Contents

Acknowledgements

I thank the following friends for their very helpful advice (any mistakes, of course, are mine): Phil Husbands, Jeremy Reffin, Anil Seth, Aaron Sloman, and Blay Whitby. And I thank Latha Menon for her understanding and patience.

List of illustrations

Chapter 1

What is artificial intelligence?

Artificial intelligence (AI) seeks to make computers do the sorts of things that minds can do.

Some of these (e.g. reasoning) are normally described as ‘intelligent’. Others (e.g. vision) aren’t. But all involve psychological skills—such as perception, association, prediction, planning, motor control—that enable humans and animals to attain their goals.

Intelligence isn’t a single dimension, but a richly structured space of diverse information-processing capacities. Accordingly, AI uses many different techniques, addressing many different tasks.

And it’s everywhere.

AI’s practical applications are found in the home, the car (and the driverless car), the office, the bank, the hospital, the sky … and the Internet, including the Internet of Things (which connects the ever-multiplying physical sensors in our gadgets, clothes, and environments). Some lie outside our planet: robots sent to the Moon and Mars, or satellites orbiting in space. Hollywood animations, video and computer games, sat-nav systems, and Google’s search engine are all based on AI techniques. So are the systems used by financiers to predict movements on the stock market, and by national governments to help guide policy decisions in health and transport. So are the apps on mobile phones. Add avatars in virtual reality, and the toe-in-the-water models of emotion developed for ‘companion’ robots. Even art galleries use AI—on their websites, and also in exhibitions of computer art. Less happily, military drones roam today’s battlefields—but, thankfully, robot minesweepers do so too.

AI has two main aims. One is *technological*: using computers to get useful things done (sometimes by employing methods very *unlike* those used by minds). The other is *scientific*: using AI concepts and models to help answer questions about human beings and other living things. Most AI workers focus on only one of these, but some consider both.

Besides providing countless technological gizmos, AI has deeply influenced the life sciences.

In particular, AI has enabled psychologists and neuroscientists to develop powerful theories of the mind–brain. These include models of *how the physical brain works*, and—a different, but equally important, question—*just what it is that the brain is doing*: what computational (psychological) questions it is answering, and what sorts of information processing enable it to do that. Many unanswered questions remain, for AI itself has taught us that our minds are very much richer than psychologists had previously imagined.

Biologists, too, have used AI—in the form of 'artificial life' (A-Life), which develops computer models of differing aspects of living organisms. This helps them to explain various types of animal behaviour, the development of bodily form, biological evolution, and the nature of life itself.

Besides affecting the life sciences, AI has influenced philosophy. Many philosophers today base their accounts of mind on AI concepts. They use these to address, for instance, the notorious mind–body problem, the conundrum of free will, and the many puzzles regarding consciousness. However, these philosophical ideas are hugely controversial. And there are deep disagreements about whether any AI system could possess *real* intelligence, creativity, or life.

Last, but not least, AI has challenged the ways in which we think about humanity—and its future. Indeed, some people worry about whether we actually have a future, because they foresee AI surpassing human intelligence across the board. Although a few thinkers welcome this prospect, most dread it: what place will remain, they ask, for human dignity and responsibility?

All these issues are explored in the following chapters.

Virtual machines

‘To think about AI’, someone might say, ‘is to think about computers’. Well, yes and no. The computers, as such, aren’t the point. It’s what they *do* that matters. In other words, although AI needs *physical* machines (i.e. computers), it’s best thought of as using what computer scientists call *virtual* machines.

A virtual machine isn’t a machine depicted in virtual reality, nor something like a simulated car engine used to train mechanics. Rather, it’s the *information-processing system* that the programmer has in mind when writing a program, and that people have in mind when using it.

A word processor, for example, is thought of by its designer, and experienced by its users, as dealing directly with words and paragraphs. But the program itself usually contains neither. And a neural network (see Chapter 4) is thought of as doing information processing *in parallel*, even though it’s usually implemented in a (sequential) von Neumann computer.

That’s not to say that a virtual machine is just a convenient fiction, a thing merely of our imagination. Virtual machines are actual realities. They can make things happen, both inside the system and (if linked to physical devices such as cameras or robot hands) in the outside world. AI workers trying to discover what’s going wrong when a program does something unexpected only rarely consider hardware faults. Usually, they’re interested in the events and causal interactions in the *virtual* machinery, or software.

Programming languages, too, are virtual machines (whose instructions have to be translated into machine code before they can be run). Some are defined in terms of lower-level programming languages, so translation is required at several levels.

That’s not true only of programming languages. Virtual machines in general are comprised of patterns of activity (information processing) that exist at various levels. Moreover, it’s not true only of virtual machines running on computers. We’ll see in Chapter 6 that *the human mind* can be understood as the virtual machine—or rather, the set of mutually interacting virtual machines, running in parallel (and developed or learned at different times)—that is implemented in the brain.

Progress in AI requires progress in defining interesting/useful virtual machines. More *physically* powerful computers (larger, faster) are all very well. They may even be necessary for certain kinds of virtual machines to be implemented. But they can't be exploited unless *informationally* powerful virtual machines can be run on them. (Similarly, progress in neuroscience requires better understanding of what *psychological* virtual machines are being implemented by the physical neurons: see Chapter 7.)

Different sorts of external-world information are used. Every AI system needs input and output devices, if only a keyboard and a screen. Often, there are also special-purpose sensors (perhaps cameras, or pressure-sensitive whiskers) and/or effectors (perhaps sound synthesizers for music or speech, or robot hands). The AI program connects with—causes changes in—these computer–world interfaces as well as processing information internally.

AI processing usually also involves *internal* input and output devices, enabling the various virtual machines within the whole system to interact with each other. For example, one part of a chess program may detect a possible threat by noticing something happening in another, and may then interface with yet another in searching for a blocking move.

The major types of AI

How the information is processed depends on the virtual machine involved. As we'll see in later chapters, there are five major types, each including many variations. One is classical, or symbolic, AI—sometimes called GOFAI (Good Old-Fashioned AI). Another is artificial neural networks, or connectionism. In addition, there are evolutionary programming; cellular automata; and dynamical systems.

Individual researchers often use only one method, but *hybrid* virtual machines also occur. For instance, a theory of human action that switches continually between symbolic and connectionist processing is mentioned in Chapter 4. (This explains why, and how, it is that someone may be distracted from following through on a planned task by noticing something unrelated to it in the environment.) And a sensorimotor device that combines 'situated' robotics, neural networks, and evolutionary programming is described in Chapter 5. (This device helps a robot to find its way 'home' by using a cardboard triangle as a landmark.) Besides their practical applications, these approaches can illuminate mind, behaviour, and life. Neural networks are helpful for modelling aspects of the brain, and for doing pattern recognition and learning. Classical AI (especially when combined with statistics) can model learning too, and also planning and reasoning. Evolutionary programming throws light on biological evolution and brain development. Cellular automata and dynamical systems can be used to model development in living organisms. Some methodologies are closer to biology than to psychology, and some are closer to non-reflective behaviour than to deliberative thought. To understand the full range of mentality, all of them will be needed—and probably more.

Many AI researchers don't care about how minds work: they seek technological efficiency, not scientific understanding. Even if their techniques originated in psychology, they now bear scant relation to it. We'll see, however, that progress in general-purpose AI (artificial general intelligence, or AGI) will require deep understanding of the computational architecture of minds.

AI foreseen

AI was foreseen in the 1840s by Lady Ada Lovelace. More accurately, she foresaw *part* of it. She focused on symbols and logic, having no glimmering of neural networks, or of evolutionary and dynamical AI. Nor did she have any leanings towards AI’s psychological aim, her interest being purely technological.

She said, for instance, that a machine ‘might compose elaborate and scientific pieces of music of any degree of complexity or extent’, and might also express ‘the great facts of the natural world’ in enabling ‘a glorious epoch in the history of the sciences’. (So she wouldn’t have been surprised to see that, two centuries later, scientists are using ‘Big Data’ and specially crafted programming tricks to advance knowledge in genetics, pharmacology, epidemiology … the list is endless.) The machine she had in mind was the Analytical Engine. This gears-and-cogwheels device (never fully built) had been designed by her close friend Charles Babbage in 1834. Despite being dedicated to algebra and numbers, it was essentially equivalent to a general-purpose digital computer.

Ada Lovelace recognized the potential generality of the Engine, its ability to process symbols representing ‘all subjects in the universe’. She also described various basics of modern programming: stored programs, hierarchically nested subroutines, addressing, microprogramming, looping, conditionals, comments, and even bugs. But she said nothing about *just how* musical composition, or scientific reasoning, could be implemented on Babbage’s machine. AI was possible, yes—but how to achieve it was still a mystery.

How AI began

That mystery was clarified a century later by Alan Turing. In 1936, Turing showed that every possible computation can in principle be performed by a mathematical system now called a universal Turing machine. This imaginary system builds, and modifies, combinations of binary symbols—represented as ‘0’ and ‘1’. After codebreaking at Bletchley Park during World War II, he spent the rest of the 1940s thinking about how the abstractly defined Turing machine could be approximated by a physical machine (he helped design the first modern computer, completed in Manchester in 1948), and how such a contraption could be induced to perform intelligently.

Unlike Ada Lovelace, Turing accepted both goals of AI. He wanted the new machines to do useful things normally said to require intelligence (perhaps by using highly unnatural techniques), and also to model the processes occurring in biologically based minds.

The 1950 paper in which he jokily proposed the Turing Test (see Chapter 6) was primarily intended as a manifesto for AI. (A fuller version had been written soon after the war, but the Official Secrets Act prevented publication.) It identified key questions about the information processing involved in intelligence (game playing, perception, language, and learning), giving tantalizing hints about what had already been achieved. (Only ‘hints’, because the work at Bletchley Park was still top-secret.) It even suggested computational approaches—such as neural networks and evolutionary computing—that became prominent only much later. But the mystery was still far from dispelled. These were highly general remarks: programmatic, not programs.

Turing’s conviction that AI must be somehow possible was bolstered in the early 1940s by the neurologist/psychiatrist Warren McCulloch and the mathematician Walter Pitts. In their paper ‘A Logical Calculus of the Ideas Immanent in Nervous Activity’, they united Turing’s work with two other exciting items (both dating from the early 20th century): Bertrand Russell’s propositional logic and Charles Sherrington’s theory of neural synapses.

The key point about propositional logic is that it’s binary. Every sentence (also called a *proposition*) is assumed to be either *true* or *false*. There's no middle way, no recognition of uncertainty or probability. Only two 'truth-values' are allowed, namely *true* and *false*.

Moreover, complex propositions are built, and deductive arguments are carried out, by using logical operators (such as *and, or,* and *if–then*) whose meanings are defined in terms of the truth/falsity of the component propositions. For instance, if two (or more) propositions are linked by *and*, it's assumed that both/all of them are true. So 'Mary married Tom and Flossie married Peter' is true if, and only if, *both* 'Mary married Tom' and 'Flossie married Peter' are true.

Russell and Sherrington could be brought together by McCulloch and Pitts because they had both described binary systems. The *true/false* values of logic were mapped onto the *on/off* activity of brain cells and the *0/1* of individual states in Turing machines. Neurons were believed by Sherrington to be not only strictly on/off, but also to have fixed thresholds. So logic gates (computing *and, or,* and *not*) were defined as tiny neural nets, which could be interconnected to represent highly complex propositions. Anything that could be stated in propositional logic could be computed by some neural network, and by some Turing machine.

In brief, neurophysiology, logic, and computation were bundled together—and psychology came along too. McCulloch and Pitts believed (as many philosophers then did) that natural language boils down, in essence, to logic. So all reasoning and opinion, from scientific argument to schizophrenic delusions, was grist for their theoretical mill. They foresaw a time when, for the whole of psychology, 'specification of the [neural] net would contribute all that could be achieved in that field'.

The core implication was clear: *one and the same theoretical approach*—namely, Turing computation—could be applied to human and machine intelligence.

Turing, of course, agreed. But he couldn't take AI much further: the technology available was too primitive. In the mid-1950s, however, more powerful and/or easily usable machines were developed. 'Easily usable', here, means that it was easier to define new *virtual* machines (e.g. programming languages), which could be more easily used to define higher-level virtual machines (e.g. programs to do mathematics, or planning).

Symbolic AI research, broadly in the spirit of Turing’s manifesto, commenced on both sides of the Atlantic. One late-1950s landmark was Arthur Samuel’s checkers (draughts) player, which made newspaper headlines because it learned to beat Samuel himself. That was an intimation that computers might one day develop *super*human intelligence, outstripping the capacities of their programmers.

The second such intimation also occurred in the late 1950s, when the Logic Theory Machine not only proved eighteen of Russell’s key logical theorems, but found a more elegant proof of one of them. This was truly impressive. Whereas Samuel was only a mediocre checkers player, Russell was a world-leading logician. (Russell himself was delighted by this achievement, but the *Journal of Symbolic Logic* refused to publish a paper with a computer program named as an author, especially as it hadn’t proved a *new* theorem.)

The Logic Theory Machine was soon outdone by the General Problem Solver (GPS)—‘outdone’ not in the sense that GPS could surpass yet more towering geniuses, but in the sense that it wasn’t limited to only one field. As the name suggests, GPS could be applied to any problem that could be represented (as explained in Chapter 2) in terms of goals, sub-goals, actions, and operators. It was up to the programmers to identify the goals, actions, and operators relevant for any specific field. But once that had been done, the *reasoning* could be left to the program.

GPS managed to solve the ‘missionaries-and-cannibals’ problem, for example. (*Three missionaries and three cannibals on one side of a river; a boat big enough for two people; how can everyone cross the river, without cannibals ever outnumbering missionaries?*) That’s difficult even for humans, because it requires one to go backwards in order to go forwards. (Try it, using pennies!)

The Logic Theory Machine and GPS were early examples of GOFAI. They are now ‘old-fashioned’, to be sure. But they were also ‘good’, for they pioneered the use of *heuristics* and *planning*—both of which are hugely important in AI today (see Chapter 2).

