, crossings, restaurants, U-turns, and so on. Most commercial activities are promoted using logos of companies which too stand for the company. Biosemiotics is the study of how complex behaviour emerges when simple systems interact with each other through signs. The pheromone trails left by ants for other ants to follow and the waggle dance of the honey bees to convey the location of food source to fellow bees are examples. The key feature is the use of words, behaviours, and shapes, collectively known as *semiosis*, as a means of transmitting meaningful information encoded in symbols and decoded by the receiver.

Human languages have evolved to describe what we see and perceive. The simplest kinds of names were probably just atomic but gradually we learnt to combine words to devise compound names, for example, *der liegestuhl* (the lounge chair) in German and *Himalaya* (the abode of snow) in Sanskrit. But at the simplest atomic level, a word, whether a noun, adjective, adverb, or verb, simply stands for something. Once many years ago a curious four-year-old had asked me: ‘Why is a (ceiling) fan called a fan, and not something else?’ The answer perhaps is that words acquire meaning via wide social agreement and also derive from related words. Words vary over regions, sometimes gradually and sometimes abruptly. The English word ‘potato’ corresponds to ‘patata’ in Sindhi, and ‘batata’ in Marathi. But some languages have a radically different name, ‘alu’ or ‘aloo’, for it. A look at the names of numbers across different languages also reveals a remarkable similarity that is unlikely to be sheer coincidence. Most languages have names starting with ‘s’ for the equivalent of the number six, which is *sechs*

[6] Even as I write this, France has scored two goals in two minutes to draw level with Argentina in the FIFA World Cup final of 2022.

in German, *seks* in Norwegian, *seis* in Spanish and Portuguese, *shash* in Sanskrit, and *sitta* in Arabic. Likewise, the number seven is *saat* in Hindi, *saith* in Welsh, *sieben* in German, *sedam* in Serbian, *septem* in Latin, and *sapta* in Sanskrit. At the same time, the same word may mean different things in different languages, much to the consternation of uninformed travellers, who may not realize that *gift* in German means poison or a toxin, and *helmet* in Finnish means pearls. In addition, the diversity in the world across regions results in communities having fine-grained words indicating small differences in what *they* encounter in their lives. Nordic countries have a multitude of words for different kinds of snow. A Swede may use *Kramsnö* for squeezy snow, perfect for making snowballs, and *Lappvante* for thick, falling snow amongst the many words that residents of Kerala may club into one word, *snow*. At the same time, the people in Kerala have different names for a variety of what the Scandinavian countries might just refer to as a *banana. Ethapazham*, for example, is the name of the longest banana available, *chenkadali* is the red banana, and *poovan* a small banana.

Observe that when we talk of trees and fruits and animals, we talk of them *as we perceive* them. There is a process of *reification* or *abstraction* that happens here. A human body is made up of about 10^{27} atoms, but we do not think of it at that level of detail. We cannot. We think of a person as an *individual* and think about the body parts as individual entities. The atoms that are part of our body are anyway transient, but our notion of the self is persistent. In his book titled *Creation*, author Steve Grand (2001) highlights the fact that when we *perceive a stationary cloud* atop a mountain pass, it is really moist wind blowing over it with water molecules condensing as they reach the top and becoming visible even as they flow on. The cloud, like our own body, is in our mind. The concepts that we form in our heads are often at a convenient level of aggregation. In any case, humans started off by giving names to what we see and what we do. Our visual perception system has a finite field of vision. A fascinating chronicle on the sizes of objects in the universe, *The Powers of Ten*, lists different physical entities that exist at different scales (Morrison et al., 1986). In the book, and a short movie of the same name, the authors zoom out from human-size objects to the very ends of our universe, and them zoom back in and onwards onto the subatomic level. Our human perception is limited from about 10^{-4} m where we can see a pollen grain shining in a ray of sunlight, to larger objects – a mustard seed (10^{-3} m), a fingernail (10^{-2} m), a sunbird (10^{-1} m), a child (10^{0} m), a small tree (10^{1} m), a pond (10^{2} m), the Golden Gate bridge (10^{3} m), and a small town seen from a hill (10^{4} m). We find it easy to give names for objects at these scales. For larger or smaller scales, we have to rely on science to inform us. We know that the diameter of solar system is 11,826,600,000 km, and the diameter of the Milky Way is about 100,000 light years across. We know that a virus is about 10^{-7} m, and the size of the carbon atom nucleus is about 10^{-14} m. All this is secondary knowledge derived from our scientific endeavour, even though we often cannot visualize very large or very small distances at the extreme scales. Quantum mechanics has further obfuscated our understanding of the world. Marcelo Gleiser (2022) writes that quantum physics has redefined our understanding of matter. In the 1920s, the wave–particle duality of light was extended to include all material objects, from electrons to you. Cutting-edge experiments now explore how biological macromolecules can behave as both particle and wave.

When we talk of the spoken word, we think of it as a symbol that stands for something. Symbols take on a life of their own when they are represented by tangible marks, which are not transitory like sounds, but have a degree of permanency associated with them.

The earliest humans known to engrave symbols on clay were the Sumerians in ancient Mesopotamia, which is often known as the cradle of civilization. The first engravings were pictographs, but soon evolved into more abstract entities like symbols from an alphabet. The earliest form of writing was cuneiform writing.

> First developed around 3200 B.C. by Sumerian scribes in the ancient city-state of Uruk, in present-day Iraq, as a means of recording transactions, cuneiform writing was created by using a reed stylus to make wedge-shaped indentations in clay tablets. Cuneiform as a robust writing tradition endured 3,000 years. The script – not itself a language – was used by scribes of multiple cultures over that time to write a number of languages other than Sumerian, most notably Akkadian, a Semitic language that was the lingua franca of the Assyrian and Babylonian Empires.[7]

It was replaced by alphabetic writing sometime after the first century AD. The breakthrough came when symbols were not only employed as images representing objects and events like a hunt, but abstract entities like sounds. A set of symbols forms an alphabet. Alphabetic symbols could now come together to form words, and words could form sentences. The spoken word became the written word. Different natural languages evolved in many regions of the world. The common theme was writing.