GOFAI wasn’t the only type of AI to be inspired by the ‘Logical Calculus’ paper. Connectionism, too, was encouraged by it. In the 1950s, networks of McCulloch–Pitts logical neurons, either purpose-built or simulated on digital computers, were used (by Albert Uttley, for instance) to model associative learning and conditioned reflexes. (Unlike today’s neural networks, these did *local*, not *distributed*, processing: see Chapter 4.)

But early network modelling wasn’t wholly dominated by neuro-logic. The systems implemented (in analogue computers) by Raymond Beurle in the mid-1950s were very different. Instead of carefully designed networks of logic gates, he started from two-dimensional (2D) arrays of randomly connected, and varying-threshold, units. He saw neural self-organization as due to dynamical waves of activation—building, spreading, persisting, dying, and sometimes interacting.

As Beurle realized, to say that psychological processes could be *modelled by* a logic-chopping machine wasn’t to say that the brain *actually is* such a machine. McCulloch and Pitts had already pointed this out. Only four years after their first groundbreaking paper, they had published another one arguing that thermodynamics is closer than logic to the functioning of the brain. Logic gave way to statistics, single units to collectivities, and deterministic purity to probabilistic noise.

In other words, they had described what’s now called distributed, error-tolerant computing (see Chapter 4). They saw this new approach as an ‘extension’ of their previous one, not a contradiction of it. But it was more biologically realistic.

Cybernetics

McCulloch’s influence on early AI went even further than GOFAI and connectionism. His knowledge of neurology as well as logic made him an inspiring leader in the budding cybernetics movement of the 1940s.

The cyberneticians focused on biological self-organization. This covered various kinds of adaptation and metabolism, including autonomous thought and motor behaviour as well as (neuro) physiological regulation. Their central concept was ‘circular causation’, or feedback. And a key concern was teleology, or purposiveness. These ideas were closely related, for feedback depended on goal differences: the current distance from the goal was used to guide the next step.

Norbert Wiener (who designed anti-ballistic missiles during the war) named the movement in 1948, defining it as ‘the study of control and communication in the animal and the machine’. Those cyberneticians who did computer modelling often drew inspiration from control engineering and analogue computers rather than logic and digital computing. However, the distinction wasn’t clear-cut. For instance, goal differences were used both to control guided missiles and to direct symbolic problem solving. Moreover, Turing—the champion of classical AI—used dynamical equations (describing chemical diffusion) to define self-organizing systems in which novel structure, such as spots or segmentation, could emerge from a homogeneous origin (see Chapter 5).

Other early members of the movement included the experimental psychologist Kenneth Craik; the mathematician John von Neumann; the neurologists William Grey Walter and William Ross Ashby; the engineer Oliver Selfridge; the psychiatrist and anthropologist Gregory Bateson; and the chemist and psychologist Gordon Pask.

Craik, who died (aged 31) in a cycling accident in 1943, before the advent of digital computers, referred to analogue computing in thinking about the nervous system. He described perception and motor action, and intelligence in general, as guided by feedback from ‘models’ in the brain. His concept of cerebral models, or representations, would later be hugely influential in AI.

Von Neumann had puzzled about self-organization throughout the 1930s, and was hugely excited by McCulloch and Pitts' first paper. Besides changing his basic computer design from decimal to binary, he adapted their ideas to explain biological evolution and reproduction. He defined various cellular automata: systems made of many computational units, whose changes follow simple rules depending on the current state of neighbouring units. Some of these could replicate others. He even defined a universal replicator, capable of copying anything—itself included. Replication errors, he said, could lead to evolution.

Cellular automata were specified by von Neumann in abstract informational terms. But they could be embodied in many ways, for example, as self-assembling robots, Turing's chemical diffusion, Beurle's physical waves, or—as soon became clear—DNA.

From the late 1940s on, Ashby developed the Homeostat, an electrochemical model of physiological homeostasis. This intriguing machine could settle into an overall equilibrium state no matter what values were initially assigned to its hundred parameters (allowing almost 400,000 different starting conditions). It illustrated Ashby's theory of dynamical adaptation—both inside the body (not least, the brain) and between the body and its external environment, in trial-and-error learning and adaptive behaviour.

Grey Walter, too, was studying adaptive behaviour—but in a very different way. He built mini-robots resembling tortoises, whose sensorimotor circuitry modelled Sherrington's theory of neural reflexes. These pioneering situated robots displayed lifelike behaviours such as light-seeking, obstacle-avoidance, and associative learning via conditioned reflexes. They were exhibited to the general public at the Festival of Britain in 1951.

Ten years later, Selfridge (grandson of the founder of the London department store) used symbolic methods to implement an essentially parallel-processing system called Pandemonium.

This GOFAI program learned to recognize patterns by having many bottom-level 'demons', each always looking out for one simple perceptual input, which passed their results on to higher-level demons. These weighed the features recognized so far for consistency (e.g. only *two* horizontal bars in an **F**), downplaying any features that didn't fit. Confidence levels could vary, and they mattered: the demons that shouted loudest had the greatest effect. Finally, a master-demon chose the most plausible pattern, given the (often conflicting) evidence available. This research soon influenced both connectionism and symbolic AI. (One very recent offshoot is the LIDA model of consciousness: see Chapter 6.)

Bateson had little interest in machines, but he based his 1960s theories of culture, alcoholism, and ‘double-bind’ schizophrenia on ideas about communication (i.e. feedback) picked up earlier at cybernetic meetings. And from the mid-1950s on, Pask—described as ‘the genius of self-organizing systems’ by McCulloch—used cybernetic and symbolic ideas in many different projects. These included interactive theatre; intercommunicating musical robots; architecture that learned and adapted to its users’ goals; chemically self-organizing concepts; and teaching machines. The latter enabled people to take different routes through a complex knowledge representation, so were suitable for both step-by-step and holistic cognitive styles (and varying tolerance of irrelevance) on the learner’s part.

In brief, all the main types of AI were being thought about, and even implemented, by the late 1960s—and in some cases, much earlier than that.

Most of the researchers concerned are widely revered today. But only Turing was a constant spectre at the AI feast. For many years, the others were remembered only by some subset of the research community. Grey Walter and Ashby, in particular, were nearly forgotten until the late 1980s, when they were lauded (alongside Turing) as grandfathers of A-Life. To understand why, one must know how the computer modellers became disunited.

How AI divided

Before the 1960s, there was no clear distinction between people modelling language or logical thinking and people modelling purposive/adaptive motor behaviour. Some individuals worked on both. (Donald Mackay even suggested building hybrid computers, combining neural networks with symbolic processing.) And all were mutually sympathetic. Researchers studying physiological self-regulation saw themselves as engaged in the same overall enterprise as their psychologically oriented colleagues. They all attended the same meetings: the interdisciplinary Macy seminars in the USA (chaired by McCulloch from 1946 to 1951), and London's seminal conference on 'The Mechanization of Thought Processes' (organized by Uttley in 1958).

From about 1960, however, an intellectual schism developed. Broadly speaking, those interested in *life* stayed in cybernetics, and those interested in *mind* turned to symbolic computing. The network enthusiasts were interested in both brain and mind, of course. But they studied associative learning in general, not specific semantic content or reasoning, so fell within cybernetics rather than symbolic AI. Unfortunately, there was scant mutual respect between these increasingly separate sub-groups.

The emergence of distinct sociological coteries was inevitable. For the theoretical questions being asked—biological (of varying kinds) and psychological (also of varying kinds)—were different. So too were the technical skills involved: broadly defined, logic versus differential equations. Growing specialization made communication increasingly difficult, and largely unprofitable. Highly eclectic conferences became a thing of the past.

Even so, the division needn't have been so ill-tempered. The bad feeling on the cybernetic/connectionist side began as a mixture of professional jealousy and righteous indignation. These were prompted by the huge initial success of symbolic computing, by the journalistic interest attending the provocative term 'artificial intelligence' (coined by John McCarthy in 1956 to name what had previously been called 'computer simulation'), and by the arrogance—and unrealistic hype—expressed by some of the symbolists.

Members of the symbolist camp were initially less hostile, because they saw themselves as winning the AI competition. Indeed, they largely ignored the early network research, even though some of their leaders (Marvin Minsky, for instance) had started out in that area.

In 1958, however, an ambitious theory of neurodynamics—defining parallel-processing systems capable of self-organized learning from a random base (and error-tolerant to boot)—was presented by Frank Rosenblatt and partially implemented in his photoelectric Perceptron machine. Unlike Pandemonium, this didn't need the input patterns to be pre-analysed by the programmer. This novel form of connectionism couldn't be ignored by the symbolists. But it was soon contemptuously dismissed. As explained in Chapter 4, Minsky (with Seymour Papert) launched a stinging critique in the 1960s claiming that perceptrons are incapable of computing some basic things.

Funding for neural-network research dried up accordingly. This outcome, deliberately intended by the two attackers, deepened the antagonisms within AI.

To the general public, it now seemed that classical AI was the only game in town. Admittedly, Grey Walter's tortoises had received great acclaim in the Festival of Britain. Rosenblatt's Perceptron was hyped by the press in the late 1950s, as was Bernard Widrow's pattern-learning Adaline (based on signal-processing). But the symbolists' critique killed that interest stone dead. It was symbolic AI which dominated the media in the 1960s and 1970s (and which influenced the philosophy of mind as well).

That situation didn't last. Neural networks—as 'PDP systems' (doing parallel distributed processing)—burst onto the public stage again in 1986 (see Chapter 4). Most outsiders—and some insiders, who should have known better—thought of this approach as utterly *new*. It seduced the graduate students, and attracted enormous journalistic (and philosophical) attention. Now, it was the symbolic AI people whose noses were put out of joint. PDP was in fashion, and classical AI was widely said to have failed.

As for the other cyberneticians, they finally came in from the cold with the naming of A-Life in 1987. The journalists, and the graduate students, followed. Symbolic AI was challenged yet again.

In the 21st century, however, it has become clear that different questions require different types of answers—horses for courses. Although traces of the old animosities remain, there’s now room for respect, and even cooperation, between different approaches. For instance, ‘deep learning’ is sometimes used in powerful systems combining symbolic logic with multilayer probabilistic networks; and other hybrid approaches include ambitious models of consciousness (see Chapter 6).

Given the rich variety of virtual machines that constitute the human mind, one shouldn’t be too surprised.

Chapter 2

General intelligence as the Holy Grail

State-of-the-art AI is a many-splendoured thing. It offers a profusion of virtual machines, doing many different kinds of information processing. There's no key secret here, no core technique unifying the field: AI practitioners work in highly diverse areas, sharing little in terms of goals and methods. This book can mention only very few of the recent advances. In short, AI's methodological range is extraordinarily wide.

One could say that it's been astonishingly successful. For its practical range, too, is extraordinarily wide. A host of AI applications exist, designed for countless specific tasks and used in almost every area of life, by laymen and professionals alike. Many outperform even the most expert humans. In that sense, progress has been spectacular.

But the AI pioneers weren't aiming only for specialist systems. They were also hoping for systems with *general* intelligence. Each human-like capacity they modelled—vision, reasoning, language, learning, and so on—would cover its entire range of challenges. Moreover, these capacities would be integrated when appropriate.

Judged by those criteria, progress has been far less impressive. John McCarthy recognized AI's need for 'common sense' very early on. And he spoke on 'Generality in Artificial Intelligence' in both of his high-visibility Turing Award addresses, in 1971 and 1987—but he was complaining, not celebrating. In 2018, his complaints aren't yet answered.

The 21st century is seeing a revival of interest in AGI, driven by recent increases in computer power. If that were achieved, AI systems could rely less on special-purpose programming tricks, benefitting instead from general powers of reasoning and perception—plus language, creativity, and emotion (all of which are discussed in Chapter 3).

However, that’s easier said than done. General intelligence is still a major challenge, still highly elusive. AGI is the field’s Holy Grail.

Supercomputers aren’t enough

Today’s supercomputers are certainly a help to anyone seeking to realize this dream. The combinatorial explosion—wherein more computations are required than can actually be executed—is no longer the constant threat that it used to be. Nevertheless, problems can’t always be solved merely by increasing computer power.

New problem-solving *methods* are often needed. Moreover, even if a particular method *must* succeed in principle, it may need too much time and/or memory to succeed in practice. Three such examples (concerning neural networks) are given in Chapter 4.

Efficiency is important, too: the fewer the number of computations, the better. In short, problems must be made tractable.

There are several basic strategies for doing that. All were pioneered by classical symbolic AI, or GOFAI, and all are still essential today.

One is to direct attention to only a part of the *search space* (the computer’s representation of the problem, within which the solution is assumed to be located). Another is to construct a smaller search space by making simplifying assumptions. A third is to order the search efficiently. Yet another is to construct a different search space, by representing the problem in a new way.

These approaches involve *heuristics, planning, mathematical simplification*, and *knowledge representation*, respectively. The next five sections consider those general AI strategies.

Heuristic search

The word 'heuristic' has the same root as '*Eureka!*': it comes from the Greek for *find*, or *discover*. Heuristics were highlighted by early GOFAI, and are often thought of as 'programming tricks'. But the term didn't originate with programming: it has long been familiar to logicians and mathematicians.

Whether in humans or machines, heuristics make it easier to solve the problem. In AI, they do this by directing the program towards certain parts of the search space and away from others.

Many heuristics, including most of those used in the very early days of AI, are rules of thumb that aren't guaranteed to succeed. The solution may lie in some part of the search space that the heuristic has led the system to ignore. For example, 'Protect your queen' is a very helpful rule in chess, but it should occasionally be disobeyed.

Others can be logically or mathematically proved to be adequate. Much work in AI and computer science today aims to identify provable properties of programs. That's one aspect of 'Friendly AI', because human safety may be jeopardized by the use of logically unreliable systems (see Chapter 7).

Whether reliable or not, heuristics are an essential aspect of AI research. The increasing AI specialism mentioned earlier depends partly on the definition of new heuristics that can improve efficiency spectacularly, but only in one highly restricted sort of problem, or search space. A hugely successful heuristic may not be suitable for 'borrowing' by other AI programs.

Given several heuristics, their order of application may matter. For instance, 'Protect your queen' should be taken into account before 'Protect your bishop'—even though this ordering will occasionally lead to disaster. Different orderings will define different search trees through the search space. Defining and ordering heuristics are crucial tasks for modern AI. (Heuristics are prominent in cognitive psychology, too. Intriguing work on 'fast and frugal heuristics', for example, indicates how evolution has equipped us with efficient ways of responding to the environment.) Heuristics make brute-force search through the entire search space unnecessary. But they are sometimes combined with (limited) brute-force search. IBM’s chess program *Deep Blue*, which caused worldwide excitement by beating world champion Gary Kasparov in 1997, used dedicated hardware chips, processing 200 million positions per second, to generate every possible move for the next eight.

However, it had to use heuristics to select the ‘best’ move within them. And since its heuristics weren’t reliable, even *Deep Blue* didn’t beat Kasparov *every* time.

Planning

Planning, too, is prominent in today's AI—not least in a wide range of military activities. Indeed, the USA's Department of Defense, which paid for the majority of AI research until very recently, has said that the money saved (by AI planning) on battlefield logistics in the first Iraq war outweighed all their previous investment.

Planning isn't restricted to AI: we all do it. Think of packing for your holiday, for instance. You have to find all the things you want to take, which probably won't all be found in the same place. You may have to buy some new items (sun cream, perhaps). You must decide whether to collect all the things together (perhaps on your bed, or on a table), or whether to put each one in your luggage when you find it. That decision will depend in part on whether you want to put the clothes in last of all, to prevent creasing. You'll need a rucksack, or a suitcase, or maybe two: how do you decide?

The GOFAI programmers who used planning as an AI technique had such consciously thought-out examples in mind. That's because the pioneers responsible for the Logic Theory Machine (see Chapter 1) and GPS were primarily interested in the psychology of human reasoning.

Modern AI planners don't rely so heavily on ideas garnered from conscious introspection or experimental observation. And their plans are very much more complex than was possible in the early days. But the basic idea is the same.

A plan specifies a sequence of actions, represented at a general level—a final goal, plus sub-goals and sub-sub-goals …—so that all details aren't considered at once. Planning at a suitable level of abstraction can lead to tree pruning within the search space, so some details never need to be considered at all. Sometimes, the final goal is *itself* a plan of action—perhaps scheduling the deliveries to and from a factory or battlefield. At other times, it's the answer to a question—for example, a medical diagnosis.

For any given goal, and expected situations, the planning program needs: a list of actions—that is, symbolic operators—or action types (instantiated by filling in parameters derived from the problem), each of which can make some relevant

[illegible]
change; for every action, a set of necessary prerequisites (cf. to grasp something, it must be within reach); and heuristics for prioritizing the required changes and ordering the actions. If the program decides on a particular action, it may have to set up a new sub-goal to satisfy the prerequisites. This goal-formulating process can be repeated again and again.

Planning enables the program—and/or the human user—to discover what actions have already been taken, and why. The ‘why’ refers to the goal hierarchy: *this* action was taken to satisfy *that* prerequisite, to achieve *such-and-such* a sub-goal. AI systems commonly employ techniques of ‘forward-chaining’ and ‘backward-chaining’, which explain how the program found its solution. This helps the user to judge whether the action/advice of the program is appropriate.

Some current planners have tens of thousands of lines of code, defining hierarchical search spaces on numerous levels. These systems are often significantly different from the early planners.

For example, most don’t assume that all the sub-goals can be worked on independently (i.e. that problems are *perfectly decomposable*). In real life, after all, the result of one goal-directed activity may be undone by another. Today’s planners can handle *partially decomposable* problems: they work on sub-goals independently, but can do extra processing to combine the resulting sub-plans if necessary.

The classical planners could tackle only problems in which the environment was fully observable, deterministic, finite, and static. But some modern planners can cope with environments that are partially observable (i.e. the system’s model of the world may be incomplete and/or incorrect) and probabilistic. In those cases, the system must monitor the changing situation during execution, so as to make changes in the plan—and/or in its own ‘beliefs’ about the world—as appropriate. Some modern planners can do this over very long periods: they engage in continuous goal formulation, execution, adjustment, and abandonment, according to the changing environment.

Many other developments have been added, and are still being added, to classical planning. It may seem surprising, then, that planning was roundly rejected by some roboticists in the 1980s, ‘situated’ robotics being recommended instead (see Chapter 5). The notion of internal representation—of goals and possible actions, for example—was rejected as well. However, that criticism was largely mistaken. Robotics often needs planning as well as purely reactive responses—to build soccer-playing robots, for instance.

Mathematical simplification

Whereas heuristics leave the search space as it is (making the program focus on only part of it), simplifying assumptions construct an unrealistic—but computationally tractable—search space.

Some such assumptions are mathematical. One example is the ‘i.i.d.’ assumption, commonly used in machine learning. This represents the probabilities in the data as being much simpler than they actually are.

The advantage of mathematical simplification when defining the search space is that mathematical—that is, clearly definable and, to mathematicians at least, readily intelligible—methods of search can be used. But that’s not to say that *any* mathematically defined search will be useful. As noted earlier, a method that’s mathematically *guaranteed* to solve every problem within a certain class may be unusable in real life, because it would need infinite time to do so. It may, however, suggest approximations that are more practicable: see the discussion of ‘backprop’ in Chapter 4.

Non-mathematical simplifying assumptions in AI are legion—and often unspoken. One is the (tacit) assumption that problems can be defined and solved without taking emotions into account (see Chapter 3). Many others are built into the general knowledge representation that’s used in specifying the task.

Knowledge representation

Often, the hardest part of AI problem solving is presenting the problem to the system in the first place. Even if it *seems* that someone can communicate directly with a program—by speaking in English to *Siri*, perhaps, or by typing French words into Google's search engine—they can't. Whether one's dealing with texts or with images, the information ('knowledge') concerned must be presented to the system in a fashion that the machine can understand—in other words, that it can deal with. (Whether that is *real* understanding is discussed in Chapter 6.)

AI's ways of doing this are highly diverse. Some are developments/variations of general methods of knowledge representation introduced in GOFAI. Others, increasingly, are highly specialized methods, tailor-made for a narrow class of problems. There may be, for instance, a new way of representing X-ray images, or photographs of a certain class of cancerous cells, carefully tailored to enable some highly specific method of medical interpretation (so, no good for recognizing cats, or even CAT scans).

In the quest for AGI, the *general* methods are paramount. Initially inspired by psychological research on human cognition, these include: sets of IF–THEN rules; representations of individual concepts; stereotyped action sequences; semantic networks; and inference by logic or probability.

Let's consider each of these in turn. (Another form of knowledge representation, namely neural networks, is described in Chapter 4.)

Rule-based programs

In rule-based programming, a body of knowledge/belief is represented as a set of IF–THEN rules linking Conditions to Actions: IF this Condition is satisfied, THEN take that Action. This form of knowledge representation draws on formal logic (Emil Post's 'production' systems). But the AI pioneers Allen Newell and Herbert Simon believed it to underlie human psychology in general.

Both Condition and Action may be complex, specifying a conjunction (or disjunction) of several—perhaps many—items. If several Conditions are satisfied simultaneously, the most inclusive conjunction is given priority. So 'IF the goal is to cook roast beef and Yorkshire pudding' will take precedence over 'IF the goal is to cook roast beef '—and adding 'and three veg' to the Condition will trump that.

Rule-based programs don't specify the order of steps in advance. Rather, each rule lies in wait to be triggered by its Condition. Nevertheless, such systems can be used to do planning. If they couldn't, they would be of limited use for AI. But they do it differently from how it's done in the oldest, most familiar, form of programming (sometimes called 'executive control').

In programs with executive control planning is represented explicitly. The programmer specifies a sequence of goal-seeking instructions to be followed step by step, in strict temporal order: '*Do this*, then *do that*; then *look to see* whether *X* is true; if it is, *do such-and-such*; if not, *do so-and-so*'.

Sometimes, the '*this*' or the '*so-and-so*' is an explicit instruction to set up a goal or sub-goal. For instance, a robot with the goal of leaving the room may be instructed to set up the sub-goal of opening the door; next, if examining the current state of the door shows it to be closed, set up the sub-sub-goal of grasping the door handle. (A human toddler may need a sub-sub-sub-goal—namely, getting an adult to grasp the unreachable door handle; and the infant may need several goals at even lower levels in order to do that.)

A rule-based program, too, could work out how to escape from the room. However, the plan hierarchy would be represented not as a temporally ordered sequence of explicit steps, but as the logical structure *implicit* in the collection of IF–THEN rules that comprise the system. A Condition may require that such-and-such a goal has already been set up (IF you want to open the door, and you aren't tall enough). Similarly, an Action can include the setting up of a new goal or sub-goal (THEN ask an adult). Lower levels will be activated automatically (IF you want to ask someone to do something, THEN set up the goal of moving near to them).

Of course, the programmer has to have included the relevant IF–THEN rules (in our example, rules dealing with doors and door handles). But he/she doesn't need to have anticipated all the potential logical implications of those rules. (That's a curse, as well as a blessing, because potential inconsistencies may remain undiscovered for quite a while.)

The active goals/sub-goals are posted on a central 'blackboard', which is accessible to the whole system. The information displayed on the blackboard includes not only activated goals but also perceptual input, and other aspects of current processing. (That idea has influenced a leading neuropsychological theory of consciousness, and an AI model of consciousness based on it: see Chapter 6.)

Rule-based programs were widely used for the pioneering 'expert systems' of the early 1970s. These included MYCIN, which offered advice to human physicians on identifying infectious diseases and on prescribing antibiotic drugs, and DENDRAL, which performed spectral analysis of molecules within a particular area of organic chemistry. MYCIN, for instance, did medical diagnosis by matching symptoms and background bodily properties (Conditions) to diagnostic conclusions and/or suggestions for further tests or medication (Actions). Such programs were AI's first move away from the hope of generalism towards the practice of specialism. And they were the first step towards Ada Lovelace's dream of machine-made science (see Chapter 1).

The rule-based form of knowledge representation enables programs to be built gradually, as the programmer—or perhaps an AGI system itself—learns more about the domain. A new rule can be added at any time. There's no need to rewrite the program from scratch. However, there's a catch. If the new rule isn't logically consistent with the existing ones, the system won't always do what it's supposed to do. It may not even *approximate* what it's supposed to do. When dealing with a small set of rules, such logical conflicts are easily avoided, but larger systems are less transparent.

In the 1970s, the new IF–THEN rules were drawn from ongoing conversations with human experts, asked to explain their decisions. Today, many of the rules don’t come from conscious introspection. But they are even more efficient. Modern expert systems (a term rarely used today) range from huge programs used in scientific research and commerce to humble apps on phones. Many outperform their predecessors because they benefit from additional forms of knowledge representation, such as statistics and special-purpose visual recognition, and/or the use of Big Data (see Chapter 4).

These programs can assist, or even replace, human experts in narrowly restricted fields. Now, there are countless examples, used to aid human professionals working in science, medicine, law … and even dress design. (Which isn’t entirely good news: see Chapter 7.)

Frames, word-vectors, scripts, semantic nets

Other commonly used methods of knowledge representation concern individual concepts, not entire domains (such as medical diagnosis or dress design).

For instance, one can tell a computer what a room is by specifying a hierarchical data structure (sometimes called a ‘frame’). This represents a *room* as having *floor, ceiling, walls, doors, windows,* and *furniture* (*bed, bath, dining table* …). Actual rooms have varying numbers of walls, doors, and windows, so ‘slots’ in the frame allow specific numbers to be filled in—and provide default assignments too (four walls, one door, one window).

Such data structures can be used by the computer to find analogies, answer questions, engage in a conversation, or write or understand a story. And they’re the basis of CYC: an ambitious—some would say vastly overambitious—attempt to represent all human knowledge.

Frames can be misleading, however. Default assignments, for instance, are problematic. (Some rooms have no window, and open-plan rooms have no door.) Worse: what of everyday concepts such as *dropping*, or *spilling*? Symbolic AI represents our common-sense knowledge of ‘naïve physics’ by constructing frames coding such facts as that a physical object will drop if unsupported. But a helium balloon won’t. Allowing explicitly for such cases is a never-ending task.

In some applications using recent techniques for dealing with Big Data, a single concept may be represented as a cluster, or ‘cloud’, made up of hundreds or thousands of sometimes-associated concepts, with the probabilities of the many paired associations being distinguished: see Chapter 3. Similarly, concepts can now be represented by ‘word-vectors’ rather than words. Here, semantic features that contribute to, and connect, many different concepts are discovered by the (deep-learning) system, and used to predict the following word—in machine translation, for instance. However, these representations aren’t yet as amenable for use in reasoning or conversation, as classical frames.

Some data structures (called ‘scripts’) denote familiar action sequences. For instance, putting a child to bed often involves tucking them up, reading a story, singing a lullaby, and switching on the night light. Such data structures can be used for question-answering, and also for *suggesting* questions. If a mother omits the night light, questions can arise about *Why?* and *What happened next?* In other words, therein lies the seed of a story. Accordingly, this form of knowledge representation is used for automatic story-writing—and would be needed by ‘companion’ computers capable of engaging in normal human conversation (see Chapter 3).

An alternative form of knowledge representation for concepts is semantic networks (these are *localist* networks: see Chapter 4). Pioneered by Ross Quillian in the 1960s as models of human associative memory, several extensive examples (e.g. *WordNet*) are now available as public data resources. A semantic network links concepts by semantic relations such as *synonymy, antonymy, subordination, super-ordination*, *part–whole*—and often also by associative linkages assimilating *factual* world-knowledge to semantics (see Chapter 3).

The network may represent words as well as concepts, by adding links coding for *syllables, initial letters, phonetics,* and *homonyms*. Such a network is used by Kim Binsted’s JAPE and Graeme Ritchie’s STAND UP, which generate jokes (of nine different types) based on puns, alliteration, and syllable-switching. For example: *Q: What do you call a depressed train? A: A low-comotive*; *Q: What do you get if you mix a sheep with a kangaroo? A: A woolly jumper*.