The faculty of language in turn created a mechanism of knowledge dissemination. Starting with stories in the oral tradition, the invention of writing made it possible for us to leave a permanent imprint for anyone to read at any time in any place. The invention of the internet made all this information available for everyone instantaneously.

The basis of the written word was the idea of symbols.

1.4.2 Symbol systems

While it is true that ANNs have knowledge about the world encoded in the weights, it is not knowledge accessible to the user. Humans beings too have knowledge encoded into our neural brains, but we have somehow evolved the ability of representing and reasoning with symbols. We have the ability to model the world around us in our heads and describe what we know in natural language.

Herbert Simon and Alan Newell (1976) proposed that the ability to represent symbolic knowledge and reason with it is sufficient and necessary for intelligence – ‘A physical symbol system has the necessary and sufficient means for general intelligent action’. This is known as the *physical symbol system hypothesis*.

- Symbol: A perceptible something that stands for something else. For example, alphabet symbols, numerals, road signs, musical notation.
- Symbol System: A collection of symbols – a pattern. For example, words, arrays, lists, even a tune.

[7] https://www.archaeology.org/issues/213-1605/features/4326-cuneiform-the-world-s-oldest-writing, accessed September 2022

- Physical Symbol System: That obeys laws of some kind, a formal system. For example, long division, computing with an abacus, an algorithm that operates on a data structure (which is a symbol system).

The idea of symbolic reasoning goes back to olden times. John Haugeland (1985) traces the evolution of the *idea* of thinking being symbolic to medieval Europe, reproduced here: Galileo Galilei (1564–1642) in *The Assayer* (published 1623) says that 'tastes, odors, colors, and so on are no more than mere names so far as the object in which we locate them are concerned, and that they reside in consciousness'. Further, that 'philosophy is written in this grand book, the universe ... It is written in the language of mathematics, and its characters are triangles, circles, and other geometric figures'. Galileo Galilei gave us what we call the laws of motion, and his explanations were expressed in geometry. The English philosopher Thomas Hobbes (1588–1679) first put forward the view that thinking itself is the manipulation of symbols. Galileo had said that all reality is mathematical in the sense that everything is made up of particles, and our sensing of smell or taste was how we reacted to those particles. Hobbes extended this notion to say that thought too was made up of (expressed in) particles which the thinker manipulated. However, he had no answer to the question of how a symbol can *mean* anything. In *De Corpore*, Hobbes first describes the view that reasoning is computation early in Chapter 1. 'By reasoning', he says, 'I understand computation.' Hobbes was influenced by Galileo. Just as geometry could represent motion, thinking could be done by manipulation of mental symbols. René Descartes (1596–1650) further extended the idea by saying that 'thoughts themselves are symbolic representations'. Descartes was the first to clarify that a symbol and what it symbolizes are two different things, but then he ran into the *mind–body dualism*. If reasoning is the manipulation of meaningful symbols according to rational rules, then who is manipulating the symbols? It can be either mechanical or meaningful, but how can it be both? How can a mechanical manipulator pay attention to meaning? These are some of the questions we are still to find answers for.

1.4.3 An architecture for cognition

It has become clear that intelligence cannot be manifested by a single algorithm or a single representation. In his influential book *The Society of Mind*, Marvin Minsky (1986) emphatically argues that a mind is a society of many processes working together and in tandem. We have already hinted that ANNs are good at certain tasks like pattern recognition, but are not easy to adapt to acting autonomously in the world managing dynamic information. We highlight the different aspects of intelligence in the next section based on the proposed architecture shown in Figure 1.6.

As one can see, this is a refinement of Figure 1.1 with *a society of algorithms* coming together to constitute an autonomous intelligent agent. AI will not be a monolithic hammer. It will be a delicate orchestra of many different processes working together, each playing its own part.

The inputs that the agent works with are the same with the different senses feeding information. Much of this information is processed by neural networks that serve as signal to symbol transducers. In practice ANNs work with discretized signals, which technically are symbols themselves. However, we prefer to think of the input as being at the signal level, because it is the raw data that the machine has to work with, for example, an image represented as an array of pixels each having a discrete value stored in it. Only when a program processes

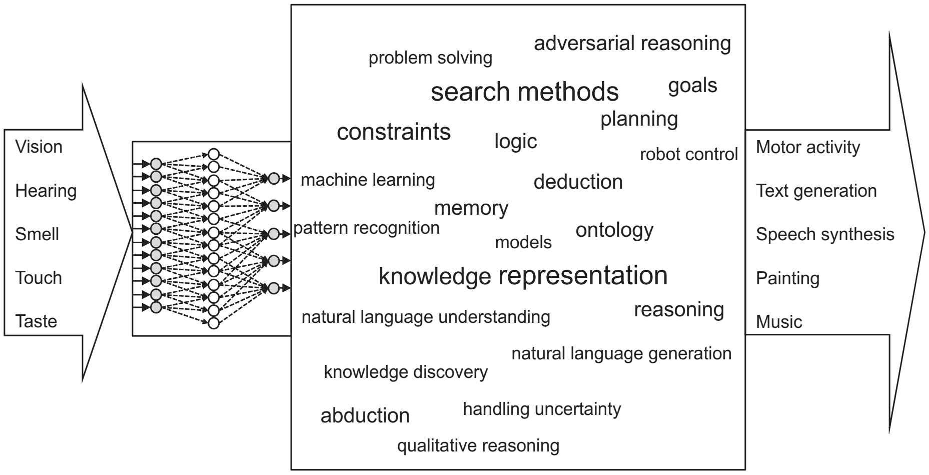

Figure 1.6 An architecture for cognition. In classical AI, an intelligent agent senses the world around it and maps it to a symbolic representation making inferences, and planning for its goals.

these pixels and extracts information about the objects in the image do we say that we have a symbolic representation. Early work on pattern recognition was syntactic in nature, for example, as described in Gonzalez and Thomason (1978). One would extract edges in the image and apply grammar rules to combine edges to (say) recognize handwritten characters. Processing complex images was not feasible at all. However, neural networks have proven to be excellent at processing images and recognizing patterns and individuals.

In our proposed architecture, the task of deliberation is done using symbolic reasoning. As shown in Figure 1.6, the deliberation phase may invoke many different algorithms. We outline the different process in the next section.

1.5 The Core of Intelligence

Given that it is hard to define what intelligence is, it is easier to adopt a behavioural characterization of intelligence – if it behaves intelligently, then it must be intelligent. Alan Turing's *imitation game* adopted the approach of conversational interaction with the idea that if it can talk intelligently, it must be intelligent. We judge a machine to be intelligent if *we* are convinced that it is behaving intelligently.

A question that has often been asked about machine intelligence is that of meaning. How could a symbol *mean* anything to a machine? We have observed that Hobbes was faced with this when he said that thinking was the manipulation of symbols. The Scottish philosopher David Hume (1711–1776) did away with the notion of meaning altogether. An admirer of Isaac Newton, Hume proposed that just like the heavenly bodies moved based on the laws of physics, impressions, and ideas were (like) the basic particles to which all mental forces and operations applied. He did not question why they did so but was satisfied by the empirical observation that they did do so.

The question of meaning was also raised by the American philosopher John Searle (1932–) with his Chinese room thought experiment in which a native English speaker locked inside a room with boxes of symbols and a set of instructions on how to manipulate them could answer questions in the Chinese language slipped below on pieces of paper without understanding a word of the Chinese language.

The digital computer manipulates symbols based on a set of instructions given to it. Does it understand the meaning of the symbols that it is manipulating? If it is adding two numbers, does it know that it is adding numbers? Do all of us understand what a number is (McCulloch, 1961)? Or if it is beating a world champion in the game of chess, does it even know that it is playing chess, or what winning is?

We will sidestep these questions on meaning and focus instead on utility and meaningful action. Build machines that operate in a purposeful goal directed manner.

In this book we assume that our goal is to build machines that autonomously solve problems for us, and that the goals of the machines are the goals we have given them to solve. Given a problem to solve, and given a set of operators in its repertoire, the task of the problem solver is to choose actions that will achieve the goal.

At the core is the ability to create a model of the world in which the agent is operating, and reason about its goals, plans, and actions with the representation. The model of the world is the base for all cognitive activity. This model contains the memories of the agent, lessons learnt, and the representation of the world in which it operates. The agent needs the ability to imagine worlds that are not immediately perceptible, or which the agent may desire to create.

Broadly speaking, there are three kinds of processes that come together to solve a problem, and which form the core of intelligent behaviour.

1.5.1 Remember the past and learn from it

Problem solving refers to the activity of making, and acting upon, decisions to transform a given situation to a desired one. The ability to solve problems is perhaps the prime hallmark of intelligent behaviour. Very often we look at a problem and a solution comes to our mind. This happens because we humans have memory, and we draw upon our past experiences, and that of others via language. We ascribe lack of intelligence to people who make the same mistakes repeatedly. And we believe that experts are those who learn even from mistakes of others. Figure 1.7 shows the learning cycle an agent may go through during problem solving. Different mechanisms for exploiting memory may have different forms of learning.

One approach that has focussed on storing past experiences is *case based reasoning* (CBR). CBR has been surprisingly effective in many industrial applications (Watson, 1997, 2002). The strategy behind CBR is simple.

- Similar problems have similar solutions, is the adage behind memory based reasoning.
- And, importantly, problems are indeed often similar.

In the simplest form, CBR maintains a *case base* of *problem solution* pairs <p, s>. The problem part of a case is a description of the problem that the case solves. The description may be attribute value pairs or it could be in natural language text. The following is the 4R cycle that CBR follows.

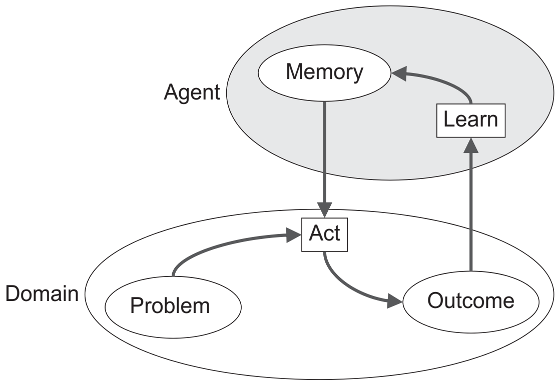

Figure 1.7 A memory based agent employs the knowledge stored in its memory to solve a problem in the domain. Based on the outcome of each instance of solving a problem, the agent refines its knowledge and improves over time.

- Retrieve: When the agent encounters a new problem, it searches the case base for the most similar problem. Often more than one case is retrieved in the style of *k nearest neighbours* retrieval.
- Reuse: The solution that is retrieved along with the case is adapted to the current problem. This could involve adjusting some parameters that are different in the current description and the retrieved one and adjusting the solution part.
- Revise: If the solution does not work, tweak it. This could involve human intervention.
- Retain: If the revised solution is significantly different, add it to the case base. The next time a similar problem shows up, this could be useful.