A *caveat:* semantic networks aren’t the same thing as neural networks. As we’ll see in Chapter 4, *distributed* neural networks represent knowledge in a very different way. There, individual concepts are represented not by a single node in a carefully defined associative net, but by the changing pattern of activity across an entire network. Such systems can tolerate conflicting evidence, so aren’t bedevilled by the problems of maintaining logical consistency (to be described in the next section). But they can’t do precise inference. Nevertheless, they’re a sufficiently important type of knowledge representation (and a sufficiently important basis for practical applications) to merit a separate chapter.

Logic and the semantic web

If one’s ultimate aim is AGI, logic seems highly appropriate as a knowledge representation. For logic is *generally* applicable. In principle, the same representation (the same logical symbolism) can be used for vision, learning, language, and so on, and for any integration thereof. Moreover, it provides powerful methods of theorem proving to handle the information.

That’s why the preferred mode of knowledge representation in early AI was the predicate calculus. This form of logic has more representational power than propositional logic, because it can ‘get inside’ sentences to express their meaning. For example, consider the sentence ‘This shop has a hat to fit everyone’. Predicate calculus can clearly distinguish these three possible meanings: ‘For every human individual, there exists in this shop some hat that will fit them’; ‘There exists in this shop a hat whose size can be varied so as to fit any human being’; and ‘In this shop there exists a hat [presumably folded up!] large enough to fit all human beings simultaneously’.

For many AI researchers, predicate logic is still the preferred approach. CYC’s frames, for example, are based in predicate logic. So are the natural language processing (NLP) representations in compositional semantics (see Chapter 3). Sometimes, predicate logic is extended so as to represent time, cause, or duty/morality. Of course, that depends on someone’s having developed those forms of modal logic—which isn’t easy.

However, logic has disadvantages, too. One involves the combinatorial explosion. AI’s widely used ‘resolution’ method for logical theorem proving can get bogged down in drawing conclusions that are true but irrelevant. Heuristics exist for guiding, and restricting, the conclusions—and for deciding when to give up (which the Sorcerer’s Apprentice couldn’t do). But they aren’t foolproof.

Another is that resolution theorem proving assumes that *not-not-X* implies *X*. If the domain being reasoned about is completely understood, that’s logically correct. But users of programs (such as many expert systems) with built-in resolution often assume that failure to find a contradiction implies that no contradiction exists—so-called ‘negation by failure’. Usually, that’s a mistake.

In real life, there’s a big difference between proving that something is false and failing to prove that it’s true (think of wondering whether or not your partner is cheating on you).

A third disadvantage is that in classical (‘monotonic’) logic, once something is proved to be true, it stays true. In practice, that’s not always so. One may accept X for good reason (perhaps it was a default assignment, or even a conclusion from careful argument and/or strong evidence), but it can turn out later that X is no longer true—or wasn’t true in the first place. If so, one must revise one’s beliefs accordingly. Given a logic-based knowledge representation, that’s easier said than done. Many researchers, inspired by McCarthy, have tried to develop ‘non-monotonic’ logics that can tolerate changing truth-values. Similarly, people have defined various ‘fuzzy’ logics, where a statement can be labelled as *probable/improbable*, or as *unknown*, rather than *true*/*false*. Even so, no reliable defence against monotonicity has been found.

AI researchers developing logic-based knowledge representation are increasingly seeking the ultimate atoms of knowledge, or meaning, *in general*. They aren’t the first: McCarthy and Hayes did so in ‘Some Philosophical Problems from an AI Standpoint’. That early paper addressed many familiar puzzles, from free will to counterfactuals. These included questions about the basic ontology of the universe: states, events, properties, changes, actions … *what?*

Unless one is a metaphysician at heart (a rare human passion), why should one care? And why should these arcane questions be ‘increasingly’ pursued today? Broadly, the answer is that trying to design AGI raises questions about what ontologies the knowledge representation can use. These questions arise also in designing the semantic web.

The semantic web isn’t the same as the World Wide Web—which we’ve had since the 1990s. For the semantic web isn’t even state of the art: it’s state of the future. If and when it exists, machine-driven associative search will be improved and supplemented by machine understanding. This will enable apps and browsers to access information from anywhere on the Internet, and to integrate different items sensibly in reasoning about questions. That’s a tall order. Besides requiring huge engineering advances in hardware and communications infrastructure, this ambitious project (directed by Sir Tim Berners-Lee) needs to deepen the Web-roaming programs’ understanding of what they’re doing.

Search engines like Google’s, and NLP programs in general, can find associations between words and/or texts—but there’s no understanding there. Here, this isn’t a philosophical point (for that, see Chapter 6), but an empirical one—and a further obstacle to achieving AGI. Despite some seductively deceptive examples—such as WATSON, *Siri*, and machine translation (all discussed in Chapter 3)—today’s computers don’t grasp the meaning of what they ‘read’ or ‘say’.

Computer vision

Today’s computers don’t understand visual images as humans do, either. (Again, this is an *empirical* point: whether AGIs could have conscious visual phenomenology is discussed in Chapter 6.)

Since 1980, the various knowledge representations used for AI vision have drawn heavily on psychology—especially the theories of David Marr and James Gibson. Despite such psychological influences, however, current visual programs are gravely limited.

Admittedly, computer vision has achieved remarkable feats: facial recognition with 98 per cent success, for instance. Or reading cursive handwriting. Or noticing someone behaving suspiciously (continually pausing by car doors) in parking lots. Or identifying certain diseased cells, better than human pathologists can. Faced with such successes, one’s mind is strongly tempted to boggle.

But the programs (many are neural networks: see Chapter 4) usually have to know exactly what they’re looking for: for example, a face *not* upside down, *not* in profile, *not* partly hidden behind something else, and (for 98 per cent success) lit in a particular way.

That word ‘usually’ is important. In 2012, Google’s Research Laboratory integrated 1,000 large (sixteen-core) computers to form a huge neural network, with over a billion connections. Equipped with deep learning, it was presented with ten million random images from YouTube videos. It wasn’t told what to look for, and the images weren’t labelled. Nevertheless, after three days one unit (one artificial neuron) had learned to respond to images of a cat’s face, and another to human faces.

Impressive? Well, yes. Intriguing, too: the researchers were quick to recall the idea of ‘grandmother cells’ in the brain. Ever since the 1920s, neuroscientists have differed over whether or not these exist. To say that they do is to say that there are cells in the brain (either single neurons or small groups of neurons) that become active when, and only when, a grandmother, or some other specific feature, is perceived. Apparently, something analogous was going on in

Google’s cat-recognizing network. And although the cats’ faces had to be full on and the right way up, they could vary in size, or appear in different positions within the 200 × 200 array. A further study, which trained the system on carefully pre-selected (but unlabelled) images of human faces, *including some in profile*, resulted in a unit that could sometimes—only *sometimes*—discriminate faces turned away from the viewer.

There are now many more—and even more impressive—such achievements. Multilayer networks have already made huge advances in face recognition, and can sometimes find the most salient part of an image and generate a verbal caption (e.g. ‘people shopping in an outdoor market’) to describe it. The recently initiated *Large Scale Visual Recognition Challenge* is annually increasing the number of visual categories that can be recognized, and decreasing the constraints on the images concerned (e.g. the number and occlusion of objects). However, these deep-learning systems will still share some of the weaknesses of their predecessors.

For instance, they—like the cat’s-face recognizer—will have no understanding of 3D space, no knowledge of what a ‘profile’, or occlusion, actually is. Even vision programs designed for robots provide only an inkling of such matters.

The Mars Rover robots, such as *Opportunity* and *Curiosity* (landed in 2004 and 2012, respectively), rely on special knowledge-representation tricks: heuristics tailored for the 3D problems they’re expected to face. They can’t do pathfinding or object manipulation in the general case. Some robots simulate *animate* vision, wherein the body’s own movements provide useful information (because they change the visual input systematically). But even they can’t notice a possible pathway, or recognize that *this* unfamiliar thing could be picked up by their robot hand whereas *that* could not.

By the time this book is published, there may be some exceptions. But they too will have limits. For instance, they won’t understand ‘I can’t pick that up’, because they won’t understand *can* and *cannot*. That’s because the requisite modal logic probably still won’t be available for their knowledge representation.

Sometimes, vision can ignore 3D space—when reading handwriting, for instance.

But even 2D computer vision is limited. Despite considerable research effort on *analogical*, or *iconic*, representations, AI can't reliably use diagrams in problem solving—as we do in geometrical reasoning, or in sketching abstract relationships on the back of an envelope. (Similarly, psychologists don't yet understand just how *we* do those things.)

In short, most human visual achievements surpass today's AI. Often, AI researchers aren't clear about what questions to ask. For instance, think about folding a slippery satin dress neatly. No robot can do this (although some can be instructed, step by step, how to fold an oblong terry towel). Or consider putting on a T-shirt: the head must go in first, and *not* via a sleeve—but *why*? Such topological problems hardly feature in AI.

None of this implies that human-level computer vision is impossible. But achieving it is much more difficult than most people believe.

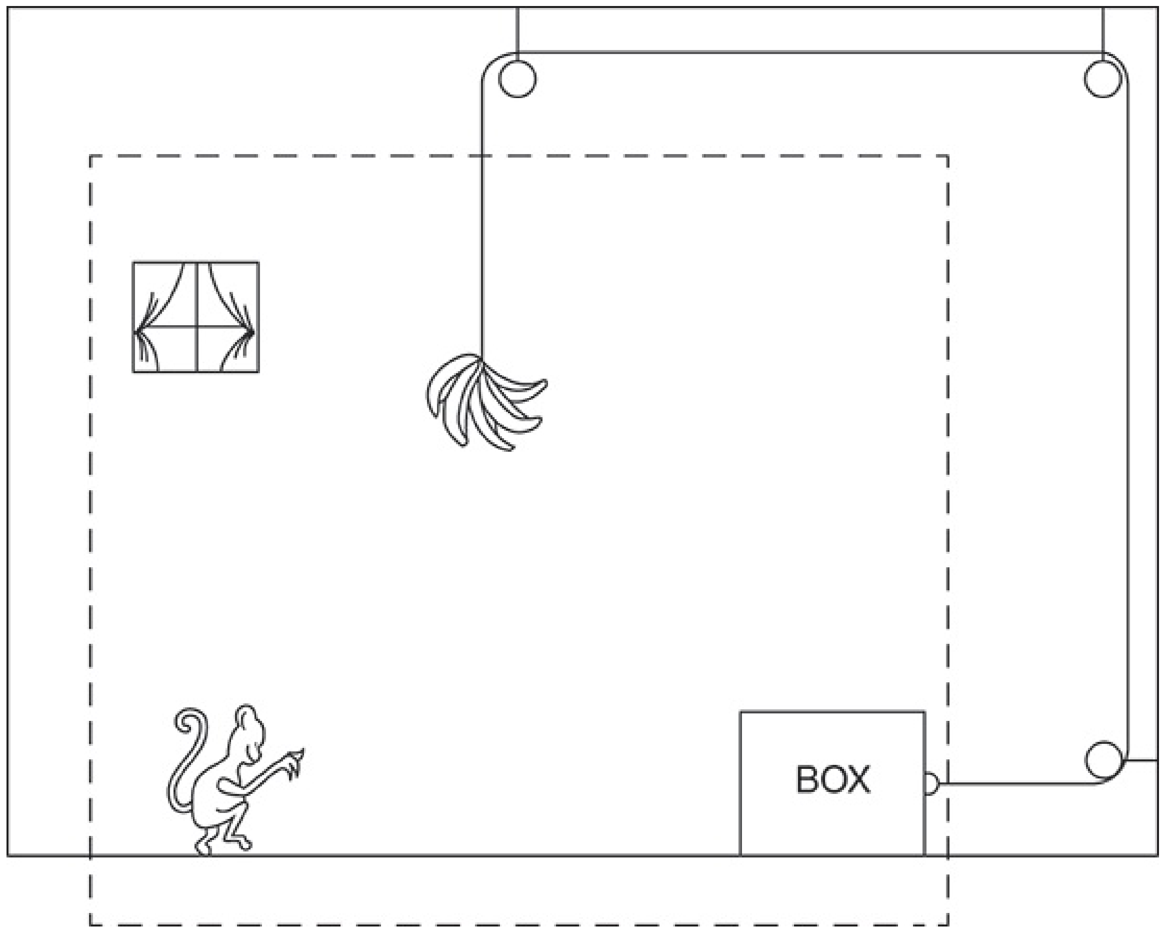
So this is a special case of the fact noted in Chapter 1: that AI has taught us that human minds are hugely richer, and more subtle, than psychologists previously imagined. Indeed, that is *the* main lesson to be learned from AI.

The frame problem

Finding an appropriate knowledge representation, in whatever domain, is difficult partly because of the need to avoid the *frame problem*. (Beware: although this problem arises when using frames as a knowledge representation for concepts, the meanings of ‘frame’ here are different.)

As originally defined by McCarthy and Hayes, the frame problem involves assuming (during planning by robots) that an action will cause only *these* changes, whereas it may cause *those* too. More generally, the frame problem arises whenever implications tacitly assumed by human thinkers are ignored by the computer because they haven’t been made explicit.

The classic case is the monkey and bananas problem, wherein the problem-solver (perhaps an AI planner for a robot) assumes that nothing relevant exists outside the frame (see Figure 1).



1. Monkey and bananas problem: how does the monkey get the bananas? (The usual approach to this problem assumes, though doesn't explicitly state, that the relevant 'world' is that shown inside the dotted-line frame. In other words, nothing exists outside this frame which causes significant changes in it on moving the box.)

My own favourite example is: *If a man of 20 can pick 10 pounds of blackberries in an hour, and a woman of 18 can pick 8, how many will they gather if they go blackberrying together?* For sure, '*18*' isn't a plausible answer. It could be much more (because they're both showing off) or, more probably, much less. Just what kinds of knowledge are involved here? And could an AGI overcome what appear to be the plain arithmetical facts?