CBR has been particularly useful in domains that are not well modelled and where the problem solving knowledge is more *experiential* than *analytical*. One of the earliest successes was the *Clavier* system developed to cure aircraft parts at Lockheed Missiles and Space Company in Sunnyvale, California (Watson, 1997). The task involved placing parts made of composite materials that kept changing in an autoclave, which is an expensive resource. The quality of the product depended on where it was placed on the tray, and operators were essentially following similar layouts from the past. Curing is an unrewarding art, rather than a science, but Clavier reduced the discrepancy reports considerably as its case base grew from an initial twenty to several thousands. Figure 1.8 shows a schematic of a CBR system employed for knowledge management in a manufacturing setting (Khemani et al., 2002).

CBR is a form of instance based learning (see also Chapter 11) in which the system memorizes past experiences and remembers them. Another approach is to assimilate the knowledge accrued from experience into compact structures that can be used. *Neural*

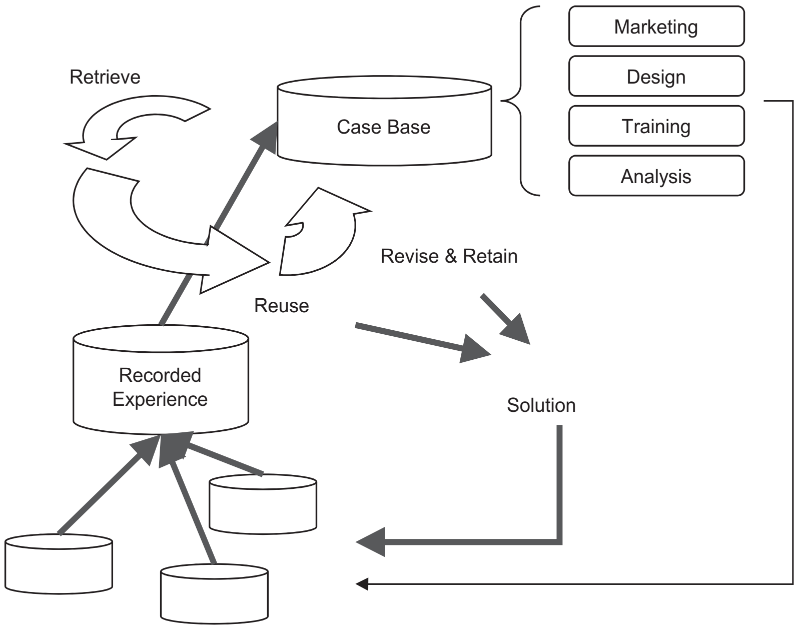

Figure 1.8 A CBR system in the manufacturing industry. The data recorded from different locations in the shop floor is assimilated into a case base. The resulting system can have multiple applications. The CBR is essentially a tool for knowledge management.

networks are examples of such learning, but there have also been more explainable structures like *decision trees*, where attributes and their values used for classification can be read from the path in the tree. Consider a small data set shown in Table 1.1 for the sake of illustration. There are three attributes A, B, and C in this data set with values {a1, a2, a3}, {b1, b2, b3}, and {c1, c2} respectively. There are two class labels ‘Yes’ and ‘No’ in each row in the table.

One could of course use CBR for prediction given a new problem in which the values of the three attributes are given. But for such well defined domains, it is convenient to build a *decision tree*. A decision tree is a discrimination tree that tests for the value of one variable at the root node, and then traverses an appropriate branch to test the value of another variable. The algorithm for constructing a decision tree with nominal attribute values is the well known *ID3* algorithm (Mitchell, 1997). The basic idea behind the algorithm is to choose that attribute that separates the two classes as best as possible. A decision tree for the data in Table 1.1 is shown in Figure 1.9.

When a new record of the values for A, B, and C comes in, it is dropped down the tree. At each node, the value of some attribute is tested and the record follows an appropriate branch. Leaves in the tree are labelled with class information. Observe that other trees may be possible, testing a different attribute at each stage. The ID3 algorithm is designed to build short trees.

Table 1.1

A	B	C	Outcome
a1	b1	c1	Yes
a1	b1	c2	No
a1	b2	c1	Yes
a1	b3	c2	No
a1	b2	c2	No
a2	b1	c1	Yes
a2	b2	c1	Yes
a2	b1	c2	Yes
a3	b1	c1	Yes
a3	b2	c1	Yes
a3	b1	c2	No
a3	b2	c2	No

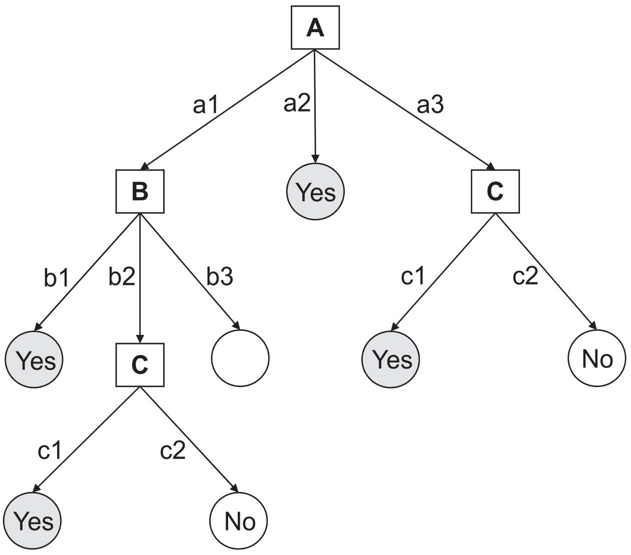

Figure 1.9 A decision tree based on the data from Table 1.1. The root nodes tests for the attribute A and traverses the appropriate branch. The leaf nodes have the class labels.

1.5.2 Understand and represent the present

The world around us and including us operates according to and can, in principle, be explained by the fundamental laws of physics. Nothing else is needed. But we do not think of the world around us as swarms of particles. This is simply because there are far too many of them, even to describe a grain of rice.

We the thinking creatures create our own worlds in our minds. And it is only our own creation that is meaningful to us. We create *categories* in our heads, for example, a human, a fox, a river, and a tree. We define an *ontology* in terms of which we represent the world. Every domain of study from physics to chemistry, to biology, to economics defines its own ontology. Or its own *terminology*. While the notion of an ontology has roots in philosophy, it has found a formal definition in computer science, as an *explicit specification of a conceptualization* (Gruber, 1993).

Behind every word of a language there sits a concept, and a knowledgeable agent relates that concept to others, within an ontology. For example, we associate the word 'banana' with a particular kind of fruit, growing on a particular kind of a single stem tree, which has a skin that can be peeled off before eating, and which has leaves that can be used to serve a meal on. Likewise, with verbs we can conjure up actions mentally, for example, jogging. In any case, whenever we use words in a language, they just stand for some concepts that an agent may have in its knowledge representation scheme. Roger Schank and his group at Yale university showed that the moment one talks of a person going into a restaurant one needs to retrieve all that one knows about what typically happens in a restaurant to make sense of the conversation (Schank and Abelson, 1977; Schank and Riesbeck, 1981). At around the same time, Marvin Minsky (1975) published his idea of *frames* for knowledge representation, which eventually led to the ideas of object oriented programming.

With the advent of the internet, when programs could talk to each other, defining ontologies gained prominence. Computational ontologies are a means to formally model the structure of a system, that is the relevant *entities* and *relations* that emerge from its observation, and which are useful to our purposes. The backbone of an ontology consists of a generalization/specialization hierarchy of concepts, that is a *taxonomy* (Guarino et al., 2009).

Figure 1.10 shows a snippet of a sample ontology represented as a frame system. The shaded squares are abstract frames, corresponding to concepts in an ontology. The *IS-A* slot in a concept defines an abstraction hierarchy, for example, 'a dog is a mammal'. The unshaded nodes represent individuals or instances of concepts. An ontology may have other kinds of links, for example, the fact that 'Ted is a dog owned by Socrates who is a human'.

The idea of *semantic networks* was already well developed. A semantic network is a graphical model in which nodes representing concepts are connected with labelled edges representing relations. Early work on semantic nets was motivated by natural language processing. Ross Quinlan is often credited with crystalizing the idea (Quillian, 1967, 1968). Subsequently, the idea of semantic nets evolved into the idea of *knowledge graphs*, which were semantic networks spread over the internet. The idea of the *Semantic Web* evolved in the twenty-first century. In 2012, Google adopted the term 'knowledge graph' (Singhal, 2012).

A knowledge graph is a collection of nodes and named edges. We can create an *abstract type* called *event* and describe the rest of the relations for an *instance* of that event. It has become common to express these as *triples* <subject, predicate, object> or <subject, property, value>. For example, here is an incident of kids fighting: 'Divya hit Atul with a stick yesterday afternoon in a park.'

(Hitting_event, Instance, *e45*)	: *e45* is an instance of Hitting_event
(*e45*, Actor, divya54)	: The actor of *e45* is Divya
(*e45*, Object, atul81)	: The object of *e45* is Atul

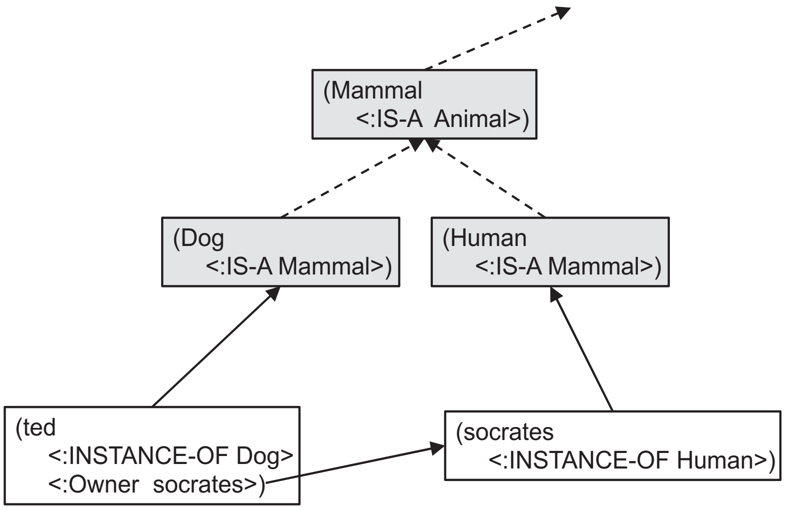

Figure 1.10 A snapshot of an ontology captured in a frame system. The shaded nodes are abstract concepts, and the unshaded ones are instances of concepts. An ontology may define an abstraction hierarchy as well as other kinds of relations.

(*e45*, Date, March_21)	: The date of *e45* is March 21
(*e45*, Loc, park28)	: The location of *e45* is the park
(*e45*, Instrument, stick14)	: The Instrument of *e45* is the stick

Figure 1.11 shows how information about musical performances and poetry could be represented as a knowledge graph.

Representation is only one side of the coin. *Reasoning* is the other.

We are never privy to everything there is to know. We have partial knowledge of the world and can try and fill in by making *inferences*. The process of making inferences is called *reasoning*. There are three kinds of inferences. All are in some way connected with a logical relation often captured as 'IF antecedent THEN consequent', expressed as a sentence in some logic. When the sentence is true, then it often expresses a causal connection from the antecedent to the consequent. But as Judea Pearl (2019) has shown to us, there can be confusion between causality and correlation, which has been exploited by the tobacco industry to contest the connection between smoking and cancer.