The frame problem arises because AI programs don't have a human's sense of *relevance* (see Chapter 3). It can be avoided if all possible consequences of every possible action are known. In some technical/scientific areas, that's so. In general, however, it isn’t. That’s a major reason why AI systems lack common sense.

In brief, the frame problem lurks all around us—and is a major obstacle in the quest for AGI.

Agents and distributed cognition

An AI *agent* is a self-contained ('autonomous') procedure, comparable sometimes to a knee-jerk reflex and sometimes to a mini-mind. Phone apps or spelling correctors could be called agents, but usually aren't—because agents normally *cooperate*. They use their highly limited intelligence in cooperation with—or anyway, alongside—others to produce results that they couldn't achieve alone. The interaction between agents is as important as the individuals themselves.

Some agent systems are organized by hierarchical control: top dogs and underdogs, so to speak. But many exemplify *distributed* cognition. This involves cooperation without hierarchical command structure (hence the prevarication, earlier, between 'in cooperation with' and 'alongside'). There's no central plan, no top-down influence, and no individual possessing *all* the relevant knowledge.

Naturally occurring examples of distributed cognition include ant trails, ship navigation, and human minds. Ant trails emerge from the behaviour of many individual ants, automatically dropping (and following) chemicals as they walk. Similarly, navigation and manoeuvring of ships results from the interlocking activities of many people: not even the captain has all the necessary knowledge, and some crew members have very little indeed. Even a single mind involves distributed cognition, for it integrates many cognitive, motivational, and emotional subsystems (see Chapters 4 and 6).

Artificial examples include neural networks (see Chapter 4); an anthropologist's computer model of ship navigation, and A-Life work on situated robotics, swarm intelligence, and swarm robotics (see Chapter 5); symbolic AI models of financial markets (the agents being banks, hedge funds, and large shareholders); and the LIDA model of consciousness (see Chapter 6).

Clearly, human-level AGI would involve distributed cognition.

Machine learning

Human-level AGI would include machine learning, too. However, this needn't be *human-like*. The field originated from psychologists' work on concept learning and reinforcement. However, it now depends on fearsomely mathematical techniques, because the knowledge representations used involve probability theory and statistics. (One might say that psychology has been left far behind. Certainly, some modern machine learning systems bear little or no similarity to what might plausibly be going on in human heads. However, the increasing use of *Bayesian* probability in this area of AI parallels recent theories in cognitive psychology and neuroscience.)

Today's machine learning is hugely lucrative. It's used for data mining—and, given supercomputers doing a million billion calculations per second, for processing Big Data (see Chapter 3).

Some machine learning uses neural networks. But much relies on symbolic AI, supplemented by powerful statistical algorithms. In fact, the statistics really do the work, the GOFAI merely guiding the worker to the workplace. Accordingly, some professionals regard machine learning as computer science and/or statistics —*not* AI. However, there's no clear boundary here.

Machine learning has three broad types: supervised, unsupervised, and reinforcement learning. (The distinctions originated in psychology, and different neurophysiological mechanisms may be involved; reinforcement learning, across species, involves dopamine.)

In *supervised* learning, the programmer 'trains' the system by defining a set of desired outcomes for a range of inputs (labelled examples and non-examples), and providing continual feedback about whether it has achieved them. The learning system generates hypotheses about the relevant features. Whenever it classifies incorrectly, it amends its hypothesis accordingly. *Specific* error messages are crucial (not merely feedback that it was mistaken).

In *unsupervised* learning, the user provides no desired outcomes or error messages. Learning is driven by the principle that co-occurring features engender expectations that they will co-occur in future. Unsupervised learning can be used to *discover* knowledge. The programmers needn't know what patterns/clusters exist in the data: the system finds them for itself.

Finally, *reinforcement* learning is driven by analogues of reward and punishment: feedback messages telling the system that what it just did was good or bad. Often, reinforcement isn't simply binary, but represented by numbers—like the scores in a video game. 'What it just did' may be a single decision (such as a move in a game), or a series of decisions (e.g. chess moves culminating in checkmate). In some video games, the numerical score is updated at every move. In highly complex situations, such as chess, success (or failure) is signalled only after many decisions, and some procedure for *credit assignment* identifies the decisions most likely to lead to success.

Symbolic machine learning in general assumes—what's not obviously true—that the knowledge representation for learning must involve some form of probability distribution. And many learning algorithms assume—what is usually false—that every variable in the data has the same probability distribution, and all are mutually independent. That's because this *i.i.d.* (independent and identically distributed) assumption underlies many mathematical theories of probability, on which the algorithms are based. The mathematicians adopted the i.i.d. assumption because it makes the mathematics simpler. Similarly, using i.i.d. in AI simplifies the search space, thus making problem solving easier.

Bayesian statistics, however, deals with *conditional* probabilities, where items/events are *not* independent. Here, probability depends on distributional evidence about the domain. Besides being more realistic, this form of knowledge representation allows probabilities to be changed if new evidence comes in. Bayesian techniques are becoming increasingly prominent in AI—and in psychology and neuroscience too. Theories of 'the Bayesian brain' (see Chapter 4) capitalize on the use of non-i.i.d. evidence to drive, and to fine-tune, unsupervised learning in perception and motor control.

Given various theories of probability, there are many different algorithms suitable for distinct types of learning and different data sets. For instance, Support Vector Machines—which accept the i.i.d. assumption—are widely used for supervised learning, especially if the user lacks specialized prior knowledge about the domain. 'Bag of Words' algorithms are useful when the *order* of features can be ignored (as in searches for words, not phrases). And if the i.i.d. assumption is dropped, Bayesian techniques (‘Helmholtz machines’) can learn from distributional evidence.

Most machine learning professionals use off-the-shelf statistical methods. The originators of those methods are highly prized by the industry: Facebook recently employed the creator of Support Vector Machines, and in 2013/14 Google hired several key instigators of *deep learning*.

Deep learning is a promising new advance based in multilayer networks (see Chapter 4), by which patterns in the input data are recognized at various hierarchical levels. In other words, deep learning *discovers* a multilevel knowledge representation—for instance, pixels to contrast detectors, to edge detectors, to shape detectors, to object parts, to objects.

One example is the cat’s-face detector that emerged from Google’s research on YouTube. Another, reported in *Nature* in 2015, is a reinforcement learner (the ‘DQN’ algorithm) that has learned to play the classic Atari 2600 2D games. Despite being given only pixels and game scores as input (and already knowing only the number of actions available for each game), this surpasses 75 per cent of humans on twenty-nine of the forty-nine games, and outperforms professional game testers on twenty-two.

It remains to be seen how far this achievement can be extended. Although DQN sometimes finds the optimal strategy, involving temporally ordered actions, it can’t master games whose planning encompasses a longer period of time.

Future neuroscience may suggest improvements to this system. The current version is inspired by the Hubel–Wiesel vision receptors—cells in the visual cortex that respond only to movement, or only to lines of a particular orientation. (That’s no big deal: the Hubel–Wiesel receptors inspired Pandemonium, too: see Chapter 1.) More unusually, this version of DQN is inspired also by the ‘experience replay’ happening in the hippocampus during sleep. Like the hippocampus, the DQN system stores a pool of past samples, or experiences, and reactivates them rapidly during learning. This feature is crucial: the designers reported ‘a severe deterioration’ in performance when it was disabled.

Generalist systems

The Atari game player caused excitement—and merited publication in *Nature*—partly because it seemed to be a step towards AGI. A single algorithm, using no handcrafted knowledge representation, learned a wide range of competences on a variety of tasks involving relatively high-dimensional sensory input. No previous program had done that.

Nor did the *AlphaGo* program, developed by the same team, which in 2016 beat the world's *Go* champion Lee Sedol. Nor *AlphaGo Zero*, which in 2017 surpassed *AlphaGo* despite having been fed no data about games of *Go* played by humans. For the record, in December 2017 *AlphaZero* also mastered chess: after only four hours of playing against itself, starting from random states but having been provided with the rules of the game, it beat the champion chess-program *Stockfish* by twenty-eight wins and seventy-two draws in one hundred games.

However (as remarked at the outset of this chapter), full AGI would do very much more. Difficult though it is to build a high-performing AI specialist, building an AI generalist is orders of magnitude harder. (Deep learning isn't the answer: its *aficionados* admit that 'new paradigms are needed' to combine it with complex reasoning—scholarly code for 'we haven't got a clue'.) That's why most AI researchers abandoned that early hope, turning instead to multifarious narrowly defined tasks—often with spectacular success.

AGI pioneers who retained their ambitious hopes included Newell and John Anderson. They originated SOAR and ACT-R respectively: systems begun in the early 1980s, and both still being developed (and used) some three decades later. However, they oversimplified the task, focusing on only a small subset of human competences.

In 1962, Newell's colleague Simon had considered the zig-zagging path of an ant on uneven ground. Every movement, he said, is a direct reaction to the situation perceived by the ant at that moment (this is the key idea of *situated* robotics: see Chapter 5). Ten years later, Newell and Simon's book *Human Problem Solving* described our intelligence as similar. According to their psychological theory, perception and motor action are supplemented by internal representations (IF–THEN rules, or ‘productions’) stored in memory, or newly built during problem solving.

‘Human beings, viewed as behaving systems’, they said, ‘are quite simple’. But the emergent behavioural complexities are significant. For instance, they showed that a system of only fourteen IF–THEN rules can solve cryptarithmetic problems (e.g. map the letters to the digits 0 to 9 in this sum: DONALD + GERALD = ROBERT, where D = 5). Some rules deal with goal/sub-goal organization. Some direct attention (to a specific letter or column). Some recall previous steps (intermediate results). Some recognize false starts. And others backtrack to recover from them.

Cryptarithmetic, they argued, exemplifies the computational architecture of *all* intelligent behaviour—so this psychological approach suited a *generalist* AI. From 1980, Newell (with John Laird and Paul Rosenbloom) developed SOAR (Success Oriented Achievement Realized). This was intended as a model of cognition as a whole. Its reasoning integrated perception, attention, memory, association, inference, analogy, and learning. Ant-like (situated) responses were combined with internal deliberation. Indeed, deliberation often resulted in reflex responses, because a previously used sequence of sub-goals could be ‘chunked’ into *one* rule.

In fact, SOAR failed to model *all* aspects of cognition, and was later extended as people recognized some of the gaps. Today’s version is used for many purposes, from medical diagnosis to factory scheduling.

Anderson’s ACT-R (Adaptive Control of Thought) family are hybrid systems (see Chapter 4), developed by combining production systems and semantic networks. These programs, which recognize the statistical probabilities in the environment, model associative memory, pattern recognition, meaning, language, problem solving, learning, imagery, and (since 2005) perceptuo-motor control.

A key feature of ACT-R is the integration of procedural and declarative knowledge. Someone may *know that* a theorem of Euclid’s is true, without *knowing how* to use it in a geometrical proof. ACT-R can learn how to apply a propositional truth, by constructing hundreds of new productions that control its use in many different circumstances. It learns which goals, sub-goals, and sub-sub-goals … are relevant in which conditions, and what results a particular action will have in various circumstances. In short, it learns by doing. And (like SOAR) it can chunk several rules that are often carried out sequentially into a single rule. This parallels the difference between how human experts and novices solve 'the same' problem: unthinkingly or painstakingly.

ACT-R has diverse applications. Its mathematics tutors offer personalized feedback, including relevant domain knowledge, and the goal/sub-goal structure of problem solving. Thanks to chunking, the grain size of their suggestions changes as the student's learning proceeds. Other applications concern NLP; human–computer interaction; human memory and attention; driving and flying; and visual web search.

SOAR and ACT were contemporaries of another early attempt at AGI: Douglas Lenat's CYC. This symbolic-AI system was launched in 1984, and is still under continuous development.

By 2015, CYC contained 62,000 'relationships' capable of linking the concepts in its database, and millions of links between those concepts. These include the semantic and factual associations stored in large semantic nets (see Chapter 3), and countless facts of naïve physics—the unformalized knowledge of physical phenomena (such as dropping and spilling) that all humans have. The system uses both monotonic and non-monotonic logics, and probabilities too, to reason about its data. (At present, all the concepts and links are hand-coded, but Bayesian learning is being added; this will enable CYC to learn from the Internet.)

It has been used by several US government agencies, including the Department of Defense (to monitor terrorist groups, for instance) and the National Institutes of Health, and by some major banks and insurance companies. A smaller version—*OpenCyc*—has been publicly released as a background source for a variety of applications, and a fuller abridgment (*ResearchCyc*) is available for AI workers. Although *OpenCyc* is regularly updated it contains only a small subset of CYC's database, and a small subset of inference rules. Eventually, the complete (or near-complete) system will be commercially available. However, that could fall into malicious hands—unless specific measures are taken to prevent this (see Chapter 7).

CYC was described by Lenat in *AI Magazine* (1986) as ‘Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks’. That is, it was specifically addressing McCarthy’s prescient challenge. Today, it’s the leader in modelling ‘common-sense’ reasoning, and also in ‘understanding’ the concepts it deals with (which even apparently impressive NLP programs cannot do: see Chapter 3).

Nevertheless, it has many weaknesses. For example, it doesn’t cope well with metaphor (although the database includes many dead metaphors, of course). It ignores various aspects of naïve physics. Its NLP, although constantly improving, is very limited. And it doesn’t yet include vision. In sum, despite its en-CYC-lopedic aims, it doesn’t really encompass human knowledge.

The dream revitalized

Newell, Anderson, and Lenat beavered away in the background for thirty years. Recently, however, interest in AGI has revived markedly. An annual conference was started in 2008, and SOAR, ACT-R, and CYC are being joined by other supposedly generalist systems.

For instance, in 2010 the machine learning pioneer Tom Mitchell launched Carnegie Mellon's NELL (Never-Ending Language Learner). This 'common-sense' system builds its knowledge by trawling the Web non-stop (for seven years at the time of writing) and by accepting online corrections from the public. It can make simple inferences based on its (unlabelled) data: for instance, the athlete Joe Bloggs plays tennis, since he's on the Davis team. Starting with an ontology of 200 categories and relations (e.g. *master, is due to*), after five years it had enlarged the ontology and amassed ninety million candidate beliefs, each with its own confidence level.