- Deduction: From a given set of facts, infer another fact that is necessarily true. Deduction is the bread and butter of logic. We study deduction in Chapter 10. Deduction is sound because it goes from the antecedent to the consequent. For example, the statement 'If X is a trapezium then X is a quadrilateral' is true by definition. So if anyone has drawn a trapezium, then she has drawn a quadrilateral. Deduction only makes explicit what is implicit in the knowledge base.

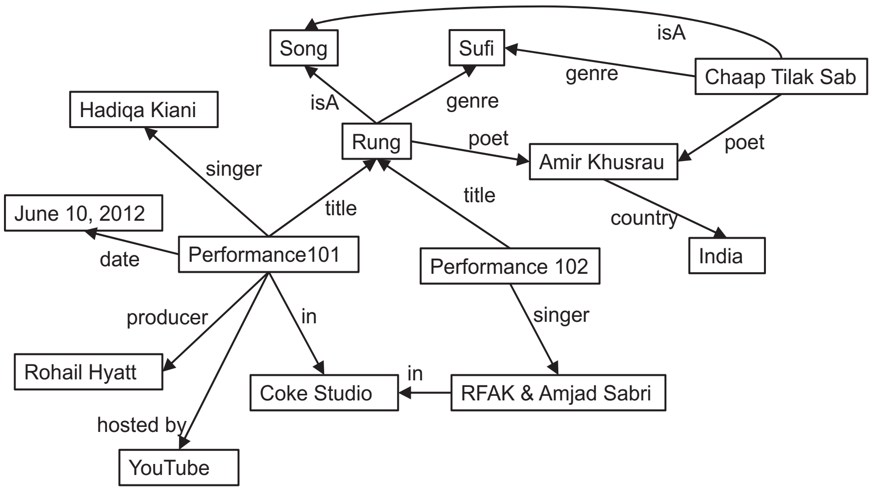

Figure 1.11 A triple subject, predicate, object store can store heterogeneous information in a knowledge graph, with nodes connected by directed edges. Each labelled edge goes from subject to object and is labelled by the property or the predicate. The figure shows a snippet of a knowledge graph relating to music and poetry.

- Induction: From a given sets of facts, infer a new fact. Also known as generalization. Induction can create new knowledge. Recognizing that a number of entities in the domain share some common property and asserting that as a general statement. For example, from the observations,

  The peepul leaf is green.
  The tamarind leaf is green.
  The neem leaf is green.
  The mango leaf is green.

  one can conclude,

  All leaves are green.

  Induction or generalization is the basis of machine learning. Such conclusions are, however, not sound, which means they are not necessarily true. One may know of a plant that does not have green leaves, for example, the Japanese maple. Nevertheless, the ability to generalize has tremendous use in practice.
- Abduction: From a given set of facts, infer another fact that is possibly true. For example, if one has cough and fever, one might hypothesize that one has COVID. But one might have a cough and fever from other causes as well. One cannot say with certainty what the cause is. Eric Larson (2021) explains this with a well known example. We know that if it rains,

the lawn will be wet. But we also know that if the sprinkler is on, the lawn will be wet, or if children are playing with a hose, the lawn will be wet, or if there was a flood (as is common in these times of climate change), the lawn will be wet. Then if we observe that the lawn is wet, how can we infer the cause? Which causal connection shall we make use of? Larson calls it the selection problem for AI. And yet, we manage to make abductive inferences all the time. Very often we use other facts we know. For example, we may know that the sky has been clear and so rain cannot be the cause of the grass being wet. Medical diagnosis, incidentally, is making abductive inferences. A doctor may suspect COVID if you have cough and cold, especially if a new wave has started. But the doctor relies on a clinical test to validate the hypothesis. Observe that we do not face this difficulty with deduction in which if we know the antecedent of a rule to be true, then the consequent necessarily follows.

Larson says that as humans the majority of the inferences we do are abductive in nature, and that is why they can be error prone too. The following scenario illustrates plausible inferences. If you are running with the ball in a football game, you need to be aware of where the other players are and what they intend to do. This inference of intention comes from background knowledge about the strategies and tactics used by the team. You should be able to imagine that if you kick the ball to where your teammate should be running to, then he would have a better shot at the goal. The opponents no doubt are thinking about it too. Why is the opposing team player running towards that spot? Making inferences is the basic cognitive act for intelligent minds and we are constantly making inferences.

Another example is the work done by Roger Schank and his group with stereotypical situations knowledge which is instrumental in generating expectations about what must have happened and what to expect. If we hear that 'John went to a restaurant. He ordered a masala dosa. He left satisfied,' we can imagine what must have happened because we have knowledge about how restaurants function, even though the story is cryptic. We know that he must eaten the dosa, and must have paid for it, because that is the normal behaviour in a restaurant.

In summary, the agent must be able to *reason* with what it knows to infer what is implicit (deduction) or even what is unknown (induction) to create new knowledge. It must be able to hypothesize connections between facts and events (abduction) to anticipate what is happening in the world around it, and what other agents are up to. It must be able to recognize intentions and plans of collaborators and adversaries, make its own plans, evaluate and choose the best ones, execute them, monitor them as they are executed, replan if necessary or take advantage of an unexpected opportunity. It must also be able to use the science of probability to judge which of its possible decisions is most likely to succeed.

The previous two subsections have described in a nutshell those aspects of AI that would each need a complete book for any justice to be done. We have briefly dwelt upon these to highlight the fact that they are necessary for building intelligent systems, along with other processes shown in Figure 1.6.

While all these are necessary, we now come to the subject matter of this book – *solving problems by first principles* by projecting decisions into the future to tease out those that solve the problem. The search methods that we study in this book arrive at solutions by trying out different options available to the agent. In the following section we outline the contents of this book.

1.5.3 Imagine the future and shape it

We have seen that having a memory enables an agent to capture and store experiences that can facilitate solving problems. Memory based reasoning allows us to reuse solutions from the past. Why would one want to reinvent the wheel every time? In addition, the faculty of language allows agents to share knowledge and experience allowing new researchers to 'stand on the shoulders of giants' (Hawking, 2003).

But the world we live in is not always so simple. For every nugget of wisdom, we can find one that says the exact opposite.[8] The Greek philosopher Heraclitus of Ephesus (535–475 BC) believed that change is a constant and famously said – *No man can step into the same river twice*. Problems are often similar, but not always. When faced with a new problem, the intelligent agent needs to solve it by first principles, *ab initio*. This book is devoted to such an approach to solving problems, in which the agent searches for solutions by projecting proposed actions into the future.

The ability to model the world is central to intelligent behaviour. This model must include not only the description of the world but also the moves available to the agent to achieve the goals it has. The algorithms that we study allow the agent to *imagine* the consequences of the actions and decisions and try out various combinations in a *mental simulation* to select the sequences that would work. The ability to imagine is perhaps the single most important characteristic of intelligence.

We now describe the chapterwise contents of this book.

Chapter 2 introduces the basic machinery of search. We begin by defining the state space, which is the space of all possible states. We define a neighbourhood function, *MoveGen*(*state*), that specifies the moves that an agent can make in any given state. The task of the search algorithm is to find a sequence of such moves from a given *start* state to a desired *goal* state. A function called *GoalTest*(*state*) inspects a state and identifies whether it is a goal state or not. The search algorithm begins at the *start* state, uses to MoveGen function to generate and navigate the state space. Then it either returns the solution or reports failure if it has exhausted all possibilities. In the process, it has generated and explored a part of the state space in the form of a *search tree*.

We also introduce the notion of a *solution space*, which is a space of *candidate solutions*. Instead of moves in the state space from one state to another, the neighbourhood function in the solution space is *perturbation* of candidates in search of the solution.

Chapter 3 introduces the simplest search strategies, *depth first search* and *breadth first search*, familiar to the students of graph algorithms. The difference in AI search is that the graphs are generated on the fly. Apart from time and space complexity, we also look at completeness and the quality of solution found. We compare the two algorithms and devise a new algorithm, *depth first iterative deepening*, that combines the best features of both. In all three cases, we recognize that the search spaces grow exponentially due to *combinatorial explosion* (*CombEx*). This leads us to explore various approaches to mitigate the effects of CombEx in the chapters that follow.

[8] Consider the two sayings – 'Out of sight, out of mind' versus 'Absence makes the heart grow fonder'.

Chapter 4 introduces *informed search* aimed to guide the search towards the goal instead of the *blind* or *uninformed* search methods of Chapter 3. We introduce the idea of a *heuristic* function $h(n)$ that looks at a state n and computes an *estimate* of its *distance* to the goal state. Search that employs such a heuristic function is called *heuristic search*. From the set of all available candidates, the algorithm *best first search* picks that node that appears to be closest to the goal. We show that if a solution exists, the algorithm will find it. That is, it is complete for finite state spaces. The performance of the algorithm depends upon the quality of heuristic estimate, and it has been empirically found that most implementations still need exponential time and space.

In an effort to save on space, we resort to *local* search. The algorithm *hill climbing* burns its bridges and considers only the neighbours of the current state (or candidate solution). Instead of using the GoalTest function, it looks for an optimum value of the heuristic function. While it does result in reduced complexity, it loses out on completeness. It does not guarantee finding a solution and can get stuck in a *local optimum*. Then begins the quest for variations in local search more likely to succeed in finding a solution. We look at the algorithm *beam search* that explores more than one path, and at *tabu search* that allows a search algorithm to get off a local optimum and continue exploration. We also introduce an aspect of stochastic search with *iterated hill climbing*.

Chapter 5 is devoted to stochastic local search methods. All the algorithms in the chapter draw inspiration from processes in nature. *Simulated annealing* begins with randomized moves and gradually makes them deterministic, reminiscent of the annealing process used to form materials in the lowest energy states. *Genetic algorithms* mimic the process of survival of the fittest in natural selection and mix and churn the components available in a population of candidates. We look at how genetic algorithms solve the travelling salesperson problem. Finally, we introduce the *ant colony optimization* algorithm that draws inspiration from how ants communicate with each other via pheromone trails and collaborate to find shortest paths. All the three algorithms studied are popular in the optimization community.

The algorithms studied so far do not guarantee an optimal solution (except breadth first search). Chapter 6 introduces the well known algorithm *A** that employs heuristic search and also guarantees an optimal solution even for infinite state spaces. We say that the algorithm is *admissible* and we present a proof of its admissibility. We introduce the problem of sequence alignment from bioinformatics where A* is applicable. We then look at space saving variations of A* that can solve much bigger problems than A* can.

Chapter 7 looks at problem decomposition and how problems can be solved in parts. We begin by looking at *pattern directed inference systems* in which an algorithm looks for patterns in the given state and triggers appropriate actions. The production system architecture lays the foundations of building *rule based expert systems*. We present the *Rete* algorithm which is an efficient implementation that is used in many *business rule management systems*. Such systems also serve as a vehicle of declarative programming exemplified by the language *OPS5* in which the user just writes the rules and an *inference engine* decides which rules to apply to what data.

We then look at a goal directed approach to breaking down problems into subproblems with *And–Or* graphs. The idea is to decompose the problem into simpler subproblems and continue the process till the reduced problems are primitive problems with trivial solutions. We present the algorithm AO* that can be used to find an optimal least cost decomposition strategy.

In Chapter 8, we turn to games as an example of adversarial reasoning. We introduce games that are popular amongst humans and computer programmers, and focus on two person board games like chess. We begin with the idea of the minimax value that is the Nash equilibrium for a game. Games like chess have game trees that are too large to be analysed completely though, and one needs to introduce the notion of an *evaluation function* that allows the *minimax algorithm* to find a move by limited lookahead. Then we look at efficient versions, *alpha beta pruning* and *SSS**, that do progressively more pruning of the game tree. We conclude with some commentary on the program AlphaGo that created a sensation by beating the *go* champion in 2016, and on the games backgammon and scrabble. Finally, we introduce contract bridge as an open challenge.