The bad news is that NELL doesn't know, for example, that you can pull objects with a string, but not push them. Indeed, the putative common sense of *all* AGI systems is gravely limited. Claims that the notorious frame problem has been 'solved' are highly misleading.

NELL now has a sister program, NEIL: Never-Ending Image Learner. Some part-visual AGIs combine a logical-symbolic knowledge representation with analogical, or graphical, representations (a distinction made years ago by Aaron Sloman, but still not well understood).

In addition, Stanford Research Institute's CALO (Cognitive Assistant that Learns and Organizes) provided the spin-off *Siri* app (see Chapter 3), bought by Apple for $200 million in 2009. Comparable currently active projects include Stan Franklin's intriguing LIDA (discussed in Chapter 6) and Ben Goertzel's *OpenCog*, which learns its facts and concepts within a rich virtual world and also from other AGI systems. (LIDA is one of two generalist systems focused on *consciousness*; the other is CLARION.)

An even more recent AGI project, started in 2014, aims at developing 'A

Computational Architecture for Moral Competence in Robots' (see Chapter 7). Besides the difficulties mentioned earlier, it will have to face the many problems that relate to morality.

A genuinely human-level system would do no less. No wonder, then, that AGI is proving so elusive.

Missing dimensions

Nearly all of today’s generalist systems are focused on *cognition*. Anderson, for instance, aims to specify ‘how all the subfields in cognitive psychology interconnect’. (‘*All*’ the subfields? Although he addresses motor control, he doesn’t discuss touch or proprioception—which sometimes feature in robotics.) A truly general AI would cover *motivation* and *emotion* as well.

A few AI scientists have recognized this. Marvin Minsky and Sloman have both written insightfully about the computational architecture of whole minds, although neither has built a whole-mind model.

Sloman’s MINDER model of anxiety is outlined in Chapter 3. His work (and Dietrich Dorner’s psychological theory) has inspired Joscha Bach’s *MicroPsi:* an AGI based on seven different ‘motives’, and using ‘emotional’ dispositions in planning and action selection. It has also influenced the LIDA system mentioned earlier (see Chapter 6).

But even these fall far short of true AGI. Minsky’s prescient AI manifesto of 1956, ‘Steps Toward Artificial Intelligence’, identified obstacles as well as promises. Many of the former have yet to be overcome. As Chapter 3 should help to show, human-level AGI isn’t within sight.

Chapter 3

Language, creativity, emotion

Some areas of AI seem especially challenging: language, creativity, and emotion. If AI can’t model these, hopes of AGI are illusory.

In each case, more has been achieved than many people imagine. Nevertheless, significant difficulties remain. These quintessentially ‘human’ areas have been modelled only up to a point. (Whether AI systems could ever have *real* understanding, creativity, or emotion is discussed in Chapter 6. Here, our question is whether they can *appear* to possess them.)

Language

Countless AI applications use natural language processing (NLP). Most focus on the computer's 'understanding' of language that is presented to it, not on its own linguistic production. That's because NLP generation is even more difficult than NLP acceptance.

The difficulties concern both thematic content and grammatical form. For instance, we saw in Chapter 2 that familiar action sequences ('scripts') can be used as the seed of AI stories. But whether the background knowledge representation includes enough about human motivation to make the story interesting is another matter. A commercially available system that writes annual summaries describing a firm's changing financial position generates very boring 'stories'. Computer-generated novels and soap-opera plots do exist—but they won't win any prizes for subtlety. (AI translations/summaries of human-generated texts may be much richer, but that's thanks to the *human* authors.)

As for grammatical form, computer prose is sometimes grammatically incorrect and usually very clumsy. Anthony Davey's AI-generated narrative of a game of noughts and crosses (tic-tac-toe) can have clausal/sub-clausal structures that match the dynamics of the game in a nicely appropriate way. But the possibilities and strategies of noughts and crosses are fully understood. Describing the succession of thoughts, or actions, of the protagonists in most human stories in a similarly elegant fashion would be much more challenging.

Turning to AI's *acceptance* of language, some systems are boringly simple: they require only keyword recognition (think of the 'menus' in e-retailing), or the prediction of words listed in a dictionary (think of the automatic completion that happens when writing text messages). Others are significantly more sophisticated.

A few require speech recognition, either of single words, as in automated telephone shopping, or of continuous speech, as in real-time TV subtitling and telephone bugging. In the latter case, the aim may be to pick out specific words (such as *bomb* and *Jihad*) or, more interestingly, to capture the sense of the sentence as a whole. This is NLP with knobs on: the words themselves—spoken by many different voices, and with different local/foreign accents—must be distinguished first. (Word distinctions come for free in printed texts.) Deep learning (see Chapter 4) has enabled significant advances in speech processing.

Impressive examples of what looks like whole-sentence understanding include machine translation; data mining from large collections of natural-language texts; summarizing articles in newspapers and journals; and free-range question answering (increasingly employed in Google searches, and in the *Siri* app for the iPhone).

But can such systems really appreciate language? Can they cope with grammar, for instance?

In AI's early days, people assumed that language understanding requires syntactic parsing. Considerable effort went into writing programs to do that. The outstanding example—which brought AI to the attention of countless people who had previously never heard of it, or who had dismissed it as impossible—was Terry Winograd's SHRDLU, written at MIT in the early 1970s.

This program accepted instructions in English telling a robot to build structures made of coloured blocks, and worked out just how certain blocks should be moved to achieve the goal. It was hugely influential for many reasons, some of which applied to AI in general. Here, what's relevant is its unprecedented ability to assign detailed grammatical structure to complex sentences, such as: *How many eggs would you have been going to use in the cake if you hadn't learned your grandmother's recipe was wrong?* (Try it!)

For technological purposes, SHRDLU turned out to be a disappointment. The program contained many bugs, so could be used only by a handful of highly skilled researchers. Various other syntax crunchers were built at around that time, but they, too, weren't generalizable to real-world texts. In short, it soon appeared that the analysis of fancy syntax is too difficult for off-the-shelf systems.

Fancy syntax wasn't the only problem. In human language-use, *context* and *relevance* matter too. It wasn't obvious that they could ever be handled by AI.

Indeed, machine translation had been pronounced impossible by the US

Indeed, machine translation had been pronounced impossible by the US government's ALPAC Report in 1964 (the acronym denoted the Automatic Language Processing Advisory Committee). Besides predicting that not enough people would want to use it to make it commercially viable (although machine aids for human translators might admittedly be feasible), the report argued that computers would struggle with syntax, be defeated by context, and—above all—be blind to relevance.

That was a bombshell for machine translation (whose funding virtually dried up overnight), and for AI in general. It was widely interpreted as showing the futility of AI. The bestseller *Computers and Common Sense* had already claimed (in 1961) that AI was a waste of taxpayers' money. Now, it seemed that top governmental experts agreed. Two US universities that were about to open AI departments cancelled their plans accordingly.

Work in AI continued nevertheless, and when the syntax-savvy SHRDLU hit the scene a few years later it seemed to be a triumphant vindication of GOFAI. But doubts soon crept in. Accordingly, NLP turned increasingly to context rather than syntax.

A few researchers had taken semantic context seriously even in the early 1950s. Margaret Masterman's group in Cambridge, England, had approached machine translation (and information retrieval) by using a thesaurus rather than a dictionary. They saw syntax as 'that very superficial and highly redundant part of language that [people in a hurry], quite rightly, drop', and focused on word clusters rather than single words. Instead of attempting word-by-word translation, they searched the surrounding text for words of similar meaning. This (when it worked) enabled ambiguous words to be translated correctly. So *bank* could be rendered (in French) as *rive* or as *banque*, depending on whether the context contained words such as *water* or *money*, respectively.

That thesaurus-based contextual approach could be strengthened by also considering words that often co-occur despite having *dissimilar* meanings (like *fish* and *water*). And this, as time passed, is what happened. Besides distinguishing various types of lexical similarity—synonyms (*empty/vacant*), antonyms (*empty/full*), class membership (*fish/animal*) and inclusion (*animal/fish*), shared class level (*cod/salmon*), and part/whole (*fin/fish*)—today's machine translation also recognizes thematic co-occurrence (*fish/water*, *fish/bank*, *fish/chips*, etc.).

It’s now clear that handling fancy syntax isn’t necessary for summarizing, questioning, or translating a natural-language text. Today’s NLP relies more on brawn (computational power) than on brain (grammatical analysis). Mathematics—specifically, statistics—has overtaken logic, and machine learning (including, but not restricted to, deep learning) has displaced syntactic analysis. These new approaches to NLP, ranging from written texts to speech recognition, are so efficient that a 95 per cent success rate is taken as the norm of acceptability for practical applications.

In modern-day NLP, powerful computers do statistical searches of huge collections (‘corpora’) of texts (for machine translation, these are paired translations done by humans) to find word patterns both commonplace and unexpected. They can learn the statistical likelihood of *fish/water*, or *fish/tadpole*, or *fish and chips/salt and vinegar*. And (as remarked in Chapter 2) NLP can now learn to construct ‘word vectors’ representing the probabilistic clouds of meaning that attend a given concept. In general, however, the focus is on words and phrases, not syntax. Grammar isn’t ignored: labels such as ADJective and ADVerb may be assigned, either automatically or by hand, to some words in the texts being examined. But syntactic *analysis* is little used.

Even detailed *semantic* analysis isn’t prominent. ‘Compositional’ semantics uses syntax in analysing the meaning of sentences; but it is found in research laboratories, not in large-scale applications. The ‘common-sense’ reasoner CYC has relatively full semantic representations of its concepts (words), and ‘understands’ them better accordingly (see Chapter 2). But that is still unusual.

Current machine translation can be astonishingly successful. Some systems are restricted to a small set of topics, but others are more open. Google Translate offers machine translation on unconstrained topics to over 200 million users every day. SYSTRAN is used daily by the European Union (for twenty-four languages) and NATO, and by Xerox and General Motors.

Many of these translations, including the EU documents, are near-perfect (because only a limited subset of words is used in the original texts). Many more are imperfect yet easily intelligible, because informed readers can ignore grammatical errors and inelegant word choices—as one does when listening to a non-native speaker. Some require minimal post-editing by humans. (With Japanese, significant pre-editing and post-editing may be needed. Japanese contains no segmented words, like the English past tense *vot-ed*, and phrase orderings are reversed. Machine-matching of languages from different language groups is usually difficult.)

In short, the results of machine translation are normally good enough for the human user to understand. Similarly, *monolingual* NLP programs that summarize journal papers can often show whether the paper merits reading in full. (*Perfect* translation is arguably impossible anyway. For example, requesting an apple in Japanese requires language reflecting the interlocutors' comparative social status, but no equivalent distinctions exist in English.)

The real-time translation available on AI applications such as Skype is less successful. That's because the system has to recognize speech, not written text (in which the individual words are clearly separated).

Two other prominent NLP applications are forms of information retrieval: *weighted search* (initiated by Masterman's group in 1976) and *data mining.* The Google search engine, for instance, searches for terms weighted by relevance—which is assessed statistically, not semantically (that is, *without* understanding). Data mining can find word patterns unsuspected by human users. Long used for market research on products and brands, it's now being applied (often using deep learning) to 'Big Data': huge collections of texts (sometimes multilingual) or images, such as scientific reports, medical records, or entries on social media and the Internet.

Applications of Big Data mining include surveillance and counter-espionage, and the monitoring of public attitudes by governments, policy-makers, and social scientists. Such enquiries can compare the shifting opinions of distinct sub-groups: men/women, young/old, North/South, and so on. For instance, the UK think tank Demos (working with an NLP data-analytics team at the University of Sussex) has analysed many thousands of Twitter messages relating to misogyny, ethnic groups, and the police. Sudden bursts of tweeting after specific events ('twitcidents') can be searched to discover, for example, changes in public opinion on the police's reaction to a particular incident.

It remains to be seen whether Big Data NLP will reliably produce useful results.

Often, data mining (using ‘sentiment analysis’) seeks to measure not only the level of public interest, but its evaluative tone. However, this isn’t straightforward. For instance, a tweet containing an apparently derogatory racial epithet, and so machine-coded as ‘negative’ in sentiment, may not in fact be derogatory. A human judge, on reading it, may see the term as being used (in this case) as a positive marker of group identity, or as a neutral description (e.g. *The Paki shop on the corner*), not as insult or abuse. (The Demos research found that only a small proportion of tweets containing racial/ethnic terms are actually aggressive.)

In such cases, the human’s judgement will rely on the context—for example, the other words within the tweet. It may be possible to adjust the machine’s search criteria so that it makes fewer ‘negative sentiment’ ascriptions. Then again, it may not. Such judgements are often contentious. Even when they are agreed, it may be difficult to identify those aspects of the context which justify the human’s interpretation.

That’s just one example of the difficulty of pinning down *relevance* in computational (or even verbal) terms.

Two well-known NLP applications may seem, at first sight, to contradict that statement: Apple’s *Siri* and IBM’s WATSON.

Siri is a (rule-based) personal assistant, a talking ‘chat-bot’ that can quickly answer many different questions. It has access to everything on the Internet—including Google Maps, Wikipedia, the constantly updated *New York Times,* and lists of local services such as taxis and restaurants. It also calls on the powerful question answerer *WolframAlpha,* which can use logical reasoning to work out—not merely *find*—answers to a wide range of factual questions.

Siri accepts a spoken question from the user (to whose voice and dialect it gradually adapts), and answers it by using web-searching and conversational analysis. Conversational analysis studies how people organize the sequence of topics in a conversation, and how they arrange interactions such as explanation and agreement. This approach enables *Siri* to consider questions such as *What does the interlocutor want?* and *How should it answer?*, and—up to a point—to adapt to the individual user’s interests and preferences.