Chapter 9 is devoted to automated planning that has emerged as an independent area of research, with its own representation and move generation schemes. The community has devised an array of domain description languages allowing for richer descriptions and actions. A distinctive feature of planning domains is that the planning operators have an inbuilt arrow of time. When an action that is applicable is applied to a given state in a forward direction, it always results in another state. Searching backwards from goal descriptions is desirable because of low branching factor but runs into inconsistencies that require double checking the plan found. We look at a variation called *goal stack planning* that gets around this problem. We also look at *plan space planning* that searches in the solution space, allowing in principle separation of action selection from action scheduling. Next, we look at two stage approaches that first encode the planning problem into an intermediate representation and then solve that. Our focus in this chapter is on the algorithms that work in the simplest planning domains called *STRIPS* domains, but we do describe the more expressive planning domains that the community has engaged with.

Chapter 10 takes a quick look into the rich world of representation and reasoning. We look at the notions of *entailment* and *proof*, and at how deduction involves search. Our focus is on formal reasoning as has evolved in the logic community with emphasis on sound and complete theorem proving. We confine ourselves to *first order logic* that occupies a major part of the representation landscape in computer science. Our goal is to show how search is instrumental in reasoning. Like in planning, we explore both *forward reasoning* from facts to conclusions and *backward goal directed reasoning* for theorems to be proved. Both turn out to be incomplete, in the sense that given a knowledge base, there can be statements that are entailed, but the two algorithms are unable to find a proof for them. We then introduce the well known *resolution refutation method* that is both sound and complete. We also introduce the idea of logic as a programming language and also look at some more expressive languages briefly.

Chapter 11, written by Sutanu Chakraborti, takes a brief look at *machine learning*, highlighting the fact that the process of learning or training involves search in the space of possible models. A model encapsulates a function from the input to the output and usually computes the output quickly. This is because the relation between the input and output is acquired and made explicit in the learning phase. Models may go through extensive training phase, but the process of using them is quick. We look at some examples of training, including Bayesian classification, *k* nearest neighbours, decision trees, and neural networks. We also look at the K-means algorithm as an example of unsupervised learning.

Finally, Chapter 12 looks at *constraint satisfaction* which offers the tantalizing possibility of integrating the different kinds of processes needed for intelligence into one. We study finite domain *constraint satisfaction problems*, and show how search and reasoning can be combined, and look at some algorithms where reasoning is effective in reducing the search effort. The most attractive feature of constraints is that they offer a unifying formalism for representation, when solutions can be found by general purpose methods. Eugene Freuder (1997), one of the founding figures in constraint programming, has said – 'Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it'.

A Note for the Reader

The proof of the pudding is in the eating. It is all very well to speculate upon the nature of intelligence and whether machines will ever be able to think or not. There is no compelling argument that they will not be able to do so. At the same time, there is no convincing demonstration of artificial general intelligence so far. Yet machines continue to excel in individual tasks doing many useful things, from diagnosing diseases from images and radiographs, finding routes for us in a new city, controlling an autonomous robot on Mars, and beating us humans on all board games. All this is done with algorithms to tackle specific problems. Will they all come together as one entity that will solve all problems that have been solved independently? A key stumbling block is going to be knowledge representation. Can we create a general representation scheme in which all problems can be posed and solved? Then the machine will be capable of analogical thinking across diverse domains, and perhaps spot connections hitherto unknown. If we can do that, then we would have surpassed humans on this aspect, because no individual human is a master of all. All of us professionals in the modern world work with our narrow domain ontologies and are specialists in our areas of expertise. The frontiers of knowledge have expanded so much that the days of the Renaissance men like Leonardo da Vinci are over.

Meanwhile, the student of AI must acquire the tools of the trade. The building blocks. Understand the algorithms that are beneath the hood of a problem solver and learn to implement them. The crux of this book is a collection of 50 odd algorithms addressing different aspects of problem solving. These algorithms are described in pseudo code that may need getting used to. S Baskaran, an expert programmer, has put together *an appendix* that is at the end of the book. We urge the reader to study that before embarking upon the book. It should go a long way in your quest of a suite of programs for sophisticated, interesting, and useful applications.

Exercises

1. Alan Turing prescribed the Imitation Game as a test of whether machines can think or not. We call the test the Turing Test. Discuss the merits and demerits of the Turing Test. Is the ability to chat intelligently a sufficient indicator of intelligence? What is the role of world knowledge in a meaningful conversation? How should a machine react to a topic it does not know about and still convince the judge that he is chatting with a human? Should a machine introduce errors or delays intentionally, for example, when given a massive arithmetic problem?

2. Devise three sets of questions for the Winograd Schema Challenge that would require world knowledge to answer correctly.
3. Natural language is notoriously ambiguous, a fact that has been widely exploited to create punch lines that surprise the listener. For example, ‘Time flies like an arrow, fruit flies like a banana’ sometimes attributed to Groucho Marx. When humans parse language, they start building a semantic picture of what they are listening to. Garden path sentences force the listener to abandon an initial likely interpretation after hearing the complete sentence. For example, ‘The old man’s glasses were filled with sherry’. Given the sentence ‘She shot the girl with the rifle’, how would a computer chat program answer the following question ‘Who had the rifle?’
4. In Chapter 8, we discuss games as models of rational behaviours aimed at maximizing the agent’s own payoff. While this is an economic model that explains why individuals, corporates, and nations behave the way they do, what does it say about the collective intelligence of humankind whose focus is on arms manufacture, sale, and use, even while climate change looms upon us?