In short, *Siri* appears to be sensitive not only to topical relevance, but to personal relevance as well. So it's superficially impressive. However, it's easily led into giving ridiculous answers—and if the user strays from the domain of facts, *Siri* is lost.

WATSON, too, is focused on facts. As an off-the-shelf resource (with 2,880 core processors) for handling Big Data, it's already used in some call centres, and it is being adapted for medical applications such as assessing cancer therapies. But it doesn't merely answer straightforward questions, as *Siri* does. It can also deal with the puzzles that arise in the general-knowledge game *Jeopardy!*.

In *Jeopardy!*, players aren't asked direct questions, but are given a clue and have to guess what the relevant question would be. For example, they are told 'On May 9, 1921, this "letter-perfect" airline opened its first passenger office in Amsterdam', and they should answer 'What is KLM?'

WATSON can meet that challenge, and many others. Unlike *Siri*, its *Jeopardy!*-playing version has no access to the Internet (although the medical version does), and no notion of the structure of conversations. Nor can it discover an answer by logical reasoning. Instead, it uses massively parallel statistical search over an enormous, but closed, database. This contains documents—countless reviews and reference books, plus the *New York Times*—providing facts about leprosy to Liszt, hydrogen to Hydra, and so on. When playing *Jeopardy!*, its search is guided by hundreds of specially crafted algorithms that reflect the probabilities inherent in the game. And it can learn from its human contestants' guesses.

In 2011, WATSON rivalled the Kasparov moment of its IBM cousin *Deep Blue* (see Chapter 2), by apparently beating the two top human champions. ('Apparently', because the computer reacts instantaneously whereas humans need some reaction time before pressing the buzzer.) But, like *Deep Blue*, it doesn't always win.

On one occasion it lost because, although it correctly focused on a particular athlete's *leg*, it didn't realize that the crucial fact in its stored data was that this person had a leg *missing*. That mistake won't reoccur, because WATSON's programmers have now flagged the importance of the word 'missing'. But others will. Even in mundane fact-seeking contexts, people often rely on relevance judgements that are beyond WATSON. For example, one clue required the identity of two of Jesus' disciples whose names are both top-ten baby names, and end in the same letter. The answer was 'Matthew and Andrew'—which WATSON got immediately. The human champion got that answer too. But his first idea had been 'James and Judas'. He rejected that, he recalled, only because 'I don't think Judas is a popular baby name, for some reason'. WATSON couldn't have done that.

Human judgements of relevance are often much less obvious than that one, and much too subtle for today's NLP. Indeed, relevance is a linguistic/conceptual version of the unforgiving 'frame problem' in robotics (see Chapter 2). Many people would argue that it will never be wholly mastered by a non-human system. Whether that's due only to the massive complexity involved, or to the fact that relevance is rooted in our specifically human form of life, is discussed in Chapter 6.

Creativity

Creativity—the ability to produce ideas or artefacts that are new, surprising, and valuable—is the acme of human intelligence, and necessary for human-level AGI. But it’s widely seen as mysterious. It’s not obvious how novel ideas could arise in *people*, never mind computers.

Even *recognizing* it isn’t straightforward: people often disagree about whether an idea is creative. Some disagreements turn on whether, and in what sense, it’s actually new. An idea may be new only to the individual involved, or new also to the whole of human history (exemplifying ‘individual’ and ‘historical’ creativity respectively). In either case, it may be *more* or *less* similar to preceding ideas, leaving room for further disagreements. Other disputes turn on valuation (which involves functional, and sometimes phenomenal, consciousness: see Chapter 6). An idea may be valued by one social group, but not others. (Think of the scorn directed by youngsters today to anyone who prizes their DVDs of Abba.)

It’s commonly assumed that AI could have nothing interesting to say about creativity. But AI technology has generated many ideas that are historically new, surprising, and valuable. These arise, for instance, in designing engines, pharmaceuticals, and various types of computer art.

Moreover, AI concepts help to explain *human* creativity. They enable us to distinguish three types: combinational, exploratory, and transformational. These involve different psychological mechanisms, eliciting different sorts of surprise.

In *combinational* creativity, familiar ideas are combined in unfamiliar ways. Examples include visual collage, poetic imagery, and scientific analogies (the heart as a pump, the atom as a solar system). The new combination provides a statistical surprise: it was improbable, like an outsider winning the Derby. But it’s intelligible, so valuable. *Just how valuable* depends on judgements of relevance, discussed earlier.

Exploratory creativity is less idiosyncratic, for it exploits some culturally valued way of thinking (e.g. styles of painting or music, or sub-areas of chemistry or mathematics). The stylistic rules are used (largely unconsciously) to produce the new idea—much as English grammar generates new sentences. The artist/scientist may explore the style's potential in an unquestioning way. Or they may deliberately push and test it, discovering what it can and cannot generate. It may even be tweaked, by slightly altering (e.g. weakening/strengthening) a rule. The novel structure, despite its novelty, will be recognized as lying within a familiar stylistic family.

Transformational creativity is a successor of exploratory creativity, usually triggered by frustration at the limits of the existing style. Here, one or more stylistic constraints are radically altered (dropped, negated, complemented, substituted, added …), so that novel structures are generated which *could not* have been generated before. These new ideas are deeply surprising, because they're seemingly *impossible.* They're often initially unintelligible, for they can't be fully understood in terms of the previously accepted way of thinking. However, they must be intelligibly close to the previous way of thinking if they are to be accepted. (Sometimes, this recognition takes many years.)

All three types of creativity occur in AI—often, with results attributed by observers to humans (in effect, passing the Turing Test: see Chapter 6). But they aren't found in the proportions one might expect.

In particular, there are very few combinational systems. One might think it's easy to model combinational creativity. After all, nothing could be simpler than making a computer produce unfamiliar associations of already stored ideas. The results will often be historically novel, and (statistically) surprising. But if they're also to be valuable, they must be mutually relevant. That's not straightforward, as we've seen. The joke-generating programs mentioned in Chapter 2 use joke templates to help provide relevance. Similarly, symbolic AI's *case-based reasoning* constructs analogies thanks to pre-coded structural similarities. So, their 'combinational' creativity has a strong admixture of exploratory creativity as well.

Conversely, one might expect that AI could never model transformational creativity. This expectation, also, is mistaken. Certainly, any program can do only what it's potentially capable of doing. But evolutionary programs can transform themselves (see Chapter 5). They can even evaluate their newly transformed ideas—but only *if* the programmer has provided clear criteria for selection. Such programs are routinely used for novelty-seeking AI applications —such as designing new scientific instruments or drugs.

This isn't a magic road to AGI, however. Valuable results are rarely guaranteed. Some evolutionary programs (in maths or science) can reliably find the optimal solution, but many problems can't be defined by optimization. Transformational creativity is risky, because previously accepted rules are broken. Any new structures must be evaluated, or chaos ensues. But current AI's fitness functions are defined by humans: the programs can't adapt/evolve them independently.

Exploratory creativity is the type best suited to AI. There are countless examples. Some exploratory AI novelties in engineering (including one generated by a program from CYC's designer: see Chapter 2) have been awarded patents. Although a patented idea isn't 'obvious to a person skilled in the art', it may unexpectedly lie within the potential of the style being explored. A few AI explorations are indistinguishable from outstanding human achievements—such as the composition, by David Cope's programs, of music in the style of Chopin or Bach. (How many *humans* can do that?)

However, even exploratory AI depends crucially on human judgement. For someone must recognize—and clearly state—the stylistic rules concerned. That's usually difficult. A world expert on Frank Lloyd Wright's Prairie Houses abandoned his attempt to describe their architectural style, declaring it 'occult'. Later, a computable 'shape-grammar' generated indefinitely many Prairie House designs, including the forty-odd originals—and *no* implausibilities. But the human analyst was ultimately responsible for the system's success. Only if an AGI could analyse styles (in art or science) *for itself* would its creative explorations be 'all its own work'. Despite some recent—very limited—examples of art styles being recognized by deep learning (see Chapters 2 and 4), that's a tall order.

AI has enabled human artists to develop a new art form: computer-generated (CG) art. This concerns architecture, graphics, music, choreography, and—less successfully (given NLP 's difficulties with syntax and relevance)—literature. In CG art, the computer isn't a mere tool, comparable to a new paintbrush, helping the artist to do things they might have done anyway. Rather, the work couldn't have been done, or perhaps even imagined, without it.

CG art exemplifies all three types of creativity. For the reasons given earlier, hardly any CG art is combinational. (Simon Colton's *The Painting Fool* has produced visual collages related to war—but it was specifically instructed to search for images associated with 'war', which were readily available in its database.) Most is exploratory or transformational.

Sometimes, the computer generates the artwork entirely independently, by executing the program written by the artist. So Harold Cohen's AARON produces line drawings and coloured images unaided (sometimes generating colours so daringly beautiful that Cohen says it's a better colourist than he is himself).

In interactive art, by contrast, the form of the final artwork depends partly on input from the audience—who may or may not have deliberate control over what happens. Some interactive artists see the audience as fellow-creators, others as mere causal factors who unknowingly affect the artwork in various ways (and some, such as Ernest Edmonds, take both approaches). In evolutionary art, exemplified by William Latham and Jon McCormack, the results are continually generated/transformed by the computer—but the *selection* is usually done by artist or audience.

In short, AI creativity has many applications. It can sometimes match, or even exceed, human standards in some small corner of science or art. But matching human creativity *in the general case* is quite another matter. AGI is as far away as ever.

AI and emotion

Emotion, like creativity, is something usually seen as utterly alien to AI. Besides the intuitive implausibility, the fact that moods and emotions depend on neuromodulators diffusing in the brain seems to rule out AI models of affect.

For many years, AI scientists themselves appeared to agree. With a few early exceptions in the 1960s and 1970s—namely Herbert Simon, who saw emotion as involved in cognitive control, and Kenneth Colby, who built interesting, although grossly overambitious, models of neurosis and paranoia—they ignored emotion.

Today, things are different. Neuromodulation has been simulated (in GasNets: see Chapter 4). Moreover, many AI research groups are now addressing emotion. Most (not quite all) of this research is theoretically shallow. And most is potentially lucrative, being aimed at developing ‘computer companions’.

These are AI systems—some screen-based, some ambulatory robots—designed to interact with people in ways that (besides being practically helpful) are affectively comfortable, even satisfying, for the user. Most are aimed at the elderly and/or disabled, including people with incipient dementia. Some are targeted on babies or infants. Others are interactive ‘adult toys’. In short: computer carers, robot nannies, and sexual playmates.

The human–computer interactions concerned include: offering reminders about shopping, medication, and family visits; talking about, and helping to compile, a continuing personal journal; scheduling and discussing TV programs, including the daily news; making/fetching food and drink; monitoring vital signs (and babies’ crying); and speaking and moving in sexually stimulating ways.

Many of these tasks will involve emotion on the person’s part. As for the AI companion, this may be able to recognize emotions in the human user and/or it may respond in apparently emotional ways. For instance, sadness in the user—caused, perhaps, by mention of a bereavement—might elicit some show of sympathy from the machine.

AI systems can already recognize human emotions in various ways. Some are

AI systems can already recognize human emotions in various ways. Some are physiological: monitoring the person’s breathing rate and galvanic skin response. Some are verbal: noting the speaker’s speed and intonation, as well as their vocabulary. And some are visual: analysing their facial expressions. At present, all these methods are relatively crude. The user’s emotions are both easily missed and easily misinterpreted.

Emotional performance on the computer companion’s part is usually verbal. It’s based in vocabulary (and intonation, if the system generates speech). But, much as the system watches out for familiar keywords from the user, so it responds in highly stereotyped ways. Occasionally, it may quote a human-authored remark or poem associated with something the user has said—perhaps in the diary. But the difficulties of NLP imply that computer-generated text is unlikely to be subtly appropriate. It may not even be acceptable: the user may be irritated and frustrated by a companion incapable of offering even the *appearance* of true companionship. Similarly, a purring robot cat may annoy the user, instead of communicating comfortably relaxed contentment.

Then again, it may not: *Paro*, a cuddly interactive ‘baby seal’ with charming black eyes and luxurious eyelashes, appears to be beneficial for many elderly people and/or people with dementia. (Future versions will monitor vital signs, alerting the person’s human carers accordingly.)

Some AI companions can use their own facial expressions, and eye gaze, to respond in seemingly emotional ways. A few robots possess flexible ‘skin’, overlying a simulacrum of human facial musculature, whose configuration can suggest (to the human observer) up to a dozen basic emotions. The screen-based systems often show the face of a virtual character, whose expressions change according to the emotions it (he/she?) is supposedly undergoing. However, all these things risk falling into the so-called ‘uncanny valley’: people typically feel uncomfortable, or even deeply disturbed, when encountering creatures that are very similar to human beings *but not quite similar enough.* Robots, or screen avatars, with not-quite-human faces may therefore be experienced as threatening.

Whether it’s ethical to offer such quasi-companionship to emotionally needy people is questionable (see Chapter 7). Certainly, some human–computer interactive systems (e.g. *Paro*) appear to provide pleasure, and even lasting contentment, to people whose lives seem otherwise empty. But is that enough?

There’s scant theoretical depth to ‘companion’ models. The emotional aspects of AI companions are being developed for commercial purposes. There’s no attempt to make them use emotions in solving their own problems, nor to illuminate the role that emotions play in the functioning of the mind as a whole. It’s as though emotions are seen by these AI researchers as optional extras: to be disregarded unless, in some messily human context, they’re unavoidable.

That dismissive attitude was widespread in AI until relatively recently. Even Rosalind Picard’s work on ‘affective computing’, which brought emotions in from the cold in the late 1990s, didn’t analyse them in depth.

One reason why AI ignored emotion (and Simon’s insightful remarks about it) for so long is that most psychologists and philosophers did so too. In other words, they didn’t think of *intelligence* as something that requires emotion. To the contrary, affect was assumed to disrupt problem solving and rationality. The idea that emotion can help one to decide what to do, and how best to do it, wasn’t fashionable.

It eventually became more prominent, thanks partly to developments in clinical psychology and neuroscience. But its entry into AI was due also to two AI scientists, Marvin Minsky and Aaron Sloman, who had long considered *the mind as a whole*, rather than confining themselves—like most of their colleagues—to one tiny corner of mentality.

For instance, Sloman’s ongoing *CogAff* project focuses on emotion’s role in the computational architecture of the mind. *CogAff* has influenced the LIDA model of consciousness, released in 2011 and still being extended (see Chapter 6). It has also inspired the MINDER program, initiated by Sloman’s group in the late 1990s.

MINDER simulates (the functional aspects of) the anxiety that arises within a nursemaid, left to look after several babies single-handedly. She/it has only a few tasks: to feed them, to try to prevent them from falling into ditches, and to take them to a first-aid station if they do. And she has only a few motives (goals): feeding a baby; putting a baby behind a protective fence, if one already exists; moving a baby out of a ditch for first aid; patrolling the ditch; building a fence; moving a baby to a safe distance from the ditch; and, if no other motive is currently activated, wandering around the nursery.

So she's hugely simpler than a real nursemaid (although more complex than a typical planning program, which has only one final goal). Nevertheless, she's prone to emotional perturbations comparable to various types of anxiety.

The simulated nursemaid has to respond appropriately to visual signals from her environment. Some of these trigger (or influence) goals that are more urgent than others: a baby crawling towards the ditch needs her attention sooner than a merely hungry baby, and one who's about to topple into the ditch needs it sooner still. But even those goals that can be put on hold may have to be dealt with eventually, and their degree of urgency may increase with time. So, a starving baby can be put back into its cot if another baby is near the ditch; but the baby who has waited longest for food should be nurtured before those fed more recently.

In short, the nursemaid's tasks can sometimes be interrupted, and either abandoned or put on hold. MINDER must decide just what the current priorities are. Such decisions must be taken throughout the session, and can result in repeated changes of behaviour. Virtually no task can be completed without interruption, because the environment (the babies) puts so many conflicting, and ever-changing, demands upon the system. As with a real nursemaid, the anxieties increase, and the performance degrades, with an increase in the number of babies—each of which is an unpredictable autonomous agent. Nevertheless, the anxiety is useful, for it enables the nursemaid to nurture the babies successfully. Successfully, but not *smoothly:* calm and anxiety are poles apart.

MINDER indicates some ways in which emotions can control behaviour, scheduling competing motives intelligently. A human nursemaid, no doubt, will experience various types of anxiety as her situation changes. But the point, here, is that emotions aren't merely *feelings*. They involve functional, as well as phenomenal, consciousness (see Chapter 6). Specifically, they are computational mechanisms that enable us to schedule competing motives—and without which we couldn't function. (So the emotionless Mr Spock of *Star Trek* is an evolutionary impossibility.)

If we are ever to achieve AGI, emotions such as anxiety will have to be included —and *used*.

Chapter 4

Artificial neural networks

Artificial neural networks (ANNs) are made up of many interconnected units, each one capable of computing only one thing. Described in this way, they may sound boring. But they can seem almost magical. They've certainly bewitched the journalists. Frank Rosenblatt's 'perceptrons', photoelectric machines that learned to recognize letters without being explicitly taught, were puffed enthusiastically in the 1960s newspapers. ANNs made an especially noisy splash in the mid-1980s, and are still regularly hailed in the media. The most recent ANN-related hype concerns deep learning.

ANNs have myriad applications, from playing the stock market and monitoring currency fluctuations to recognizing speech or faces. But it's *the way they work* that is so intriguing.

A tiny handful are run on specifically parallel hardware—or even on a hardware/wetware mix, combining real neurons with silicon circuits. Usually, however, the network is simulated by a von Neumann machine. That is, ANNs are parallel-processing virtual machines implemented on classical computers (see Chapter 1).

They are intriguing partly because they are very different from the virtual machines of symbolic AI. Sequential instructions are replaced by massive parallelism, top-down control by bottom-up processing, and logic by probability. And the dynamical, continuously changing aspect of ANNs contrasts starkly with symbolic programs.

Moreover, many networks have the uncanny property of self-organization from a random start. (The 1960s perceptrons had this too: hence their high press profile.) The system starts with a random architecture (random weights and profile.) The system starts with a random architecture (random weights and connections), and gradually adapts itself to perform the task required.

Neural networks have many strengths, and have added significant computational capabilities to AI. Nevertheless, they also have weaknesses. So they can’t deliver the truly *general* AI envisaged in Chapter 2. For instance, although some ANNs can do approximate inference, or reasoning, they can’t represent precision as well as symbolic AI can. (*Q: What’s 2 + 2? A: Very probably 4*. Really?) Hierarchy, too, is more difficult to model in ANNs. Some (*recurrent*) nets can use interacting networks to represent hierarchy—but only to a limited degree.

Thanks to the current enthusiasm for deep learning, networks of networks are less rare now than they used to be. However, they are still relatively simple. The human brain must comprise countless networks, on many different levels, interacting in highly complex ways. In short, AGI is still far distant.

The wider implications of ANNs

ANNs are a triumph of AI considered as computer science. But their theoretical implications go much further. Because of some general similarities to human concepts and memory, ANNs are of interest to neuroscientists, psychologists, and philosophers.

The neuroscientific interest isn't new. Indeed, the pioneering perceptrons were intended by Rosenblatt not as a source of practically useful gizmos, but as *a neuropsychological theory*. Today's networks—despite their many differences from the brain—are important in computational neuroscience.

Psychologists, too, are interested in ANNs—and the philosophers haven't been far behind. For instance, one mid-1980s example caused a furore well outside the ranks of professional AI. This network apparently learned to use the past tense much as children do, starting by making no mistakes but then over-regularizing—so that *go/went* gives way to *go/goed*—before achieving correct usage for both regular and irregular verbs. That was possible because the input that was provided to it mirrored the changing probabilities of the words typically heard by a child: the network *wasn't* applying innate grammatical rules.

This was important because most psychologists (and many philosophers) at the time had accepted Noam Chomsky's claims that children *must* rely on inborn linguistic rules in order to learn grammar, and that infantile over-regularizations were irrefutable evidence of those rules being put to work. The past-tense network proved that neither of these claims is true. (It didn't prove, of course, that children don't have innate rules: merely that they don't *need* to have them.)

Another widely interesting example, which was originally inspired by developmental psychology, is research on 'representational trajectories'. Here (as also in deep learning), input data that are initially confusing are recoded on successive levels, so that less obvious regularities are captured in addition to the prominent ones. This relates not only to child development, but also to psychological and philosophical debates about inductive learning. For it shows that prior expectations (computational structure) are needed in order to learn patterns in the input data, and that there are unavoidable constraints on the order in which different patterns are learned.

In short, this AI methodology is theoretically interesting in many ways, as well as being hugely important commercially.

Parallel distributed processing

One category of ANNs in particular attracts huge attention: those doing PDP. Indeed, when people refer to ‘neural networks’ or ‘connectionism’ (a term less often used today), they usually mean PDP.

Because of the way they work, PDP networks share four major strengths. These relate to both technological applications and theoretical psychology (and also to the philosophy of mind).

The first is their ability to learn patterns, and associations between patterns, by being shown examples instead of being explicitly programmed.

The second is their tolerance of ‘messy’ evidence. They can do *constraint satisfaction*, making sense of partially conflicting evidence. They don’t demand rigorous definitions, expressed as lists of necessary and sufficient conditions. Rather, they deal with overlapping sets of family resemblances—a feature of human concepts, too.

Yet another strength is their ability to recognize incomplete and/or partly damaged patterns. That is, they have *content-addressable* memory. So do people: think of identifying a melody from the first few notes, or played with many mistakes.

And fourth, they are robust. A PDP network with some nodes missing doesn’t spout nonsense, or halt. It shows *graceful degradation*, in which performance worsens gradually as the damage increases. So they aren’t brittle, as symbolic programs are.

These benefits result from the D in PDP. Not all ANNs involve distributed processing. In *localist* networks (such as *WordNet*: see Chapter 2), concepts are represented by single nodes. In *distributed* networks, a concept is stored across (distributed over) the whole system. Localist and distributed processing are sometimes combined, but that’s uncommon. Purely localist networks are uncommon too, because they lack the major strengths of PDP.

One could say that distributed networks are localist *at base*, for each unit corresponds to a single microfeature—for example, a tiny patch of colour, at a particular place in the visual field. But these are defined at a much lower level than are concepts: PDP involves ‘sub-symbolic’ computation. Moreover, each unit can be part of many different overall patterns, so contributes to many different ‘meanings’.

There are many types of PDP systems. All are made of three or more layers of interconnected units, each unit capable of computing only one simple thing. But the units differ.

A unit in the input layer fires whenever its microfeature is presented to the network. An output unit fires when it’s triggered by the units connected to it, and its activity is communicated to the human user. The hidden units, in the middle layer(s), have no direct contact with the outside world. Some are *deterministic*: they fire, or not, depending only on the influences from their connections. Others are *stochastic*: whether they fire depends partly on some probability distribution.

The connections differ, too. Some are *feedforward*, passing signals from a lower layer to a higher one. Some send *feedback* signals in the opposite direction. Some are *lateral*, linking units within the same layer. And some, as we’ll see, are both *feedforward* and *feedback*. Like brain synapses, connections are either excitatory or inhibitory. And they vary in strength, or *weight*. Weights are expressed as numbers between +1 and −1. The higher the weight of an excitatory (or inhibitory) link, the higher (or lower) the probability that the unit receiving the signal will fire.

PDP involves *distributed* representation, for each concept is represented by the state of the entire network. This may seem puzzling, even paradoxical. It’s certainly very different from how representations are defined in symbolic AI.

People interested only in technological/commercial applications don’t care about that. If they’re satisfied that certain obvious questions—such as how a single network can store several different concepts, or patterns—aren’t problematic in practice, they’re happy to leave it at that.

People concerned with the psychological and philosophical implications of AI ask that ‘obvious question’, too. The answer is that the possible overall states of a PDP network are so multifarious that only a few will involve simultaneous activation in *this* or *that* scattering of units. An activated unit will spread activation only to *some* other units. However, those ‘other units’ vary: any given unit can contribute to many different patterns of activation. (In general, ‘sparse’ representations, with many unactivated units, are more efficient.) The system will saturate eventually: theoretical research on associative memories asks how many patterns can, in principle, be stored by networks of a certain size.

But those engaged with the psychological and philosophical aspects aren’t happy to leave it at that. They are interested also in the concept of *representation* itself, and in debates about whether human minds/brains actually do contain internal representations. PDP devotees argue, for example, that this approach refutes the Physical Symbol System hypothesis, which originated in symbolic AI and rapidly spread into the philosophy of mind (see Chapter 6).

Learning in neural networks

Most ANNs can learn. This involves making adaptive changes in the weights, and sometimes also in the connections. Usually, the network's anatomy—the number of units, and the links between them—is fixed. If so, learning alters only the weights. But sometimes, learning—or evolution (see Chapter 5)—can add new links and prune old ones. *Constructive* networks take this to the extreme: starting with no hidden units at all, they add them as learning proceeds.

PDP networks can learn in many different ways—and exemplify all the types distinguished in Chapter 2: supervised, unsupervised, and reinforcement learning.

In supervised learning, for instance, they come to recognize a class by being shown various examples of it—none of which needs to possess *every* 'typical' feature. (The input data may be visual images, verbal descriptions, sets of numbers …) When an example is presented, some input units respond to 'their' microfeatures, and activations spread until the network settles down. The resulting state of the output units is then compared with the desired output (identified by the human user), and further weight changes are instigated (perhaps by *backprop*) so as to make those errors less probable. After many examples, differing slightly from each other, the network will have developed an activation pattern that corresponds to the typical case, or 'prototype', even if no such case has actually been encountered. (If a damaged example is now presented, stimulating many fewer of the relevant input units, this pattern will be completed automatically.)

Most ANN learning is based on the *fire together, wire together* rule, stated in the 1940s by the neuropsychologist Donald Hebb. Hebbian learning strengthens often-used connections. When two linked units are activated simultaneously, the weights are adjusted to make this more likely in future.

Hebb expressed the *ft/wt* rule in two ways, which were neither precise nor equivalent. Today's AI researchers define it in many different ways, based perhaps on differential equations drawn from physics, or on Bayesian probability theory. They use theoretical analysis to compare, and improve, the various versions. So, PDP research can be fiendishly mathematical.

Given that a PDP network is using some Hebbian learning rule to adapt its weights, when does it stop? The answer isn’t *When it has achieved perfection (all inconsistencies eliminated),* but *When it has achieved maximum coherence.*

An ‘inconsistency’ occurs, for instance, when two microfeatures that aren’t usually present together are simultaneously signalled by the relevant units. Many symbolic AI programs can do constraint satisfaction, approaching the solution by eliminating contradictions between evidence on the way. But they don’t tolerate inconsistency as part of the solution. PDP systems are different. As the PDP strengths listed earlier show, they can perform successfully even if discrepancies persist. Their ‘solution’ is the overall state of the network when inconsistencies have been minimized, not abolished.

One way of achieving that is to borrow the idea of *equilibrium* from thermodynamics. Energy levels in physics are expressed numerically, as are the weights in PDP. If the learning rule parallels the physical laws (and if the hidden units are stochastic), the same statistical Boltzmann equations can describe the changes in both cases.

PDP can even borrow the method used to cool metals rapidly but evenly. Annealing starts at a high temperature and cools down gradually. PDP researchers sometimes use *simulated annealing,* wherein the weight changes in the first few cycles of equilibration are much larger than those in later cycles. This enables the network to escape from situations (‘local minima’) where overall consistency has been achieved relative to what went before, but even greater consistency (and a more stable equilibrium) could be reached if the system were disturbed. Compare shaking a bag of marbles, to dislodge any marbles resting on an internal ridge: one should start by shaking forcefully, but end by shaking gently.

A faster, and more widely used, way of achieving maximal consistency is to employ backprop. But whichever of the many learning rules is employed, the state *of the whole network* (and especially of the output units), at equilibrium, is taken to be the representation of the concept involved.